RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
OPTION AUTOMATIQUE

Année Universitaire
2004 - 2005

L'AUTOMATISATION D'UNE CHAÎNE DE PRODUCTION INDUSTRIELLE PAR AUTOMATE PROGRAMMABLE INDUSTRIEL

Dirigé par :
Mr. B HEMICI
Mr. M BELGACEM

Projet de Fin d'Études Présenté par :

Mr. BOURZAK Abdelmoumen

En Vue de l'Obtention du Diplôme
d'Ingénieur d'État en AUTOMATIQUE

REMERCIEMENTS

Au terme de ce projet de fin d'études, je tiens à formuler mes chaleureux remerciements à Mr. B. HEMICI Professeur à l'ENP, mon encadreur, qui a dirigé mon travail avec dévouement jusqu'à son achèvement.

Je remercie également Mr. M. BELGACEM Chef département de qualité à l'unité ORSIM groupe BCR sise à Oued-Rhiou, mon co-encadreur, pour ces directives, ses conseils combien utiles, sa très grande disponibilité et ses qualités humaines.

Je tiens à remercier aussi, l'ensemble du personnel de l'unité ORSIM, ingénieurs, électriciens, mécaniciens, chimistes, pour leurs aides, directives, conseils et surtout leurs gentillesses.

J'adresse mes vifs remerciements au président du jury ainsi qu'aux examinateurs qui ont accepté de juger ce travail.

Je tiens aussi à exprimer ma profonde gratitude à l'ensemble de nos enseignants d'automatique pour leurs efforts déployés à mon égard tout au long de mon parcours universitaire.

Enfin, je remercie le personnel de la bibliothèque de l'école nationale polytechnique en particulier NAOUEL, FAIZA, MADAME, HAFIDA et tous ceux qui ont contribué de prés ou de loin à l'élaboration de ce travail.

DEDICACE

Je dédie ce modeste travail,

À Ceux à qui mon cœur depuis sa naissance n'a pas pu éprouver qu'amour et reconnaissance, à ceux qui ont donné un sens à mon existence en m'offrant une éducation digne de confiance
À mes chères Parents.

A tout les membres de ma famille ; mon frère, mes sœurs, mon beau frère et son frère

A la famille BOURZAK toute entière,

À tous mes amis d'enfance,

A toute la promo d'automatique 2003 – 2004 et 2004 – 2005,

A la clic de Setif, Galema, Kharrata,

A ma clic de l'Ouest,

À mes amis de la cité universitaire BOURAOUI Amar,

Et à tous qui m'ont connu et aidé de prés ou de loin dans la réalisation de ce travail.

Et un grand merci pour « the Amenofils », et sa main magique

ملخص

يكمن هدف هذا العمل في بحث نهاية الدراسة لالتشغيل الألي لسلسلة إضافة الكروم المزين لوحدة الإنتاج ORSIM التابعة للمجمع BCR الواقعة بمنطقة وادارهيو ولاية غيليزان ، و ذلك باستعمال المشغل الآلي الصناعي المبرمج للمنتج SIEMENS و بالأخص النوع S7-314 IFM و محيطه البرمجي STEP7 من أجل تحديث تقنية التحكم المستعملة حاليا و كذا تطوير إمكانيات الإنتاج من حيث الجودة و الكمية.

الكلمات المفتاحية : المشغل الآلي الصناعي المبرمج ،التحكم ،الكروم المزين ، المحيط البرمجي STEP7

Résumé

Ce projet de fin d'étude se porte sur l'automatisation de la chaîne du chromage décoratif de l'unité de production ORSIM du groupe BCR sise à Oued-Rhiou W. Relizane, en utilisant les automates programmables industriels (API) de la firme SIEMENS et particulièrement le S7-314 IFM et son environnement de programmation STEP7, dans le but d'améliorer la technique de commande utilisée ainsi les performances de la chaîne

Mots clés : Automate programmable, API, automatisation, processus industriel, chaîne de production, chromage décoratif, commande, STEP7, GRAFCET, S7-314 IFM

Abstract

This project of end of studies has as object the automation of the chain of the decorative chromium plating of the manufacturing unit ORSIM from the group BCR located on Ouedi-Rhiou W Relizane, by using the Programmable Logic Controller (PLC) of the SIEMENS firm and particularly S7-314 IFM and its programming environment STEP7, in order to improve the used control technique thus the performances of the chain.

Key words : Programmable logic controller, PLC, automation, industrial process, decorative chromium, control.

SOMMAIRE

CHAPITRE -1-

DESCRIPTION DE LA CHAINE DU CHROMAGE DECORATIF

CHAPITRE -2-

LES AUTOMATES PROGRAMMABLES INDUSTRIELS ET ENVIRONNEMENT DE PROGRAMMATION STEP7

CHAPITRE -3-

LA CPU S7-314 IFM
ET SES FONCTIONS INTEGREES

CHAPITRE -4-

COMMANDE PAR API DU PROCESSUS DE CHROMAGE

INTRODUCTION GENERALE

L'évolution rapide des technologies nouvelles a permis de contourner la plupart des difficultés rencontrées dans le monde industriel, et a fourni plusieurs possibilités pour satisfaire les exigences et les critères demandés tels que la productivité, la sécurité, l'optimisation des coûts de production et l'amélioration des conditions de travail.

L'automatisation des procédés industriels est actuellement l'un des axes où on fait appel, de plus en plus, aux technologies évoluées à mesure que les exigences du monde industriel ont aussi évolué. Parmi celles-ci, figurent les Automates Programmables Industriels (API), qui offrent la solution adaptée aux besoins exigés.

L'automate programmable industriel est l'organe principal de la boucle de réglage placée dans un procédé industriel, en vue de le contrôler. Il a pour tâche principale, la récolte des informations relatives à l'état du système, à partir des différents capteurs via ses interfaces d'entrées, et les traiter pour prendre une décision ; et ainsi commander les actionneurs via ses interfaces de sorties suivant une logique de fonctionnement mise en évidence, par un programme inscrit dans la mémoire.

Le but de ce travail est l'élaboration d'une solution à base d'API pour automatiser une chaîne de production industrielle traditionnelle d'une unité de production de robinetterie ORSIM du groupe BCR (Boulonnerie, Coutellerie, Robinetterie) sise à Oued-Rhiou à 50km du chef lieu de Relizane.

Le présent travail s'articule autour de quatre parties. La première partie présente une description du processus industriel effectué, les différents constituants de la chaîne, les différentes phases de production et la méthode de contrôle utilisée.

