

INTRODUCTION GENERALE

Un automate programmable industriel (ou API) est un dispositif électronique programmable destiné à automatiser des processus tels que la commande de machines au sein d'une usine et à piloter des robots industriels par exemple. L'automate programmable reçoit des données par ses entrées, celles-ci sont ensuite traitées par un programme défini, le résultat obtenu étant délivré par ses sorties. Ce cycle de traitement est toujours le même, quel que soit le programme, néanmoins le temps d'un cycle d'API varie selon la taille du programme et la puissance de l'automate.

C'est l'unité centrale qui gère l'automate programmable : elle reçoit, mémorise et traite les données entrantes et détermine l'état des données sortantes en fonction du programme établi.

Ce dispositif a l'avantage d'être composé d'éléments particulièrement robustes et possède d'énormes capacités d'exploitation. En contrepartie, il est beaucoup plus cher que les solutions informatiques classiques comme les micro-ordinateurs. L'automate programmable a été spécialement conçu pour l'industrie afin de pouvoir automatiser certains processus.

Dans notre travail nous avons proposé un sujet Ce projet permet l'étude des systèmes automatisés gérer par automate programmables S7-300. Conception et réalisation d'une maquette pour le labo d'automatisme, d'un carrefour, nécessite un bon fonctionnement avec le carrefour.

Ce travail est organisé sur quatre chapitre le premier est conçu pour les propriétés des automates programmables et le principe de fonctionnement, le deuxième chapitre consacré pour les différentes modules des step7 avec présentation des éléments de commande et

d'affichage. Le troisième chapitre et pour les caractéristiques des modules et leurs programmations et enfin nous avons exposé les simulations des résultats.

Comme perspectif nous proposons d'utilisés les automates pour commander les vérins des robots.

Chapitre 1

L'automate programmable industriel API

0. Introduction :

L'automate programmable industriel API (ou Programmable Logic Controller PLC) est aujourd'hui le constituant le plus répandu des automatismes. On le trouve non seulement dans tous les secteurs de l'industrie, mais aussi dans les services (gestion de parkings, d'accès à des bâtiments) et dans l'agriculture (composition et délivrance de rations alimentaires dans les élevages). Il répond aux besoins d'adaptation et de flexibilité de nombres d'activités économiques actuelles. Cette place majeure soulève bien sûr un certain nombre de questions. C'est à ces questions que nous allons essayer de répondre ici, en mettant en évidence: Ses caractéristiques propres, matérielles et logicielles ; sa capacité à s'intégrer dans un ensemble plus large, et donc à répondre aux besoins d'un système automatisé de production SAP.[1]

1. Besoins de l'automatisation :

L'évolution des techniques s'est traduite pour l'automatisation par :

— un développement massif ; une approche de plus en plus globale des problèmes ; une intégration dès la conception de l'installation.

On est ainsi passé du stade de la machine automatisée à celui du système automatisé de production. [1]

2. Place des automates programmables :

Dans ces systèmes de traitement de l'information, les API occupent une place de choix. Les équipements notés « commande » sont souvent des automates. Remplaçant initialement des ensembles en technologie câblée (relais électromagnétiques ou statiques, composants pneumatiques), ils constituent de plus en plus un maillon fiable et efficace entre le calculateur, qui a plutôt un rôle de gestion et l'appareillage de terrain (capteurs et actionneurs). Cet appareillage pouvant lui-même aujourd'hui contenir un processeur, il nous faut préciser la définition de l'API, tout en sachant que dans ce domaine comme dans beaucoup d'autres touchant à l'informatique, les frontières sont floues et mouvantes.

Nous considérerons comme automate programmable un système : construit autour d'un processeur numérique, spécifique ou non ; pouvant être relié à de nombreux signaux physiques ; fonctionnant grâce à une protection adaptée dans des conditions industrielles ; doté d'un logiciel de programmation permettant un traitement simple des variables booléennes (Tout ou Rien, TOR en abrégé); doté de possibilités d'échanges avec d'autres processeurs. Cela constitue un « noyau » susceptible de nombreuses extensions et compatible avec la définition assez lourde de la norme française EN 61131-1: « Système électronique fonctionnant de manière numérique, destiné à être utilisé dans un environnement industriel, qui utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur aux fins de mise en œuvre de fonctions spécifiques, telles que des fonctions de logique, de mise en séquence, de temporisation, de comptage et de calcul arithmétique, pour commander au moyen d'entrées et de sorties Tout-ou-Rien ou analogiques divers types de machines ou de processus. L'automate programmable et ses périphériques associés sont

conçus pour pouvoir facilement s'intégrer à un système d'automatisme industriel et être facilement utilisés dans toutes leurs fonctions prévues. »

À partir de ces définitions, nous distinguerons dans les fonctions que l'automate doit remplir : Un rôle de commande où il est un composant d'automatisme, élaborant des actions, suivant une algorithmique appropriée, à partir des informations que lui fournissent des détecteurs (Tout ou Rien) ou des capteurs (analogiques ou numériques) ; Un rôle de communication dans le cadre de la production : Avec des opérateurs humains : c'est le dialogue d'exploitation, avec d'autres processeurs, hiérarchiquement supérieurs (calculateur de gestion de production), égaux (autres automates intervenant dans la même chaîne) ou inférieurs (instrumentation intelligente). Se poseront aussi, pour le rendre ou le faire demeurer opérationnel, les problèmes de la programmation et de la maintenance. [1]

3. API, composant d'automatisme :

Ce rôle constitue sa base historique et sa spécificité par rapport à d'autres matériels. Pour étudier cet équipement connecté à des systèmes réels en milieu industriel, il nous faut passer en revue l'aspect matériel, l'aspect logiciel et, devant les conditions d'emploi, nous poser la question de la sûreté de fonctionnement. [1]

3.1. Matériel :

La structure matérielle interne d'un API obéit au schéma donné sur la figure, nous analyserons successivement chacun des composants qui apparaissent sur ce schéma

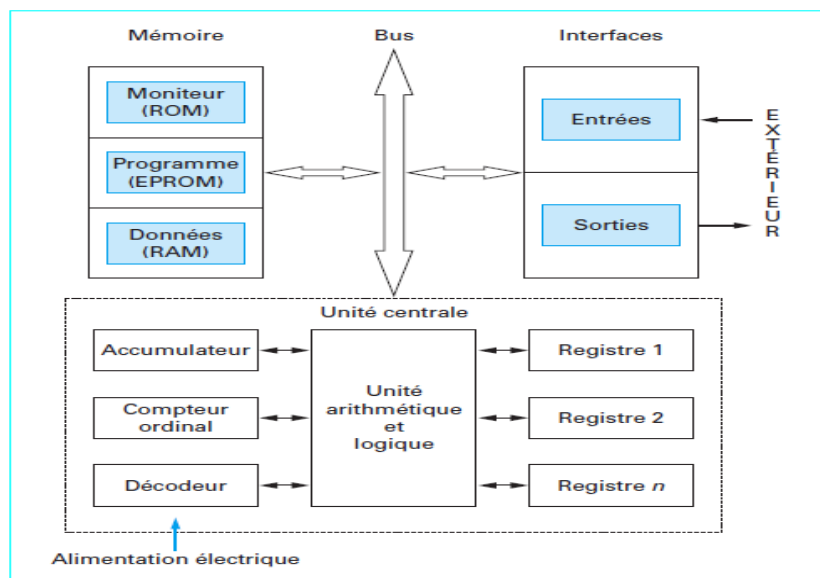


Figure0 : La structure matérielle interne d'un API [1]

3.1.1. Processeur :

Cœur de l'appareil, dans l'unité centrale, ce n'est sans doute pas, paradoxalement, le point le plus caractéristique, mais il conditionne tout de même largement les performances.

Les premiers API étaient équipées de processeurs spécifiques, à cycle de scrutation unique : on exécutait en permanence un programme gérant essentiellement des variables binaires.

On est passé ensuite à des processeurs plus performants, issus du monde de l'informatique. Cette évolution a permis de baisser les coûts, d'accroître les possibilités opérationnelles. Les processeurs « généralistes » étant toutefois ponctuellement moins efficaces que les spécialisés, on peut penser que c'est aussi une des raisons pour laquelle les progrès (en temps de calcul par exemple) sur un ensemble d'opérations de base ont été moins spectaculaires qu'en informatique générale, avec une conséquence heureuse pour les utilisateurs : une longévité supérieure du matériel. Des progrès importants ont été accomplis sur d'autres points, sur lesquels nous reviendrons.

L'unité centrale UC est une carte électronique bâtie autour de la (ou des) « puce(s) » processeur(s), qui assure au moins les fonctions suivantes : opérations logiques sur bits (le bit, contraction de « binary digit », étant l'information élémentaire à deux états) ou sur mots (ensemble de bits, le plus souvent 16 pour les API) ; temporisation et comptage.

Il existe trois technologies de réalisation : la technologie câblée, la plus rapide mais aussi la plus coûteuse, réservée à des usages particuliers ; la technologie à microprocesseur, la plus économique dès lors que l'on utilise un microprocesseur standard produit en grande série ; la technologie mixte, certaines opérations étant réalisées en câblé pour en accroître la rapidité.

Pour assurer la liaison entre l'UC et les cartes d'entrées/sorties, un réseau, certaines consoles ou unités de dialogue, il faut une carte électronique spécialisée d'interfaçage : le coupleur. Il existe au moins un coupleur de base, éventuellement des coupleurs vers d'autres châssis contenant des entrées/sorties supplémentaires. [1]

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

Ils assurent le rôle d'interface de la partie commande (PC) dans le schéma classique de la figure 2, qui distingue une partie opérative, où les actionneurs agissent physiquement sur le processus, et une partie commande récupérant les informations sur l'état de ce processus et coordonnant en conséquence les actions pour atteindre les objectifs prescrits (matérialisés par des consignes). [1]

Pour ce faire, ils doivent : regrouper des variables de même nature, pour diminuer complexité et coût ; assurer le dialogue avec l'UC ; « traduire » les signaux industriels en informations API et réciproquement, avec une protection de l'UC et un traitement adéquats. Beaucoup d'automates assurent cet interfaçage par des modules amovibles comportant un nombre fixé de voies d'un certain type (du point de vue nature informationnelle, mais aussi électrique, du signal). [1]

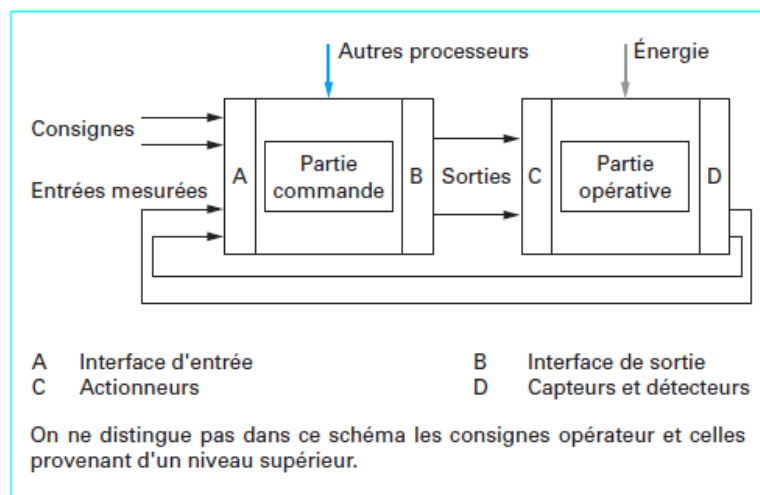


Figure1 : schéma classique modules d'E/S

D'autres automates, tels le TSX17 de Schneider Electric, ont une structure figée monobloc, avec des modules intégrés dans le châssis de base. Le nombre de voies de ces modules reflète alors le rapport moyen entre entrées et sorties dans les problèmes industriels, qui se traduit par un déséquilibre au profit des entrées. [1]

Le nombre total de modules est évidemment limité, pour des problèmes physiques (taille du châssis et/ou de l'alimentation électrique) ou de gestion informatique. La possibilité de configurer des voies d'accès en entrée ou en sortie est rarement utilisée, pour des raisons de sécurité. [1]

a. Modules de base

Il s'agit principalement des entrées/sorties Tout ou Rien (E/S TOR), la gestion de ce type de variables constituant le point de départ historique des API et restant une de leurs activités majeures. [1]

Dans le cas de modules séparés, la modularité dépend des constructeurs, les valeurs 8 et 16 se rencontrant fréquemment. On trouve des modules pour tensions continues (24 V, 48 V) et alternatives (24 V, 48 V, 100/120 V, 220/240 V), les valeurs de ces tensions étant conformes à la norme EN 61131-2. Pour les automates à E/S intégrées, les caractéristiques des entrées et sorties sont choisies parmi les standards les plus répandus (entrées 24 V isolées, sorties transistorisées à alimentation continue, ou entrées 110 V alternatif isolées, sorties relais, par exemple, pour le TSX17). La protection des circuits logiques d'entrée contre les parasites électriques est souvent résolue par découplage optoélectronique. Le passage des signaux par un stade de faisceau lumineux assure en effet une séparation entre les circuits électriques internes et externes. Du côté sorties, on doit assurer le même type de protection, mais aussi une amplification de puissance, avec au final un courant continu ou alternatif selon les cas. Deux types principaux de cartes électroniques sont utilisés :

— Les modules à sorties statiques (relais statiques intégrant des composants spécialisés : transistors bipolaires, IGBT, thyristors), sans usure mécanique et dont les caractéristiques de commutation se maintiennent dans le temps; les modules à relais électromagnétiques, où le découplage résulte de l'existence de deux circuits électriques (bobine – circuit d'excitation – et contacts – circuits de puissance –). D'une durée de vie plus limitée que les relais statiques (moins de 100 000 cycles pour les contacts soumis à 10 A sous 125 V alternatif) et plus lents, les relais électromagnétiques ont aussi des avantages : faible résistance de contact, faible capacité de sortie, faible coût. [1]

b. Modules spécialisés

Nous rangerons dans cette catégorie des cartes qui assurent non seulement une liaison avec le monde extérieur mais aussi une partie du traitement pour soulager le processeur et donc améliorer les performances. De tels modules comportent donc un processeur spécifique ou une électronique spécialisée. Nous pouvons citer les suivants. [1]

Cartes de comptage rapide : Elles permettent de saisir des événements plus courts que la durée du cycle, travaillant à des fréquences qui peuvent dépasser 10 kHz. Une des applications du comptage rapide, qui exige des cartes spécifiques, est le positionnement, expliqué ci-après. [1]

Cartes d'axe : Il s'agit de positionner un élément mécanique avec plus ou moins de vitesse et de précision en commandant la rotation d'un moteur électrique. Plusieurs cas sont à distinguer. Le positionnement par moteur pas à pas, en boucle ouverte (sans contrôle effectif de la position atteinte). Il faut alors, en réalité, compter les impulsions appliquées au moteur, jusqu'à obtenir théoriquement le nombre de pas désiré. Quelques sorties TOR (validation de

la génération d'impulsions, sens de rotation) sont nécessaires. Ce type de positionnement est le plus sommaire ; Le pilotage par variateur préréglé : le comptage y sert cette fois à enregistrer les impulsions d'un codeur incrémental lié à l'axe du moteur. Ce dernier est piloté via un variateur à plusieurs vitesses. L'algorithme choisit une des vitesses et la met en oeuvre par un jeu de sorties TOR réalisant un codage. Il n'y a pas de contrôle de l'accélération ;

— le pilotage continu par variateur reprend le même principe que précédemment mais le variateur reçoit une tension variant continûment suivant l'algorithme de commande, qui peut tenir compte de la charge, de l'accélération, d'autres facteurs encore. Cette solution est la plus performante mais aussi la plus coûteuse car exigeant une sortie analogique. [1]

Cartes d'acquisition : Sous cette appellation nous rangerons non seulement les acquisitions de mesures (entrées analogiques), mais aussi les sorties analogiques ; il existe des modules mixtes regroupant entrées et sorties. Dans les deux cas, il faut une conversion entre les valeurs analogiques et les mots manipulables par l'API. La valeur analogique correspond à un nombre de bits (8 à 12) tel que l'erreur de quantification correspondante, toujours donc $\ll 1\%$, s'avère généralement inférieure à la résolution des capteurs ou à l'effet de seuil sur la commande des actionneurs. Les grandeurs analogiques reçues ou fournies obéissent à des standards électriques dont les plus fréquemment rencontrés sont le 4-20 mA, transmission en courant qui évite l'affaiblissement du signal et permet de différencier le 0 d'échelle de la rupture de la liaison, et le 0-10 V.

À noter aussi que les modules analogiques présentent généralement plusieurs entrées (8, voire 16) multiplexées pour n'utiliser qu'un seul convertisseur analogique/numérique, alors que les sorties exigent un convertisseur numérique/analogique par voie. [1]

Entrées/sorties déportées : Leur intérêt est de diminuer le câblage en réalisant la liaison avec détecteurs, capteurs et actionneurs au plus près de ceux-ci, la liaison entre le boîtier déporté et l'unité centrale s'effectuant par une liaison spéciale moins gourmande en filerie, particulièrement si elle est de type série. Elles sont souvent raccordées aujourd'hui à l'automate par un réseau de terrain, suivant des protocoles bien définis et multifournisseurs à défaut d'être universels. L'utilisation de fibres optiques permet de porter la distance de déport à plusieurs kilomètres. [1]

On peut encore citer les cartes d'E/S numériques (pour des afficheurs ou des capteurs à sortie numérique), de détection de « fronts » (changements d'état d'un signal binaire), d'aide au dépannage – bien que les outils logiciels soient de plus en plus efficaces dans ce domaine –, de gestion de réseau, les cartes de traitement des codes-barres, les cartes à processeur autonome de régulation (PMX57 de Schneider Electric). [1]

3.1.3. Éléments de stockage et de liaison :

Le stockage des données et des programmes s'effectue dans des **mémoires**. La mémoire vive (RAM) est volatile mais secourue par batterie. La mémoire morte (ROM) dont l'utilisateur ne peut que lire le contenu, éventuellement programmable (PROM) à l'aide d'outils spéciaux, contient le système d'exploitation, tandis que les programmes au point et utilisables peuvent se stocker dans des mémoires reprogrammables (EPROM), là encore avec un matériel spécifique. La mémoire de données contient elle-même plusieurs zones :

- une zone de bits, dont certains peuvent être secourus en cas de défaillance de l'alimentation électrique ;
- une zone de mots, permettant de soumettre des données à un traitement plus large que le traitement booléen (traitement numérique ou alphanumérique).

La capacité de stockage d'une mémoire s'exprime en kilooctets (Ko) : 1 Ko = 1024 × 8 bits). Il faut connaître la capacité minimale utile de l'API, mais aussi la capacité maximale que l'on peut obtenir par diverses extensions. La mémoire des automates est très inférieure à celle des microordinateurs.

C'est la spécialisation des traitements qui permet de se satisfaire de telles capacités. Le système d'exploitation impose aussi des limites par type, dans la programmation comme dans les données. Les liaisons s'effectuent : Avec l'extérieur par des borniers (à visser, à clipser, etc.) sur lesquels arrivent des câbles transportant le signal électrique ; avec l'intérieur par des bus, liaisons parallèles entre les divers éléments, il peut y avoir plusieurs bus, car on doit transmettre des données, des états, des adresses. Les informations présentes sur le(s) bus sont souvent partiellement ou totalement codées, d'où des systèmes d'encodage et de décodage. [1]

3.1.4. Auxiliaires :

Il s'agit principalement : de l'alimentation électrique qui a pour rôle de fournir les tensions continues que nécessitent les composants (5 V, 12 V...) avec de bonnes performances, notamment face aux microcoupures du réseau. Sa source d'énergie est normalement le réseau électrique, parfois du 24 V continu. Il ne faut pas oublier que les châssis d'extension et les entrées/sorties déportées doivent aussi disposer d'une alimentation. Il est parfois nécessaire, pour lutter contre les perturbations électriques, d'introduire un transformateur d'isolement. C'est notamment le cas pour les raccordements à un réseau électrique à « neutre flottant ». Un onduleur évite les risques de coupure si celles-ci risquent de dépasser les tolérances admises; d'un ventilateur indispensable dans les châssis comportant de nombreux modules ou dans le cas où la température ambiante est susceptible de devenir assez élevée (plus de 40 °C); du support mécanique. Il peut s'agir d'un rack (structure métallique accueillant des cartes avec généralement un raccordement arrière), l'automate se présentant alors sous forme d'un ensemble de cartes, d'une armoire, d'une grille et des fixations correspondantes; d'indicateurs d'état concernant la présence de tension, l'exécution du programme (mode RUN), la charge de la batterie, le bon fonctionnement des coupleurs. [1]

3.2.Programmation :

C'est un des atouts majeurs des API puisqu'elle permet une multitude de traitements des informations reçues sans toucher à la configuration matérielle. Certaines modifications peuvent même s'effectuer alors que l'automate est en marche. Il faut toutefois comprendre le fonctionnement du processeur, qui impose certaines contraintes et choisir le langage le plus approprié dans le cadre du problème à résoudre. [1]

3.2.1. Déroulement du programme :

Il doit assurer en permanence un cycle opératoire qui comporte trois types de tâches

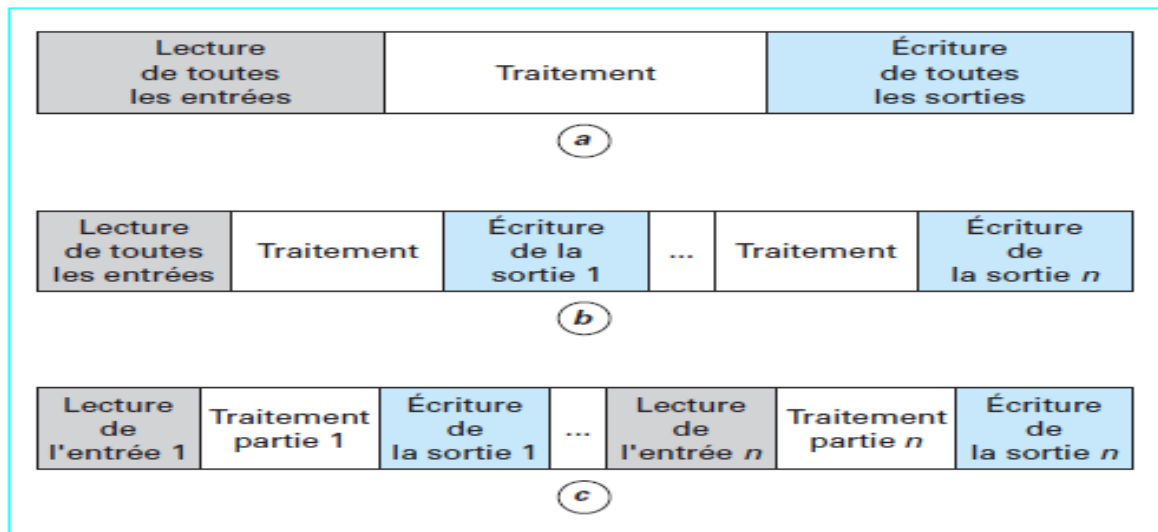


Figure2 : déroulement du programme

- L'acquisition de la valeur des entrées (lecture)
- le traitement des données
- l'affectation de la valeur des sorties (écriture)

Le premier et le troisième point s'effectuent par mise en communication successive de chacune des voies d'entrée ou de sortie (scrutation). L'exécution des instructions se fait dans l'ordre de leur rangement dans la mémoire programme ; l'incrément d'un compteur ordinal contenant initialement l'adresse du début de mémoire permet le passage à l'instruction suivante, sauf si un « saut avant » permet de se positionner plus loin, à une adresse plus élevée, en n'exécutant pas un ensemble d'instructions. [1]

Le cas le plus simple est celui où les tâches s'effectuent dans l'ordre indiqué figure2, cas **a**.

D'autres modes de gestion des tâches sont possibles (figure2, cas **b** et **c**). Ces procédures posent le problème du synchronisme ou de l'asynchronisme du fonctionnement de l'automate. Dans le premier cas, le système est dit synchrone par rapport aux entrées et aux sorties, ce qui signifie que tous les changements se font en même temps : pendant le traitement, les E/S conservent la même valeur. Dans les autres cas, une variable peut prendre différentes valeurs en différents points du programme. Par contre, ces solutions peuvent raccourcir le temps de réaction de l'automate à un événement, matérialisé par le changement d'état d'une variable TOR. Cela explique des aberrations apparentes telles le fait qu'une équation fournisse un résultat faux pendant un cycle ou une portion de cycle. Vis-à-vis de l'extérieur, la référence de cadence n'est pas l'horloge du processeur, beaucoup trop rapide, mais le temps de scrutation (temps de cycle). [1]

Le temps de scrutation, qui sépare deux lancements successifs de la même portion du programme d'application, comprend non seulement le temps de traitement mais aussi le temps de mise à jour de la mémoire des E/S. [1]

Le temps de réponse total (TRT) est le temps qui s'écoule entre un changement d'état du signal aux bornes d'entrée jusqu'au changement d'état correspondant aux bornes de sortie, vers l'installation. Il est la somme de tous les intervalles de temps nécessaires pour que des informations passent des bornes d'entrée aux bornes de sortie. Malheureusement, tous ces intervalles de temps peuvent varier en fonction de divers paramètres. [1]

Le temps de transfert du système d'entrée est le temps qui sépare l'apparition de l'information d'entrée (front montant sur le signal fourni par le détecteur) aux bornes d'entrée et son enregistrement dans la mémoire de l'automate. Il comprend le temps de passage par les filtres d'entrée et le temps de conversion. [1]

Le temps d'exécution dépend de la taille du programme, la plupart des machines pouvant ne lire que la partie de la mémoire programme utilisée par le programme en cours, mais change par exemple en cas de présence d'un saut avant ; il peut s'apprécier à partir d'une vitesse de référence exprimée en millisecondes par kilomots de la mémoire programme.

Le temps de transfert du système de sortie, séparant l'apparition des sorties dans la mémoire image de l'API de celle du signal sur les actionneurs, dépend notamment du type de sortie. Il est plus long pour les sorties à relais électromagnétiques. [1]

En dehors de la scrutation périodique, et de ses différentes modalités, il existe une possibilité de ne faire exécuter une portion de programme que dans certaines circonstances : c'est l'interruption. On a alors une exécution événementielle. Elle permet d'accélérer la réponse à une information jugée majeure – la coupure d'alimentation électrique en premier lieu – mais pose davantage de problèmes de gestion. [1]

3.2.2. Langages de programmation :

L'une des particularités des API est de se programmer dans des langages spécifiques adaptés à leur champ d'activités, orientés problème, souvent tournés prioritairement vers la manipulation de variables TOR ou numérisées. Les premiers langages étaient inspirés des schémas à contacts car s'adressant prioritairement à des personnels ayant reçu une formation d'électriciens. En France, dès le début des années 1980, sont apparus des langages issus du GRAFCET. Dans tous les cas régnait la plus grande hétérogénéité entre constructeurs, et même, chez un même constructeur, entre différentes séries. Le besoin de cohérence et la percée des outils informatiques (PC) à possibilités graphiques ont entraîné une évolution qui s'est traduite par la promulgation de la norme CEI 1131-3¹ reprise dans la norme française EN 61131-3. Celle-ci n'unifie pas les langages, n'assure pas la portabilité d'une machine à une autre car elle ne définit pas de classes de conformité, mais assure un minimum de clarté et instaure des règles dans ce domaine. Elle définit le logiciel suivant trois entités, présentées ciaprès dans un ordre ascendant : La fonction (par exemple, le ET booléen, la comparaison de valeurs numériques), à sortie unique ; le bloc fonction, composant logiciel réutilisable dans tous les langages (par exemple, un compteur), qui peut comporter plusieurs sorties ; le programme. Elle fixe le mode de représentation des variables, des constantes numériques, des dates et durées, des fonctions, des commentaires non exécutables facilitant la lecture du programme, etc. Elle autorise quatre types de langages, deux littéraux, deux graphiques, ainsi que des « éléments de schéma » structurant les programmes (c'est à ce niveau qu'apparaît le GRAFCET). [1]

3.2.2.1. Liste d'instructions (IL Instruction List) :

C'est un langage textuel, qui rappelle par certains aspects l'assembleur employé pour la programmation des microprocesseurs. Une instruction débute sur une ligne, comporte un opérateur, un ou plusieurs opérandes. On peut introduire des étiquettes et des commentaires. La structure des champs est donc la suivante :

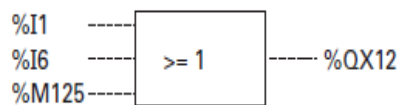
ÉTIQUETTE (facultatif)	OPÉRATEUR	OPÉRANDE(S)	COMMENTAIRE (facultatif)
---------------------------	-----------	-------------	-----------------------------

intervient implicitement le temps, les sorties dépendant des entrées mais aussi de leur état précédent). [1]

3.2.2.4. Blocs fonctionnels (FBD Function Bloc Diagram) :

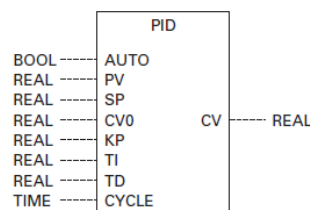
Cette représentation obéit aux mêmes lois fondamentales que la précédente, mais les blocs fonctionnels sont désormais la règle. Graphiquement, les entrées d'un bloc doivent se trouver à gauche, les sorties à droite.

Ainsi trois contacts en parallèle dans le LD peuvent se remplacer par un bloc OU généralisé, avec un symbole déjà employé pour les opérateurs câblés :



Les blocs traduisent une analyse du procédé, permettant une structuration hiérarchisée où se retrouve en quelque sorte la structure de l'application, voire une approche obéissant à la même inspiration que la programmation objet. [1]

Les blocs fonctionnels peuvent être fournis par une bibliothèque ou personnalisés par l'utilisateur. Ces derniers peuvent contenir eux-mêmes des blocs fonctionnels élémentaires, du langage à contacts ou des listes d'instructions. Un des intérêts de ce langage réside dans la réduction du nombre d'erreurs, par l'utilisation de blocs validés. Le réemploi de sous-ensembles se trouve également facilité. Ainsi, le bloc fonctionnel PID (Proportionnel, Intégral, Dérivé), représenté ci-après, et qui fait apparaître le type de variable à employer sur chaque entrée/sortie, s'obtient-il par combinaison fonctionnelle de blocs plus restreints, correspondant par exemple à chacune des composantes P, I et D et peut-il être combiné à un autre pour former un bloc « régulation cascade » :



3.2.2.5. Diagramme fonctionnel en séquences (SFC Sequential Function Chart) :

Il s'agit de ce que l'on dénomme couramment les langages « GRAFCET ». Ils sont particulièrement adaptés à la commande de cycles opératoires. En France et maintenant dans d'autres pays, ils s'accordent à une culture GRAFCET que possèdent beaucoup d'utilisateurs potentiels. Il ne faudrait pas en toute rigueur parler d'un langage, mais d'une structuration, d'une segmentation d'un programme en un ensemble d'étapes et de transitions. Graphiquement, on se trouve devant une alternance étape(s)-transition-étape(s) etc., avec des liaisons dirigées (le flux traduisant l'évolution par l'activation d'étapes circulant cette fois de haut en bas). À chaque transition est associée une condition de franchissement de cette transition ; à chaque étape peuvent être définies des actions à entreprendre. Conditions de franchissement et actions s'expriment dans un des langages décrits ci-avant. Les langages

GRAFCET [R 7 250], qu'il vaut mieux appeler langages inspirés du GRAFCET (ils n'appartiennent pas au même univers : l'un est un modèle de spécification, les autres des langages conduisant à une exécution qui elle-même se traduit par des actions physiques), tels PL7 de Schneider Electric et GRAPH 5 proposé par Siemens, respectent avec plus ou moins de bonheur l'ergonomie de cet outil de description, mais ne peuvent respecter strictement les hypothèses temporelles associées : la durée de franchissement d'une transition ne peut être « aussi petite que l'on veut », pas plus que deux événements distincts ne peuvent toujours être séparés, au niveau prise en compte, par un intervalle de temps. Il faut aussi noter, en ce qui concerne la traduction du GRAFCET, que les algorithmes d'interprétation, pour simplifier le traitement informatique, sont généralement du type SRS (sans recherche de stabilité) : si, à un moment donné, l'état des entrées et des variables internes permet le franchissement consécutif de plusieurs transitions, ce franchissement se fera au rythme d'une transition par cycle machine, ce qui pose problème si l'environnement change avant le franchissement de la dernière transition. La plupart des constructeurs, d'ailleurs membres des comités de normalisation, offrent plusieurs possibilités de langages sur une même machine et même parfois les « panachent » : sur les automates TSX, dans le langage PL7, les fonctions auxiliaires (réceptivités, actions) d'une application décrite par un grafcet s'écrivent en langage à contacts. [1]

3.2.3. Fonctions disponibles dans les langages

On peut les répartir comme suit :

Fonctions sur bits : On trouve là les fonctions booléennes combinatoires et notamment la trilogie NON, ET, OU, ces deux dernières généralisables à plus de deux entrées. Le DILEMME (OU exclusif, XOR) est parfois présent. Presque toujours aussi existe au moins une fonction séquentielle du type bascule Set/Reset (mémoire à deux ordres, marche et arrêt) ; attention dans ce cas à la priorité lorsque les deux ordres sont simultanément à 1. On peut y rattacher aussi le saut conditionnel sur valeur d'un bit, qui positionne le compteur ordinal à l'adresse d'une étiquette donnée lorsque l'expression booléenne associée vaut 1.

Fonctions sur mots : Il s'agit de la comparaison (recherche de l'égalité, parfois du plus grand ou plus petit), du décalage bit à bit (avec introduction ou non d'une nouvelle valeur), parfois du masquage, d'opérations (OU) bit à bit sur mots.

Fonctions de comptage : Le comptage est une des activités majeures des automates ; outre ses applications directes (détermination d'un nombre de pièces, de couches, de défauts...) il a des applications en positionnement, que nous avons déjà vues, et dans la gestion du temps, car mis à part les temporisateurs analogiques qui complètent l'équipement de certains automates, les temporisateurs fonctionnent par comptage des impulsions d'une base de temps interne à cadence variable (minute, seconde, 1/10e de seconde, 1/100e de seconde...) : en fixant le nombre d'impulsions attendu (présélection), on peut ainsi retarder le début ou la fin d'une action.

Fonctions numériques : C'est là qu'on va trouver les fonctions algébriques (addition, totalisation, multiplication), mais aussi les fonctions de régulation.

Celles-ci se présentent souvent sous forme de blocs fonctionnels ; le classique PID est maintenant rejoint par d'autres possibilités : régulation en tendance, commande prédictive, par exemple. Il faut encore mentionner les fonctions de traitement auxiliaires telles que lissage, linéarisation des thermocouples, extraction de racine carrée pour mesure de débit, qui accompagnent le traitement des mesures analogiques. On n'a pas décrit dans ce paragraphe les fonctions spécifiques à un langage (fonctions GRAFCET, instructions IF ... THEN des langages ST, etc.), pas plus que les fonctions de communication. [1]

3.3. Sûreté de fonctionnement :

La sûreté de fonctionnement qui préoccupe l'utilisateur est celle de l'installation dans son ensemble. L'API étant un élément déterminant dans cet ensemble, on doit examiner sa propre sûreté de fonctionnement.

Si elle est très grande, il pourra même intervenir dans la gestion de la sûreté de l'ensemble. Sachant que la sûreté de fonctionnement dépend de la disponibilité et de la sécurité nous étudierons successivement :

- les conditions d'emploi, facteur essentiel de la disponibilité ;
- la sécurité propre de l'API ;
- les automates de sécurité. [1]

3.3.1. Conditions d'emploi :

La norme CEI 1131-2 (norme française EN 61131-2) précise les conditions extrêmes de fonctionnement que doit supporter un API : En température, 5 à 40 °C pour un équipement fermé, même ventilé, 5 à 55 °C pour un équipement ouvert; en humidité, 50 à 95 % d'humidité relative, sans condensation ; en alimentation électrique des tensions variant au plus de -15 à +20 % en amplitude pour les alimentations continues, de -10 à +15 % pour les alternatives, de -5 à +5 % en fréquence, toujours par rapport aux valeurs nominales, les coupures d'alimentation ne doivent pas dépasser 10 ms (à 50 Hz) avec au moins 1 s entre 2 coupures. La norme définit aussi, quantitativement, la tenue aux vibrations, la tenue aux chocs, la sauvegarde mémoire. Elle précise, pour les modules d'entrée/sortie TOR, les tensions et courants nominaux ; de même pour les E/S analogiques, mais elle oblige aussi les constructeurs à fournir dans ce cas des indications concernant l'échantillonnage et les conversions indispensables dans ce traitement. Enfin, comme pour les autres matériels électriques, la compatibilité électromagnétique doit maintenant être étudiée de près. Dans le cas des API, ces appareils, considérés comme récepteurs, peuvent être affectés par des champs électromagnétiques rayonnants et par des décharges électrostatiques. Suivant la protection contre les risques d'électrocution, les API sont répartis en trois classes I, II, III. La norme définit dans ce cas, mais aussi dans plusieurs des précédents, les méthodes d'essai. [1]

3.3.2. Sécurité propre de l'API :

On a longtemps reproché aux API – et aux systèmes programmables en général – de manquer de sécurité intrinsèque. Ce problème important mérite d'être examiné de près.

Les reproches faits à l'API portent sur deux points :

- Le nombre de sorties, donc d'actions, affectées par une défaillance, peut être très important.
- les conséquences d'une défaillance sont parfois imprévisibles, contrairement à certains organes TOR dont on connaît au moins, avec une très grande probabilité, la position lors d'une panne. C'est pourquoi les normes de sécurité exigent que les arrêts d'urgence agissant sur la partie opérative, le plus souvent par coupure des sources d'énergie, soient réalisés en technologie câblée. On notera au demeurant qu'aujourd'hui une information relative à ces arrêts est renvoyée vers l'API.

Ces problèmes concernent tous les systèmes à processeur et on peut remarquer qu'ils n'ont pas de solution parfaite pour toutes les installations à forte proportion de traitement numérique et de régulation, quasiment toujours effectués par des processeurs numériques.

Il faut en relativiser la portée en remarquant que :

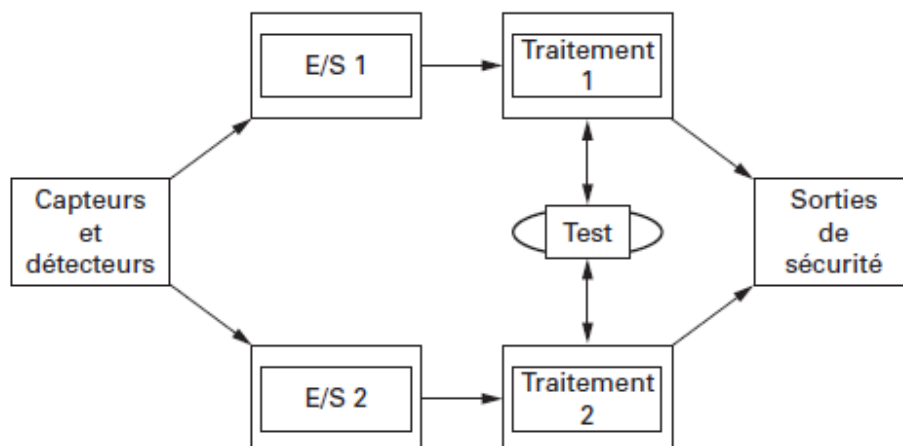
- Les défaillances de l'unité centrale, les plus graves par leurs conséquences possibles, sont très rares (moins de 5 % des défaillances de la partie commande, plus rares déjà que celles de la partie opérative) ;
- l'API possède des mécanismes de test
- le chien de garde surveille la durée du cycle de scrutation et enclenche une procédure d'alarme si le programme n'a pas « rebouclé » au bout d'un temps prédéterminé,
- mémoires, batterie, tensions d'alimentation, fonctionnement des horloges internes, couplage avec les E/S sont testés et un défaut fait l'objet d'un affichage par voyants sur le plastron de l'automate, avec lancement possible de programmes de repli assurant une marche dégradée de l'installation avec les « meilleures » valeurs possibles pour les sorties.

La sécurité du programme est du ressort de l'utilisateur et ne peut faire l'objet de tests indépendants de l'application. Notons que des simulateurs de parties opératives (PO), effectivement reliés à l'automate, permettent, sans danger pour le personnel et le matériel, de réaliser a priori de nombreux tests, faisant gagner à la fois en temps de mise au point et en sécurité. [1]

3.3.3. Automates de sécurité

Les capacités de test et de gestion de l'API amènent par contre à se poser la question de son utilisation pour contrer les défaillances survenant dans la PO ou dans des interfaces, et qui sont rappelons les plus nombreuses. D'où l'apparition d'automates dédiés à la sécurité. Ceux-ci possèdent une fiabilité supérieure à la normale : par la redondance de l'unité centrale (UC) assortie d'un mécanisme de comparaison des résultats permettant de détecter la défaillance d'une des UC ; par la redondance d'autres éléments essentiels tels les modules d'entrée, les coupleurs... avec test par le bloc UC, ainsi que l'illustre la figure 3 ; par la mise en oeuvre périodique de programmes élaborés d'autotest, capables de mettre en évidence des pannes dormantes, c'est-à-dire dont les effets ne sont pas encore apparus.

À partir de ces machines, on peut gérer toutes les fonctions de sécurité indirectes : gestion des accès à des zones dangereuses, analyse et prévention des défauts de la PO... donc améliorer fiabilité et sécurité du système de production. La question de l'utilisation en sécurité directe – celle qui concerne la protection des personnes – pose davantage de problème et fait l'objet de discussions au sein des organismes spécialisés tels l'Institut national de recherche sur la sécurité, surtout à cause des difficultés à tester les logiciels applicatifs, vu la multiplicité des situations possibles[1]



- Figure 3 : principe de l'automate de sécurité

4. Conclusion :

L'automate est un bon produit, facile à programmer, à connecter, adapté aux conditions industrielles. L'expansion considérable de ses possibilités, et celle corrélative de son marché, le prouvent. Il ne faut pas vouloir en faire une solution miracle. Dans tous les cas : Une bonne analyse du problème à résoudre ; Le respect des règles d'installation ; Un léger surdimensionnement pour préserver des marges de modification, sont les conditions d'une implantation réussie, dont la durée de vie dépassera largement celles habituelles dans le monde informatique, dont l'API est pour partie issu. [1]

CHAPITRE 2
SIMATIC S7-300
CPU 31XC ET CPU 31X
ET CARACTERISTIQUES DES
MODULES

I- SIMATIC S7-300 CPU 31Xc et CPU 31x

0. Introduction :

Le contrôleur Programmable Logique (PLC) est couramment utilisé dans les lignes automatisées industrielles dans de nombreuses applications telles que : l'emballage, l'extrusion, le tri, la manutention ... etc.

Automates sont disponibles auprès de nombreux fabricants tels que : Siemens, Allen Bradley, Mitsubishi, Schneider et Omron. Et chacune de ces entreprises a de nombreux modules pour le PLC.[2]

Pendant le reste du semestre, vous étudierez les modèles de stations industriels assemblés par Festo Société. Ce poste peut travailler ensemble pour réaliser un processus industriel complet. Toutes les stations sont fournies par Siemens SIMATIC S7-300 qui est programmé en utilisant le logiciel SIMATIC STEP 7. [2]

1. Eléments de commande et d'affichage : CPU 312, 314, 315-2 DP :

1.1. Eléments de commande et de signalisation :

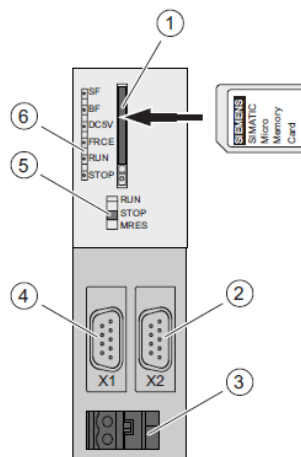


Figure4 : élément de commande et de signalisation CPU 312, 314,315-2DP [2]

Chiffre Désignation :

- ① Logement de la micro-carte mémoire SIMATIC avec éjecteur
- ② 2. ème interface X2 (uniquement CPU 315-2 DP)
- ③ Raccordement de la tension d'alimentation
- ④ 1. ère interface X1 (MPI)
- ⑤ Commutateur de mode de fonctionnement
- ⑥ Indicateurs d'état et d'erreur[2]

Logement de la micro-carte mémoire SIMATIC :

Une micro-carte mémoire SIMATIC est utilisée comme cartouche mémoire. Elle peut faire office de mémoire de chargement et de support de données amovible.

Remarque ces CPU ne possédant pas de mémoire de chargement intégré, vous devez insérer une Micro Memory Card SIMATIC dans la CPU pour le fonctionnement. [2]

Commutateur de mode de fonctionnement :

Le mode de fonctionnement de la CPU est réglé grâce au commutateur de mode de fonctionnement. [2]

Tableau 1 Positions du commutateur de mode de fonctionnement

Position	Signification	Explications
RUN	Mode de fonctionnement RUN	La CPU traite le programme utilisateur.
STOP	Mode de fonctionnement STOP	La CPU ne traite aucun programme utilisateur.
MRES	Effacement général	Position du commutateur de mode de fonctionnement pour l'effacement général de la CPU. L'effacement général à l'aide du commutateur de mode de fonctionnement nécessite une séquence d'actions particulière de votre part.

- Tableau1 : Positions du commutateur de mode de fonctionnement

Raccordement de la tension d'alimentation :

Chaque CPU dispose d'une prise à 2 points pour le raccordement de la tension d'alimentation. A l'état de livraison, le connecteur avec raccords filetés est déjà enfiché sur cette prise. [2]

1.2. Signalisations d'état et d'erreur de la CPU 31x :

Indicateurs généraux d'état et d'erreur

Tableau 2 Indicateurs d'état et d'erreur généraux de la CPU 31x [2]

Désignation de LED	Couleur	Signification
SF	rouge	Erreur matérielle ou logicielle.
DC5V	verte	Alimentation 5V de la CPU et du bus S7-300.
FRCE	jaune	DEL allumée : tâche de forçage permanent active LED clignote à 2 Hz : fonction de test de clignotement du participant
RUN	verte	CPU en mode RUN. La LED clignote à 2 Hz au démarrage, à 0,5 Hz en attente.
STOP	jaune	CPU en mode STOP ou HALT ou démarrage. La LED clignote à 0,5 Hz lorsque la CPU requiert un effacement général, et à 2 Hz pendant l'effacement général.

- Tableau2 : Indicateurs d'état et d'erreur généraux de la CPU 31x

2. Communication :

2.1. Interface multipoint (MPI) :

Disponibilité :

Toutes les CPU décrites disposent d'une interface MPI.

Si votre CPU possède une interface MPI/DP, elle sera paramétrée comme interface MPI à la livraison. [2]

Propriétés :

L'interface multipoint (MPI) est l'interface de la CPU avec un PG/OP ou pour la communication dans un sous-réseau MPI.

La vitesse de transmission par défaut pour toutes les CPU est de 187,5 kbauds. Pour la communication avec un S7-200, vous pouvez également régler 19,2 kbauds. Des vitesses de transmission allant jusqu'à 12 Mbauds au maximum sont possibles pour les CPU 315-2 PN/DP, CPU 317-2 et pour la CPU 319-3 PN/DP. La CPU envoie automatiquement à l'interface MPI ses paramètres de bus réglés (p. ex. la vitesse de transmission). Ainsi, une console de programmation peut, par exemple, avoir les bons paramètres et se connecter automatiquement à un sous-réseau MPI. [2]

Appareils raccordables via MPI :

- PG/PC
- OP/TP
- S7-300/S7-400 avec interface MPI
- S7-200 (uniquement avec 19,2 kbauds)

Synchronisation d'horloge : La synchronisation d'horloge est possible via l'interface MPI de la CPU. La CPU peut être paramétrée comme maître d'horloge (avec intervalles de synchronisation par défaut) ou comme esclave d'horloge. Paramétrage par défaut : pas de synchronisation d'horloge. Vous modifiez le paramétrage du type de synchronisation dans la boîte de dialogue des propriétés de la CPU ou de l'interface (onglet "Horloge") de HW Config. [2]

CPU comme horloge esclave : En tant qu'esclave d'horloge, la CPU reçoit des télégrammes de synchronisation d'un maître d'horloge et un seul et elle utilise cette heure comme sa propre heure interne. [2]

CPU comme horloge maître : En tant que maître d'horloge, la CPU envoie sur l'interface MPI, selon l'intervalle de synchronisation paramétré, des télégrammes afin de synchroniser d'autres stations dans le sous-réseau MPI raccordé.

Condition : l'horloge de la CPU ne doit plus se trouver à l'état par défaut. Elle doit être mise à l'heure une fois. [2]

La synchronisation d'horloge comme maître d'horloge démarre :

- Dès que vous réglez l'heure la première fois à l'aide de SFC 0 "SET_CLK" ou d'une fonction PG,
- Par un autre maître d'horloge si la CPU est paramétrée aussi comme esclave d'horloge via l'interface MPI/DP ou PROFINET. [2]

Interfaces pour la synchronisation d'horloge :

La synchronisation d'horloge est possible aux interfaces suivantes :

- Sur l'interface MPI
- Sur l'interface DP
- Sur l'interface PROFINET
- Dans le système d'automatisation en configuration centralisée [2]

2.2. Communication PG :

Propriétés :

La communication PG permet de réaliser l'échange de données entre les stations d'ingénierie (par exemple PG, PC) et les modules SIMATIC aptes à la communication. Le service est possible via les sous-réseaux MPI, PROFIBUS et Industrial Ethernet. Le passage entre les différents sous-réseaux est également pris en charge. [2]

La communication PG met à disposition des fonctions qui sont nécessaires pour charger les programmes et les données de configuration, exécuter les tests et évaluer les informations de diagnostic. Ces fonctions sont intégrées dans le système d'exploitation des Modules SIMATIC S7. Une CPU peut maintenir simultanément plusieurs liaisons en ligne avec un ou différents PG. [2]

2.3. Communication OP

Propriétés :

La communication OP permet de réaliser l'échange de données entre les stations opérateur (par exemple OP, TP) et les modules SIMATIC aptes à la communication. Le service est possible via les sous-réseaux MPI, PROFIBUS et Industrial Ethernet.

La communication OP met à disposition toutes les fonctions nécessaires au contrôle-commande.

Ces fonctions sont intégrées dans le système d'exploitation des modules S7 SIMATIC. Une CPU peut maintenir simultanément plusieurs liaisons en ligne avec un ou différents OP. [2]

2.4. Les données échangées via la communication de base S7 :

Propriétés :

La communication de base S7 permet de réaliser l'échange de données entre les CPU S7 et les modules SIMATIC aptes à la communication au sein d'une station S7 (échange de données acquitté). L'échange de données s'effectue par des liaisons S7 non configurées. Le service est possible par le sous-réseau MPI ou dans la station avec les modules de fonction (FM).

La communication de base S7 met à disposition toutes les fonctions nécessaires à l'échange de données. Ces fonctions sont intégrées au système d'exploitation des CPU. L'utilisateur peut utiliser le service via l'interface utilisateur "Fonction système" (SFC). [2]

3. Liaisons S7 :

3.1. Liaison S7 en tant que chemin de communication

Si les modules S7 communiquent entre eux, il s'établit ce que l'on appelle une liaison S7 entre les modules. Cette liaison S7 constitue la voie de communication.

Chaque liaison nécessite des ressources de liaison S7 sur la CPU, et ce pour la durée pendant laquelle cette liaison existe.

C'est pourquoi un certain nombre de ressources S7 sont mises à disposition sur chaque CPU S7 ; ces ressources sont occupées par différents services de communication (communication PG/OP, communication S7 ou communication de base S7). [2]

Points de liaison :

Une liaison S7 entre modules aptes à la communication s'établit entre des points de liaison. La liaison S7 possède toujours deux points de liaison, le point actif et le point passif.

- Le point de liaison actif est affecté au module qui établit la liaison S7.
- Le point de liaison passif est affecté au module qui reçoit la liaison S7.

Chaque module apte à la communication peut alors être le point de liaison d'une liaison S7.

Sur le point de liaison, la liaison de communication établie occupe alors toujours une liaison S7 du module concerné. [2]

Point de transition :

Si vous utilisez la fonctionnalité Routage, la liaison S7 entre deux modules aptes à la communication est établie via plusieurs sous-réseaux. Ces sous-réseaux sont reliés entre eux par une passerelle. Le module qui réalise cette passerelle est appelé routeur. Le routeur est ainsi le point de transition d'une liaison S7.

Chaque CPU dotée d'une interface DP ou PN peut être un routeur pour une liaison S7. Vous pouvez établir un nombre déterminé de liaisons par routage. Les capacités fonctionnelles des liaisons S7 ne s'en trouvent pas restreintes. [2]

4. Concept de mémoire

4.1. Zones de mémoire et rémanence

4.1.1. Zones de mémoire de la CPU

Les trois zones de mémoire de votre CPU

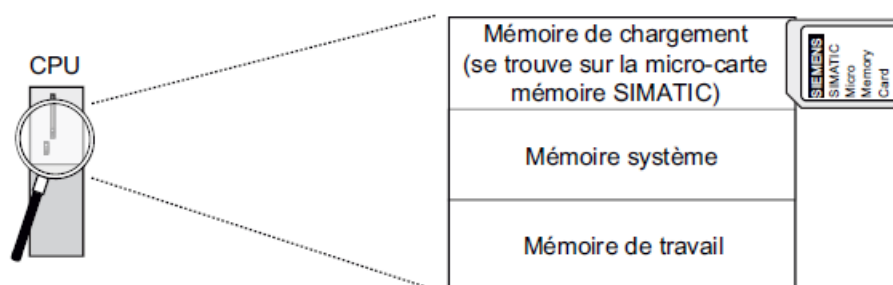


Figure5 : Les trois zones de mémoire de CPU

Mémoire de chargement : La mémoire de chargement se trouve sur la micro-carte mémoire SIMATIC et correspond exactement à la taille de cette carte. Elle sert à mémoriser les blocs de codes et les blocs de données ainsi que les données système (configuration, liaisons, paramètres de modules etc.). Les blocs qui sont repérés comme n'intervenant pas dans l'exécution sont copiés uniquement dans la mémoire de chargement. En plus, il est possible de stocker toutes les données de configuration d'un projet sur la micro-carte mémoire SIMATIC. [2]

Remarque : Le chargement de programmes utilisateur et donc l'utilisation de la CPU ne sont possibles que si vous avez enfiché une micro-carte mémoire SIMATIC dans la CPU. [2]

Mémoire système : La mémoire système est intégrée à la CPU et ne peut pas être étendue. Elle contient.

- Les plages d'opérandes Mémentos, Temporisations et Compteurs
- Les mémoires images des entrées et des sorties
- Les données locales

Mémoire de travail : La Mémoire de travail est intégrée à la CPU et ne peut pas être étendue. Elle sert à exécuter le code et à traiter les données du programme utilisateur. Le traitement du programme s'effectue exclusivement au niveau de la mémoire de travail et de la mémoire système. [2]

4.1.2. Rémanence de la mémoire de chargement, système et vive :

Votre CPU possède une mémoire rémanente ne nécessitant pas de maintenance, ce qui veut dire que vous n'avez pas besoin de pile de sauvegarde. Grâce à la rémanence, le contenu de la mémoire rémanente est conservé, même suite à une MISE HORS TENSION et un démarrage à chaud. [2]

Données rémanentes dans la mémoire de chargement : Votre programme dans la mémoire de chargement est toujours rémanent : dès le chargement, il est stocké sur la micro-carte mémoire SIMATIC, ce qui le met à l'abri des coupures de courant et des effacements généraux. [2]

Données rémanentes dans la mémoire système : Pour les mémentos, les temporisations et les compteurs, vous déterminez par la configuration (propriétés de la CPU, onglet rémanence) quelles parties doivent être rémanentes et quelles parties doivent être initialisées avec "0" en cas de démarrage à chaud. Les tampons de diagnostic, l'adresse MPI (et la vitesse de transmission) ainsi que les compteurs d'heures de fonctionnement sont généralement stockés dans la zone de mémoire rémanente sur la CPU. La rémanence de l'adresse MPI et de la vitesse de transmission garantissent que votre CPU reste apte à la communication après une panne de secteur, un effacement général ou une perte des paramètres de la communication (par débrogage de la micro-carte mémoire SIMATIC ou par effacement des paramètres). [2]

Données rémanentes dans la mémoire de travail : Le contenu des DB rémanents reste rémanent en cas de redémarrage et de MISE HORS TENSION/SOUS TENSION. Les blocs de données rémanentes peuvent être chargés dans la mémoire de travail jusqu'à la limite de maximale de rémanence de cette dernière.

Pour les CPU à partir de la version V2.0.12, les DB non rémanents sont également pris en charge. Lors d'un démarrage ou d'une mise hors puis sous tension, les DB non rémanents sont initialisés avec leurs valeurs initiales de la mémoire de chargement. Les blocs de données et

blocs de code non rémanents peuvent être chargés jusqu'à la limite maximale de la mémoire de travail. [2]

Les CPU	Taille de la mémoire rémanente (pour blocs de données rémanents)
CPU 312	32 Ko
CPU 313, 314	64 Ko
CPU 315	128 Ko
CPU 317	256 Ko
CPU 319	700 Ko

Tableau3 : Rémanence de la mémoire de travail

4.1.3. Rémanence des objets mémoire :

Comportement de rémanence des objets de mémoire :

Le tableau suivant présente le comportement de rémanence des objets de mémoire pour les changements des différents états de fonctionnement. [2]

Objet mémoire	Changement de l'état de fonctionnement		
	MISE SOUS TENSION / MISE HORS TENSION	STOP → RUN	Effacement général
Programme/données utilisateur (mémoire de chargement)	X	X	X
Comportement de rémanence des DB pour les CPU avec version de firmware < V2.0.12	X	X	-
Comportement de rémanence des DB pour les CPU avec version de firmware >= V2.0.12	Réglable dans les propriétés des DB dans STEP 7 à partir de V5.2 + SP1.		-
Mémentos, temporisations et compteurs configurés rémanents	X	X	-
Tampon de diagnostic, compteur d'heures de fonctionnement	X	X	X
Adresse MPI, vitesse de transmission (ou adresse DP, vitesse de transmission de l'interface MPI/DP des CPU 315-2 PN/DP, CPU 317 et CPU 319 quand elles sont paramétrées comme partenaire DP).	X	X	X

- Tableau4 : Comportement de rémanence des objets de mémoire (s'applique à toutes les CPU avec interface DP/MPI)

x = rémanent ; – = non rémanent

Comportement de rémanence d'un DB avec CPU avec version de firmware < V2.0.12 :

Avec ces CPU, le contenu des DB reste rémanent en cas de MISE HORS TENSION/MISE SOUS TENSION et de STOP-RUN. [2]

Comportement de rémanence d'un DB avec CPU à partir de la version de firmware >= V2.0.12 :

Dans ces CPU, vous pouvez créer des blocs de données avec la propriété "NON-Retain" (nom rémanent).

Les blocs de données avec la propriété "NON-Retain" reprennent leur valeurs initiales après chaque mise hors tension et mise sous tension, ainsi qu'à chaque passage ARRETMARCHE de la CPU.

Vous avez deux possibilités d'affecter la propriété "NON-Retain" à un bloc de données :

- STEP 7 (à partir de la version 5.2 + SP 1) : activation du DB, NON-Retain
- SFC 82 "Crea_DBL" (génération d'un DB dans la mémoire de chargement) : dans le paramètre ATTRIB, mise à "1" du bit 2 [2]

Au passage de HORS TENSION à SOUS TENSION ou au démarrage de la CPU, le DB doit	
reprenre les valeurs initiales (DB non rémanent)	garder les dernières valeurs effectives (DB rémanent)
Que se passe-t-il : Au passage de HORS TENSION à SOUS TENSION et au démarrage (STOP-RUN) de la CPU, les valeurs effectives du DB ne sont pas rémanentes. Le DB reçoit les valeurs initiales mémorisées dans la mémoire de chargement	Que se passe-t-il : Au passage de HORS TENSION à SOUS TENSION et au démarrage (STOP-RUN) de la CPU, les valeurs effectives du DB sont conservées.
Conditions requises dans STEP 7 : • dans les propriétés du bloc du DB, la case à cocher "Non-Retain" est activée ou • un DB non rémanent a été généré avec la SFC 82 "CREA_DBL" et l'attribut de bloc correspondant (ATTRIB -> bit NON_RETAIN).	Conditions requises dans STEP 7 : • dans les propriétés du bloc du DB, la case à cocher "Non-Retain" est activée ou • ou DB rémanent a été généré avec la SFC 82 « CREA_DBL ».