Dans la deuxième et la troisième partie, on présentera une étude générale sur les automates programmables industriels ainsi que l'API choisi pour effectuer l'automatisation. On présentera aussi, l'interface de programmation STEP7 ainsi que les avantages apportés.

Quant à la quatrième et dernière partie, elle sera consacrée à l'élaboration du programme de commande de la chaîne de chromage avec tous les détails utiles.

Enfin, on terminera l'étude par une conclusion générale qui discutera les avantages apportés et les perspectives visées en terme de réalisation et installation.

CHAPITRE -1-

DESCRIPTION DE LA CHAINE DU CHROMAGE DECORATIF

INTRODUCTION

L'unité de chromage que nous avons à automatisée, paraît si complexe que nous ne puissions pas la décrire ici en détail. Néanmoins nous jugeons très nécessaire de décrire le processus de chromage comme étant l'élément de base, et cela sans oublier l'appareillage utilisé et les opérations et étapes que subit le produit brut avant qu'il soit une pièce finie.

1. DESCRIPTION DU PROCESSUS

1.1. Le chromage :

Le chromage est un procédé de revêtement par voie électrolytique sur une surface d'un métal moins résistant et oxydable, généralement un acier en fer par une couche fine et adhérente en chrome métal, dans le but de le protéger contre la corrosion et de lui procurer un aspect esthétique décoratif remarquable par sa brillance métallique éclatante, ainsi que par sa forte résistance à l'usure et à l'abrasion. De telles propriétés confèrent au chrome une grande importance industrielle et le rendent performant.

Conditions opératoires et paramètres de l'électrolyte :

Acide Chromique 180 à 300 g/l
Sulfates 0.8 à 1.1 g/l
Rapport SO_4^{-2}/CrO_3 0.4 à 0.5 %
Température 36° à 46°
Densité de courant cathodique 5 à 25 A/dm²

1.2. Le Nickelage :

Les dépôts de nickel s'effectuent à partir de solutions de sels simples : sulfate, chlorure, fluo borate, … Ces dépôts peuvent être mats ou brillants, durs ou tendres, ductiles ou tendus..

Actuellement le nickel brillant trouve son utilisation dans plusieurs domaines tel que l'industrie automobile, électrotechnique, l'industrie des appareils ménagers et la fabrication des appareillages divers et cela par raison d'assurer aux pièces traitées une bonne brillance en vue du chromage consécutif et en outre de les adapter aux exigences de corrosion.

Conditions Opératoires :

Température 35° à 65° optimum 50°
Densité de courant cathodique 2 à 10 A/dm² optimum 4 A/dm²
Rapport surfacique anode/cathode 1/1 à 2/1
Agitation
- air surprimé 20 à 30 $m^3.n^{-1}.m^{-2}$
- mécanique 2 à 6 m/min

Filtration continue
Tension superficielle 30 à 40 mN/m

1.3. Opérations complémentaires :

Préalablement à un dépôt métallique de chrome, il est important de bien débarrasser la surface à traiter de souillures et de tout corps étranger tel que graisses et oxydes, afin de permettre au dépôt final une bonne adhérence, une uniformité et une dureté remarquable, c'est pour sa qu'on remarque que la chaîne considérée contient plusieurs bains de traitement de surface, tel que le rinçage, le décapage, le dégraissage, ...

1.4. Processus de chromage décoratif :

Opération	Concentration (g/l)	Temps de maintien (min)	Température (°C)	Ampérage (A/dm²)	Remarque
Dégraissage chimique	50	3 – 15	40 – 80		
Dégraissage ultrason	25 – 50	5 – 10	50 – 80		
Rinçage de récupération					
Rinçage à froid					
Dégraissage électrolytique	20 – 50	1 – 3	25 – 40	3 – 4	
Rinçage de récupération					
Rinçage a froid					
Dépassivation	50 – 100		Ambiante		Par Acide
Rinçage à froid					
Activation	10 %		Ambiante		Hcl
Nickelage	Ni^{2+} : 85-90 $NiCl_2$: 70-80 H_3BO_3 : 50-60 $NiSO_4$: 300		55	2 – 10	
Rinçage de récupération					
Rinçage à froid					
Activation	70				Acide chromique à faible concentration
Chromage	Cr^{6+} : 180-220 SO_4^{2-} : 0.8-1.2 Rpt : 04-0.5%				
Rinçage de récupération					
Passivation					Réduction bisulfate de sodium
Rinçage à froid					
Rinçage à chaud					
Séchage					

Tableau 1.1. Les différentes opérations du processus de chromage.

2. CONSTITUTION DE LA CHAINE DE CHROMAGE

La chaîne de chromage installée au sain de l'unité de production de robinetterie est constituée de plusieurs éléments dont :

2.1. Les bains :

Elle comporte 30 bains métalliques répartis sur deux axes parallèles pour optimiser l'espace de travail. Ils peuvent contenir des solutions chimiques tel que les acides, les bases ...pour le traitement des pièces accrochées à la barre.

Il y a 19 barres qui circulent à la fois dans la chaîne, et cela pour optimiser le temps de traitement des pièces.

Il y a des bains qui sont équipés avec des capteurs de proximité pour détecter la présence de la bar (support portant les pièces a traitées), des capteurs de température, … (Voir Fig)

> **Remarque :**
> On a constaté que les ne sont pas dotés de capteurs de position permettant de détecter la positon des chariots pendant leurs déplacement. Donc un autre système est utilisé (comptage). Mais on remarque aussi que au dessus de chaque on a placé un boue de fer qui sert comme repère pour le bain désigné.

La disposition actuelle des 30 bains est représentée selon la figure qui suit :