- Tableau 5 Comportement de rémanence des DB pour les CPU à partir de la version de firmware >=V2.0.12

Rémanence de la mémoire de travail :

Les CPU	Taille de la mémoire rémanente (pour blocs de données rémanents)
CPU 312	32 Ko
CPU 313, 314	64 Ko
315	128 Ko
317	256 Ko
319	700 Ko

Tableau 6 : Rémanence de la mémoire de travail

Le reste de la mémoire de travail n'est utilisable que pour des blocs de code ou des DB non rémanents.

4.1.4. Plages d'opérands de la mémoire système :

La mémoire système des CPU S7 est divisée en plages d'opérands. En utilisant les opérations appropriées, vous adressez les données directement dans la plage d'opérands respective dans votre programme. [2]

Plages d'opérands de la mémoire système :

Plages d'opérands	Description
Mémoire image des entrées	La CPU lit au début de chaque cycle de l'OB 1 les entrées depuis les modules d'entrées et enregistre les valeurs dans la mémoire image des entrées.
Mémoire image de sorties	Le programme calcule les valeurs pour les sorties pendant le cycle et les archive dans la mémoire image des sorties. A la fin du cycle OB 1, la CPU écrit les valeurs de sortie calculées dans les modules de sorties.
Mémento	Cette plage met à disposition l'espace mémoire pour les résultats intermédiaires calculés dans le programme.
Temporisations	Les temporisations sont disponibles dans cette plage
Compteur	Les compteurs sont disponibles dans cette plage.
Données locales	Cette plage de mémoire est réservée aux données temporaires d'un bloc de code (OB, FB, FC) pour la durée du traitement de ce bloc.

Tableau 7 : Plages d'opérands de la mémoire système

Mémoire image des entrées et des sorties :

Si les plages d'opérands Entrées (E) et Sorties (A) sont adressées dans le programme utilisateur, les états de signaux ne sont pas interrogés sur les modules de signaux TOR, mais il y a accès à une zone de mémoire dans la mémoire système de la CPU. On désigne cette zone de mémoire par mémoire image. La mémoire image de processus est divisée en deux parties : la mémoire image des entrées et la mémoire image des sorties. [2]

- **Avantages de la mémoire image :** Contrairement à l'accès direct aux modules d'entrées/de sorties, l'accès à la mémoire image présente l'avantage suivant : une image cohérente des signaux de processus est à la disposition de la CPU pendant la durée du traitement cyclique du programme. En cas de changement de signal sur un module d'entrées durant le traitement du programme, l'état logique est conservé dans la mémoire image jusqu'à l'actualisation de cette dernière dans le cycle suivant. En outre, l'accès à la mémoire image prend beaucoup moins de temps que l'accès direct aux modules de signaux, puisqu'elle se trouve dans la mémoire système de la CPU.
- **Actualisation de la mémoire image :** La mémoire image est actualisée de façon cyclique par le système d'exploitation. La figure suivante présente les phases de traitement au cours d'un cycle.

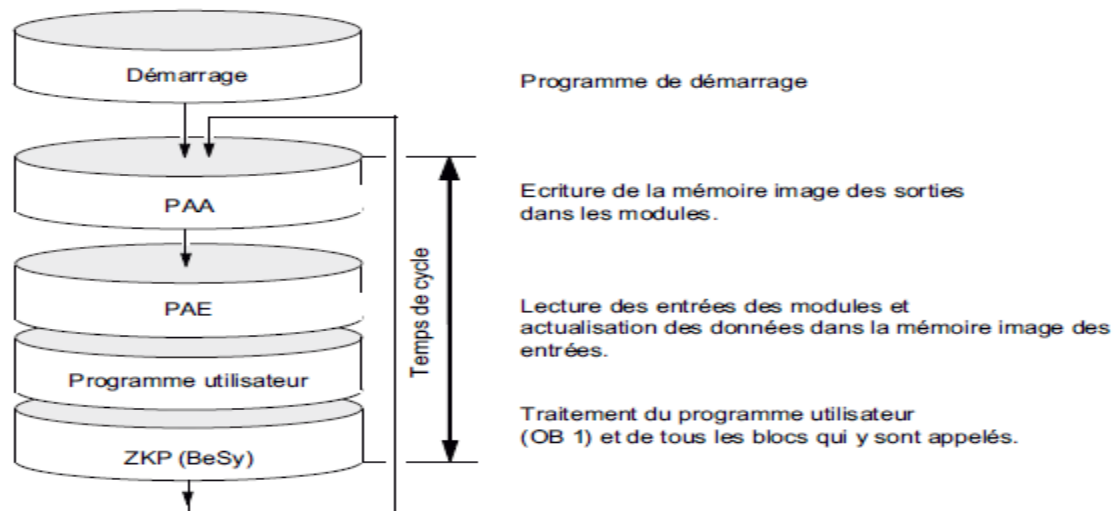


Figure6 : actualisation cyclique de la mémoire image

4.1.5. Propriétés de la micro-carte mémoire SIMATIC :

La micro carte mémoire SIMATIC comme cartouche mémoire de la CPU, votre CPU utilise une microcarte mémoire SIMATIC comme cartouche mémoire. Vous pouvez utiliser cette dernière comme mémoire de chargement et comme support de données amovible.

Quelles sont les données enregistrées dans la micro-carte SIMATIC ?

Les données suivantes peuvent être enregistrées dans la micro-carte SIMATIC :

- Programme utilisateur, c'est-à-dire tous les blocs (OB, FC, FB, DB) et données système
- Archives et recettes
- Données relatives à la configuration (projets STEP 7)
- Données pour une mise en jour du système d'exploitation, sauvegarde du système d'exploitation

II- Système d'automatisation S7-300 Caractéristiques des modules

1. Modules d'alimentation :

Introduction

Divers modules d'alimentation sont mis à disposition pour l'alimentation du S7-300 et des capteurs/actionneurs en 24 V cc [3]

Modules d'alimentation

Ce chapitre fournit les caractéristiques techniques des modules d'alimentation du S7-300.

Outre les caractéristiques techniques des modules d'alimentation, ce chapitre contient :

- Les propriétés
- Schéma de branchement
- Schéma de principe
- La protection des conducteurs
- Les réactions en cas de conditions de fonctionnement atypiques[3]

1.1. Module d'alimentation PS 307 ; 2 A ; (6ES7307-1BA01-0AA0)

N° de référence :

6ES7307-1BA01-0AA0

Propriétés :

Le module d'alimentation PS 307 ; 2 A présente les propriétés suivantes :

- Courant de sortie 2 A
- Tension nominale de sortie 24 Vcc, stabilisée, tenue aux courts-circuits et à la marche à vide
- Raccordement à un réseau alternatif monophasé (tension nominale d'entrée 120/230 V ca, 50/60 Hz)
- Séparation de sécurité des circuits
- Peut servir de tension d'alimentation des capteurs et actionneurs

Schéma de raccordement du PS 307 ; 2 A :

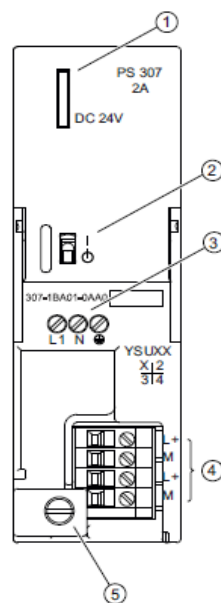
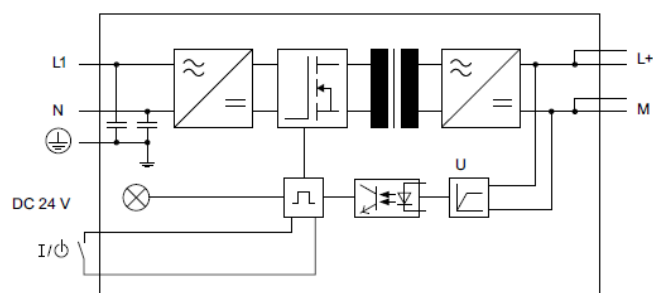


Figure 7 : module d'alimentation PS 307 2A

- ① Signalisation "Tension de sortie 24 V cc présente"
- ② Commutateur EN/HORS du 24 V cc
- ③ Bornes pour la tension secteur et le conducteur de protection
- ④ Bornes pour la tension de sortie 24 V cc
- ⑤ Arrêt de traction [3]

Schéma de principe du module d'alimentation PS 307 ; 2 A :



- Figure8 : principe du module d'alimentation PS 307 ; 2 A

La protection des conducteurs :

Pour protéger la ligne du module d'alimentation PS 307 ; 2 A en provenance du réseau, il est conseillé de brancher un petit disjoncteur (p. ex. série 5SN1 de Siemens) ayant les caractéristiques suivantes :

- Courant nominal pour 230 V ca : 3 A
- Caractéristique de déclenchement (type) : C. [3]

Réaction dans des conditions de fonctionnement atypiques :

Condition atypique	Résultat	LED 24 V cc
Surcharge du circuit de sortie • $I > 2,6 \text{ A}$ (dynamique) • $2 \text{ A} < I \leq 2,6 \text{ A}$ (statique)	Creux de tension, rétablissement automatique de la tension Chute de tension, influence la durée de vie	Clignote
Court-circuit d'une sortie	Tension de sortie 0 V, rétablissement automatique de la tension après avoir éliminé le court-circuit	éteint
Surtension au niveau du primaire	Destruction possible	-
Sous-tension au niveau du primaire	Coupure automatique, rétablissement automatique de la tension	éteint

Tableau 8 : Réaction dans des conditions de fonctionnement atypiques

Caractéristiques techniques du PS 307 ; 2 A (6ES7307-1BA01-0AA0) :

Caractéristiques techniques	
Dimensions, poids	
Dimensions l x h x p (mm)	40 x 125 x 120
Poids	400 g env.
Valeurs d'entrée	
tension d'entrée • valeur nominale Fréquence de réseau • valeur nominale • plage admissible	120 V/230 V ca (commutation automatique) 50 Hz ou 60 Hz de 47 Hz à 63 Hz
Courant d'entrée nominal • sous 230 V • sous 120 V	0,5 A 0,9 A
Courant d'appel à l'enclenchement (à 25 °C)	22 A
I^2t (pour le courant d'appel à l'enclenchement)	1 A ² s
Valeurs de sortie	
tension de sortie • valeur nominale • plage admissible • durée d'établissement	24 V cc 24 V \pm 3 %, tenue à la marche à vide max. 2,5 s
courant de sortie • valeur nominale	2 A, montage parallèle possible
Protection contre les courts-circuits	électronique, sans mémorisation de 1,1 à 1,3 x I_N
Ondulation résiduelle	max. 150 mV _c à c.
Valeurs caractéristiques	
Classe de protection selon CEI 536 (DIN VDE 0106, partie 1)	I, avec conducteur de protection
Isolément • Tension nominale d'isolément (entre 24 V et L1) • tension d'essai	250 V ca 4200 V cc
Séparation de sécurité descircuits	Circuit SELV
Maintien en cas de coupure secteur (pour 93 V ou 187 V) • Taux de répétition	min. 20 ms min. 1 s
Rendement	84 %
Dissipation	57 W
Dissipation du module	typ. 9 W
Diagnostic	
Signalisation de la présence d'une tension de sortie	Oui, LED verte

Tableau 9 Caractéristiques techniques du PS 307 [3]

2. Modules TOR :

2.1. Modules d'entrées TOR :

Tableau des propriétés :

Le tableau suivant regroupe les principales propriétés des modules d'entrées TOR.

Propriétés	Module			
	SM 321; DI 64 x 24V; Sinking/Sourcing	SM 321; DI 32 x DC24V	SM 321; DI 32 x AC120V	SM 321; DI 16 x DC24V
	(-1BH00-)	(-1BL00-)	(-1EL00-)	(-1BH02-)
Nombre d'entrées	64 DO ; séparation galvanique par groupes de 16	32 DI ; séparation galvanique par groupes de 16	32 DI ; séparation galvanique par groupes de 8	16 DI ; séparation galvanique par groupes de 16
Tension d'entrée nominale	24 V cc	24 V cc	120 V ca	24 V cc
Convient pour ...	-	détecteurs de proximité 2, 3 ou 4 fils (BERO).		
Prend en charge l'isochronisme	non	non	non	non
Diagnostic paramétrable	non	non	non	non
Alarme de diagnostic	non	non	non	non
Alarme de processus au changement de front	non	non	non	non
Retards d'entrée réglables	non	non	non	non
Particularités	-	-	-	-

Propriétés	Module				
	SM 321; DI 16 x DC24V High Speed	SM 321; DI 16 x DC24V avec alarme de diagnostic et de processus	SM 321; DI 16 x DC 24V/125V avec alarme de diagnostic et de processus	SM 321; DI 16 x DC24V; de type M	SM 321; DI 16 x UC24/48V
	(-1BH10-)	(-7BH01-)	(-7EH00-)	(-1BH50-)	(-1CH00-)
Nombre d'entrées	16 DI ; séparation galvanique par groupes de 16	16 DI ; séparation galvanique par groupes de 16	16 DI ; séparation galvanique par groupes de 16	16 DI de type M, séparation galvanique par groupes de 16	16 DI ; séparation galvanique par groupes de 1
Tension d'entrée nominale	24 V cc	24 V cc	24 V CC - 125 V	24 V cc	24 à 48 V CC 24 à 48 V CA
Convient pour ...	Interrupteur, détecteurs de proximité 2, 3 ou 4 fils (BERO).				
Prend en charge l'isochronisme	oui	oui	non	non	non
Diagnostic paramétrable	non	oui	oui	non	non
Alarme de diagnostic	non	oui	oui	non	non
Alarme de processus au changement de front	non	oui	oui	non	non
Retards d'entrée réglables	non	oui	oui	non	non
Particularités	Module rapide ; en particulier pour l'isochronisme	2 alimentations capteurs résistantes aux courts-circuits pour 8 voies chacune ; possibilité d'arrivée redondante externe de l'alimentation de capteurs	---	---	---

Propriétés	Module				
	SM 321; DI 16 x DC48-125V	SM 321; DI 16 x AC120/230 V	SM 321; DI 16 x NAMUR	SM 321; DI 8 x AC 120/230V	SM 321; DI 8 x AC 120/230 V ISOL
	(-1CH20-)	(-1FH00-)	(-7TH00-) *	(-1FF01-)	(-1FF10-)
Nombre d'entrées	16 DI ; séparation galvanique par groupes de 4	16 DI ; séparation galvanique par groupes de 4	16 DI ; séparation galvanique par groupes de 2	8 DI ; séparation galvanique par groupes de 2	8 DI ; séparation galvanique par groupes de 2
Tension d'entrée nominale	120/230 V CA	120/230 V CA	24 V cc	120/230 V CA	120/230 V CA
Convient pour ...	Interrupteur, détecteurs de proximité 2, 3 ou 4 fils (BERO).	Interrupteur ; détecteurs de proximité CA 2, 3 fils	Capteur NAMUR	Interrupteur ; détecteurs de proximité CA 2, 3 fils	
Prend en charge l'isochronisme	non	non	non	non	non
Diagnostic paramétrable	non	non	non	non	non
Alarme de diagnostic	non	non	oui	non	non
Alarme de processus au changement de front	non	non	non	non	non
Retards d'entrée réglables	non	non	non	non	non
Particularités	---	---	Module avec diagnostic de voie et de nombreuses fonctions de contrôle-commande	---	---

Tableau 10 : les principales propriétés des modules d'entrées TOR [3]

2.2. Modules de sorties TOR :

Résumé des caractéristiques : Le tableau suivant regroupe les principales propriétés des modules de sortie TOR

Propriétés	Module				
	SM 322; DO 64 x DC24V/0,3A Sourcing	SM 322; DO 64 x DC 24V/0,3A Sinking	SM 322; DO 32 x DC24V/0,5A	SM 322; DO 32 x AC120/230V/1A	SM 322; DO 16 x DC24V/0,5A
	(-1BH00-)	(-1BP50-)	(-1BL00-)	(-1FL00-)	(-1BH01-)
Nombre de sorties	64 DO ; séparation galvanique par groupes de 16	64 DO ; séparation galvanique par groupes de 16	32 DO ; séparation galvanique par groupes de 8	32 DO ; séparation galvanique par groupes de 8	16 DO ; séparation galvanique par groupes de 8
Courant de sortie	3 A	3 A	0,5 A	1,0 A	0,5 A
Tension assignée de charge	24 V cc	24 V cc	24 V cc	24 V cc	24 V cc
Convient pour ...	électro-vannes, contacteurs pour courant continu et DEL de signalisation				
Prend en charge le fonctionnement en synchronisme d'horloge	non	non	non	non	non
Diagnostic paramétrable	non	non	non	non	non
Alarme de diagnostic	non	non	non	non	non
Sortie de la valeur de remplacement	non	non	non	non	non
Particularités	-				

Propriétés	Module				
	SM 322; DO 16 x DC24V/0,5A High Speed (-1BH10-)	SM 322; DO 16 x UC24/48 V (-5GH00-)	SM 322; DO 16 x AC120/230V/1A (-1FH00-)	SM 322; DO 16 x DC24V/0,5A (-8BH00-)* (-8BH01-)* (-8BH10-)	SM 322; DO 8 x DC24V/2A (-1BF01-)
Nombre de sorties	16 DO ; séparation galvanique par groupes de 8	16 DO ; séparation galvanique par groupes de 1	16 DO ; séparation galvanique par groupes de 8	16 DO ; séparation galvanique par groupes de 4	8 DO ; séparation galvanique par groupes de 4
Courant de sortie	0,5 A	0,5 A	1 A	0,5 A	2 A
Tension assignée de charge	24 V cc	24V à 48V cc 24V à 48V ca	120/230V ca	24 V cc	24 V cc
Convient pour ...	électro-vannes, contacteurs pour courant continu et DEL de signalisation				
Prend en charge le fonctionnement en synchronisme d'horloge	oui	non	non	non	non
Diagnostic paramétrable	non	oui	non	oui	non
Alarme de diagnostic	non	oui	non	oui	non
Sortie de la valeur de remplacement				oui	non
Particularités	Module rapide ; en particulier pour le fonctionnement en synchronisme d'horloge	-	-	possibilité de commande redondante de la charge ; nombreuses fonctions de contrôle- commande	-

Propriétés	Module			
	SM 322; DO 8 x DC24V/0,5A avec alarmes de diagnostic (-8BF00-)	SM 322; DO 8 x DC48- 125V/1,5A (-1CF00-)	SM 322; DO 8 x AC120/230 V/2A (-1FF01-)	SM 322; DO 8 x AC120/230 V/ 2A ISOL (-5FF00-)
Nombre de sorties	8 DO ; séparation galvanique par groupes de 8	8 DO ; séparation galvanique et protection contre les erreurs de polarité par groupes de 4	8 DO ; séparation galvanique par groupes de 4	8 DO ; séparation galvanique par groupes de 1
Courant de sortie	0,5 A	1,5 A	2 A	2 A
Tension assignée de charge	24 V cc	48 à 125 V cc	120/230 V CA	120/230 V CA
Convient pour ...	électro-vannes, contacteurs pour courant continu et DEL de signalisation		bobines, contacteurs, démarreurs de moteurs, micro-moteurs et voyants à courant alternatif.	
Prend en charge le fonctionnement en synchronisme d'horloge	non	non	non	non
Diagnostic paramétrable	oui	non	non	oui
Alarme de diagnostic	oui	non	non	oui
Sortie de la valeur de remplacement	oui	non	non	oui
Particularités	Possibilité de commande redondante de la charge	-	Indicateur de fusion du fusible. Fusible changeable pour chaque groupe	-

Tableau 11 : les principales propriétés des modules de sortie TOR [3]

2.3. Module d'entrées TOR SM 321 ; DI 16 x 24 V cc ; (6ES7321-1BH02-0AA0) :

N° de référence :

"Module standard" 6ES7321-1BH02-0AA0

Propriétés :

Le module SM 321 ; DI 16 x 24 V cc dispose des propriétés suivantes :

- 16 entrées, séparation galvanique par groupes de 16
- Tension d'entrée nominale : 24 V cc
- convenant pour commutateurs et contacts de détecteurs de proximité 2, 3 ou 4 fils (BERO)

Schéma de branchement et de principe du SM 321 ; DI 16 x 24 V cc :

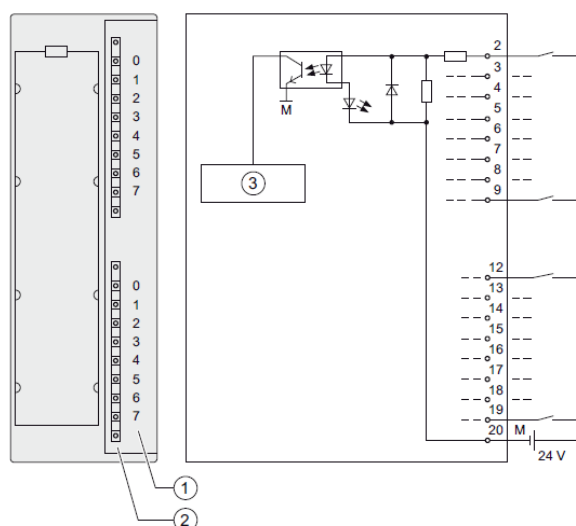


Figure 9 : branchement et principe du module SM 321 ; DI 16 x 24 Vcc

- ① N° de voie
- ② Signalisation d'état - vert
- ③ Coupleur de bus interne [3]

Caractéristiques techniques du SM 321 ; DI 16 x 24 V cc :

Caractéristiques techniques	
Dimensions et poids	
Dimensions l x h x p (mm)	40 x 125 x 117
Poids	environ 200 g
Caractéristiques spécifiques du module	
Prend en charge le fonctionnement en synchronisme d'horloge	non
Nombre d'entrées	16
Longueur de câble <ul style="list-style-type: none"> non blindé blindé 	max. 600 m max. 1000 m
Tensions, courants, potentiels	
Nombre d'entrées en commande simultanée <ul style="list-style-type: none"> montage horizontal jusqu'à 60 °C montage vertical jusqu'à 40°C 	16 16
Séparation de potentiel <ul style="list-style-type: none"> entre voies et bus interne entre les voies par groupes de 	oui oui 16
Différence de potentiel admissible <ul style="list-style-type: none"> entre différents circuits 	75 V cc/ 60 V ca
Isolation testée avec	500 V cc
Consommation <ul style="list-style-type: none"> sur bus interne 	max. 10 mA
Dissipation du module	typ. 3,5 W
Etat, alarmes, diagnostics	
Signalisation d'état	une LED verte par voie
Alarmes	Néant
Fonctions de diagnostic	Néant
Caractéristiques pour la sélection d'un capteur	
tension d'entrée <ul style="list-style-type: none"> valeur nominale pour signal "1" pour signal "0" 	24 V cc 13 à 30 V - 30 à + 5 V
courant d'entrée <ul style="list-style-type: none"> pour signal "1" 	typ. 7 mA

Temporisation d'entrée <ul style="list-style-type: none"> de "0" à "1" de "1" à "0" 	1,2 à 4,8 ms 1,2 à 4,8 ms
Caractéristique d'entrée	type 1, selon CEI 61131
Raccordement de BERO 2 fils <ul style="list-style-type: none"> courant de repos admissible 	possible max. 1,5 mA
Raccordement des capteurs de signaux	avec connecteur frontal à 20 points

Tableau12 : Caractéristiques techniques du SM 321 ; DI 16 x 24 V cc

2.4. Module de sorties TOR SM 322 ; DO 16 x 24 V cc/ 0,5 A ; (6ES7322-1BH01-0AA0) :

N° de référence : "Module standard"

6ES7322-1BH01-0AA0

Propriétés :

Le module SM 322 ; DO 16x24 V cc/0,5 A se distingue par les propriétés suivantes :

- 16 sorties, séparation galvanique par groupes de 8
- Courant de sortie 0,5 A
- Tension d'alimentation nominale 24 V cc
- Convenant pour électrovannes, contacteurs pour courant continu et LED [3]

Utilisation du module avec des compteurs rapides

Si le module est utilisé avec des compteurs rapides, veuillez tenir compte de la remarque suivante :

Remarque : Lors de l'activation de la tension d'alimentation 24 V via un contact mécanique, les sorties du SM 322 ; DO 16 x 24 V cc/0,5 A conduisent un signal "1" pendant environ 50µs. [3]

Schéma de branchement et de principe du module SM 322 ; DO 16x24Vcc/0,5A

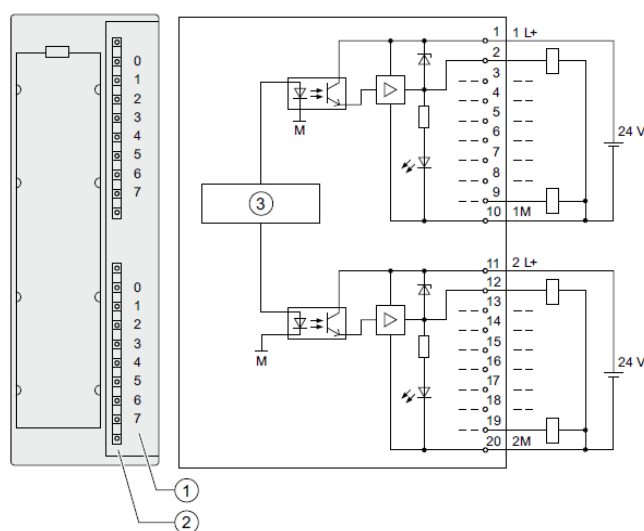


Figure 10 : branchement et principe du module SM 322 ; DO 16x24Vcc/0,5A

- ① N° de voie
- ② Signalisation d'état - vert
- ③ Coupleur de bus interne[3]

Caractéristiques techniques du SM 322 ; DO 16 x 24 V cc/0,5 A :

Caractéristiques techniques	
Dimensions et poids	
Dimensions l x h x p (mm)	40 x 125 x 117
Poids	environ 190 g
Caractéristiques spécifiques du module	
Prend en charge le fonctionnement en synchronisme d'horloge	non
Nombre de sorties	16
Longueur de câble <ul style="list-style-type: none">• non blindé• blindé	max. 600 m max. 1000 m
Tensions, courants, potentiels	
Tension nominale de charge L +	24 V cc
Courant total des sorties (par groupe)	
<ul style="list-style-type: none">• montage horizontal jusqu'à 40°C jusqu'à 60 °C	max. 4 A max. 3 A
<ul style="list-style-type: none">• montage vertical jusqu'à 40°C	max. 2 A
Séparation de potentiel	
<ul style="list-style-type: none">• entre voies et bus interne	oui
<ul style="list-style-type: none">• entre les voies par groupes de	oui 8
Différence de potentiel admissible <ul style="list-style-type: none">• entre différents circuits	75 V cc/ 60 V ca
Isolation testée avec	500 V cc
Consommation <ul style="list-style-type: none">• sur bus interne• sur tension d'alimentation L + (sauf charge)	max. 80 mA max. 80 mA
Dissipation du module	typ. 4,9 W
Etat, alarmes, diagnostics	
Signalisation d'état	une LED verte par voie
Alarmes	Néant
Fonctions de diagnostic	Néant

Caractéristiques techniques	
Caractéristiques pour la sélection d'un actionneur	
tension de sortie	min. L + (-0,8 V)
• pour signal "1"	
courant de sortie	
• pour signal "1"	
valeur nominale	0,5 A
plage admissible	5 mA à 0,6 A
• pour signal "0" (courant résiduel)	Max. 0,5 mA
Temporisation de sortie (avec charge résistive)	
• de "0" à "1"	max. 100 µs
• de "1" à "0"	max. 500 µs
Plage de résistance de charge	48 Ω à 4 kΩ
Charge de lampes	max. 5 W
Mise en parallèle de 2 sorties	
• pour commande redondante d'une charge	Possible (seulement sorties d'un même groupe)
• pour élévation de la puissance	Pas possible
Rebouclage sur une entrée TOR	possible
Fréquence de commutation	
• pour charge résistive	max. 100 Hz
• pour charge inductive, selon CEI 947-5-1, CC 13	max. 0,5 Hz
• pour charge de lampes	max. 10 Hz
Limitation (interne) de la tension de coupure inductive	typ. L + (- 53 V)
Protection contre les court-circuits de la sortie	oui, par hachage électronique
• seuil d'action	typ. 1 A
Raccordement des actionneurs	avec connecteur frontal à 20 points

Tableau 13 : Caractéristiques techniques du SM 322 ; DO 16 x 24 V cc/0,5 A [3]

III-Conclusion :

Dans ce chapitre, nous avons donné une vue générale sur les CPU de l'automate de SIEMENS et leurs spécification en suit une vue sur les modules d'E/S et leurs caractéristique. Le but de cette partie est de connu comment choisi les CPU et les modules de l'automate programmable de SIEMENS pour notre projet.

CHAPITRE 3

Programmer avec STEP 7 Et SIMATIC HMI WinCC flexible 2008

I. Programmer avec STEP 7

1. STEP 7 :

C'est le logiciel SIMATIC de base pour la conception de programmes pour systèmes d'automatisation SIMATIC S7-300/400 dans les langages de programmation CONT, LOG ou LIST.[4]

Le progiciel de base STEP 7 existe en plusieurs versions :

- STEP 7-Micro/DOS et STEP 7-Micro/Win pour des applications autonomes simples sur SIMATIC S7 - 200. [4]

- STEP 7 pour des applications sur SIMATIC S7-300/400, SIMATIC M7-300/400 et SIMATIC C7 présentant des fonctionnalités supplémentaires :

- Possibilité d'extension grâce aux applications proposées par l'industrie logicielle SIMATIC

- Possibilité de paramétrage de modules fonctionnels et de modules de communication

- Forçage et fonctionnement multiprocesseur

- Communication par données globales

- Transfert de données commandé par événement à l'aide de blocs de communication et de blocs fonctionnels

- Configuration de liaisons

Tâches fondamentales :

La mise en place d'une solution d'automatisation avec STEP 7 nécessite la réalisation de tâches fondamentales. La figure suivante indique les tâches à exécuter dans la plupart des projets et les classe selon la marche à suivre. Ce guide renvoie aux chapitres respectifs, vous permettant ainsi de vous déplacer dans le manuel selon la tâche que vous avez à réaliser. [4]

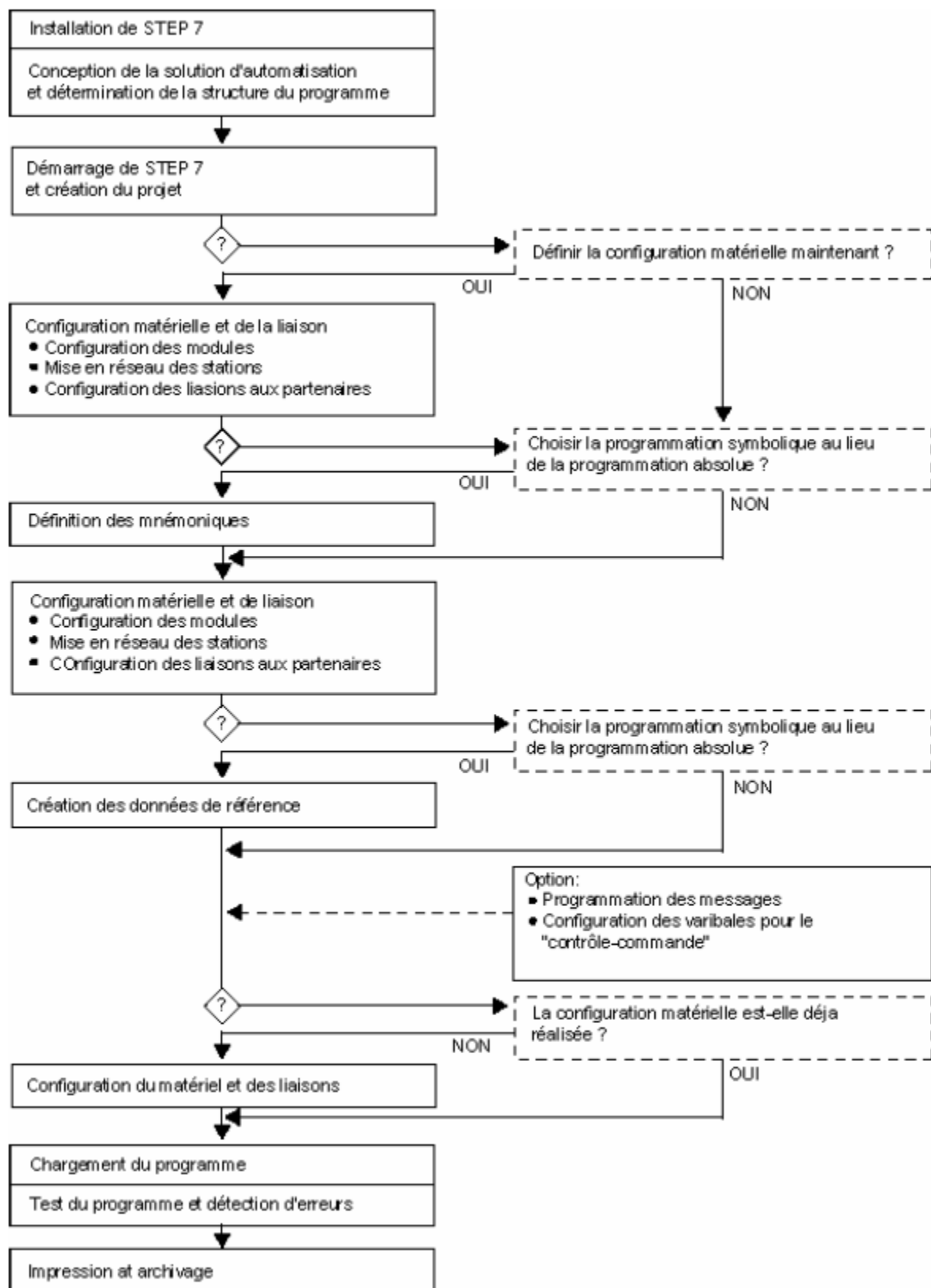


Figure 11 : les tâches à exécuter dans les plupart des projets [4]

Solutions de rechange : Comme le montre la figure précédente, vous pouvez procéder de deux manières différentes :

- Vous pouvez configurer le matériel en premier lieu, puis programmer les blocs.

- Mais vous pouvez aussi programmer d'abord les blocs sans avoir à configurer auparavant le matériel. Ceci est particulièrement recommandé pour les tâches de maintenance. En effet, vous avez ainsi la possibilité d'intégrer des blocs programmés dans un projet existant. [4]

Brève description des diverses étapes :

- **Installation de STEP 7 et des "License Keys"**

Pour une première utilisation, vous devez installer STEP 7 et transférer les "License Keys" depuis la disquette sur le disque dur. [4]

- **Conception de la solution d'automatisation**

Avant d'utiliser STEP 7, vous devez planifier votre solution d'automatisation depuis la division du processus en tâches individuelles jusqu'à la réalisation d'un schéma de configuration [4]

- **Conception de la structure du programme**

En utilisant les blocs mis à votre disposition par STEP 7, vous transposez les tâches décrites lors de la conception de votre solution d'automatisation en structure de programme. [4]

- **Démarrage de STEP 7**

Vous démarrez STEP 7 depuis l'interface utilisateur de Windows. [4]

- **Définition de la structure du projet**

Un projet peut être comparé à un dossier dans lequel toutes les données sont organisées de manière hiérarchique et sont toujours disponibles. Dès lors que vous avez créé un projet, toutes les tâches suivantes y seront exécutées. [4]

- **Création de la station**

En créant la station, vous définissez l'automate programmable : p.ex. SIMATIC 300, SIMATIC 400, SIMATIC S5. [4]

- **Configuration matérielle**

Dans une table de configuration, vous définissez les modules que vous allez mettre en œuvre dans votre solution d'automatisation ainsi que les adresses permettant d'y accéder depuis le programme utilisateur. Vous pouvez en outre y paramétrer les caractéristiques des modules[4]

- **Configuration de réseaux et de liaisons de communication**

La condition requise à l'établissement d'une communication est l'existence d'un réseau préalablement configuré. Vous devez à cet effet créer les réseaux auxiliaires nécessaires à votre solution d'automatisation, définir leurs propriétés et pour les stations mises en réseau, les caractéristiques de connexion au réseau ainsi que, le cas échéant, les liaisons de communication. [4]

- **Définition de mnémoniques**

Dans une table des mnémoniques, vous pouvez remplacer des adresses par des mnémoniques locaux ou globaux de désignation plus évocatrice afin de les utiliser dans votre programme [4]

- **Création du programme**

En utilisant l'un des langages de programmation mis à votre disposition, vous créez un programme affecté ou non à un module, que vous enregistrez sous forme de blocs, de sources ou de diagrammes. [4]

- **S7 uniquement :**

Création et exploitation de données de référence

Vous pouvez utiliser ces données de référence afin de vous faciliter le test et la modification de votre programme utilisateur. [4]

- **Configuration de messages**

Créez par exemple des messages sur bloc avec leurs textes et attributs. En utilisant le programme de transfert, vous transférez ensuite les données de configuration de messages dans la base de données du système de contrôle-commande (p.ex. SIMATIC WinCC, SIMATIC ProTool) [4]

- **Configuration de variables de contrôle-commande**

Vous définissez une fois pour toutes les variables de contrôle-commande dans STEP 7 et leur affectez les attributs souhaités. En utilisant le programme de transfert, vous transférez les variables de contrôle-commande créées dans la base de données du système de contrôle-commande WinCC. [4]

- **Chargement de programmes dans le système cible**

S7 uniquement : une fois la configuration, le paramétrage et la création du programme terminés, vous pouvez transférer votre programme utilisateur complet ou des blocs individuels dans le système cible (module programmable de votre solution matérielle). La CPU contient déjà le système d'exploitation. [4]

M7 uniquement : parmi différents systèmes d'exploitation, vous sélectionnez celui qui s'adapte à votre solution d'automatisation et le transférez seul ou avec le programme utilisateur sur le support de données souhaité du système cible M7. [4]

- **Test de programmes**

S7 uniquement : pour effectuer un test, vous avez la possibilité d'afficher les valeurs de variables depuis votre programme utilisateur ou depuis une CPU, d'affecter des valeurs à ces variables et de créer une table des variables que vous souhaitez afficher ou forcer.

M7 uniquement : test du programme utilisateur à l'aide d'un programme de débogage en langage évolué. [4]

2. Logiciel de base STEP 7 :

Normes en vigueur :

Les langages de programmation SIMATIC intégrés à STEP 7 répondent à la norme DIN EN 6.1131-3.

Le logiciel de base s'exécute sous les systèmes d'exploitation MS Windows 2000 Professional (que par la suite nous appellerons Windows 2000) ainsi que MS Windows XP Professional (que par la suite nous appellerons Windows XP) ainsi que MS Windows Server 2003 ainsi que MS Windows 7 Business, Ultimate et Enterprise et s'adapte à son organisation graphique orientée objet. [4]

Fonctions du logiciel de base :

Le logiciel de base vous assiste dans toutes les phases du processus de création de vos solutions d'automatisation, comme par exemple :

- la création et la gestion de projets,

- la configuration et le paramétrage du matériel et de la communication,
- la gestion des mnémoniques,
- la création de programmes, par exemple pour les systèmes cible S7,
- le chargement de programmes dans des systèmes cible,
- le test de l'installation d'automatisation,
- le diagnostic lors de perturbations de l'installation.

La conception de l'interface utilisateur du logiciel STEP 7 répond aux connaissances ergonomiques modernes et son apprentissage est très facile. [4]

Applications disponibles :

Le logiciel de base STEP 7 met à votre disposition différentes applications : [4]

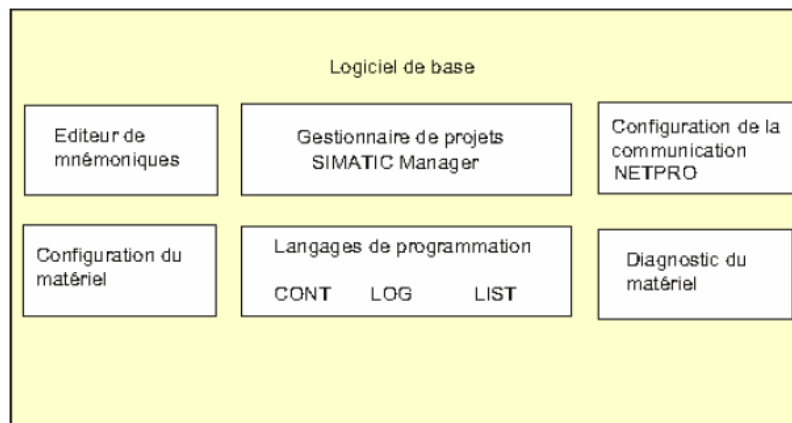


Figure12 : La différente application de logiciel step7

Il n'est pas nécessaire d'appeler séparément chaque application, car elles sont démarrées automatiquement lorsque vous sélectionnez une fonction correspondante ou ouvrez un objet.

Gestionnaire de projets SIMATIC :

Le gestionnaire de projets SIMATIC gère toutes les données relatives à un projet d'automatisation – quel que soit le système cible (S7/M7/C7) sur lequel elles ont été créées. Le gestionnaire de projets SIMATIC démarre automatiquement les applications requises pour le traitement des données sélectionnées. [4]

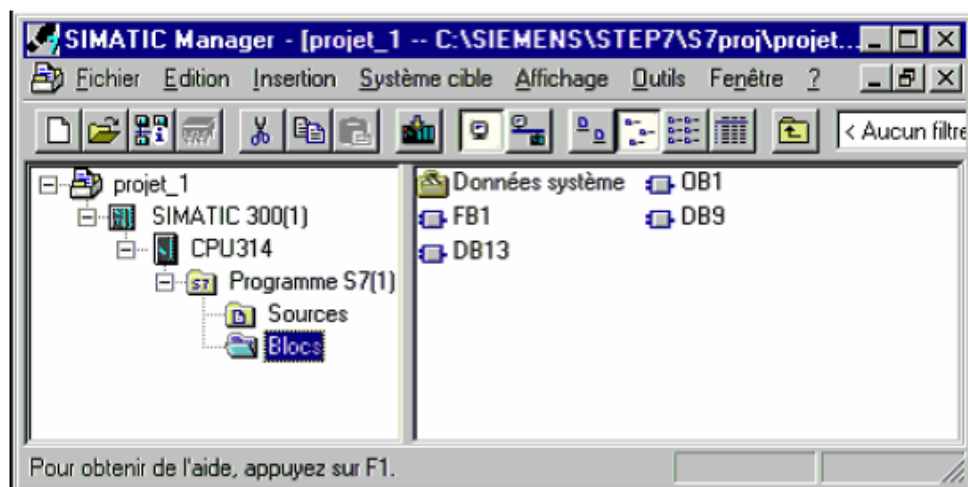


Figure13 : Gestionnaire de projet SIMATIC

Editeur de mnémoniques :

L'éditeur de mnémoniques vous permet de gérer toutes les variables globales. Vous disposez des fonctions suivantes :

- Définition de désignations symboliques et de commentaires pour les signaux du processus (entrées/sorties), mémentos et blocs,
- Fonctions de tri,
- Importation/exportation avec d'autres programmes Windows.

La table des mnémoniques qui en résulte est mise à disposition de toutes les applications. La modification de l'un des paramètres d'un mnémonique est de ce fait reconnue automatiquement par toutes les applications. [4]

Diagnostic du matériel :

Le diagnostic du matériel fournit un aperçu de l'état du système d'automatisation. Dans une représentation d'ensemble, un symbole permet de préciser pour chaque module, s'il est défaillant ou pas. Un double clic sur le module défaillant permet d'afficher des informations détaillées sur le défaut. [4]

Les informations disponibles dépendent des différents modules :

- Affichage d'informations générales sur le module (p.ex. numéro de commande, version, désignation) et son état (p.ex. défaillant), [4]
- Affichage d'erreurs sur les modules (p.ex. erreur de voie) de la périphérie centrale et des esclaves DP, [4]
- Affichage des messages de la mémoire tampon de diagnostic.

Pour les CPU, des informations supplémentaires s'affichent :

- causes de défaillance dans le déroulement d'un programme utilisateur
- Durée de cycle (le plus long, le plus court et dernier),
- Possibilités et charge de la communication MPI,
- Performances (nombre d'entrées/sorties, de mémentos, de compteurs, de temporisations et de blocs possibles). [4]

Langages de programmation :

Les langages de programmation CONT, LIST et LOG pour S7-300/400 font partie intégrante du logiciel de base.

- Le schéma à contacts (CONT) est un langage de programmation graphique. La syntaxe des instructions fait penser aux schémas de circuits. CONT permet de suivre facilement le trajet du courant entre les barres d'alimentation en passant par les contacts, les éléments complexes et les bobines. [4]
- La liste d'instructions (LIST) est un langage de programmation textuel proche de la machine. Dans un programme LIST, les différentes instructions correspondent, dans une large mesure, aux étapes par lesquelles la CPU traite le programme. Pour faciliter la programmation, LIST a été complété par quelques structures de langage évolué (comme, par exemple, des paramètres de blocs et accès structurés aux données).
- Le logigramme (LOG) est un langage de programmation graphique qui utilise les boîtes de l'algèbre de Boole pour représenter les opérations logiques. Les fonctions complexes, comme

par exemple les fonctions mathématiques, peuvent être représentées directement combinées avec les boîtes logiques. [4]

Configuration matérielle :

Vous utilisez cette application pour configurer et paramétrer le matériel d'un projet d'automatisation.

Vous disposez des fonctions suivantes :

- Pour configurer le système d'automatisation, vous sélectionnez des châssis (Racks) dans un catalogue électronique et affectez les modules sélectionnés aux emplacements souhaités dans les racks.
- La configuration de la périphérie décentralisée est identique à celle de la périphérie centralisée.
- Pour le paramétrage de la CPU, des menus vous permettent de définir des caractéristiques telles que le comportement à la mise en route et la surveillance du temps de cycle. Le fonctionnement multiprocesseur est possible. Les données saisies sont enregistrées dans des blocs de données système.
- Pour le paramétrage des modules, des boîtes de dialogue vous permettent de définir tous les paramètres modifiables. Les réglages à l'aide de commutateurs DIP s'avèrent inutiles. Le paramétrage des modules est réalisé automatiquement au démarrage de la CPU. L'avantage suivant en résulte. Le remplacement d'un module est ainsi possible sans nouveau paramétrage.
- Le paramétrage de modules fonctionnels (FM) et de processeurs de communication (CP) s'effectue de manière identique à celui des autres modules dans la configuration matérielle. A cet effet, des boîtes de dialogues ainsi que des règles spécifiques aux modules sont ainsi mises à disposition pour chaque FM et CP (fournies dans le logiciel fonctionnel du FM/CP). Dans les boîtes de dialogue, le système ne propose que des saisies possibles, ce qui empêche les entrées erronées. [4]

3. Principes de conception d'une structure de programme :

3.1. Programmes dans une CPU :

Deux programmes différents s'exécutent dans une CPU :

- Le système d'exploitation et
- Le programme utilisateur.

Système d'exploitation :

Le système d'exploitation, contenu dans chaque CPU, organise toutes les fonctions et procédures dans la CPU qui ne sont pas liées à une tâche d'automatisation spécifique. Ses tâches sont les suivantes :

- Le déroulement du démarrage à chaud et du redémarrage,
- L'actualisation de la mémoire image des entrées et l'émission de la mémoire image des sorties,
- L'appel du programme utilisateur,
- L'enregistrement des alarmes et l'appel des OB d'alarme,
- La détection et le traitement d'erreurs,
- La gestion des zones de mémoire,

- La communication avec des consoles de programmation et d'autres partenaires de communication. [4]

La modification des paramètres par défaut du système d'exploitation permet d'influer sur le comportement de la CPU dans des domaines précis. [4]

Programme utilisateur :

Vous devez créer votre programme utilisateur et le charger dans la CPU. Il contient toutes les fonctions nécessaires au traitement de votre tâche d'automatisation spécifique. Il doit entre autres :

- Déterminer les conditions pour le démarrage à chaud et le redémarrage de la CPU (par exemple, initialiser des signaux),
- Traiter des données du processus (par exemple, combiner des signaux binaires, lire et exploiter des valeurs analogiques, définir des signaux binaires pour la sortie, écrire des valeurs analogiques),
- Réagir aux alarmes,
- Traiter les perturbations dans le déroulement normal du programme. [4]

3.2. Blocs dans le programme utilisateur :

Le logiciel de programmation STEP 7 vous permet de structurer votre programme utilisateur, c'est-à-dire de le subdiviser en différentes parties autonomes. Il en résulte les avantages suivants :

- Écrire des programmes importants mais clairs,
- Standardiser certaines parties du programme,
- Simplifier l'organisation du programme,
- Modifier facilement le programme,
- Simplifier le test du programme, car vous pouvez l'exécuter section par section,
- Faciliter la mise en service. [4]

Types de bloc :

Vous pouvez utiliser différents types de bloc dans un programme utilisateur S7 :

Bloc	Brève description de la fonction	Pour plus de détails, voir
Blocs d'organisation (OB)	Les OB déterminent la structure du programme utilisateur.	Blocs d'organisation et structure du programme
Blocs fonctionnels système (SFB) et fonctions système (SFC)	Les SFB et SFC sont intégrés à la CPU S7 et vous permettent de réaliser quelques fonctions systèmes importantes.	Blocs fonctionnels système (SFB) et fonctions système (SFC)
Blocs fonctionnels (FB)	Les FB sont des blocs avec "mémoire" que vous programmez vous-même.	Blocs fonctionnels (FB)
Fonctions (FC)	Les FC contiennent des routines de programmes pour les fonctions fréquemment utilisées.	Fonctions (FC)
Blocs de données d'instance (DB d'instance)	Les DB d'instance sont affectés au bloc FB/SFB appelé. Ils sont générés automatiquement lors de la compilation.	Blocs de données d'instance
Blocs de données (DB)	Les DB sont des zones de données dans lesquelles l'on enregistre les données utilisateur. Outre les données affectées respectivement à un bloc fonctionnel, vous pouvez définir des données globales utilisables par tous les blocs.	Blocs de données globaux (DB)

Tableau 14 :les différents bloc dans un programme utilisateur S7

Les OB, FB, SFB, FC et SFC contiennent des parties de programme et sont de ce fait également désignés comme blocs de code. Le nombre de blocs autorisés par type de bloc ainsi que la longueur maximale de chaque bloc dépendent de la CPU. [4]

3.2.1. Blocs d'organisation et structure du programme :

Les blocs d'organisation (OB) constituent l'interface entre le système d'exploitation et le programme utilisateur. Ils sont appelés par le système d'exploitation et gèrent le traitement de programme cyclique et déclenché par alarme, ainsi que le comportement à la mise en route de l'automate programmable et le traitement des erreurs. Vous pouvez programmer les blocs d'organisation et déterminer ainsi le comportement de la CPU. [4]

Priorité des blocs d'organisation :

Les blocs d'organisation définissent l'ordre (événements de déclenchement) dans lequel les différentes parties du programme sont traitées. L'exécution d'un OB peut être interrompue par l'appel d'un autre OB. Cette interruption se fait selon la priorité : les OB de priorité plus élevée interrompent les OB de priorité plus faible. La priorité la plus faible est celle de l'OB d'arrière-plan. [4]

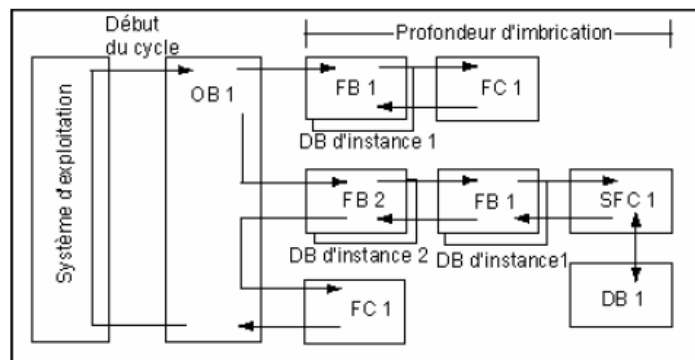
3.2.2. Hiérarchie d'appel dans le programme utilisateur :

Pour faire fonctionner le programme utilisateur, vous devez appeler les blocs qui le composent. C'est ce que vous réalisez à l'aide d'opérations STEP 7 spéciales, les appels de blocs que vous ne pouvez programmer et démarrer que dans des blocs de code. [4]

Ordre et profondeur d'imbrication :

On appelle hiérarchie d'appel l'ordre et l'imbrication des appels de blocs. Le niveau de profondeur autorisé pour les imbrications dépend de la CPU.

L'exemple de la figure suivante illustre l'ordre et l'imbrication des appels de blocs dans un cycle de traitement. [4]



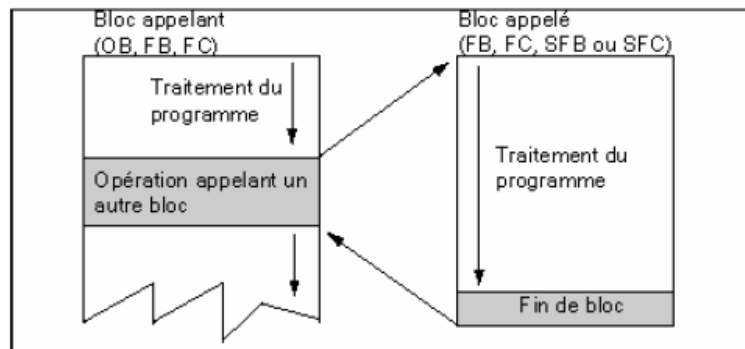
Règles relatives à l'ordre de création des blocs :

- Vous créez les blocs de haut en bas, Ce qui signifie que vous commencez par la rangée de blocs supérieure.
- Tout bloc appelé doit déjà exister, Ce qui signifie que dans une rangée de blocs, le sens de création est de droite à gauche.
- En dernier, vous créez l'OB1. [4]

Appels de blocs :

La figure suivante montre comment s'exécute un appel de bloc au sein d'un programme utilisateur : le programme appelle le deuxième bloc dont les opérations sont alors traitées dans

leur intégralité. Une fois le bloc appelé achevé, le traitement se poursuit avec l'opération suivant l'appel de bloc dans le bloc appelant. [4]



Avant de programmer un bloc, vous devez déterminer les données que le programme doit traiter : vous déclarez les variables du bloc. [4]

3.2.3. Catégories de blocs :

3.2.3.1. Bloc d'organisation pour le traitement de programme cyclique (OB1) :

Le traitement de programme cyclique constitue le traitement normal pour les automates programmables. Le système d'exploitation appelle l'OB1 cycliquement et déclenche ainsi le traitement cyclique du programme utilisateur. [4]

Déroulement du traitement de programme cyclique :

Le tableau suivant montre les phases du traitement de programme cyclique :

Phase	Déroulement dans les CPU jusqu'à 10/98	Déroulement dans les CPU à partir de 10/98
1	Le système d'exploitation démarre la surveillance du temps de cycle.	Le système d'exploitation démarre la surveillance du temps de cycle.
2	La CPU lit l'état des entrées dans les modules d'entrées et met à jour la mémoire image des entrées.	Elle écrit ensuite les valeurs de la mémoire image des sorties dans les modules de sorties.
3	Puis, elle traite le programme utilisateur et exécute les opérations indiquées dans le programme.	La CPU lit l'état des entrées dans les modules d'entrées et met à jour la mémoire image des entrées.
4	Elle écrit ensuite les valeurs de la mémoire image des sorties dans les modules de sorties.	Puis, elle traite le programme utilisateur et exécute les opérations indiquées dans le programme.
5	A la fin d'un cycle, le système d'exploitation exécute les travaux en attente, par exemple le chargement et l'effacement de blocs ou la réception et l'émission de données globales.	A la fin d'un cycle, le système d'exploitation exécute les travaux en attente, par exemple le chargement et l'effacement de blocs ou la réception et l'émission de données globales.
6	La CPU revient alors au début du cycle et démarre à nouveau la surveillance du temps de cycle.	La CPU revient alors au début du cycle et démarre à nouveau la surveillance du temps de cycle.

Tableau 15 : les phases du traitement de programme cyclique

Mémoires image du processus :

Pour disposer d'une image cohérente des signaux du processus pendant la durée du traitement de programme cyclique, la CPU n'accède pas directement aux modules de signaux lors de l'utilisation des plages d'opérandes Entrées (E) et Sorties (A), mais à une zone de mémoire interne de la CPU qui contient une image des entrées et sorties. [4]

Programmation du traitement de programme cyclique :

Pour programmer le traitement cyclique, vous écrivez votre programme utilisateur avec STEP 7 dans l'OB1 et les blocs qui y sont appelés.

Le traitement de programme cyclique commence dès que le programme de mise en route s'est achevé sans erreur. [4]

Possibilités d'interruption :

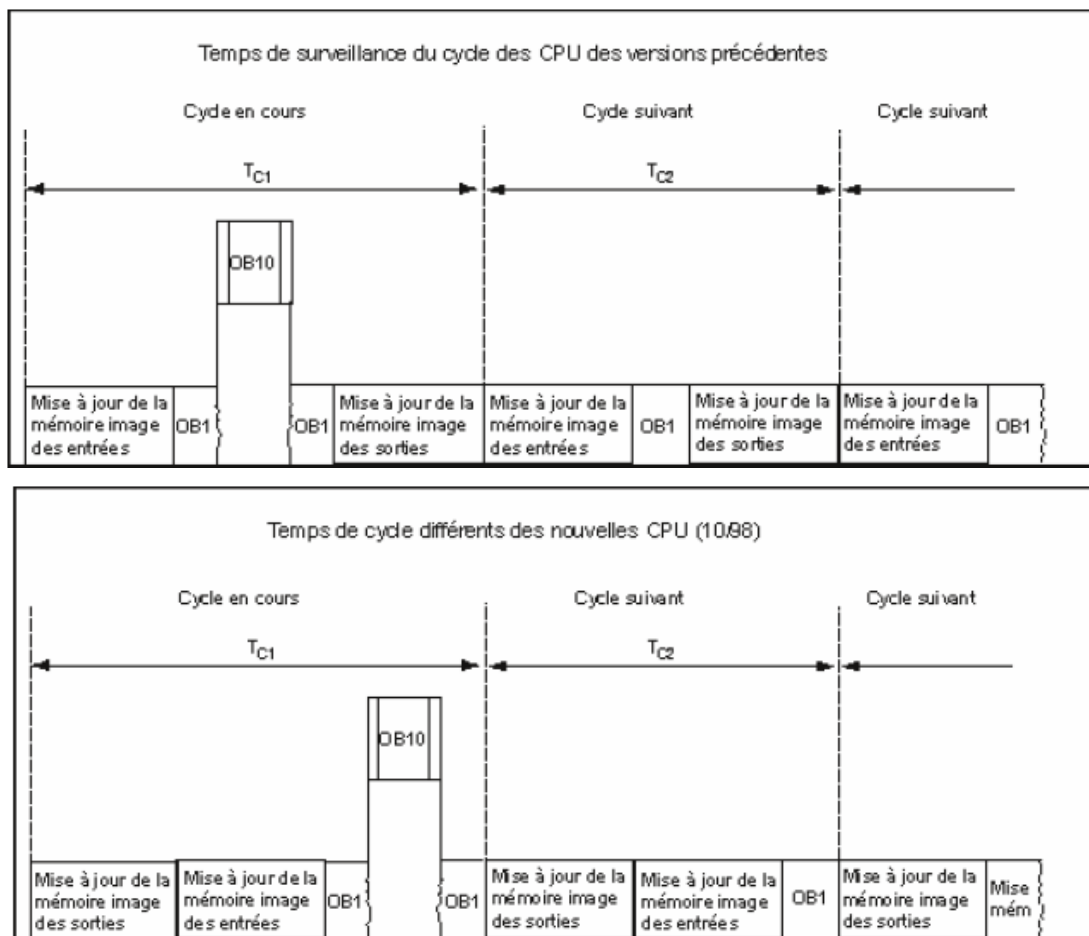
Le traitement de programme cyclique peut être interrompu par :

- Une alarme,
- Une commande STOP (commutateur de mode, commande de menu depuis la PG, SFC46 STP, SFB20 STOP),
- Une coupure de tension secteur,
- l'apparition d'une erreur de matériel ou de programme.