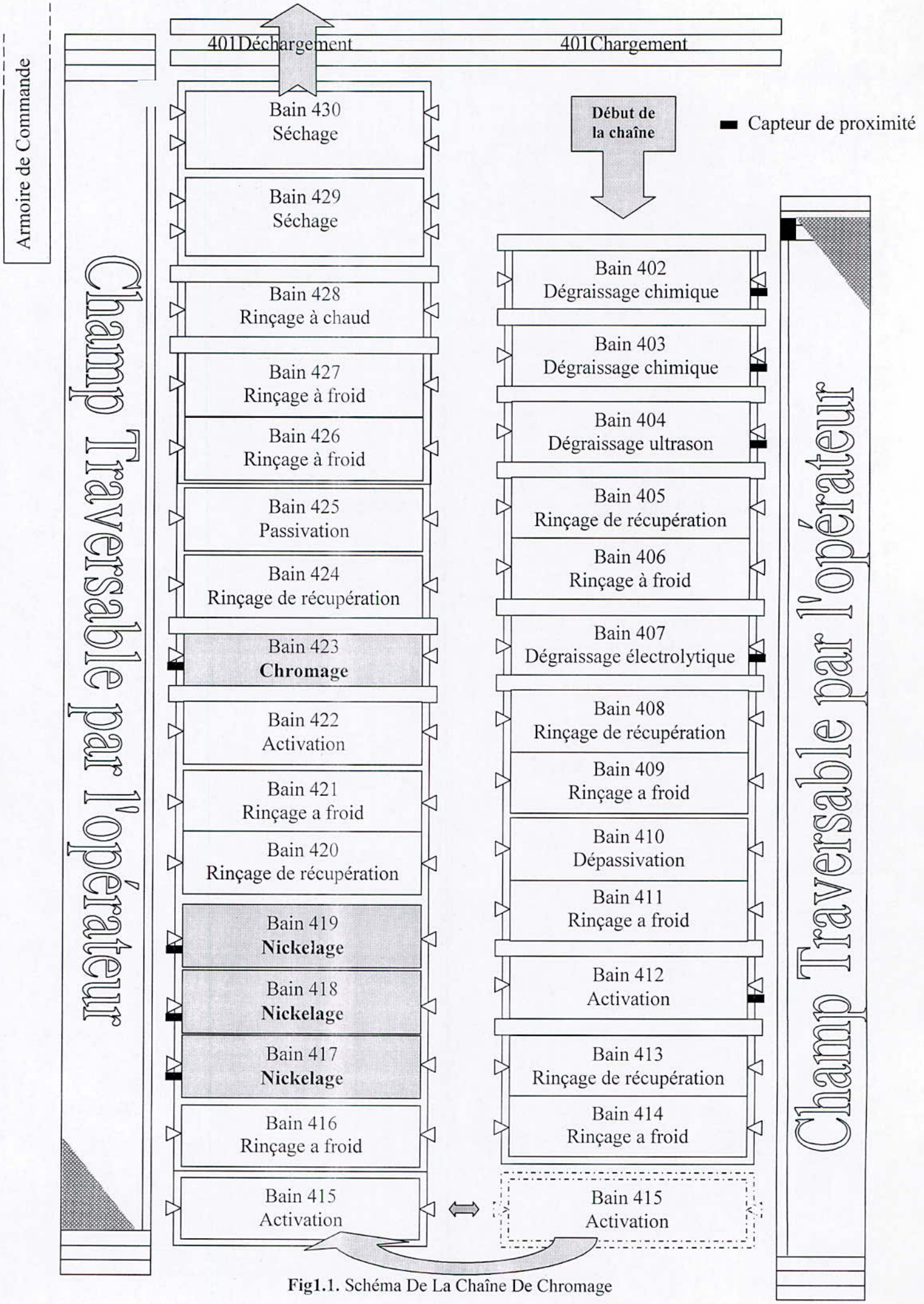


Fig1.1. Schéma De La Chaîne De Chromage

2.2. Les chariots :

La chaîne comporte six chariots, trois sur le premier axe et trois sur le deuxième, montés sur des rails juste au dessus des bains.

Chaque chariot est doté de plusieurs de capteurs, boutons, …. Mentionnés dans le tableau suivant :

Désignation	Type	Rôle
Xb0	Détecteur de proximité	Détection de vitesse lente (avance)
Xb1	Détecteur de proximité	Détection de vitesse lente (recule)
Xb2	Détecteur de proximité	Fin de course (pour mouvement transversal)
Xb3a	Bouton	Commande manuelle pour le levage
Xb3b	Bouton	Commande manuelle pour l'abaissement
Xb3c	Bouton	Commande manuelle pour l'avance
Xb3d	Bouton	Commande manuelle pour le recule
Xb4	Bouton poussoir	Déclenchement de secours
Xb7	Détecteur de proximité	Détection de l'état initiale
Xb8	Détecteur de proximité	Détection de la présence de la barre
Xb9	Came	Limite de levage
Xb10	Came	Limite d'abaissement
Xb11	Came	Détection de la vitesse lente (abaissement)
Xb12	Came	Détection de la vitesse lente (levage)
Xb17	Détecteur de proximité	Stationnement

Tableau 1.2. Ensemble de capteurs et boutons d'un chariot

(X) désigne le numéro du chariot.
Ex : pour le chariot 1 c'est on désigne : 1b0, 1b1, …

Afin d'expliquer l'emplacement des capteurs et des boutons Sur chaque chariot, on va les présentés dans la figure suivante :