Temps de cycle :

Le temps de cycle est le temps dont a besoin le système d'exploitation pour le traitement du programme cyclique ainsi que de toutes les parties de programme interrompant ce cycle (par exemple, traitement des autres blocs d'organisation) et des activités du système (par exemple, mise à jour de la mémoire image). Ce temps est contrôlé.

Ce temps (T_c) n'est pas identique à chaque cycle. Les figures suivantes indiquent différents temps de cycle ($T_{c1} \neq T_{c2}$) pour les anciennes et les CPU jusqu'à 10/98 et les CPU à partir de 10/98. [4]



L'OB1 est interrompu par une alarme horaire dans le cycle en cours. [4]

3.2.3.2. Fonctions (FC) :

Les fonctions font partie des blocs que vous programmez vous-même. Une fonction est un bloc de code sans mémoire. Les variables temporaires d'une fonction sont sauvegardées dans la pile des données locales. Ces données sont perdues à l'achèvement de la fonction. Les fonctions peuvent faire appel à des blocs de données globaux pour la sauvegarde des données.

Comme une fonction ne dispose pas de mémoire associée, vous devez toujours indiquer des paramètres effectifs pour elle. Vous ne pouvez pas affecter de valeur initiale aux données locales d'une FC. [4]

Domaine d'application :

Une fonction contient un programme qui est exécuté quand cette fonction est appelée par un autre bloc de code. Vous pouvez faire appel à des fonctions pour :

- Renvoyer une valeur de fonction au bloc appelant (exemple : fonctions mathématiques),
- Exécuter une fonction technologique (exemple : commande individuelle avec combinaison binaire). [4]

Affectation de paramètres effectifs aux paramètres formels :

Un paramètre formel sert de paramètre générique au paramètre "réel", le paramètre effectif. Les paramètres effectifs remplacent les paramètres formels lors de l'appel d'une FC. Vous devez toujours affecter des paramètres effectifs aux paramètres formels d'une FC (par exemple, le paramètre effectif

"E3.6" au paramètre formel "Démarrage"). Les paramètres d'entrée, de sortie et d'entrée/sortie utilisés par la FC sont sauvegardés comme pointeurs désignant les paramètres effectifs du bloc de code qui a appelé la fonction. [4]

Différence importante entre les paramètres de sortie des FC et des FB :

Dans les blocs fonctionnels (FB), la copie du paramètre actuel figurant dans le DB d'instance est utilisée lors de l'accès aux paramètres. Si un paramètre d'entrée n'est pas transmis ou si un paramètre de sortie n'est pas affecté dans le bloc lors de l'appel d'un FB, ce sont les anciennes valeurs encore disponibles dans le DB d'instance qui sont utilisées (DB d'instance = mémoire du FB).

Les fonctions (FC) ne disposent pas de mémoire (FC). Contrairement au FB, l'affectation des paramètres formels n'est de ce fait pas optionnelle, mais indispensable. L'accès aux paramètres de la FC s'effectue via des adresses (pointeur interzone). Si un opérande de la zone de données (bloc de données) ou une variable locale du bloc appelant sont utilisés comme paramètre actuel, une copie de ce paramètre actuel est enregistrée temporairement dans les données locales du bloc appelant, lors de la transmission de paramètres. [4]

3.2.3.3. Blocs fonctionnels (FB) :

Les blocs fonctionnels font partie des blocs que vous programmez vous-même. Un bloc fonctionnel est un bloc avec rémanence. Un bloc de données d'instance lui est associé qui en constitue la mémoire. Les paramètres transmis au FB ainsi que les variables statiques sont sauvegardés dans le bloc de données d'instance. Les variables temporaires sont rangées dans la pile des données locales.

Les données sauvegardées dans le bloc de données d'instance ne sont pas perdues à l'achèvement du traitement du FB. En revanche, les données sauvegardées dans la pile des données locales le sont. [4]

Domaine d'application :

Un bloc fonctionnel contient un programme qui est exécuté quand ce bloc fonctionnel est appelé par un autre bloc de code. Les blocs fonctionnels facilitent la programmation de fonctions complexes souvent utilisées. [4]

3.2.3.4. Blocs de données globaux (DB) :

Contrairement aux blocs de code, les blocs de données ne contiennent pas d'instructions STEP 7. Ils servent à l'enregistrement de données utilisateur : ils contiennent des données variables que le programme utilisateur utilise. Les blocs de données globaux servent à l'enregistrement de données utilisateur pouvant être utilisées par tous les autres blocs.

La taille des DB peut varier. Vous trouverez la taille maximale autorisée dans les descriptions de CPU /70/ et /101/. [4]

C'est vous qui définissez l'organisation des blocs de données globaux.

DB globaux dans le programme utilisateur :

Lorsqu'il est appelé, un bloc de code (FC, FB ou OB) peut occuper temporairement de l'espace mémoire dans la zone des données locales (pile L). En plus de cette zone de données locales, ce bloc de code peut ouvrir une autre zone de mémoire sous la forme d'un DB. Contrairement aux données dans la zone des données locales, les données du DB ne sont pas effacées à la fermeture du DB ou à la fin du traitement du bloc de code correspondant.

Tout FB, FC ou OB peut lire les données contenues dans un DB global ou écrire des données dans un DB global. Ces données sont conservées dans les blocs de données même lorsqu'on quitte le DB. [4]

Il est possible d'ouvrir simultanément un DB global et un DB d'instance. La figure ci-après présente les différents accès aux blocs de données. [4]

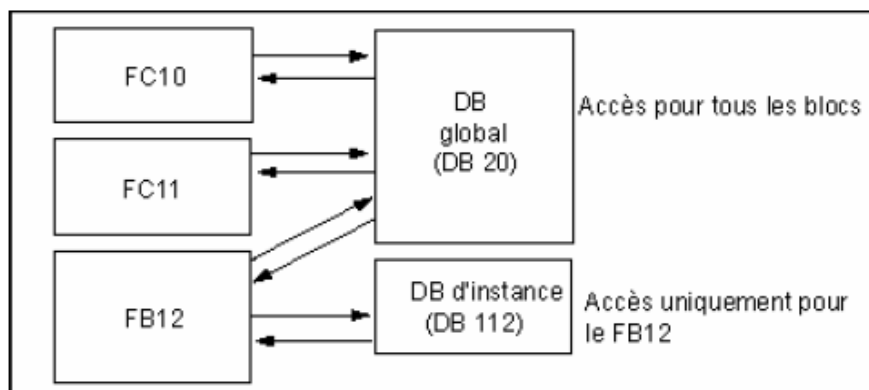


Figure 14 : les différents accès aux blocs

4. SIMATIC Manager :

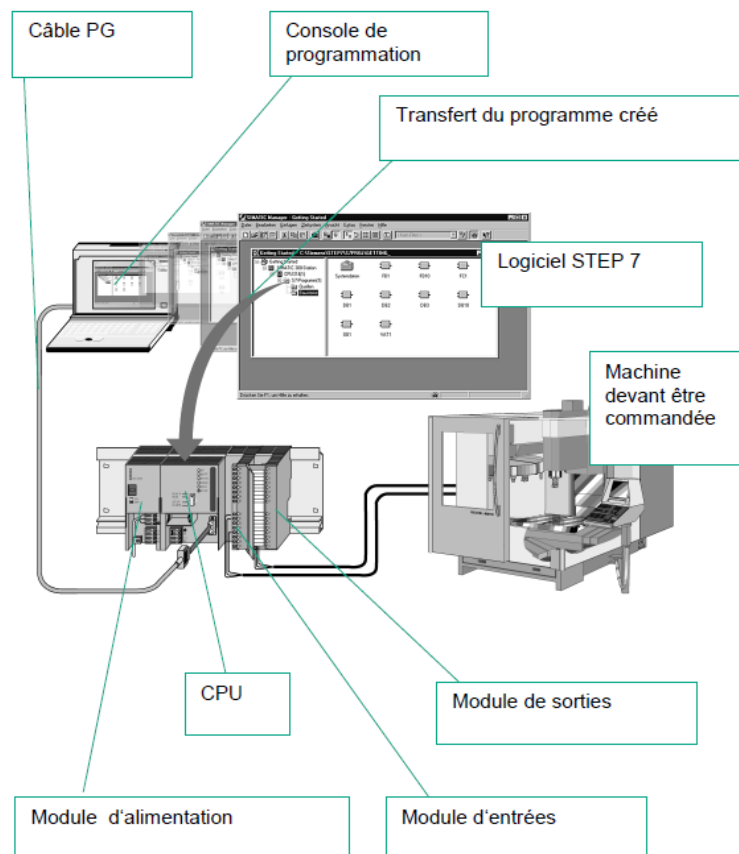
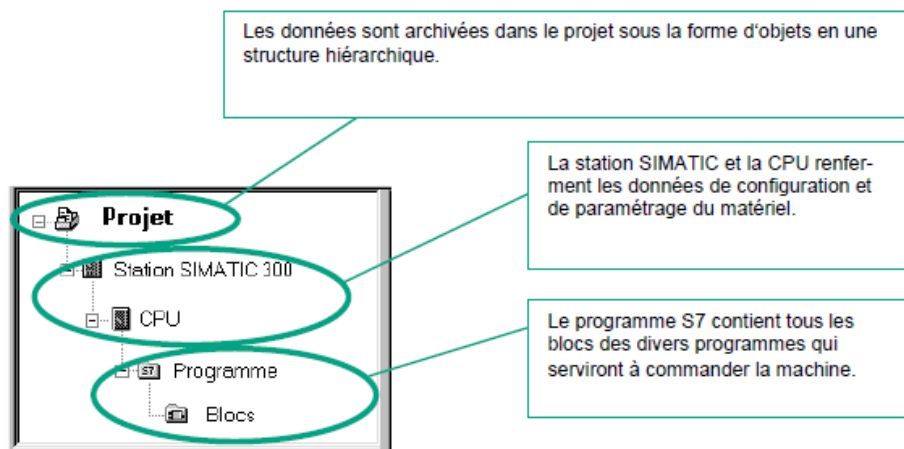


Figure15 : Schéma générale d'un système automatisé par l'automate SIEMENS

4.1. Lancer SIMATIC Manager et créer un projet :

Le lancement de STEP 7 fait s'ouvrir le gestionnaire de projets SIMATIC Manager.

L'assistant de STEP 7 est par défaut toujours activé. Celui-ci a pour but de vous assister dans la création de votre projet STEP 7. La structure du projet sert à ordonner les données et programmes créés au cours du projet. [5]



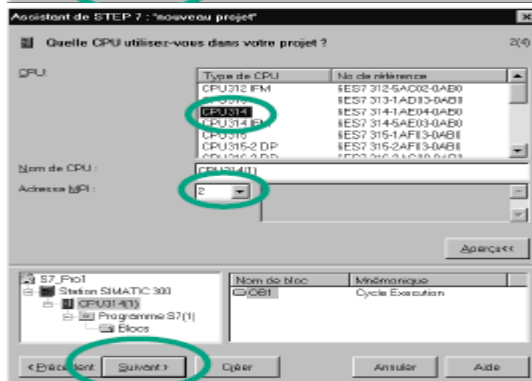
SIMATIC Manager

Double-cliquez sur l'icône **SIMATIC Manager**. Ceci lance l'assistant de STEP 7. [5]



Avec **Aperçu**, vous pouvez afficher ou masquer la structure du projet créé.

Avec **Suivant**, vous passez à la feuille suivante de l'assistant. [5]

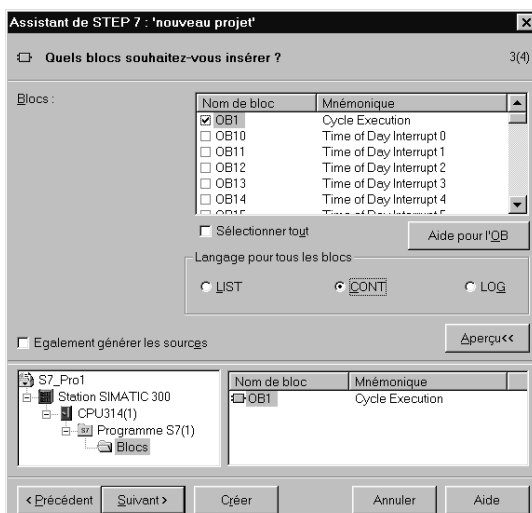


Sélectionnez pour l'exemple de projet de notre "Getting Started" la CPU 314.

Cet exemple a été conçu de telle sorte que vous pouvez sélectionner la CPU qui vous a été livrée.

L'adresse MPI est réglée par défaut sur 2.

Confirmez vos sélections et passez au prochain dialogue avec **Suivant**[5]



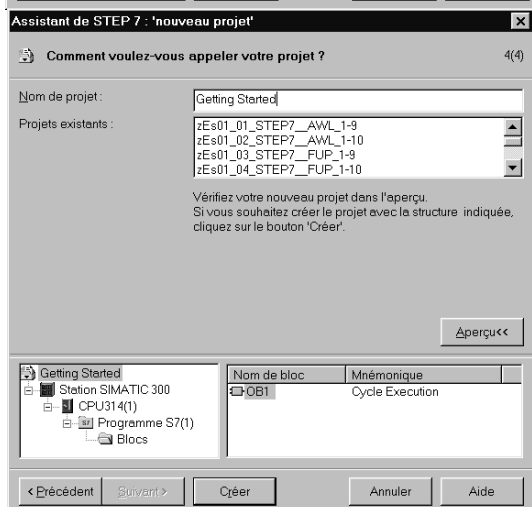
Sélectionnez le bloc d'organisation

OB1 (s'il n'est déjà sélectionné).

Choisissez votre langage de programmation : **CONT, LOG** ou **LIST**. [5]

Confirmez vos sélections avec

Suivant. [5]



Sélectionnez en double-cliquant dans la zone de texte "Nom du projet" le nom proposé et entrez à la place de celui-ci "Getting Started".[5]

Si vous cliquez sur **Créer**, votre nouveau projet sera créé selon la structure que vous pouvez voir avec **Aperçu**. [5]

Après l'exécution de la commande **Créer**, SIMATIC Manager s'ouvre avec la fenêtre du projet „Getting Started“ nouvellement créé. La signification et la manipulation des fichiers et dossiers créés sera expliquée dans les pages suivantes.

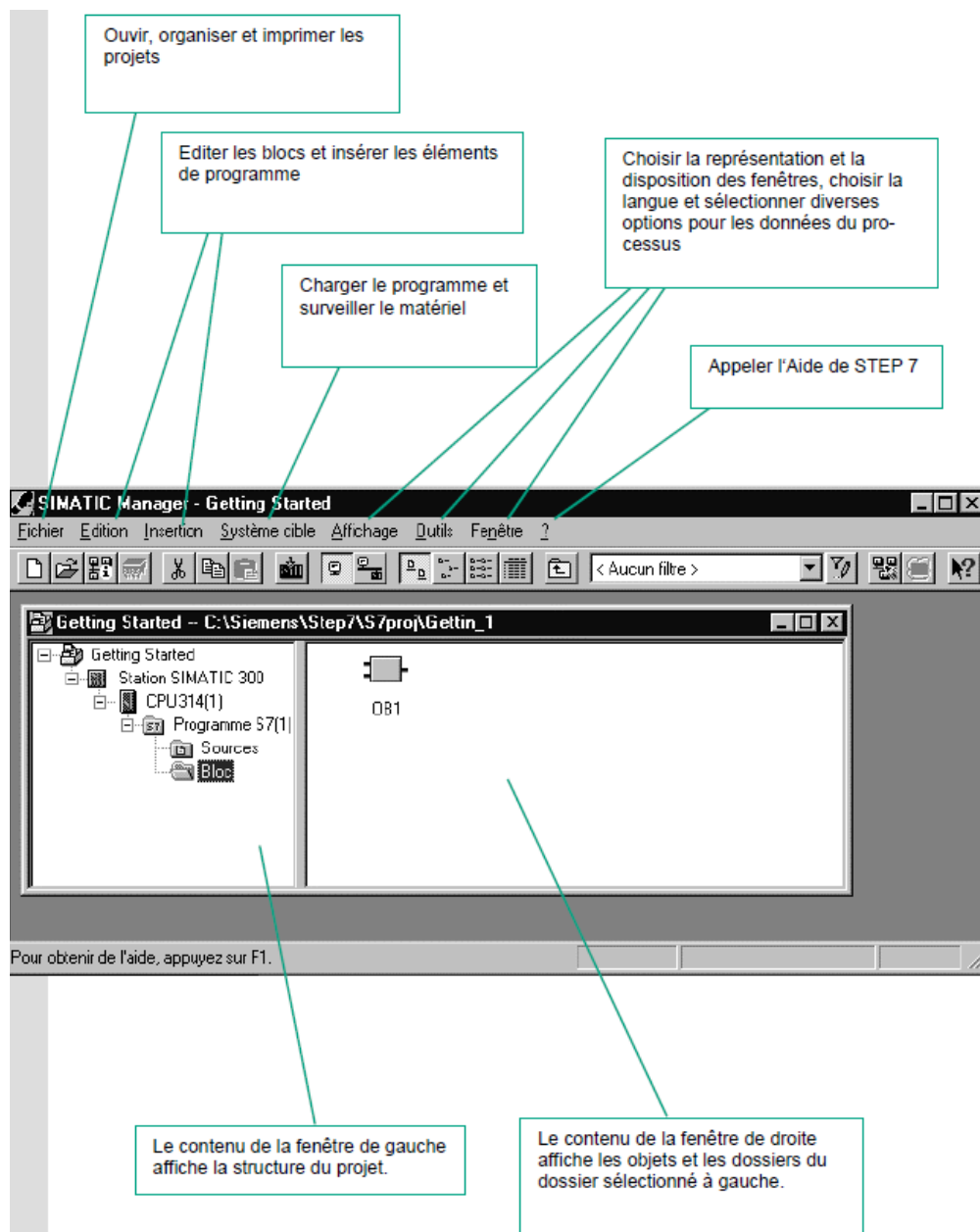
L'assistant de STEP 7 est activé par défaut à chaque nouveau lancement du programme.

Si vous voulez le désactiver, vous pouvez le faire dans le premier dialogue de l'assistant.

Sachez toutefois qu'il vous faudra créer manuellement chaque dossier du projet que vous créez sans l'assistant. [5]

4.2. Structure du projet dans SIMATIC Manager et appel de l'aide de STEP7 :

Dès que l'Assistant est refermé, SIMATIC Manager apparaît de nouveau avec la fenêtre du projet "Getting Started" qui vient d'être créée ouverte. C'est à partir de cette fenêtre que vous allez appeler toutes les fonctions et les autres fenêtres de STEP 7. [5]



5. Programmation symbolique :

5.1. Adresse absolue :

Chaque entrée et chaque sortie possède par défaut une adresse absolue déterminée par la configuration matérielle. Celle-ci est indiquée de manière directe, c'est-à-dire absolue.

L'adresse absolue peut être remplacée par des noms symboliques pouvant être librement choisis. [5]

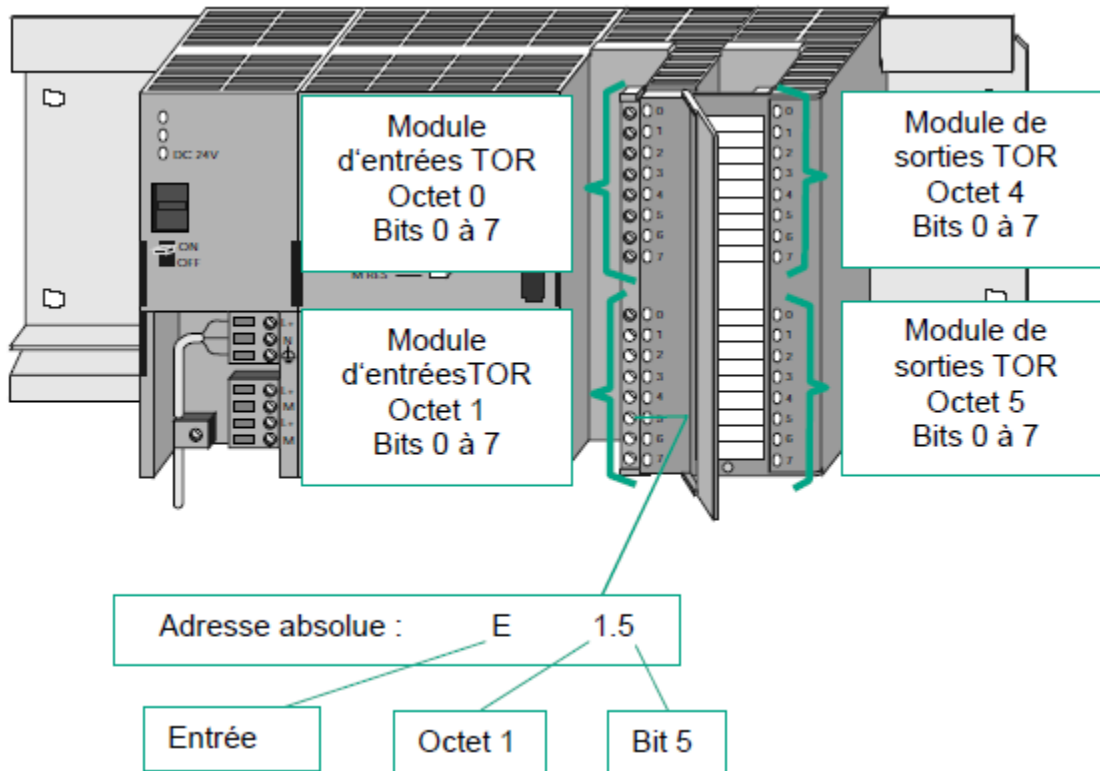


Figure16 : l'adresse absolue d'E/S

5.2. Programmation symbolique :

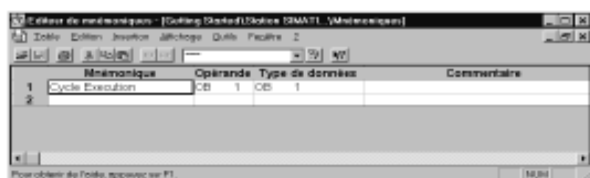
Vous affectez dans la table des mnémoniques un nom symbolique à toutes les adresses absolues que vous voulez appeler dans le programme ainsi que le type de données, par exemple pour l'entrée E0.1 le mnémonique Commutateur 1. Ces noms valent pour toutes les sections du programme. C'est pourquoi on les appelle des variables globales.

La programmation symbolique permet d'alléger l'écriture de votre programme qui y gagne en clarté. [5]

Travailler avec l'éditeur de mnémoniques :



Pour ouvrir celui-ci, naviguez dans la fenêtre de projet "Getting Started" jusqu'au **Programme S7 (1)** et double-cliquez sur **Mnémoniques**. [5]



Mnémonique	Opérande	Type de données	Commentaire
Cycle Execution	OB 1	OB 1	

Mnémonique	Opérande	Type de données	Commentaire
Programme principal	OB 1	OB 1	
Lampe verte	A 4.0	BOOL	

Commentaire

Mnémonique	Opérande	Type de données	Commentaire
1 Programme principal	OB 1	OB 1	
2 Lampe rouge	A 4.1	BOOL	
3 Lampe verte	A 4.0	BOOL	

La table des mnémoniques ne contient pour l'instant que le bloc d'organisation défini par défaut, l'**OB1**. [5]

Cliquez sur **Cycle Execution** et écrivez à la place de celui-ci "Programme principal".

Entrez dans la ligne 2 "Feu vert" et "A 4.0". Le type de données s'inscrit automatiquement dans la colonne du type.

Cliquez dans la ligne 1 ou 2 sur la colonne du commentaire pour entrer éventuellement un commentaire de mnémonique. L'action de la touche

Entrée clôt la ligne ou l'enregistrement et insère une nouvelle ligne de mnémonique.

Entrez dans la ligne 3 "Feu rouge" et "A 4.1" et confirmez la saisie avec **Entrée**[5]

6. Création d'un programme dans l'OB1 :

6.1. Ouvrir l'éditeur de programme dans la vue CONT, LIST ou LOG et ouvrir l'OB1 :

Choisissez votre langage de programmation : CONT, LIST ou LOG : Pour créer vos programmes S7, vous disposez dans STEP 7 de trois langages de programmation CONT, LIST ou LOG. Dans la pratique, vous devez vous décider pour l'un de ces langages. [5]

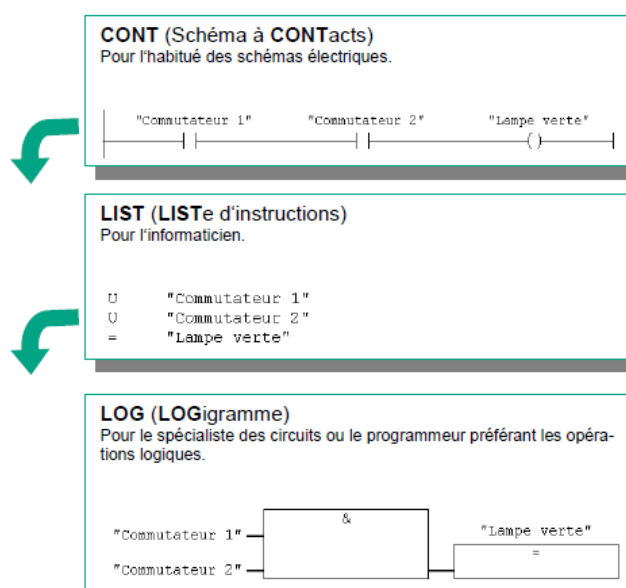


Figure 17 : les trois langages de programmation CON, LIST, LOG

L'éditeur de programme CONT/LIST/LOG :

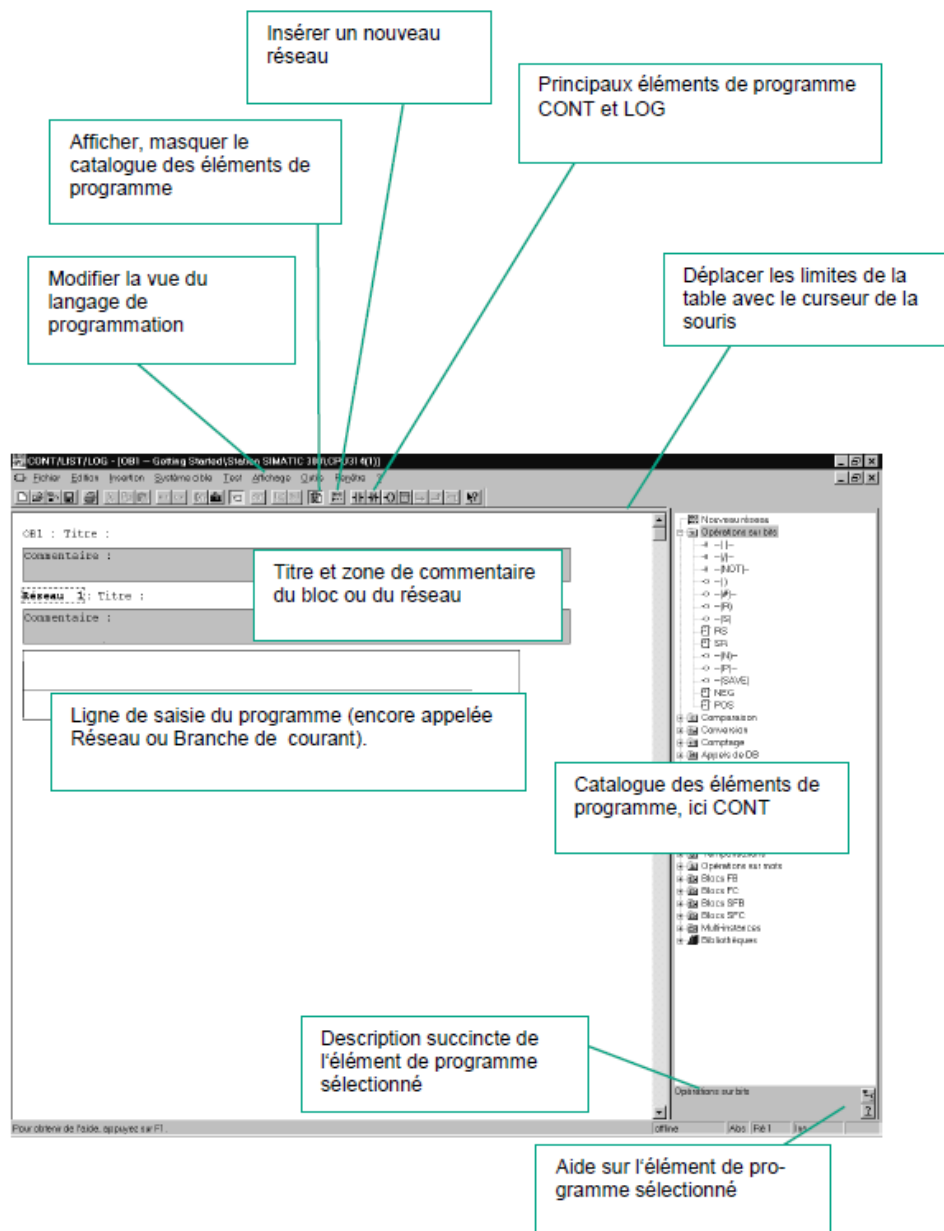


Figure 18 : L'éditeur de programme CONT/LIST/LOG [5]

6.2. Programmation de l'OB1 en CONT :

Vous apprendrez dans les pages suivantes à programmer un circuit série, un circuit parallèle et une bascule Mise à 1 /Remise à 0 en langage de programmation CONT (Schéma à **CONTacts**). [5]

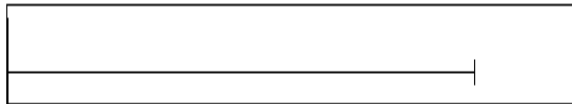
Programmation d'un circuit série en CONT :

Affichage	...
...	
<u>C</u> ONT	Ctrl+1
<u>L</u> IST	Ctrl+2
<u>L</u> OG	Ctrl+3
...	

Si vous ne l'avez pas encore fait, sélectionnez via le menu **Affichage** le langage de programmation **CONT**. [5]

OB1 : Titre :
Commentaire :

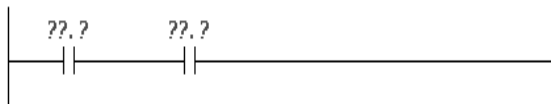
Cliquez dans la zone **Titre** de l'OB1 et entrez comme titre pour celui-ci "Exécution cyclique". [5]



Sélectionnez la position voulue de la branche de courant pour y insérer le premier élément. [5]



Cliquez dans la barre d'outils sur le bouton représenté ici et insérez un contact à fermeture.



Insérez de la même manière un second contact à fermeture.



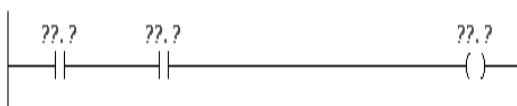
Insérez une bobine à l'extrémité droite de la branche de courant.



Pour achever notre circuit série, il manque encore les adresses des contacts et de la bobine.

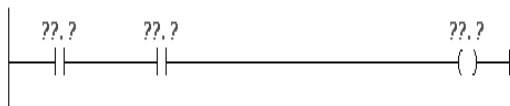


Vérifiez si vous avez activé la représentation symbolique. [5]

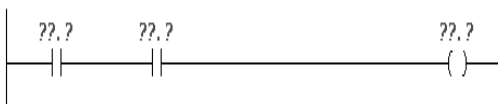


Cliquez sur '??.' et entrez le nom symbolique "Commutateur 1" (entre guillemets !).

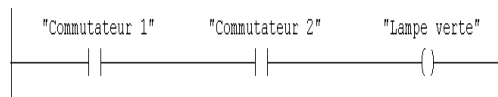
Confirmez avec la touche **Entrée**. [5]



Introduisez pour le second contact à fermeture le nom symbolique "Commutateur 2". [5]



Entrez pour la bobine le nom "Feu vert".



Votre circuit série est maintenant programmé.



Enregistrez le bloc lorsque le programme ne signale plus aucune erreur. [5]

Programmation d'un circuit parallèle en CONT :

 Titre : Sélectionnez le **Réseau 1**.

Commentaire :



Insérez un nouveau réseau.



Sélectionnez à nouveau la branche de courant.



Insérez un contact à fermeture et une bobine.



??,?

??,?

Sélectionnez la branche verticale du réseau.



Insérez une branche parallèle.



??,?

??,?

Insérez dans la branche parallèle un second contact à fermeture.



Fermez la branche en cliquant le cas échéant sur l'extrémité de la flèche).



??,?

??,?

Il ne reste plus qu'à compléter les adresses.

Procédez pour cela comme pour le circuit série. [5]



"Commutateur 3"

"Lampe rouge"

Entrez pour le contact du haut "Commutateur 3", pour le contact du bas "Commutateur 4" et pour la bobine "Feu rouge". [5]



Enregistrez le bloc.

II. SIMATIC HMI WinCC flexible 2008

1. Introduction à WinCC flexible

Introduction à SIMATIC HMI

Lorsque la complexité des processus augmente et que les machines et installations doivent répondre à des spécifications de fonctionnalité toujours plus sévères, l'opérateur a besoin d'un maximum de transparence. Cette transparence s'obtient au moyen de l'Interface Homme-Machine (IHM). [6]

Un système IHM constitue l'interface entre l'homme (opérateur) et le processus (machine/installation). Le contrôle proprement dit du processus est assuré par le système d'automatisation. Il existe par conséquent une interface entre l'opérateur et WinCC flexible (sur le pupitre opérateur) et une interface entre WinCC flexible et le système d'automatisation. Un système IHM se charge des tâches suivantes:

- **Représentation du process**

Le processus est représenté sur le pupitre opérateur. Lorsqu'un état du processus évolue p. ex., l'affichage du pupitre opérateur est mis à jour. [6]

- **Commande du processus**

L'opérateur peut commander le processus via l'interface utilisateur graphique. Il peut p. ex. définir une valeur de consigne pour un automate ou démarrer un moteur. [6]

- **Vue des alarmes**

Lorsque surviennent des états critiques dans le processus, une alarme est immédiatement déclenchée, p. ex. lorsqu'une valeur limite est franchie. [6]

- **Archivage de valeurs processus et d'alarmes**

Les alarmes et valeurs processus peuvent être archivées par le système IHM. Vous pouvez ainsi documenter la marche du processus et accéder ultérieurement aux données de la production écoulée. [6]

- **Documentation de valeurs processus et d'alarmes**

Les alarmes et valeurs processus peuvent être éditées par le système IHM sous forme de journal. Vous pouvez ainsi consulter les données de production à la fin d'une équipe p. ex. [6]

- **Gestion des paramètres de processus et de machine**

Les paramètres du processus et des machines peuvent être enregistrés au sein du système IHM dans des recettes. Ces paramètres sont alors transférables en une seule opération sur l'automate pour démarrer la production d'une variante du produit p. ex. [6]

SIMATIC HMI

SIMATIC HMI offre une gamme complète permettant de couvrir toutes les tâches de contrôle-commande. SIMATIC HMI vous permet de maîtriser le processus à tout instant et de maintenir les machines et installations en état de marche. Les systèmes SIMATIC HMI simples sont p. ex. de petites consoles à écran tactile mises en oeuvre sur site.

A l'autre extrémité de la gamme SIMATIC HMI se trouve des systèmes utilisés pour la conduite et la surveillance de chaînes de production. Il s'agira en l'occurrence des puissants systèmes client-serveur. [6]

Utilisation de SIMATIC WinCC flexible

WinCC flexible est le logiciel IHM pour la réalisation, par des moyens d'ingénierie simples et efficaces, de concepts d'automatisation évolutifs, au niveau machine. WinCC flexible réunit les avantages suivants:

- Simplicité
- Ouverture
- Flexibilité

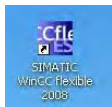
2. WinCC flexible Engineering System :

2.1. Notions élémentaires de l'interface utilisateur de configuration

Principe

WinCC flexible est le logiciel IHM pour la réalisation, par des moyens d'ingénierie simples et efficaces, de concepts d'automatisation évolutifs, au niveau machine.

Vous avez accès à toutes les fonctionnalités prises en charge par le pupitre opérateur connecté. Vous démarrez WinCC flexible, soit par l'icône placée sur le bureau de votre ordinateur de configuration, soit par le menu Démarrer de Windows. [6]



Sous WinCC flexible, vous ne pouvez ouvrir qu'un seul projet à la fois. Pour pouvoir traiter plusieurs projets simultanément, vous devrez démarrer WinCC flexible plusieurs fois. [6]

2.2. Interface logicielle de WinCC flexible

2.2.1. Eléments de l'interface utilisateur de WinCC flexible

Introduction :

L'environnement de travail de WinCC flexible se compose de plusieurs éléments. Certains de ces éléments sont liés à des éditeurs particuliers et uniquement visibles lorsque cet éditeur est activé. [6]

Eléments de WinCC flexible : WinCC flexible se compose des éléments suivants :

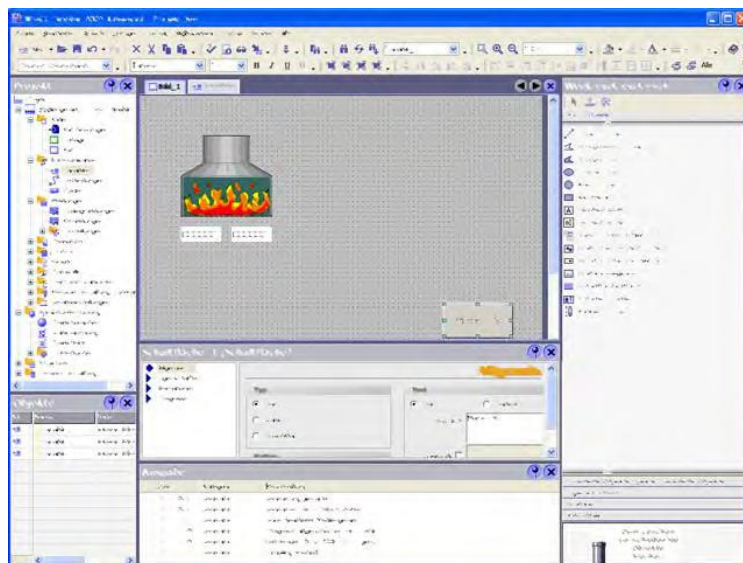


Figure 19 : L'environnement de travail de WinCC flexible

Menus et barres d'outils : Les menus et barres d'outils vous donnent accès à toutes les fonctions disponibles sous WinCC flexible. Lorsque vous positionnez le pointeur de la souris sur une fonction, vous obtenez une info-bulle. [6]

Zone de travail : La zone de travail sert à éditer les objets du projet. Tous les éléments de WinCC flexible sont disposés autour de la zone de travail. A l'exception de la zone de travail, vous pouvez disposer et configurer, déplacer ou masquer p. ex. tous les éléments comme bon vous semble. [6]

Fenêtre de projet : Tous les éléments et tous les éditeurs disponibles d'un projet sont affichés sous forme d'arborescence dans la fenêtre du projet et peuvent être ouverts à partir de cette fenêtre. [6]

Sous chaque éditeur se trouvent les dossiers, dans lesquels un stockage structuré des objets est possible. Pour les vues, les recettes, les scripts, les journaux et les dictionnaires personnalisés, vous pouvez en outre accéder directement aux objets configurés. Dans la fenêtre de projet, vous pouvez accéder aux paramètres du pupitre, à la localisation et à la gestion de versions. [6]

Fenêtre des propriétés : La fenêtre des propriétés vous permet d'éditer les propriétés des objets, p. ex. la couleur des objets de vue. Elle n'est disponible que dans certains éditeurs. [6]

Boîte à outils : La fenêtre d'outils vous propose un choix d'objets que vous pouvez insérer dans vos vues, p. ex. des objets graphiques et éléments de commande. La fenêtre d'outils contient en outre des bibliothèques d'objets et collections de blocs d'affichage prêts à l'emploi.

Bibliothèque : La bibliothèque fait partie de la fenêtre d'outils. La bibliothèque vous donne accès aux objets de vue préconfigurés. Les objets de la bibliothèque permettent d'augmenter la quantité d'objets de vue disponibles et d'améliorer votre productivité lors de la configuration par la réutilisation d'objets préconfigurés. La bibliothèque est le lieu central d'enregistrement des objets fréquemment utilisés tels que les objets graphiques et variables. [6]

Fenêtre des erreurs et avertissements : La fenêtre des erreurs et avertissements affiche les alarmes système générées p. ex. lors du test d'un projet. [6]

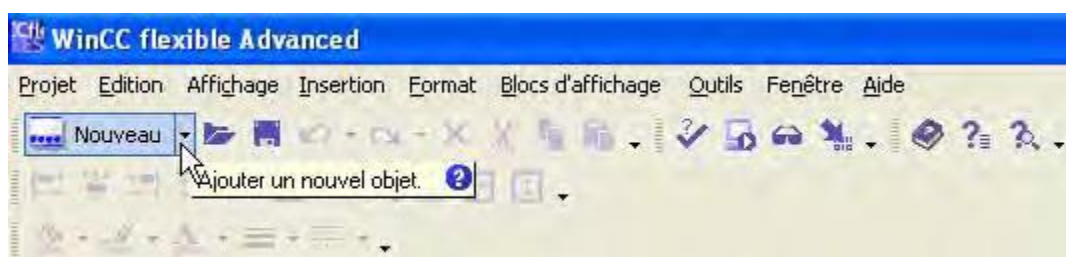
Fenêtre des objets : La fenêtre des objets affiche les éléments de la zone que vous avez sélectionnée dans la fenêtre de projet. [6]

2.2.2. Menus et barres d'outils

Introduction

Vous trouverez dans les menus et barres d'outils toutes les fonctions dont vous avez besoin pour la configuration de votre pupitre opérateur. Lorsqu'un éditeur est actif, les commandes de menu ou barres d'outils correspondantes sont visibles.

Lorsque vous positionnez le pointeur de la souris sur une commande, vous obtenez un infobulles correspondant à chaque fonction. [6]



Positionnement des barres d'outils :

Lors de la création d'un nouveau projet, les barres d'outils sont positionnées par défaut au bord supérieur de l'écran. La position des barres d'outils est liée à l'utilisateur qui est connecté sous Windows. Lorsque vous démarrez WinCC flexible et que, lors de la session précédente, vous aviez déplacé des barres d'outils avec la souris, ces barres d'outils reprennent la position qu'elles avaient à la fermeture de la dernière session. [6]

Menus

Les menus suivants sont disponibles sous WinCC flexible :

Menu	Descriptif technique
"Projet"	Contient des commandes de gestion de projets.
"Edition"	Contient des commandes servant à utiliser le presse-papiers ainsi que des fonctions de recherche.
"Affichage"	Contient des commandes permettant d'ouvrir et de fermer des éléments ainsi que des paramètres des fonctions zoom et plans. Un élément fermé peut être rouvert via le menu "Affichage".
"Insertion"	Contient des commandes pour l'insertion de nouveaux objets.
"Format"	Contient des commandes servant à disposer et à formater des objets de vue.
"Blocs d'affichage"	Contient des commandes servant à créer et éditer des blocs d'affichage.
"Outils"	Contient, entre autres, des commandes servant à changer de langue d'interface et à configurer les paramètres de base de WinCC flexible.
"Script"	Contient des commandes permettant de synchroniser et de vérifier la syntaxe de scripts.
"Fenêtre"	Contient des commandes de gestion de plusieurs vues de la zone de travail, permettant p. ex. de changer de vue.
"Aide"	Contient des commandes d'accès aux fonctions d'aide.

Tableau 16 : les menus disponibles sous Win CC

Les menus et les options de menus disponibles dépendent de l'éditeur utilisé. [6]

Barres d'outils

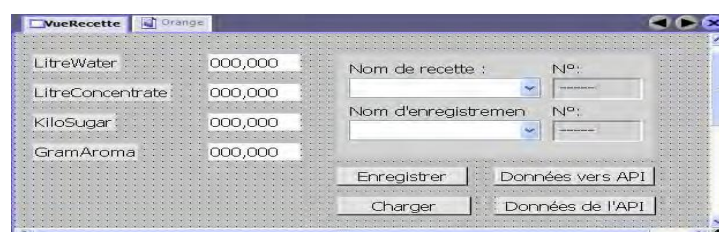
Les barres d'outils vous permettent d'accéder rapidement à des fonctions importantes dont vous avez fréquemment besoin. Vous pouvez configurer chaque barre d'outils comme suit :

- Ajouter ou supprimer des boutons
- Modifier la position[6]

2.2.3. Zone de travail

Introduction

Dans la zone de travail, vous éditez les données de projet soit sous forme de tableau, ce qui est le cas des variables p. ex., soit sous forme graphique, ce qui est le cas des vues de process p. ex. [6]



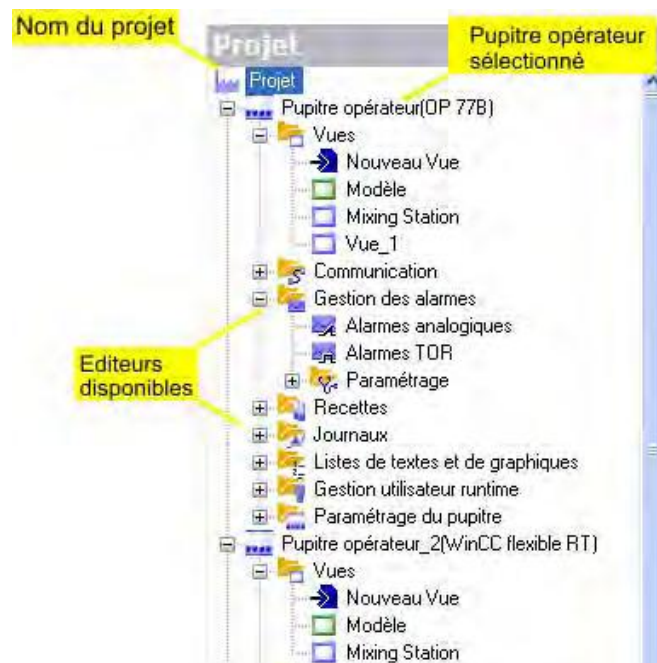
Description

Chaque éditeur ouvert est représenté dans la zone de travail dans un onglet particulier. Dans le cas d'éditeurs graphiques, chaque élément est représenté dans un onglet distinct. Si vous avez ouvert plusieurs éditeurs simultanément, un seul onglet est actif. Pour changer d'éditeur, il suffit de cliquer avec la souris sur l'onglet voulu. Vous pouvez ouvrir au maximum 20 éditeurs simultanément. [6]

2.2.4. Fenêtre de projet

Introduction

La fenêtre du projet est le poste central de traitement du projet. Tous les éléments et tous les éditeurs disponibles d'un projet sont affichés sous forme d'arborescence dans la fenêtre du projet et peuvent être ouverts à partir de cette fenêtre. A chaque éditeur correspond une icône qui vous permet d'identifier les objets qui lui sont associés. Seuls les éléments pris en charge par le pupitre opérateur sélectionné apparaissent dans la fenêtre de projet. Dans la fenêtre de projet, vous pouvez accéder aux paramètres du pupitre, à la localisation et à la gestion de versions. [6]



Description

La fenêtre de projet affiche la structure hiérarchique du projet :

- Projet
- Pupitres opérateurs
- Dossier
- Objets

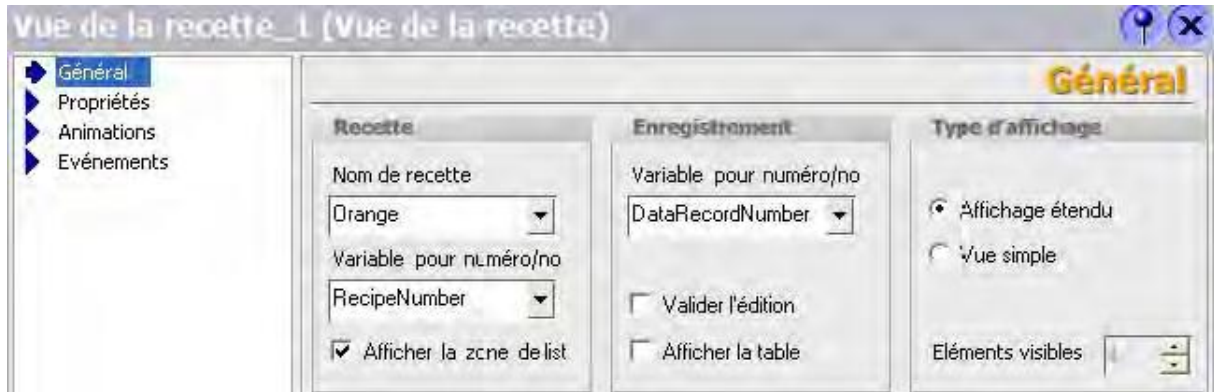
La fenêtre de projet sert à créer des objets et à les ouvrir pour les éditer. Vous pouvez créer des dossiers pour structurer les objets de votre projet. La fenêtre de projet s'utilise de façon analogue à l'explorateur Windows. Vous pouvez ouvrir pour chaque objet un menu contextuel qui regroupe les principales commandes. [6]

Les éléments des éditeurs graphiques sont affichés dans la fenêtre de projet et dans la fenêtre des objets. Les éléments de tableurs sont uniquement affichés dans la fenêtre des objets. [6]

2.2.5. Fenêtre des propriétés

Introduction

La fenêtre des propriétés permet de modifier les propriétés d'un objet sélectionné dans la zone de travail. Le contenu de la fenêtre des propriétés dépend de l'objet sélectionné. [6]



Description

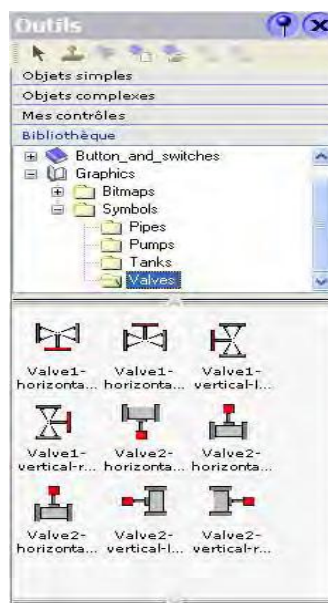
La fenêtre des propriétés affiche les propriétés de l'objet sélectionné classées par catégories. Aussitôt que vous quittez une zone de saisie, les modifications de valeurs effectuées sont actives. [6]

Si vous entrez une valeur inadmissible, celle-ci s'affiche sur fond de couleur. L'info-bulles vous fournit des informations, p. ex. sur la plage des valeurs valides. [6]

2.2.6. Bibliothèque

Introduction

La bibliothèque fait partie de la fenêtre d'outils. La bibliothèque est le lieu central d'enregistrement des objets fréquemment utilisés. Tout objet enregistré dans la bibliothèque ne doit être configuré qu'une seule fois. Vous pouvez ensuite le réemployer à volonté. Les objets de la bibliothèque permettent d'augmenter la quantité d'objets de vue disponibles et d'améliorer votre productivité lors de la configuration par la réutilisation d'objets préconfigurés. [6]



Description : Sous WinCC flexible, on distingue la bibliothèque globale de la bibliothèque spécifique au projet :

- **Bibliothèque globale**

La bibliothèque globale n'est pas enregistrée avec le projet dans la base de données mais sous forme de fichier. Le fichier enregistré est stocké par défaut dans le répertoire d'installation de WinCC flexible. La bibliothèque globale est disponible pour tous les projets. [6]

- **Bibliothèque spécifique projet**

La bibliothèque de projet qui est enregistrée avec les données de projet dans la base de données, est uniquement disponible dans le projet où elle a été créée.

Dans ces deux bibliothèques, vous pouvez créer des dossiers pour structurer les objets qu'elles contiennent. Vous pourrez par ailleurs transférer à tout moment les éléments d'une bibliothèque spécifique dans la bibliothèque globale. [6]

2.2.7. Fenêtre des erreurs et avertissements

Introduction

La fenêtre des erreurs et avertissements affiche les événements système générés p. ex. lors du test d'un projet. [6]



Description : Dans la fenêtre des erreurs et avertissements, les événements système sont affichées par défaut dans leur ordre d'apparition. Les catégories désignent respectivement le module WinCC flexible qui a généré un événement système. Les alarmes système de la catégorie "Compilateur" sont p. ex. générées durant le contrôle de cohérence.

Pour trier les événements système, cliquez avec la souris sur la barre de titre de la colonne correspondante. A partir du menu contextuel, vous pouvez atteindre le lieu d'occurrence de l'erreur ou une variable, vous pouvez aussi copier ou supprimer une alarme système.

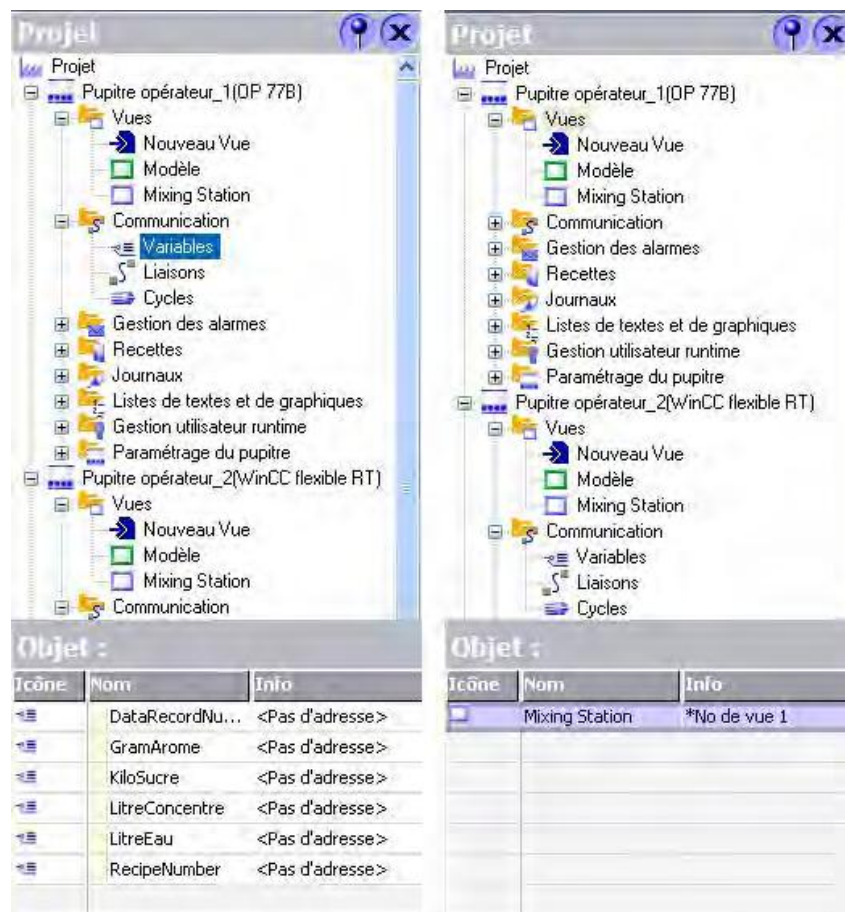
La fenêtre des erreurs et avertissements affiche tous les événements système se rapportant à la dernière action. A chaque nouvelle action, tous les événements système précédents sont écrasés. Afin que vous ayez tout de même accès à tous les événements système, ceux-ci sont enregistrés dans un fichier journal distinct. [6]

2.2.8. Fenêtre des objets

Introduction

Lorsque vous sélectionnez des dossiers ou des éditeurs dans la fenêtre de projet, leur contenu s'affiche dans la fenêtre des objets.

La figure suivante montre comment la sélection dans la fenêtre de projet influence l'affichage dans la fenêtre des objets. [6]



Description : Dans la fenêtre des objets, un double clic sur un objet permet d'ouvrir l'éditeur correspondant. Vous pouvez effectuer des actions de glisser-déplacer avec tous les objets affichés dans la fenêtre des objets. Les actions de glisser-déplacer suivantes sont p. ex. prises en charge :

- Placer une variable dans la vue de process qui se trouve dans la zone de travail : Créé un champ d'E/S lié à la variable.
- Placer une variable sur un champ d'E/S : Liaison de la variable au champ d'E/S.
- Placer une vue de process dans la vue de process qui se trouve dans la zone de travail : Création d'un bouton de raccourci vers cette vue de process.

Les noms d'objet longs sont représentés en abrégé dans la fenêtre des objets. Lorsque vous positionnez le pointeur de la souris sur l'objet, le nom de l'objet s'affiche en entier sous forme d'info-bulles.

Pour trouver rapidement l'objet recherché parmi un grand nombre d'objets, entrez sur le clavier la première lettre de l'objet recherché. [6]

III. Conclusion :

Ce chapitre est une étude générale de l'environnement SIMATIC managed STEP7 et l'environnement Win CC flexible qui nous avons utilisé pour la programmation de notre CPU et pour faire un interface homme machine HMI dans notre PC portable.

CHAPITRE 4

FEU DE CIRCULATION

1- Introduction :

Un feu de circulation routière est un dispositif permettant la régulation du trafic routier entre les usagers de la route, les véhicules et les piétons. [7]

Les feux destinés aux véhicules à moteurs sont généralement de type tricolore, auxquels peuvent s'ajouter des flèches directionnelles. Ceux destinés aux piétons sont bicolores et se distinguent souvent par la reproduction d'une silhouette de piétons. Les feux tricolores pour cyclistes se distinguent par la reproduction d'une bicyclette. [7]

En Europe, la convention européenne sur la signalisation routière (convention de Vienne sur la signalisation routière) de 1968, à laquelle se sont depuis ralliés de nombreux États, contient des dispositions qui fixent les catégories, formes et couleurs des signaux routiers, dont les signaux lumineux. [7]

Un carrefour à feux tricolores est commandé par un contrôleur de feux, appareil électronique de contrôle/commande. [7]

Les feux sont généralement déclinés à partir de deux couleurs de base : le rouge pour fermer, le vert (ou encore le bleu plus rarement) pour ouvrir. L'orange (ou parfois le jaune) est également utilisé et sert à signaler le passage du feu vert au feu rouge. Ces couleurs ont l'avantage d'être très différentes, sauf pour la plupart des daltoniens. [7]

2- Histoire

Bien qu'il existe très peu de sources historiques sur les signaux routiers, il semblerait que ce soit à Londres, au coin de Bridge Street et de Palace Yard, le 10 décembre 1868, qu'un feu de signalisation ait été utilisé pour la première fois, sous la forme d'une lanterne à gaz pivotante aux couleurs rouge et verte nécessitant la présence d'un agent de police pour le manœuvrer (ce dernier sera grièvement blessé le 2 janvier 1869). Ce n'est que bien plus tard que les feux, après leur généralisation sont devenus tricolores par l'adjonction d'une phase intermédiaire marquée par la couleur jaune-orange. [7]

Aux États-Unis, les premiers feux - bicolores - seront installés à Cleveland en 1914.

En France, l'inventeur du feu tricolore est Léon Foenquinos, lequel le décrit ainsi dès 1920 : « on installera, aux angles des croisements de rues, des poteaux ayant trois mètres de hauteur, sur lesquels seront fixés des signaux électriques lumineux et sonores (...) ». Léon Foenquinos diffusera ses idées et cédera toutes ses inventions à la France par amour de son pays. [7]

En France, le 5 mai 1923, au croisement des boulevards Saint-Denis et Sébastopol, à Paris, est posé un feu de signalisation. Il est rouge et accompagné d'une sonnerie. C'est le premier en France. Il faudra attendre dix ans avant que n'apparaissent les feux vert et orange. [7]

Dès à présent, on peut voir des feux tricolores dont les ampoules sont remplacées par des diodes électroluminescentes et qui affichent le décompte des secondes restant avant le changement d'état. [7]

3- Usage1

L'emploi des feux de circulation a pour but d'assurer la sécurité de tous les usagers de la voirie, piétons et conducteurs, et de faciliter l'écoulement des flux de circulation denses. On peut citer comme exemples d'emploi :

- la gestion du trafic aux intersections ;
- la traversée des piétons, autour des intersections gérées par des feux et où le moment de trafic est élevé ou le sentiment d'insécurité des piétons important ;
- l'exploitation par sens uniques alternés d'une section où le croisement est impossible ou dangereux (ouvrage d'art étroit, emprise de travaux, etc.) ;
- l'affectation de certaines voies d'une chaussée à un sens de circulation en fonction des besoins, ou leur condamnation momentanée ;
- le contrôle d'accès à certaines voies rapides ;
- la gestion d'un point de contrôle des personnes ou des véhicules nécessitant leur arrêt (péage) ;
- la protection d'obstacles intermittents (passages à niveau, traversées de voies de tramways, ponts mobiles, passages d'avions, avalanches, etc.). [7]

4- Séquences de feux

Il existe trois principales séquences de feux :

4-1- Feux de signalisation à trois états

Utilisé en France, aux États-Unis, au Canada, en Nouvelle-Zélande, au Sénégal et en Belgique.