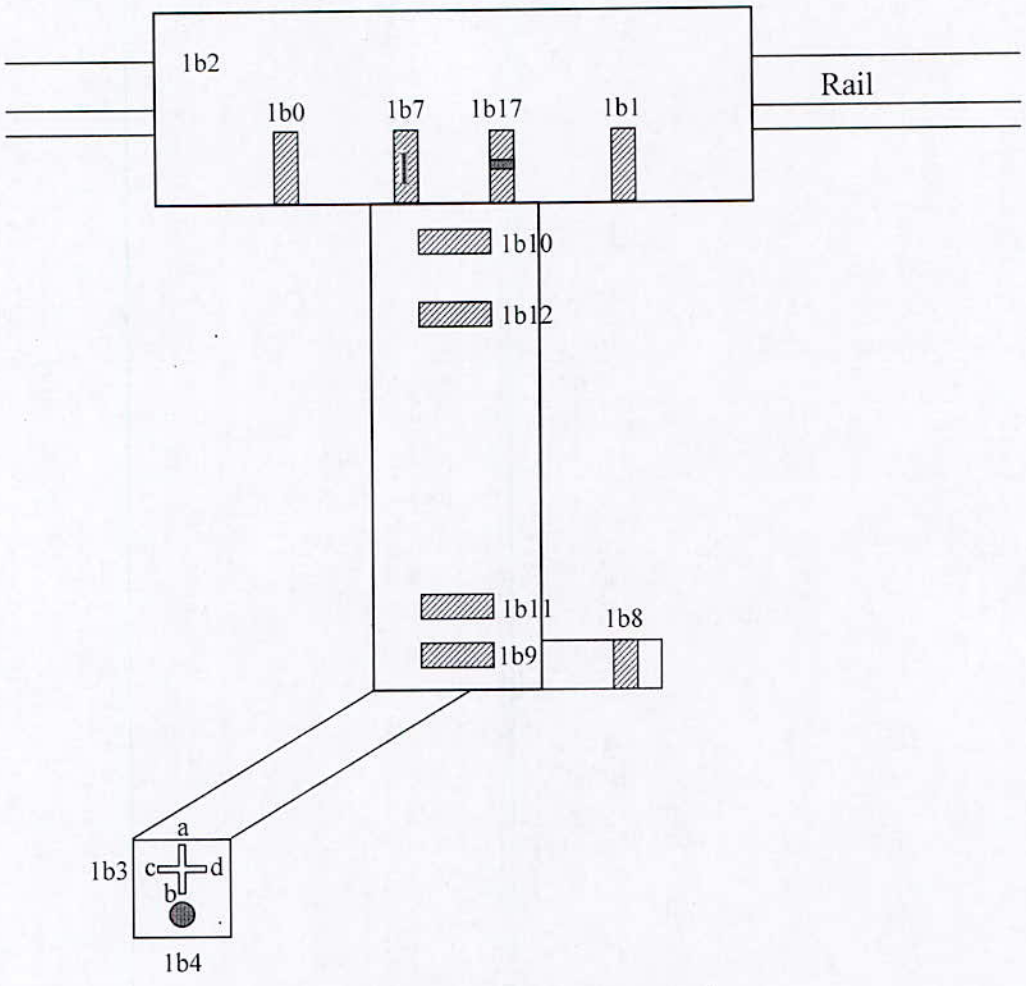


Fig1.2. Vue de face d'un chariot

Chaque chariot est entraîné par deux moteurs de puissance de 20 watts (l'un pour le mouvement transversal et le deuxième pour le mouvement vertical), ces moteurs sont triphasés asynchrone qui peuvent tourner dans deux sens (à gauche et droite pour le moteur transversal, en haut et en bas pour le moteur vertical), et avoir deux vitesse (vitesse lente et vitesse rapide).

Chacun des deux moteurs est dotés aussi d'un frein mécanique alimenté par un courant continue venant d'un redresseur. Ils offrent ainsi au chariot un cycle de déplacement qu'on peu le résumer comme suit :

- Départ avec une vitesse lente (déplacement lent)
- Démarrage de la vitesse rapide (déplacement rapide)
- redémarrage de la vitesse lente et arrêt de la vitesse rapide (déplacement lent)
- Arrêt du chariot (stationnement)

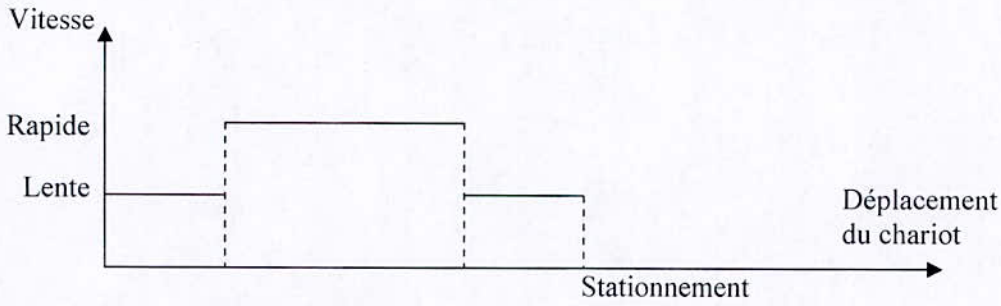


Fig1.3. Cycle de déplacement du chariot

Schéma des moteurs :

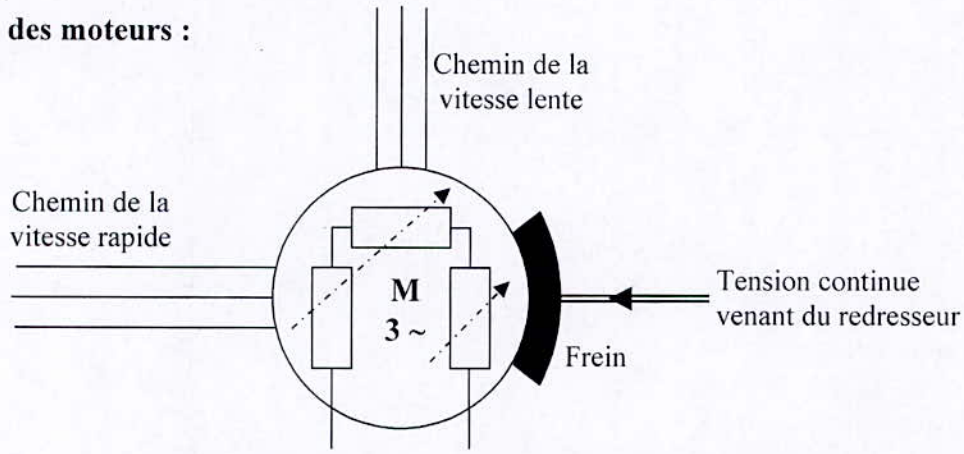


Fig1.4. L'un des deux moteurs installés sur chaque chariot

2.3. Chariots transversaux :

2.3.1. Le bain 415 (bain de rinçage) :

Il se déplace entre les deux axes de la chaîne, permettant ainsi le déplacement des barres du premier axe au deuxième. On remarque aussi qu'il est doté de deux détecteurs de proximité (Un pour la détection de la barre et l'autre pour indiquer la fin de course du bain).

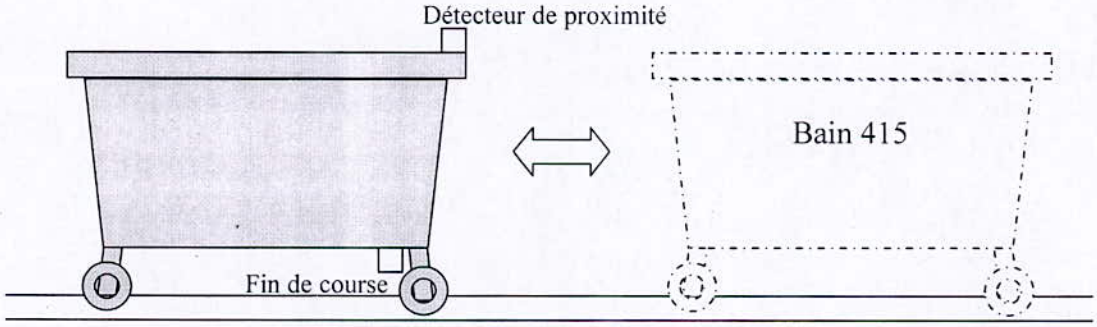


Fig1.5. Le déplacement du bain 415

2.3.2. Chariots de chargement :

Il existe deux chariots de chargement, situés au début de la chaîne et qui travaillent en alternance.

2.4. Armoire de commande :

Elle contient :

La commande des six chariots :

A base de tapis, chaque tapi porte des papillons et roule avec un régime constant autour d'une succession de cames, où la configuration des papillons détermine une séquence d'étapes a suivre par le chariot à commander.

- Bouton de marche/arrêt
- Bouton de marche manuel/automatique de la chaîne
- Bouton de marche manuel/automatique de chaque chariot

Les relais :

Des relais de 24 volts assurant le lien entre la commande et la puissance

Les alimentations :

- De puissance (380 volts)
- De la commande et des relais (24 volts)

Les mesures de sécurité :

- Bouton d'arrêt de secours principal
- Bouton d'arrêt de secours de chaque chariot, …

Des LEDs d'indication :

- Marche/arrêt de la chaîne
- Fonctionnement manuel/automatique de la chaîne
- Marche/arrêt des redresseurs, …

CONCLUSION

Dans ce chapitre nous avons décrit la situation actuelle de la chaîne de chromage installée au sain de la BCR, ce qui nous a permis de localiser les principaux défauts, causés, en grande partie, par la technologie de contrôle adoptée.

Nous avons au cours de ce chapitre cité les différentes composantes du matériel utilisé pour l'opération de chromage (chaîne de chromage), ainsi que celles du processus correspondant. Tout cela en vu de modéliser le système pour pouvoir ensuite concevoir un système de commande moderne du processus, basé essentiellement sur l'entité ***Automate Programmable Industriel***, pour lequel nous allons consacrer tout le chapitre suivant.

CHAPITRE -2-

LES AUTOMATES PROGRAMMABLES INDUSTRIELS ET ENVIRONNEMENT DE PROGRAMMATION STEP7

INTRODUCTION

L'industrie moderne que, l'on peut qualifier d'industrie de qualité et de quantité, ne cesse d'exiger un matériel de contrôle de plus en plus performant afin de réaliser les deux objectifs, simultanément. Et c'est pour cette raison qu'on voulait remplacer les dispositifs de commande classiques avec tous les inconvénients qui en découlent (logique câblée très compliquée, encombrement, difficulté d'entretien ...) par un autre beaucoup plus performant et avantageux.

Ce serait certainement l'Automate Programmable Industriel qui devient de nos jours le cœur de toute unité industrielle moderne.

Voici donc une description plus ou moins détaillée de l'API et de tout ce qui y est lié en terme de soft et de hard.

HISTORIQUE

Au début des années 50, les ingénieurs étaient déjà confrontés à des problèmes d'automatismes, les composants de base de l'époque étaient les relais électromagnétique à un ou plusieurs contacts. Les circuits conçus comportaient des centaines voire des milliers de relais. Le transistor n'était connu que comme un composant d'avenir et les circuits intégrés étaient inconnus.

Vers 1960, les semi-conducteurs (transistors, diodes) sont apparus dans les automatismes sous forme de circuits digitaux. Ce n'est quelques années plus tard, que l'apparition des circuits intégrés a amorcé une révolution dans la façon de concevoir les automatismes. Ceux-ci étaient très peu encombrants et leur consommation était des plus réduite. On pouvait alors concevoir des fonctions de plus en plus complexes à des coûts toujours décroissants.

C'est en 1969 que les constructeurs américains d'automobiles (General Motors en particulier) ont demandé aux firmes fournissant le matériel d'automatisme des systèmes plus évolués et plus souples pouvant être modifiés simplement sans coût exorbitants.

Les ingénieurs américains ont résolu le problème en créant un nouveau type de produit nommé automates programmables. Ils n'étaient rentables que pour des installations d'une certaine complexité, mais la situation a très vite changée, ce qui a rendu les systèmes câblés obsolètes.

De nombreux modèles d'automates sont aujourd'hui disponible ; depuis les nano automates bien adaptés aux machines et aux installations simples avec un petit nombre d'entrées/sorties, jusqu'aux automates multifonctions capables de gérer plusieurs milliers d'entrées/sorties, et destinés au pilotage de processus complexes.

2. DEFINITION GENERALE

Un Automate Programmable Industriel API dit Programmable Logic Controller PLC dans le langage industriel, est une machine électronique spécialisée dans la conduite et la surveillance en temps réel de processus industriels et tertiaires. Il exécute une suite d'instruction introduite dans ses mémoires sous forme de programme, et s'apparente par conséquent aux machines de traitement de l'information.

Trois caractéristiques fondamentales le distinguent des outils informatiques tels que les ordinateurs utilisés dans les entreprises et le tertiaire :

- il peut être directement connecté aux capteurs et pré actionneurs grâce à ses Entrées/sorties industrielles.
- il est conçu pour fonctionner dans des ambiances industrielles sévères, (Température, vibration, micro-coupures de la tension d'alimentation, Parasite, etc.).
- enfin, sa programmation à partir de langages spécialement développés pour le Traitement de fonctions d'automatisme facilite son exploitation et sa mise en oeuvre[1]

Selon la norme française EN 61131-1, un automate programmable est un :

« 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 utilisateurs aux fins de mise en oeuvre des 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. »[2]

3. ARCHITECTURE DES API

3.1. Le Processeur :

Le processeur a pour rôle principal le traitement des instructions qui Constituent le programme de fonctionnement de l'application. Mais en dehors De cette tâche de base, il réalise également d'autres fonctions :

- Gestion des entrées/sorties.
- surveillance et diagnostique de l'automate par une série de teste lancés à la mise sous tension ou cycliquement en cours de fonctionnement.
- Dialogue avec le terminal programmation aussi bien pour l'écriture et la mise au point du programme qu'en cours d'exploitation pour des réglages ou des vérifications de données.[3]

---

[1] Automates Nano et Plate-forme d'automatisme micro (104) Schneider Electrique 1999.
[2] M.Bertrand, Automates programmables industriels S 8015, Techniques de l'ingénieur.
[3] Automate Nano et Plate- forme d'automatisme Micro (104) Schneider Electric 1999.

Le processeur est organisé autour d'un certain nombre de registres, ce sont des mémoires rapides permettant la manipulation des informations qu'elles retiennent, ou leur combinaison avec des informations extérieures.

Les principaux registres existants dans un processeur sont :

3.1.1. L'accumulateur

C'est le registre où s'effectuent les opérations du jeu d'instruction, les résultas sont contenus dans ce registre spécial.

3.1.2. Le registre d'instruction :

Il reçoit l'instruction à exécuter et décode le code opération. Cette instruction est désignée par le pointeur.

3.1.3. Le registre d'adresse :

Ce registre reçoit, parallèlement au registre d'instruction, la partie opérande de l'instruction. Il désigne le chemin par lequel circulera l'information lorsque le registre d'instruction validera le sens et ordonnera le transfert.