Rouge : « Tout conducteur doit marquer l'arrêt absolu devant un feu de signalisation rouge, fixe ou clignotant. »

Vert : « Les feux de signalisation verts autorisent le passage des véhicules [...]. »

Jaune : « Tout conducteur doit marquer l'arrêt devant un feu de signalisation jaune fixe, sauf dans le cas où, lors de l'allumage dudit feu, le conducteur ne peut plus arrêter son véhicule dans des conditions de sécurité suffisantes. » [7]

4-2- Feux de signalisation à quatre états

Utilisé au Royaume-Uni, en Allemagne, en Suisse, en République tchèque, en Hongrie, en Slovaquie, en Croatie, en Bosnie-Herzégovine, en Serbie, en Turquie, en Israël, en Chine et au Canada. [7]

Le deuxième état, jaune bref sans extinction du rouge, permet aux conducteurs de se préparer à démarrer mais n'autorise pas le passage.

En Italie, le jaune clignote avec le vert pour prévenir du passage au rouge.

Au Canada, une flèche verte peut être présente en plus des trois feux de couleur, indiquant la priorité pour un virage (généralement à gauche). La priorité de virage à gauche est souvent indiquée simplement par le clignotement du feu vert, d'une durée limitée, lors du passage du feu rouge au feu vert (la circulation à contre-sens est alors maintenue au feu rouge). [7]

4-3- Feux de signalisation à cinq états

Utilisé en Arménie, en Autriche, en Lituanie et au Canada.

L'état supplémentaire, vert clignotant entre les états du vert et du jaune, prévient la fin de l'état vert. [7]

Au Canada, si le droit de passage à droite est interdit aux feux rouges (au Nouveau-Brunswick et au Québec, le droit de passage à droite sur feux rouges est autorisé) les feux de circulation contiennent deux flèches vertes (gauche et droite) est contenu avec le rouge, jaune et vert. [7]

5- Carrefour à feux

Un carrefour à feux est une intersection dont le trafic est réglé par des feux de signalisation lumineux pilotés par un contrôleur. Le réglage des cycles de feux doit permettre d'assurer la sécurité des automobilistes et des piétons tout en permettant un débit maximal. [7]

5-1- Définitions

Un certain nombre de notions permettent de décrire et d'organiser les carrefours à feu :

- Conflit : croisement de deux mouvements, de véhicules ou de piétons.
- Courant : ensemble de mouvements réunis sur une même voie.
- Cycle de feux : enchaînement des différentes phases des feux.
- Durée de vert minimum : durée minimale de vert nécessaire pour écouler le stock d'usagers constitué pendant la durée de rouge.
- Mouvement : déplacement de véhicules dont on distingue l'origine et la destination, déplacement de piétons sans origine ni destination. Pour calculer le débit d'un mouvement en compte le nombre d'u.v.p. auquel on applique un coefficient d'équivalence en fonction du type de mouvement. [7]
 - Direct : mouvement consistant pour les véhicules à traverser le carrefour tout droit. Son coefficient d'équivalence est de 1
 - Tourne-à-Droite : mouvement consistant pour les véhicules à tourner à droite dans le carrefour. Son coefficient d'équivalence est de 1,1 quand l'angle est de 90° et de 1,2 si le mouvement tournant présente un angle aigu difficile à négocier. Un mouvement tournant non prioritaire ayant un conflit avec un flux de piétons supérieur à 250 piétons par heure aura un coefficient de 1,3 ou plus.
 - Tourne-à-Gauche : mouvement consistant pour les véhicules à tourner à gauche dans le carrefour. Ce mouvement est le plus complexe à gérer puisqu'il croise le mouvement direct face auquel il n'a pas la priorité et le tourne-à-gauche du sens opposé. Le coefficient d'équivalence est très variable et dépend essentiellement du débit et du nombre de mouvements antagonistes

prioritaires. Le coefficient peut ainsi généralement varier de 1,1 à 1,7. Plus le cycle est court, plus le coefficient du tourne à gauche sera faible.

- Tourne-à-Gauche à l'indonésienne : mouvement dans lequel les tourne-à-gauche ne sont pas antagonistes. Les tourne-à-gauche à l'indonésienne s'accompagnent souvent d'une voie de stockage dédiée au tourne-à-gauche.
- Phase : période durant laquelle un ou plusieurs courants sont admis dans le carrefour.
- Temps de dégagement : temps de latence entre deux phases qui admettent des courants antagonistes dans le carrefour afin de permettre la libération complète du point de conflit (par exemple, les véhicules effectuant un tourne-à-gauche, stockés au centre du carrefour).
- u.v.p : Unité de Voiture Particulière. Unité permettant de mesurer les débits d'un mouvement, d'une voie ou du carrefour. Un deux roues vaut 0,3 U.v.p., un véhicule léger (< à 3,5 tonnes) vaut 1 u.v.p., un poids lourd ou un bus valent 2 u.v.p., un semi-remorque ou un bus articulé valent 3 u.v.p.
- Vitesse moyenne : le réglage de la durée des différentes phases se fait en considérant des vitesses moyennes d'1m/s pour les piétons (sauf devant une école, un hôpital ou une maison de retraite) et de 10m/s pour les véhicules.
- Voie : portion de la chaussée dédiée à un courant. [7]

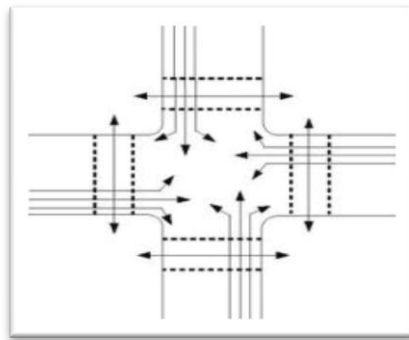


Figure 20 : Les 16 mouvements possibles sur un carrefour à quatre branches

6- Contrôleur de carrefour à feux

Un contrôleur de carrefour à feux est un automate spécialisé destiné à la gestion des feux de circulation en régulation de trafic. [7]

6-1- Fonctions de régulation

La fonction principale d'un contrôleur est d'assurer la commande des feux en respectant les contraintes de sécurité définies entre les lignes de feux antagonistes. Les feux sont généralement pilotés en 230 volts, la basse tension est très peu utilisée. [7]

Afin d'effectuer une régulation dynamique en fonction du trafic, le contrôleur est capable de détecter les véhicules (généralement via des boucles électromagnétiques placées dans la chaussée ou parfois par des radars à effet Doppler fixés en haut des poteaux) et les piétons (via des boutons-poussoirs d'appels). [7]

Un contrôleur de carrefour peut être capable de gérer plusieurs carrefours élémentaires totalement indépendants (notion appelée multi-carrefours). [7]

Les contrôleurs peuvent être secondés avec d'autres appareils compagnons auxquels ils sont raccordés dans l'armoire. [7]

6-2- Fonctions de sécurité

Le contrôleur doit vérifier que les signaux soient correctement allumés ou éteints suivant ce qui est commandé (mesure du courant et de la tension). [7]

La réglementation en vigueur impose au minimum que le signal de rouge principal soit contrôlé à des fins de sécurité. [7]

En cas de défaut sur un de ces signaux, le contrôleur passe en clignotant de sécurité (feux jaunes qui clignotent à la seconde). [7]

En cas de défaillance générale détectée, un dispositif dit de "chien de garde" matériel se doit également d'effectuer la gestion du clignotant de sécurité. [7]

Chapitre 5

Réalisation du projet

1. Introduction

Dans ce chapitre, nous allons présenter notre travail qui est la réalisation d'une carte simulé un carrefour a quatre branches commandé par un automate SIEMENS qui intègre un programme CON produit de la traduction d'une solution pour géré le carrefour par RDP.

Nous apportons aussi une description détaillée sur les divers changements de notre solution par l'RDP.

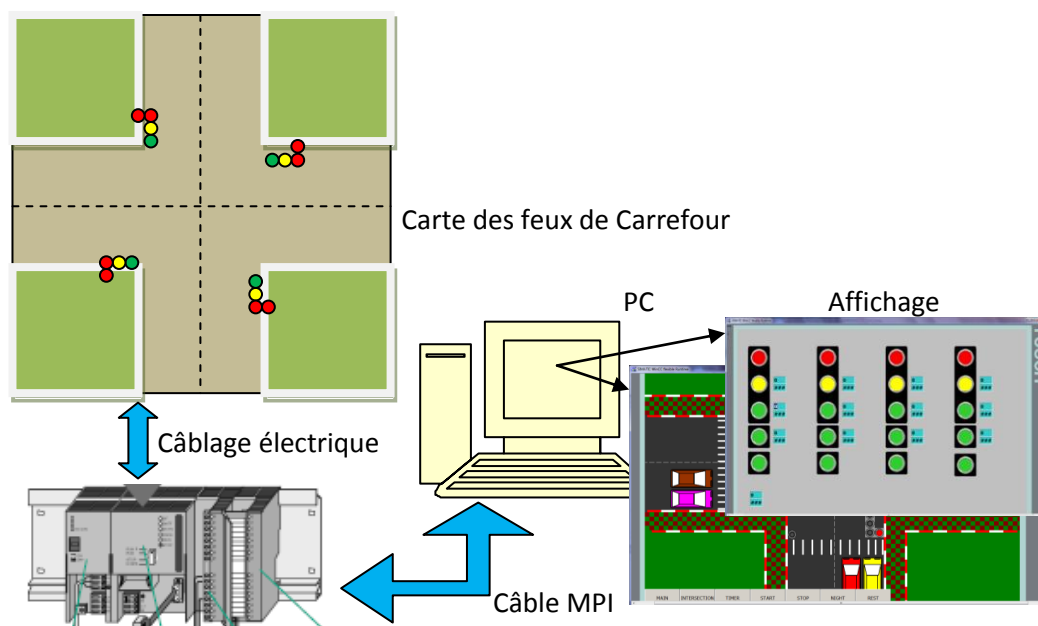
2. Représentation générale de projet

Notre projet consiste a réalisé un carte simuler un carrefour à quatre bronche commander par un automate SIEMENS et on a utilisé un PC pour l'interface homme machine HMI.

Cet automate contient un CPU314 V1.2, deux modules d'entré SM321 DI16X24V et deux modules de sortie SM322 DO16X24V, capable a gérée le carrefour à quatre branches

L'objectif de PC est pour faire l'interface homme machine HMI et pour la programmation de l'automate doit être contient logiciel SIMATIC Manager et Win CC flexible

La figure suivant montre le schéma bloc du système à réaliser



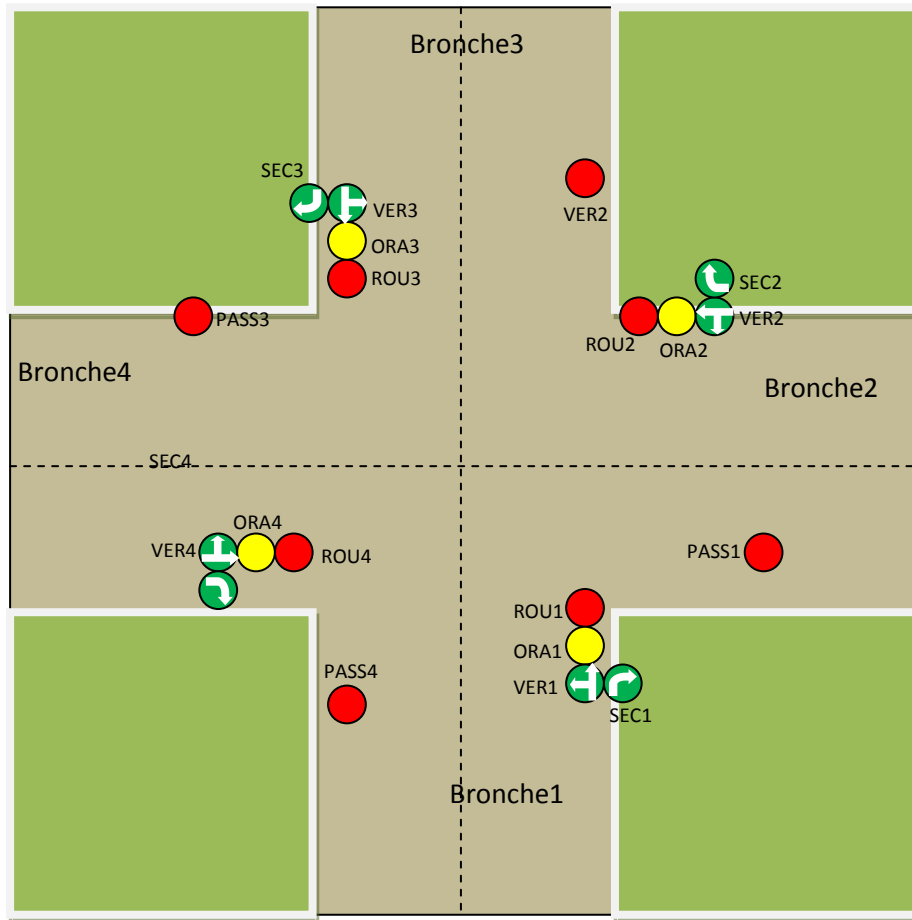
3. Fonctionnement de feux de signalisation

Notre intersection contient quatre branches

- Branche 1 possède les lompes suivantes : vert1, orange1, rouge1, passage1 et secondaire1.
- Branche 2 possède les lompes suivantes : vert2, orange2, rouge2, passage2 et secondaire2.
- Branche 3 possède les lompes suivantes : vert3, orange3, rouge3, passage3 et secondaire3.

- Branche 4 possède les lampes suivantes : vert4, orange4, rouge4, passage4et secondaire4.

Le schéma suivant montre les quatre branches et le codage des lampes de chaque branche :



Le fonctionnement des lampes de l'intersection se fait selon les états suivant :

Etat 1 :

- VER1 lumineux
- SEC1 lumineux
- SEC4 lumineux
- ROU2 lumineux
- ROU3 lumineux
- ROU4 lumineux
- VER2 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU1 éteint
- SEC2 éteint
- SEC3 éteint
- PASS1 éteint

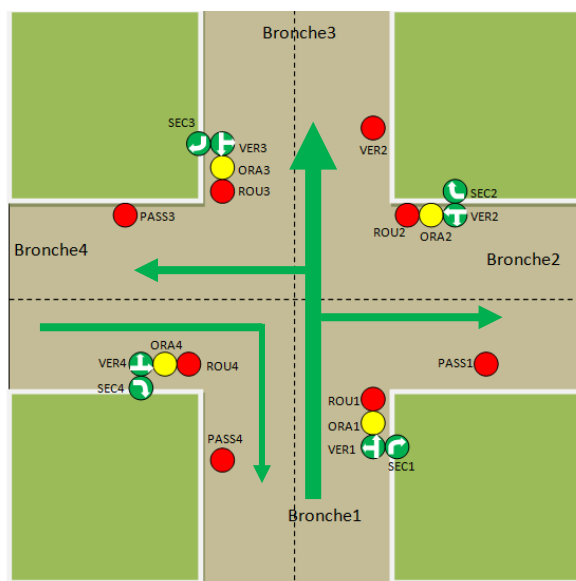


figure 23 : état1 de l'intersection

- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

Cet état reste pendant une période de temps TVER1 pour retardé l'éteint du lompe VET1

Etat2 :

- ORA1 lumineux
- ROU2 lumineux
- ROU3 lumineux
- ROU4 lumineux
- VER1 éteint
- VER2 éteint
- VER3 éteint
- VER4 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU1 éteint
- SEC1 éteint
- SEC2 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

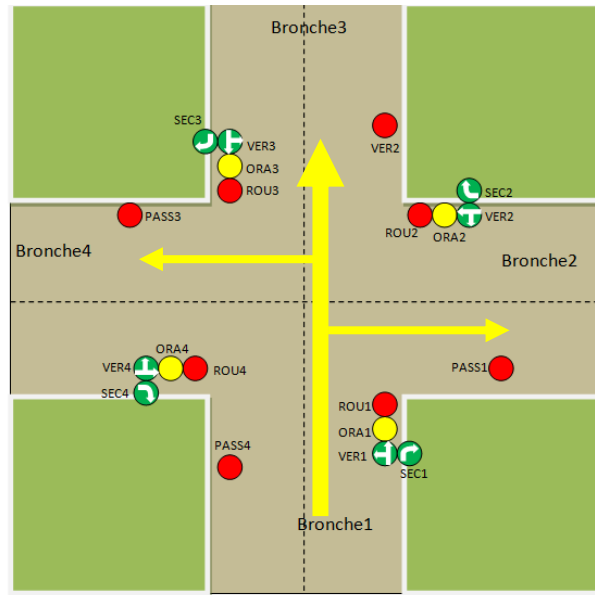


figure 24 : état 2 de l'intersection

Cet état reste pendant une période de temps TORA1 pour retardé l'éteint du lompe ORA1

Etat 3 :

- VER2 lumineux
- PASS2 lumineux
- SEC1 lumineux
- ROU1 lumineux
- ROU3 lumineux
- ROU4 lumineux
- VER1 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU2 éteint
- SEC2 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS3 éteint
- PASS4 éteint

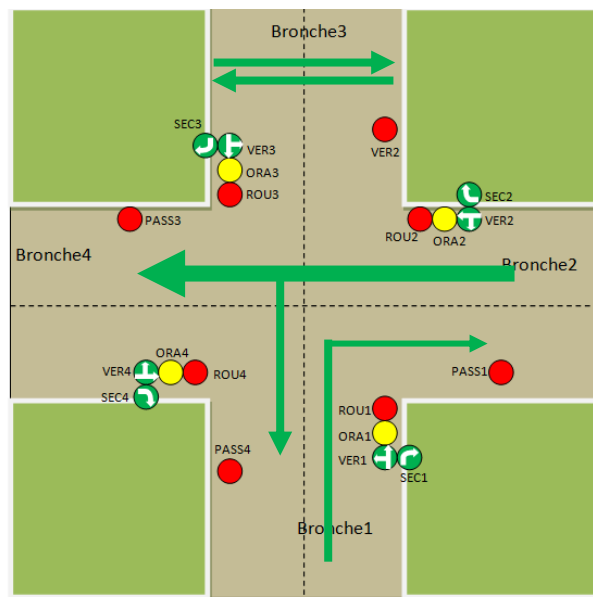


figure 25 : état 3 de l'intersection

Cet état reste pendant une période de temps TPASS2 pour retardé l'éteint du lompe PASS2

Etat 4 :

- VER2 lumineux
- SEC1 lumineux
- SEC 2 lumineux
- ROU1 lumineux
- ROU3 lumineux
- ROU4 lumineux
- VER1 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU2 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

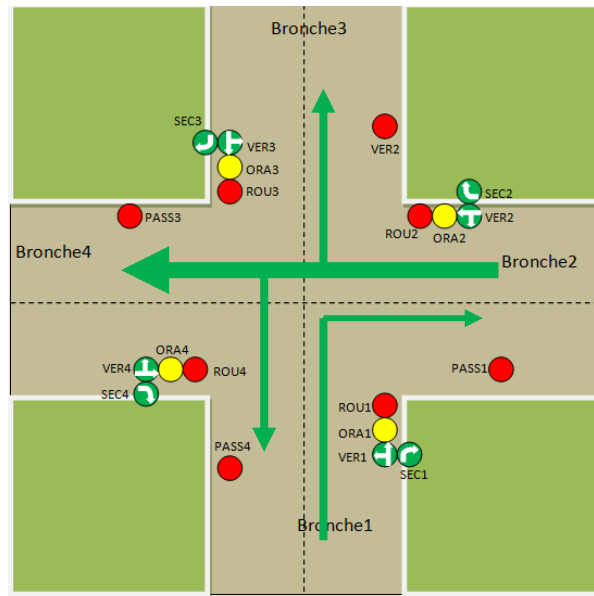


figure 26 : état4 de l'intersection

Cet état reste pendant une période de temps TVER2 pour retardé l'éteint du lompe VER2

Etat 5 :

- ORA2 lumineux
- ROU1 lumineux
- ROU3 lumineux
- ROU4 lumineux
- VER1 éteint
- VER2 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA3 éteint
- ORA4 éteint
- ROU2 éteint
- SEC1 éteint
- SEC2 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

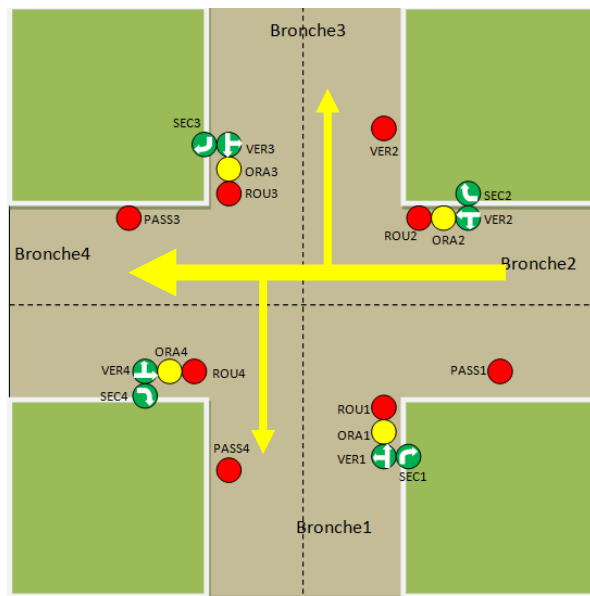


figure 27 : état5 de l'intersection

Cet état reste pendant une période de temps TORA2 pour retardé l'éteint du lompe ORA2

Etat 6 :

- VER3 lumineux
- PASS3 lumineux
- SEC2 lumineux
- ROU1 lumineux
- ROU2 lumineux
- ROU4 lumineux
- VER1 éteint
- VER2 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU3 éteint
- SEC1 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS3 éteint
- PASS4 éteint

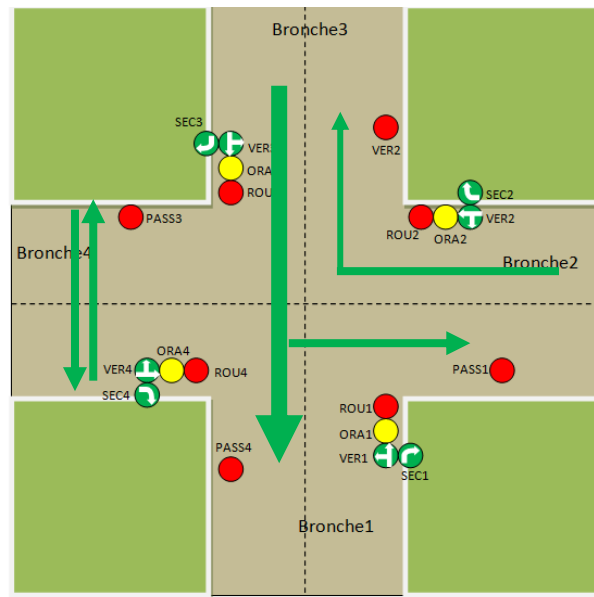


figure 28 : état 6 de l'intersection

Cet état reste pendant une période de temps TPASS3 pour retardé l'éteint du lompe PASS3

Etat 7 :

- VER3 lumineux
- SEC3 lumineux
- SEC 2 lumineux
- ROU1 lumineux
- ROU2 lumineux
- ROU4 lumineux
- VER1 éteint
- VER2 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU3 éteint
- SEC1 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

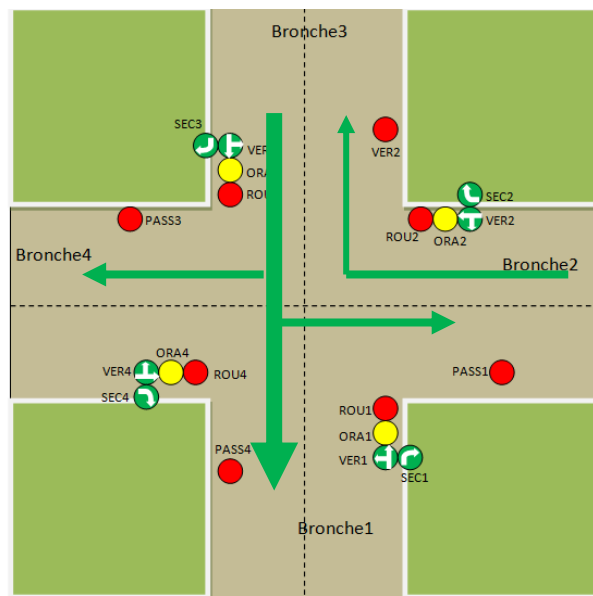


figure 29 : état 7 de l'intersection

Cet état reste pendant une période de temps TVER3 pour retardé l'éteint du lompe VER3

Etat 8 :

- ORA3 lumineux
- ROU1 lumineux
- ROU2 lumineux
- ROU4 lumineux
- VER1 éteint
- VER2 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA4 éteint
- ROU3 éteint
- SEC1 éteint
- SEC2 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

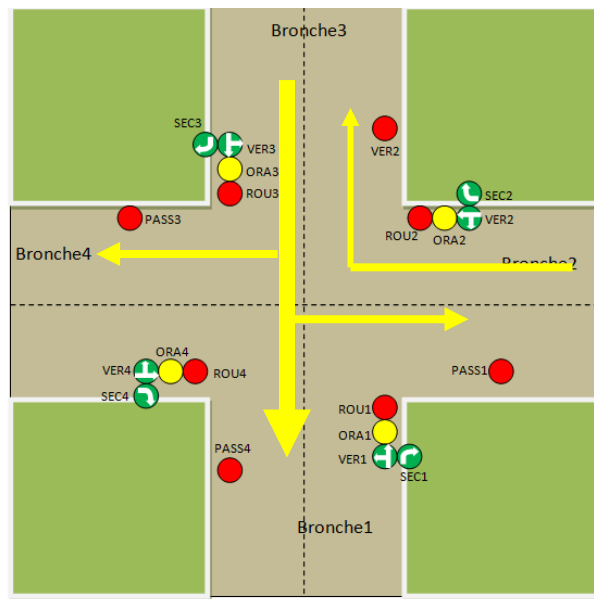


figure 30 : état 8 de l'intersection

Cet état reste pendant une période de temps TORA3 pour retardé l'éteint du lompe ORA3

Etat 9 :

- VER4 lumineux
- PASS4 lumineux
- SEC3 lumineux
- ROU1 lumineux
- ROU2 lumineux
- ROU3 lumineux
- VER1 éteint
- VER2 éteint
- VER3 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU4 éteint
- SEC1 éteint
- SEC2 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint

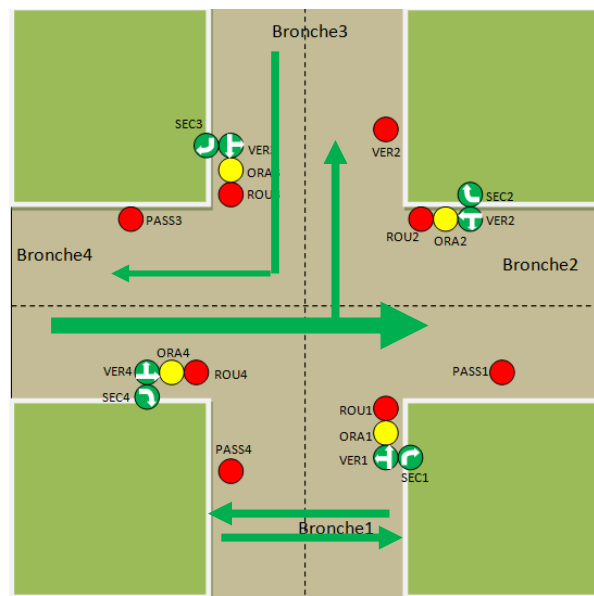


figure 31 : état 9 de l'intersection

Cet état reste pendant une période de temps TPASS4 pour retardé l'éteint du lompe PASS4

Etat 10 :

- VER4 lumineux
- SEC3 lumineux
- SEC 4 lumineux
- ROU1 lumineux
- ROU2 lumineux
- ROU3 lumineux
- VER1 éteint
- VER2 éteint
- VER3 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU4 éteint
- SEC1 éteint
- SEC2 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

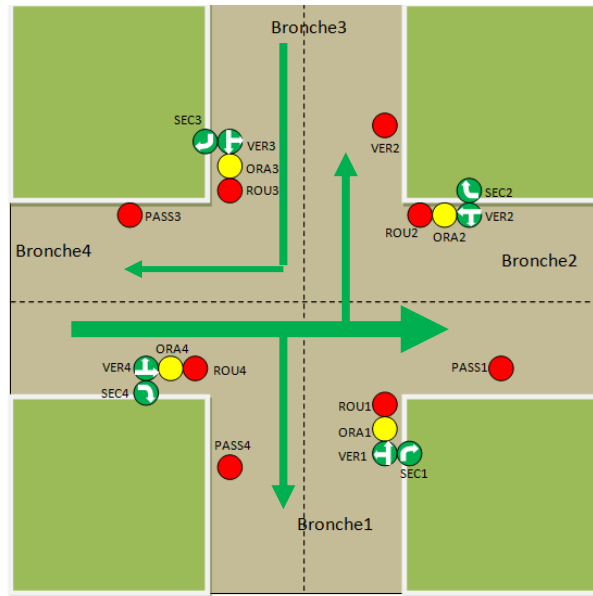


figure 32: état 10 de l'intersection

Cet état reste pendant une période de temps TVER4 pour retardé l'éteint du lompe VER4