3.1.4. Le registre d'état :

C'est un ensemble de positions binaires décrivant, à chaque instant, la situation dans laquelle se trouve précisément la machine.

3.1.5. Les piles :

Une organisation spéciale de registres constitue une pile, ces mémoires sont utilisées pour contenir le résultas de chaque instruction après exécution.

Ce résultat sera utilisé ensuite par d'autre instruction, et cela pour faire place à la nouvelle information dans l'accumulateur.

3.2. Les mémoires :

Un système à processeur est toujours accompagné d'un ou de plusieurs types de mémoires. Les automates programmables industriels possèdent pour la plupart les mémoires suivantes :

3.2.1. Mémoire de travail :

La mémoire de travail (mémoire vive) contient les parties du programme significatif pour son exécution. Le traitement du programme a lieu exclusivement dans la mémoire de travail et dans la mémoire système.

3.2.2. Mémoire système :

La mémoire système (mémoire vive) contient les éléments de mémoire que chaque CPU met à la disposition du programme utilisateur comme, par exemple, mémoire image des entrées, mémoire image des sorties, mémentos, temporisation et compteurs. La mémoire système contient, en outre, la pile des blocs et la pile des interruptions.

Elle fournit aussi la mémoire temporaire allouée au programme (pille des données locales).

3.2.3. Mémoire de chargement :

La mémoire de chargement sert à l'enregistrement du programme utilisateurs sans affectation de mnémonique ni de commentaires (ces derniers restent dans la mémoire de la console de programmation).

La mémoire de chargement peut être soit une mémoire vive (RAM), soit une mémoire EPROM.

3.2.4. Mémoire RAM non volatile :

Zone de mémoire configurable pour sauvegarder des données en cas de défaut d'alimentation.

3.2.5. Mémoire ROM :

Contient le système d'exploitation qui gère la CPU.

3.3. Les mémoires d'entrée/sortie :

Ils traduisent les signaux industriels en informations API réciproquement, appelées aussi coupleurs.

Beaucoup d'automates assurent cet interfaçage par des modules amovibles être modulaires par carte ou par rack. D'autres automates ont une structure monobloc, avec des modules intégrés dans un châssis de base, (cas des automates de télémécanique TSX17 et SIMATIC S7-314 IFM).

Le nombre total de modules est évidement limité, pour des problèmes physiques :

- Alimentation électrique.
- Gestion informatique.
- Taille du châssis.

Différents types de modules sont disponibles sur le marché selon l'utilisation souhaitée, les plus répandus sont :

3.3.1. Entrées/sorties TOR (Tout ou Rien) :

La gestion de ce type de variables constituant le point de départ historique des API reste une de leurs activités majeures.

Leur nombre est en général de 8,16, 24 ou 32 entrées/sorties, qui peuvent fonctionner :

- En continu : 24V, 48V.
- En alternatif : 24V, 48V, 100/120V, 2210/240V

3.3.2. Entrées/Sorties analogique :

Elles permettent l'acquisition de mesures (entrées analogique), et la commande (sorties analogiques). Ces modules comportent un ou plusieurs convertisseurs Analogique/Numérique (A/N) pour les entrées et Numérique / Analogique (N/A) pour les sorties dont la résolution est de 8 à 16 bits.

Les standards les plus utilisés sont : ± 10V, 0-10V, ±20mA, 0-20mA et 4-20mA.

Ces modules sont en général multiplexés en entrée pour n'utiliser qu'un seul convertisseur A/N, alors que les sorties exigent un convertisseur N/A par voie pour pouvoir garder la commande durant le cycle de l'API.

3.3.3. Module spécialisés :

Ils assurent non seulement une liaison avec le monde extérieur, mais aussi une partie du traitement pour soulager le processeur et donc améliorer les performances.
Ces modules peuvent posséder un processeur embarqué ou une électronique spécialisée.
On peut citer :

Les cartes de comptage rapide :
Elles permettent saisir des événements plus courts que la durée du cycle travaillant à des fréquences qui peuvent dépasser 10kHz.

Les Entrées/Sorties déportées :
Leur intérêt est de diminuer le câblage en réalisant la liaison avec détecteur, capteurs ou actionneurs au plus prés de ceux-ci, ce qui a pour effet d'améliorer la précision de mesure.

La liaison entre le boîtier déporté et l'unité centrale s'effectue par le biais d'un réseau de terrain suivant des protocoles bien définis.
L'utilisation de la fibre optique permet de porter la distance à plusieurs kilomètres.

3.4. L'alimentation électrique :

Elle a pour rôle de fournir les tensions continues nécessaires aux composants avec de bonnes performances, notamment face aux micro-coupures du réseau électrique qui constitue la source d'énergie principale.

La tension d'alimentations peut être de 5V, 12V ou 24V.

D'autres alimentations peuvent être nécessaires pour les châssis d'extension et pour les modules entrées/sorties.

Un onduleur est nécessaire pour éviter les risques de coupures non tolérées.

3.5. Les liaison :

Elles s'effectuent :

- avec l'extérieur par des **borniers** (à vitesse, à clipser, ...), sur lesquels arrivent des câbles transportant les signaux électriques.
- avec l'intérieur par des **bus**, liaisons parallèles entre les divers éléments. Il existe plusieurs types de bus, car on doit transmettre des données, des états, des adresses.

3.6. Eléments auxiliaires :

- Un ventilateur est indispensable dans les châssis comportant de nombreux modules ou dans le cas où la température ambiante est susceptible de devenir assez élevée.
- Un support mécanique : il peut s'agir d'un rack, l'automate se présente alors.
- Sous forme d'un ensemble de cartes, d'une armoire, d'une grille, et des fixations correspondantes.
- Des indicateurs d'état : concernant la présence de tension, la charge de la batterie, le bon fonctionnement de l'automate etc.

4. PROTECTION DE L'AUTOMATE

La protection des circuits d'entrée contre les parasites électriques est souvent résolue par découplage optoélectrique. Le passage des signaux par un stade de faisceau lumineux assure en effet une séparation entre les circuits internes et externes.

Du coté sorties, on doit assurer le même type de protection, mais amplification de puissance, avec au final un courant continu ou alternatif selon les cas.

Deux types de cartes électroniques sont utilisés :

4.1. Les modules à sorties statiques :

Relais statiques intégrant des composants spécialisés : transistor bipolaires, thyristors. Ces composants n'ont aucune usure mécanique et leurs caractéristiques de commutation se maintiennent dans le temps.

4.2. Les modules à relais électromagnétique :

Où le découplage résulte de l'existence de deux circuits électriques (Bobines d'excitation, circuits de puissance), ces relais électromagnétique ont l'avantage d'avoir une faible résistance de contact, une faible capacité de sortie et surtout un faible coût, mais une durée de vie et une vitesse de commutation inférieures aux sorties statiques.

5. ENVIRONNEMENT

Dans le cadre d'une évolution conduisant à une automatisation de plus en plus globale, l'automate est de moins en moins acquis seul, et même si c'est le cas, il doit pouvoir se connecter à d'autre matériels à processeur, et dialoguer avec les agents d'exploitation.

Les types de communication supportés par les API modernes sont : la communication avec un opérateur par un pupitre ou un terminal industriel : ils permettent une communication homme-machine, et ce dans les deux sens (clavier, ...)

6. LE STEP 7

STEP7 fait partie de l'industrie logicielle SIMATIC. Il représente le logiciel de base pour la configuration et la programmation de systèmes d'automatisation.

Les tâches de bases qu'il offre à son utilisateur lors de la création d'une solution d'automatisation sont :

- La création et gestion de projets.
- La configuration et le paramétrage du matériel et de la communication.
- La gestion des mnémoniques.
- La création des programmes.
- Le chargement de programmes dans les systèmes cibles.
- Le test de l'installation d'automatisation.
- Le diagnostic lors des perturbations dans l'installation.

Il s'exécute sous les systèmes d'exploitation de Microsoft à partir de la version Windows 95. Et s'adapte par conséquent à l'organisation graphique orientée objet qu'offre ces systèmes d'exploitation.

7. APPLICATIONS DE STEP7:

Il est constitué de deux types d'application ; applications de base et applications en options.

7.1. Applications du logiciel de base STEP7 :

Le logiciel STEP7 met à disposition les applications de base suivantes :

- Le gestionnaire de projets.
- La configuration du matériel.
- L'éditeur de mnémoniques.
- L'éditeur de programmes CONT, LOG, LIST.
- Le S7GRAPH pour la programmation séquentielle.
- La configuration de la communication NETPRO.
- Le diagnostic du matériel.

7.1.1. Gestionnaire de projets SIMATIC Manager :

Le gestionnaire de projets SIMATIC Manager gère toutes les données relatives à un projet d'automatisation, il démarre automatiquement les applications requises pour le traitement des données sélectionnées.

Le schéma suivant représente la fenêtre qui apparaît à l'ouverture du SIMATIC Manager

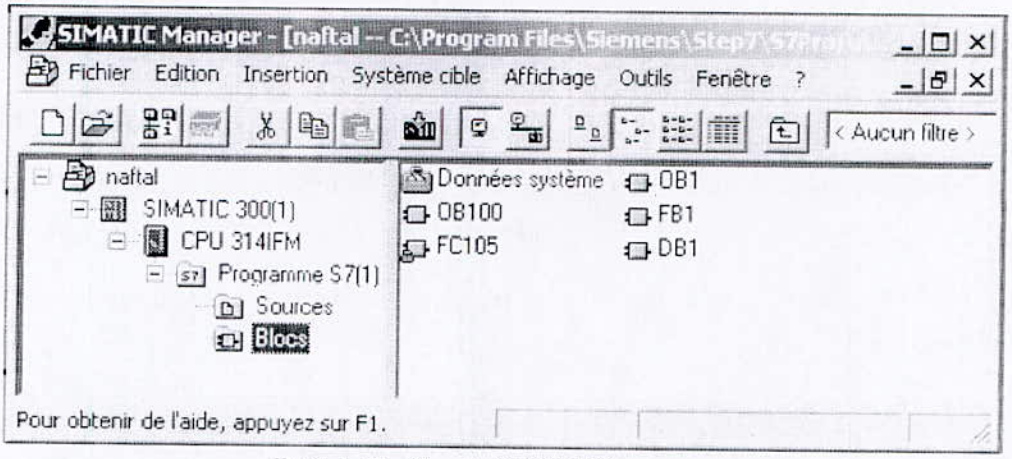


Fig2.1. Fenêtre du SIMATIC manager.

7.1.2. Configuration du matériel HW Config :

HW Config est utilisé pour configurer et paramétrer le support matériel dans un projet d'automatisation.

7.1.3. Editeur de mnémoniques :

Il permet la gestion de toutes les variables globales. En effet, il définit des désignations symboliques et des commentaires pour les signaux du processus (Entrées/Sorties), les mémentos, les blocs de données, les temporisations et les compteurs.

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.

7.1.4. Editeur de programme :

Les langages de base proposés sont :

- Le schéma à contact (CONT), langage graphique similaire aux schémas de circuits à relais, il permet de suivre facilement le trajet du courant.

- La liste d'instruction (LIST), langage textuel de bas niveau, à une instruction par ligne, similaire au langage assembleur.

- Le logigramme (LOG), langage de programmation graphique qui utilise les boîtes de l'algèbre de BOOLE pour représenter les opérations logiques.

L'éditeur de programmes permet aussi la visualisation et forçage de variables,

7.1.5. Le S7Graph :

Le S7Graph est conçu pour la programmation graphique des processus, en plus des possibilités que nous offre le STEP7, il nous permet une programmation claire et rapide

On va évoquer les avantages de la programmation avec le S7Graph dans le chapitre 2

7.1.6. Configuration de la communication Net Pro :

La configuration et le paramétrage de réseaux se font à l'aide de l'application NetPro. Elle permet de :

- Créer une vue graphique du réseau en question ainsi que les sous-réseaux qui le constituent.
- Déterminer les propriétés et les paramètres de chaque sous-réseau.

7.1.7. 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.

Avec le diagnostic, on peut avoir des informations générales sur les modules, les causes des erreurs, comme on peut détecter les causes des défaillances dans un programme.

7.2. Possibilité d'extension du logiciel de base STEP7 :

Cette extension est réalisée à l'aide de logiciels optionnels, regroupés dans trois catégories

7.2.1. Applications techniques :

Ensemble de langages de programmation évolués pour programmeur, des langages graphiques pour l'ingénieur en technologie et enfin des logiciels complémentaires pour le diagnostic, la simulation, la maintenance à distance, et la documentation de l'installation.

7.2.2. Logiciels exécutables :

Solutions finies programmées pouvant être appelées dans le programme utilisateur, et d'être directement intégrées dans la solution d'automatisation.

Exemple :

- Le contrôle PID qui permet l'intégration de régulateurs à action continue et de régulateurs à impulsion dans le programme utilisateur. L'application de paramétrage à laquelle la définition du régulateur est intégrée, permet le paramétrage rapide et le réglage optimal du régulateur.