Etat 11 :

- ORA4 lumineux
- ROU1 lumineux
- ROU2 lumineux
- ROU3 lumineux
- VER1 éteint
- VER2 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ROU4 éteint
- SEC1 éteint
- SEC2 éteint
- SEC3 éteint
- SEC4 éteint
- PASS1 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

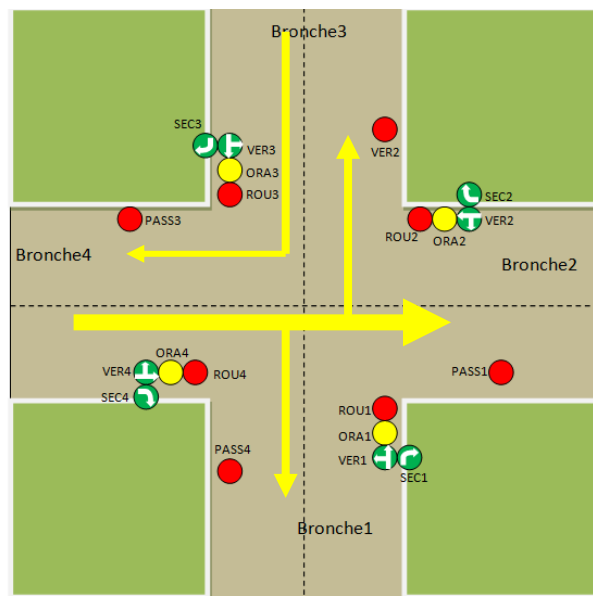


figure 33 : état 11 de l'intersection

Cet état reste pendant une période de temps TORA4 pour retardé l'éteint du lompe ORA4

Etat 12 :

- VER1 lumineux
- PASS1 lumineux
- SEC4 lumineux
- ROU2 lumineux
- ROU3 lumineux
- ROU4 lumineux
- VER2 éteint
- VER3 éteint
- VER4 éteint
- ORA1 éteint
- ORA2 éteint
- ORA3 éteint
- ORA4 éteint
- ROU1 éteint
- SEC1 éteint
- SEC2 éteint
- SEC3 éteint
- PASS2 éteint
- PASS3 éteint
- PASS4 éteint

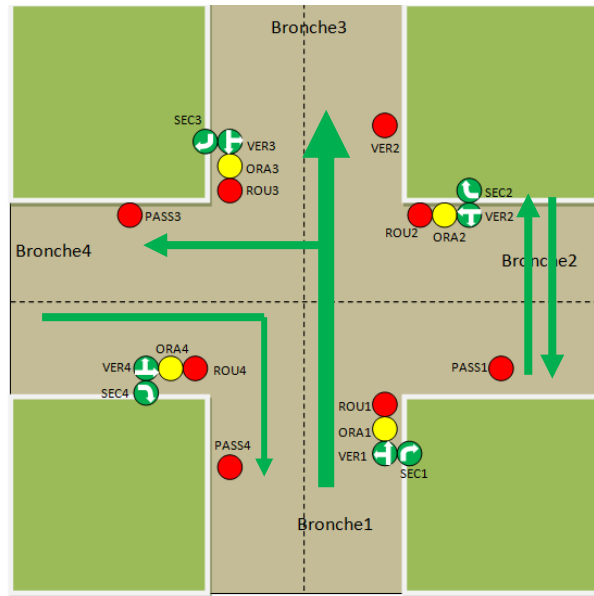


figure 34 : état 12 de l'intersection

Cet état reste pendant une période de temps TPASS1 pour retarder l'éteint du lompe PASS1

Ces états produire avec manière répétitive et successive :

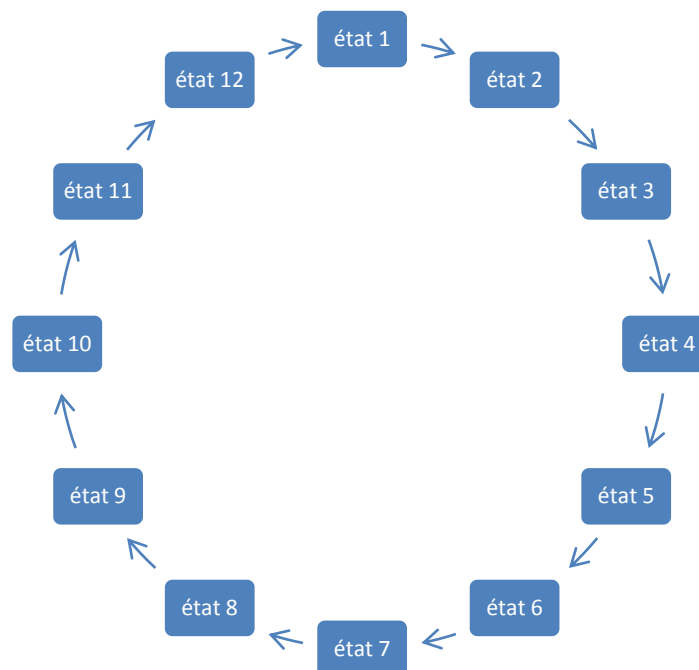


Figure 35 : cycle des états

4. Géré le fonctionnement de feux de carrefour par réseaux de PETRI :

Ver1 : lompe vert de bronche 1.

Rou1 : lompe rouge bronche 1.

Pass1 : passage piéton bronche1

Ver2 : lompe vert de bronche 2.

Rou2: lompe rouge bronche 2.

Pass2 : passage piéton bronche2

Ver3 : lompe vert de bronche 3.

Rou3 : lompe rouge bronche 3.

Pass3 : passage piéton bronche3

Pass4 : passage piéton bronche4

BM : bouton marche

Sec2 : lompe vert secondaire bronche2

Sec3 : lompe vert secondaire bronche3

Sec4 : lompe vert secondaire bronche4

La figure ci-dessous représente une solution par réseaux de PETRI :

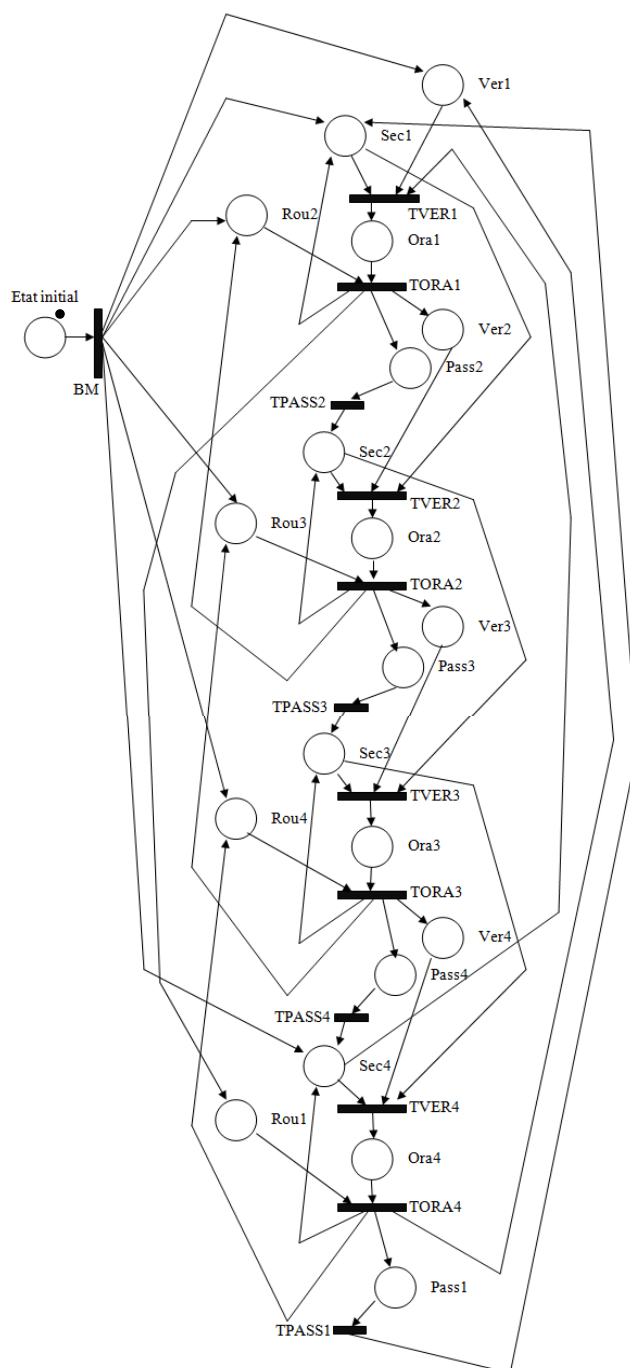
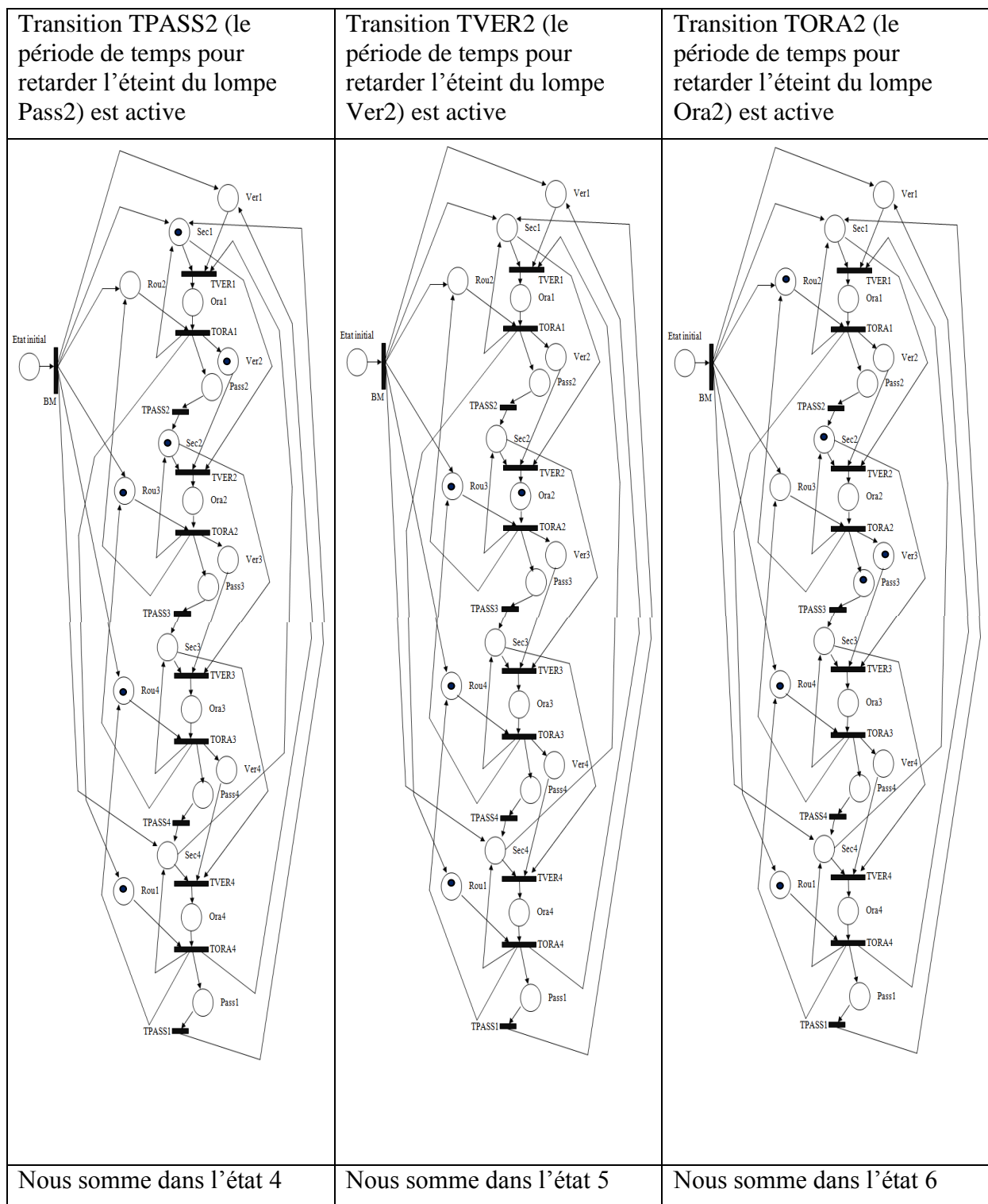


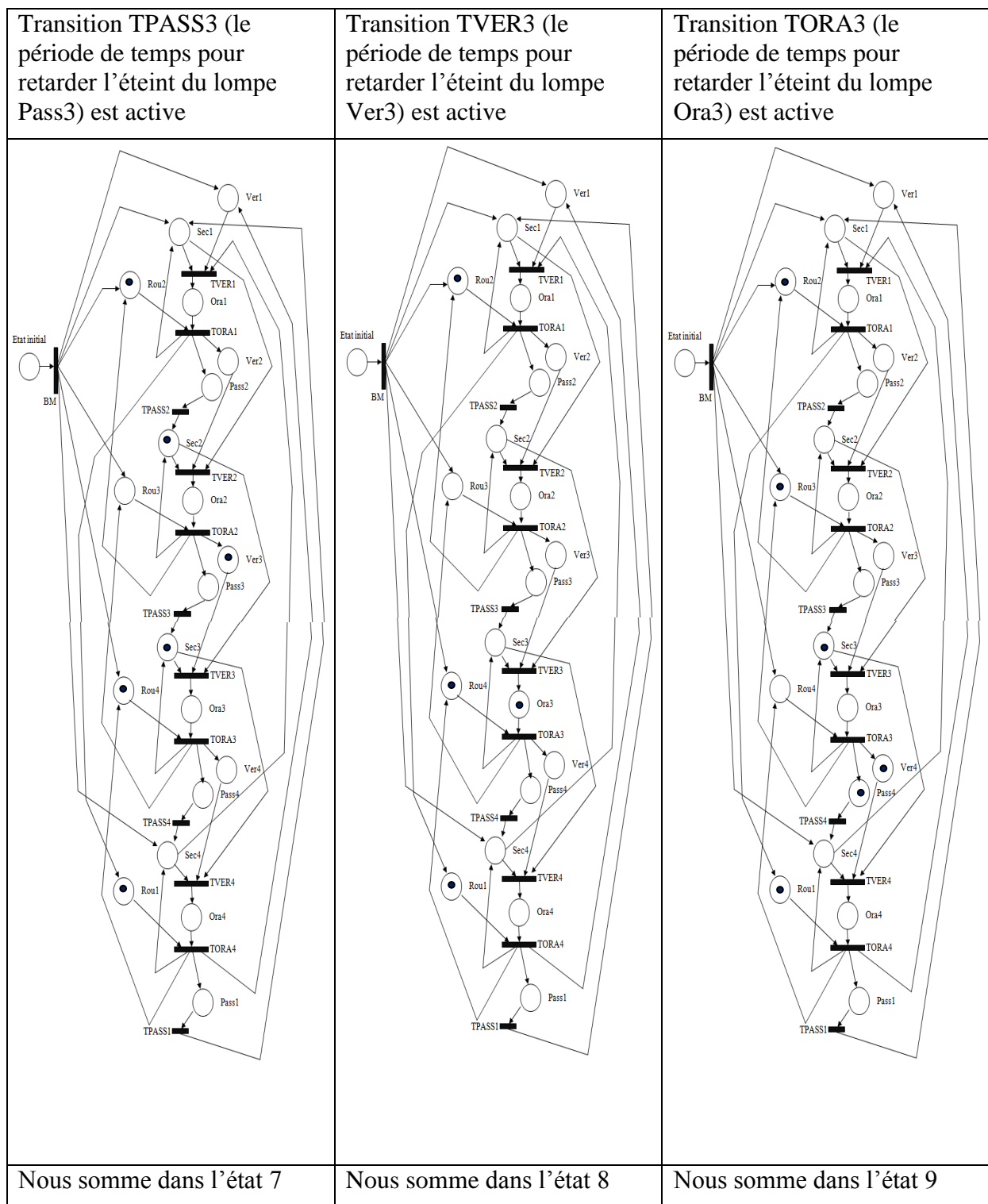
Figure 36 : RDP de carrefour

5. Franchissement graphique de réseaux :

Les figures dans les tableaux suivant représentent le franchissement de chaque transition et les mouvements des jetons

Transition BM (bouton marche) est active	Transition TVER1 (le période de temps pour retarder l'éteint du lompe Ver1) est active	Transition TORA1 (le période de temps pour retarder l'éteint du lompe Ora1) est active
cette transition prendre à l'état1	Nous sommes dans l'état 2	Nous sommes dans l'état 3





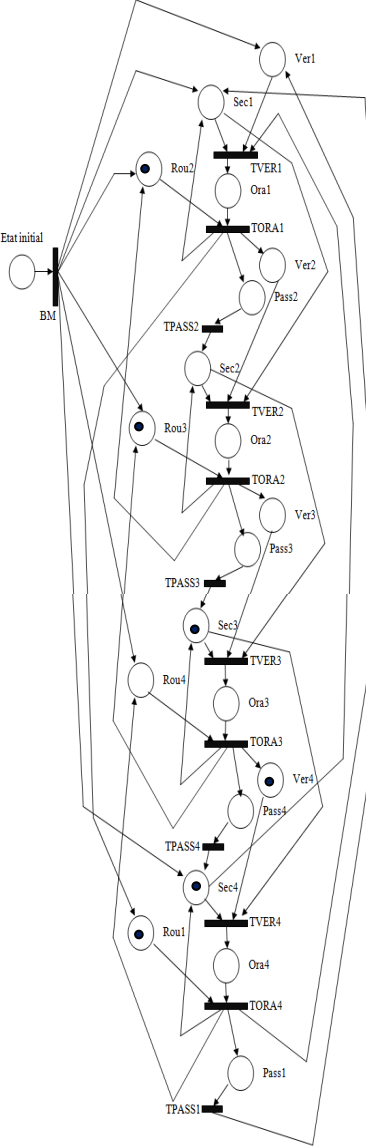
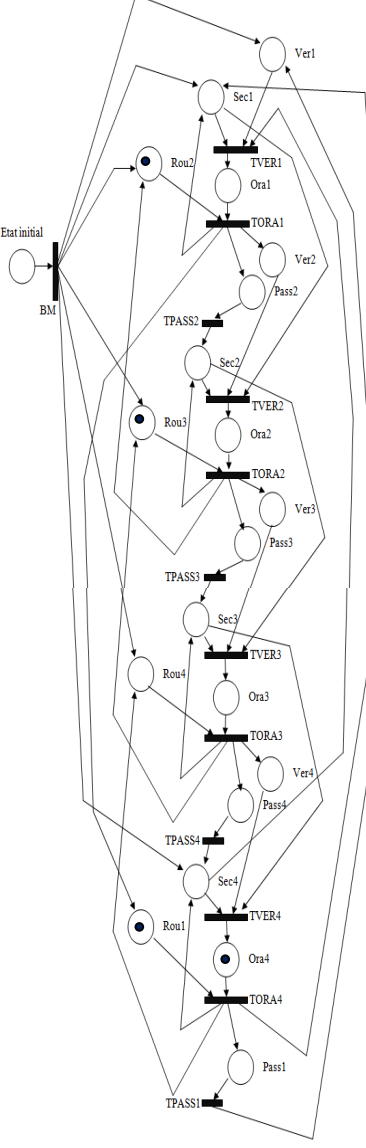
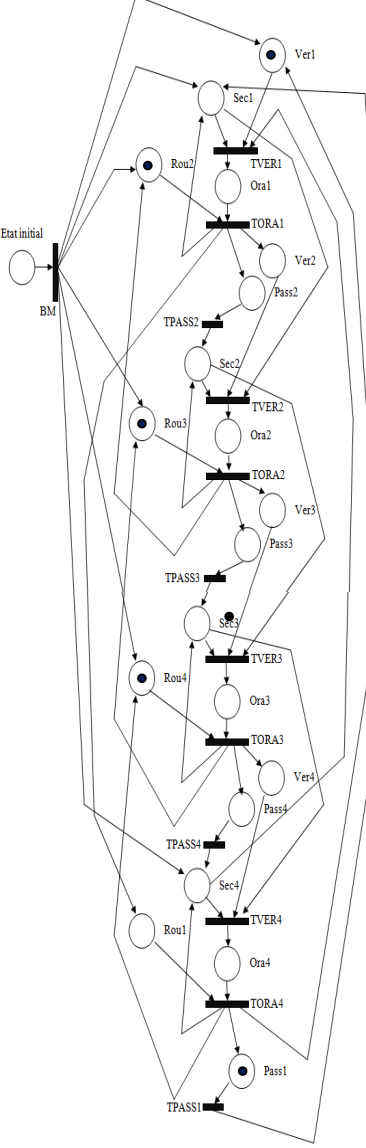
Transition TPASS4 (le période de temps pour retarder l'êteint du lompe Pass4) est active	Transition TVER4 (le période de temps pour retarder l'êteint du lompe Ver4) est active	Transition TORA4 (le période de temps pour retarder l'êteint du lompe Ora4) est active
		
Nous somme dans l'état 10	Nous somme dans l'état 11	Nous somme dans l'état 12

Tableau 17: franchissement graphique de RdP

Transition TPASS1 (le période de temps pour retarder l'êteint du lompe Pass1) est active :

Nous somme retourne l'état 1

6. Système avec mode nuit (les lompes orange clignoté) et bouton d'arrête :

La figure suivent présent le système avec le mode nuit et on a ajoute aussi un bouton d'arrête

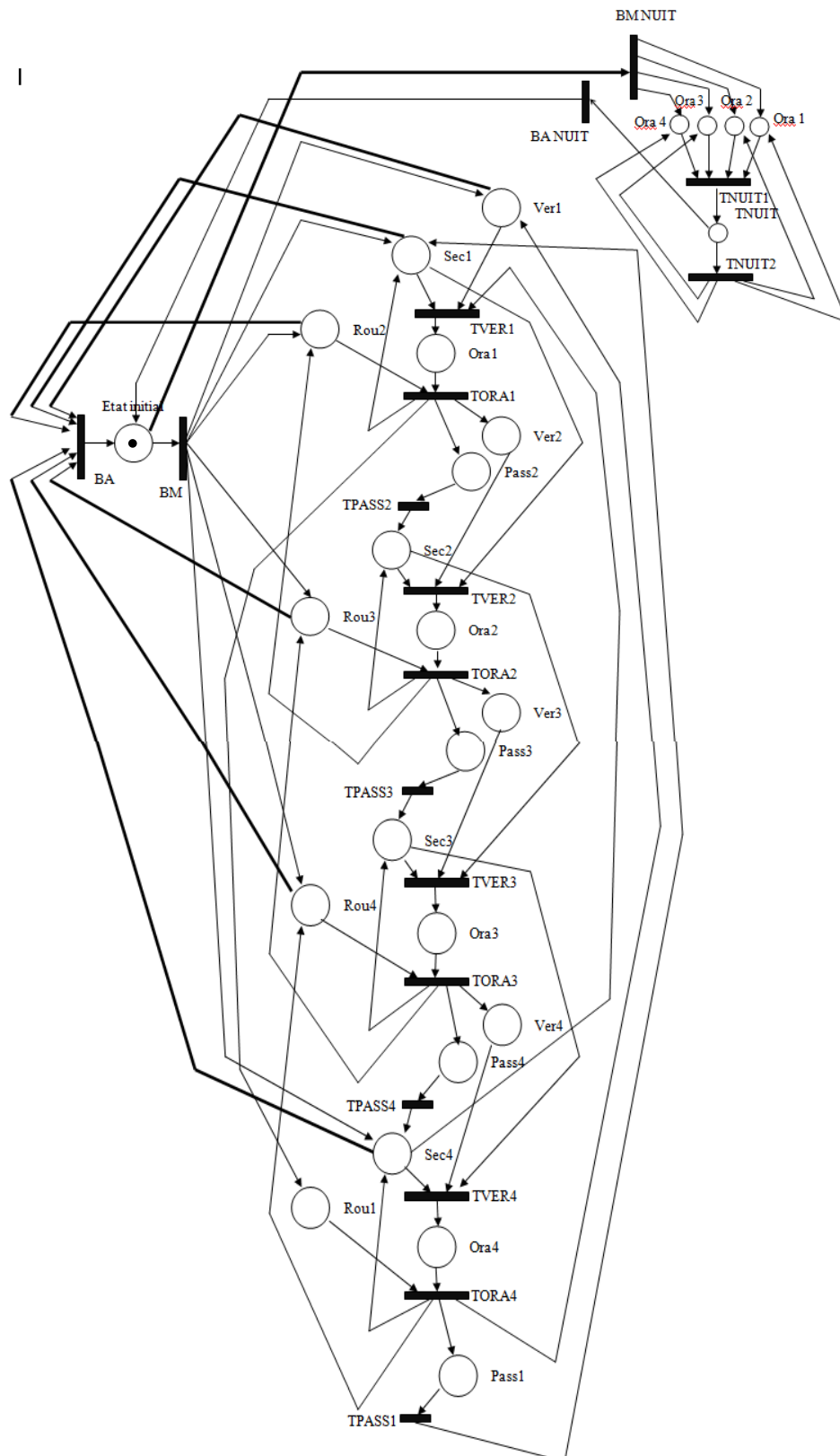


Figure 36 : RDP de carrefour avec le mode nuit et bouton d'arrêt

7. Traduction réseaux de PETRI ver langage CON de step7:

L'exemple suivant montre comment faire la traduction

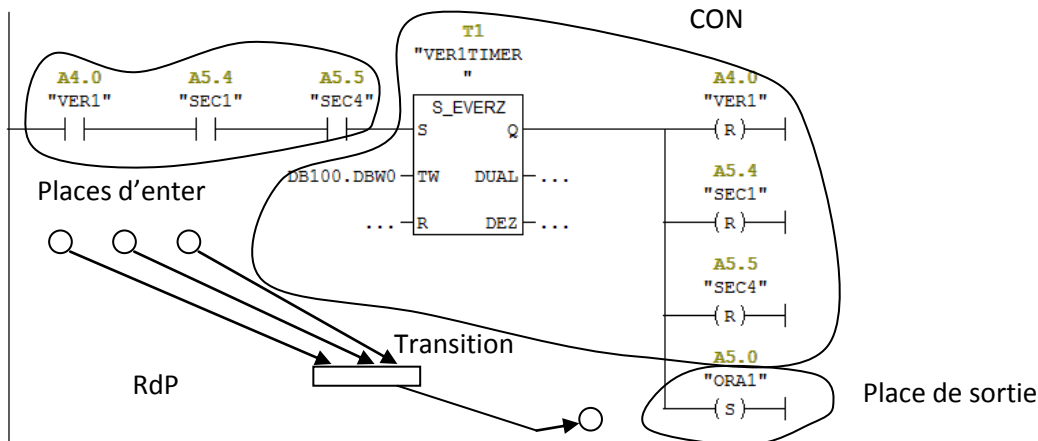


Figure 37 : exemple de traduction l'RDP de carrefour vers langage CON

Pour faire la traduction on a basé sur les notions suivantes :

- Le temporisateur T1(VER1TIMER) et la mise à zéro de VER1 (A4.0) SEC1 (A5.4) SEC4 (A5.5) considérés comme une transition
- Les contacts à fermetures VER1 SEC1 et SEC4 considérés comme des places d'entrée
- La mise à 1 d'ORA1 (A5.0) considérée comme une place de sortie
- Si un contact est actif donc il y a un jeton

Explication :

Si les contacts à fermetures VER1 SEC1 et SEC4 (places d'entrées) est active (il y a des jetons), le temporisateur T1 commence à compter, puis il désactive les contacts à fermetures VER1 SEC1 et SEC4 (places d'entrées) par la mise à zéro (enlève les jetons) et active la mise à 1 (ajoute les jetons) d'ORA1 (place de sortie)

8. L'interface homme machine de carrefour :

La figure suivante représente l'interface homme machine pour manipuler l'intersection



Figure 39 l'interface homme machine pour manipuler l'intersection

Les bouton de l'interface :

- Main : main vue
- Intersection : donne l'intersection de carrefour
- Start : comme bouton marche de système
- Stop : comme bouton arrête de système
- Night : mode nuit de carrefour
- Rest : reste système

9. Conclusion :

Dans ce chapitre, nous avons expliqué les détails de notre travail à savoir : l'étude de fonctionnement de carrefour et faire un solution par RDP en suit la traduction vers langage CON de step7 et à la fin un interface HMI par logiciel WIN cc pour manipuler l'intersection et modifié les temporisateurs de système.

Conclusion générale :

Notre travail consiste à gérer un carrefour à quatre branches par un automate SIEMENS, on utilise le réseau de PETRI et le langage CON de step7 et manipuler l'intersection de ce carrefour par un PC on utilise l'interface homme machine de Win CC flexible.

Nous avons commencé ce travail par donner une définition et un tour sur les automates programmable industriel afin de comprendre leur fonctionnement et leur rôle.

Ensuite nous avons fait un passage sur l'automate SIEMENS leur CPU et leur caractéristique des modules. Nous avons choisi l'automate qui contient le CPU 314 et les modules d'E/S SM321 DI16X24VCC SM322 DO16X24VCC0.5A. Cet automate capable de commander le carrefour.

Le programme que nous avons développé durant ce travail a montré la puissance de l'automate en terme rapidité de traitement et en terme simplicité à programmer par logiciel STEP7.

Ce travail et en développement continu, on compte prochainement gérer un carrefour plus compliqué qui possède plusieurs voies dans les branches et on compte aussi pour l'interface homme machine on peut utiliser un écran supervision tactile à la place de PC.

Nous souhaitons que ce modeste travail soit agréé comme étant une plateforme référence de au niveau du laboratoire d'électronique de M'sila, et qu'il survive pour les futurs éventuels projets de fin d'études autour des systèmes automatisés.

BIBLIOGRAPHIE

- [1] Automates programmables Industriels Techniques de l'Ingénieur
- [2] SIMATIC S7-300 CPU 31xC et CPU 31x : Caractéristiques techniques
- [3] SIMATIC S7-300 Système d'automatisation S7-300 Caractéristiques des modules
- [4] SIMATIC Programmer avec STEP 7
- [5] SIMATIC STEP 7 V5.1 Getting Started
- [6] SIMATIC HMI WinCC flexible 2008 Compact / Standard / Advanced
- [7] Web site Wikipedia.ORG