- Le contrôle avec logique floue. Utilisé lorsque des processus ne peuvent pas ou peuvent difficilement être décrits mathématiquement, lorsque le déroulement de mécanismes et

de processus est imprévisible, lorsque des comportements non linéaires surviennent alors que l'on dispose d'une connaissance acquise par expérience du processus.

7.2.3. Interfaces homme/machine :

Ce sont des logiciels spécifiques au contrôle-commande dans SIMATIC. On peut citer :

- Le système de visualisation du processus SIMATIC WinCC qui est un système de base indépendant des branches et technologies d'utilisation et qui comporte toutes les fonctions indispensables au contrôle-commande.
- ProAgent qui permet un diagnostic précis et rapide du processus dans les installations et les machines en fournissant des informations relatives à la localisation et à la cause des erreurs.

8. PREMIER PAS VERS STEP7 :

Pour concevoir un projet avec STEP7, il existe 2 approches :

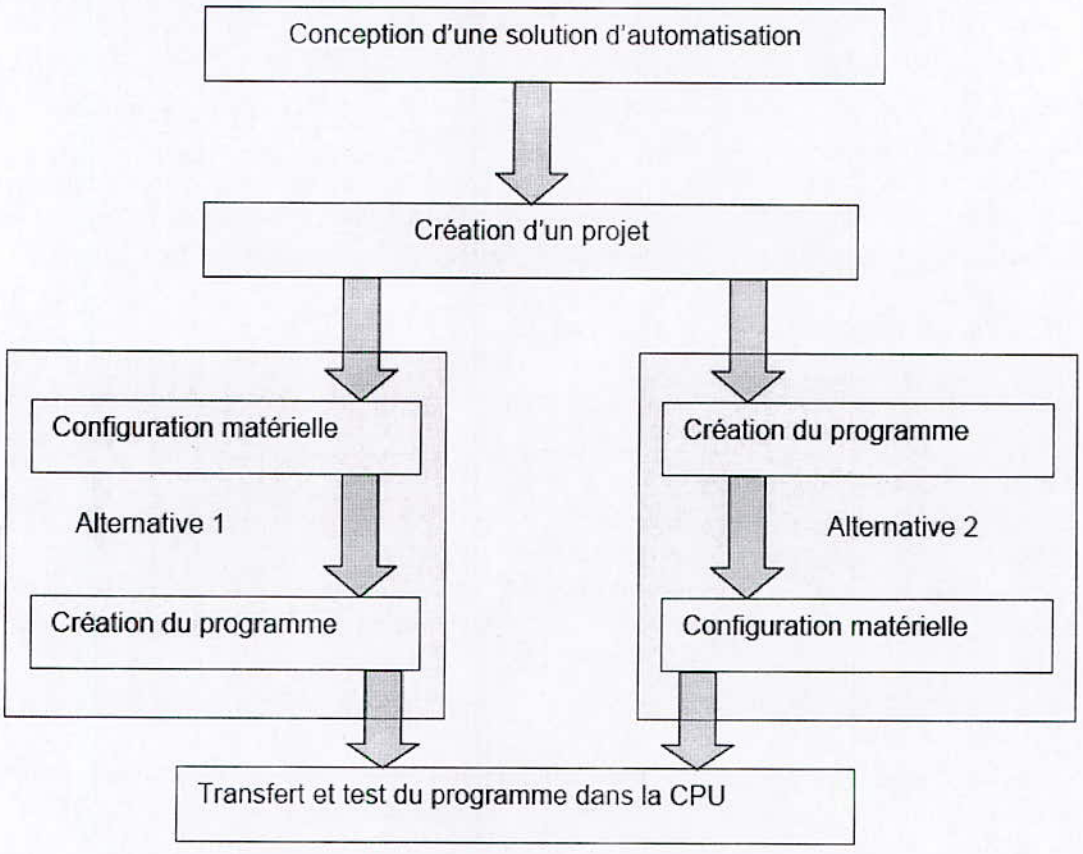


Fig2.2. Organigramme pour la création de projets sous STEP 7.

Il faut noter que pour un système contenant beaucoup de variables, la seconde alternative n'est pas très pratique.

Nous optons donc pour la première approche.

8.1. Création du projet avec STEP7 :

Pour créer un projet avec STEP 7, on peut lancer l'assistant de création de projet de STEP7, ou créer directement un projet que l'on configurera soi même.

8.1.1 Utilisation de l'assistant de création de projets :

Par défaut l'assistant de création de projets apparaît à chaque démarrage du SIMATIC Manager, si ce n'est pas le cas, son lancement se fait en passant par le menu Fichier>Assistante Nouveau projet'.

Cet assistant permet de créer avec une interface simple.

Les étapes à suivre sont les suivantes :

Etape1 :

Cliquer sur le bouton « suivant »

Etape 2 :

- Il faut choisir la CPU utilisée pour le projet, la liste contient normalement toutes les CPU supportées par la version de STEP7 utilisée.
- Dans le champ « Nom de CPU », il faut donner un nom à la CPU. Cela peut
- S'avérer utile dans le cas où l'on utilise plusieurs CPU dans un même projet.
- Il faut aussi choisir une adresse MPI pour la CPU. Si on utilise une seule CPU
- La valeur par défaut est 2.
- Cliquer sur suivant.

Remarque :
Lors de la sélection de la CPU une brève description est disponible dans la petite fenêtre à coté du choix de l'adresse MPI.

Etape 3 :

Cet écran permet d'insérer des blocs. Ces blocs seront décrits plus loin.

- Pour commencer on se contentera de OB1 seulement qui est le bloc principal (Équivalent du MAIN pour le langage C).
- On doit aussi choisir un langage de programmation parmi les trois proposés (LISTE, CONT ou LOG).
- Cliquer sur suivant.

Etape 4 :

Nommer le projet et cliquer sur créer.
Le projet est maintenant créé, on peut voir arborescence à gauche de la fenêtre
Qui s'est ouverte.

8.1.2. Création d'un nouveau projet sans l'assistant de création de projet :

Cette méthode est un peu plus compliquée, mais permet de mieux gérer le projet. Dans la fenêtre SIMATIC Manager, cliquer sur Fichier>Nouveau (ou encore CTRL+N), une fenêtre demandant un nom de projet s'ouvre. Il faut donc donner un nom au projet puis valider par OK.

La fenêtre du projet s'ouvre.

Le projet est vide, il faut lui insérer une station SIMATIC, cela est possible en cliquant sur le projet avec le bouton droit puis Insérer un nouvel objet >station SIMATIC 300.

La station SIMATIC n'est toujours pas configurée, il faut passer à l'étape de configuration matérielle.

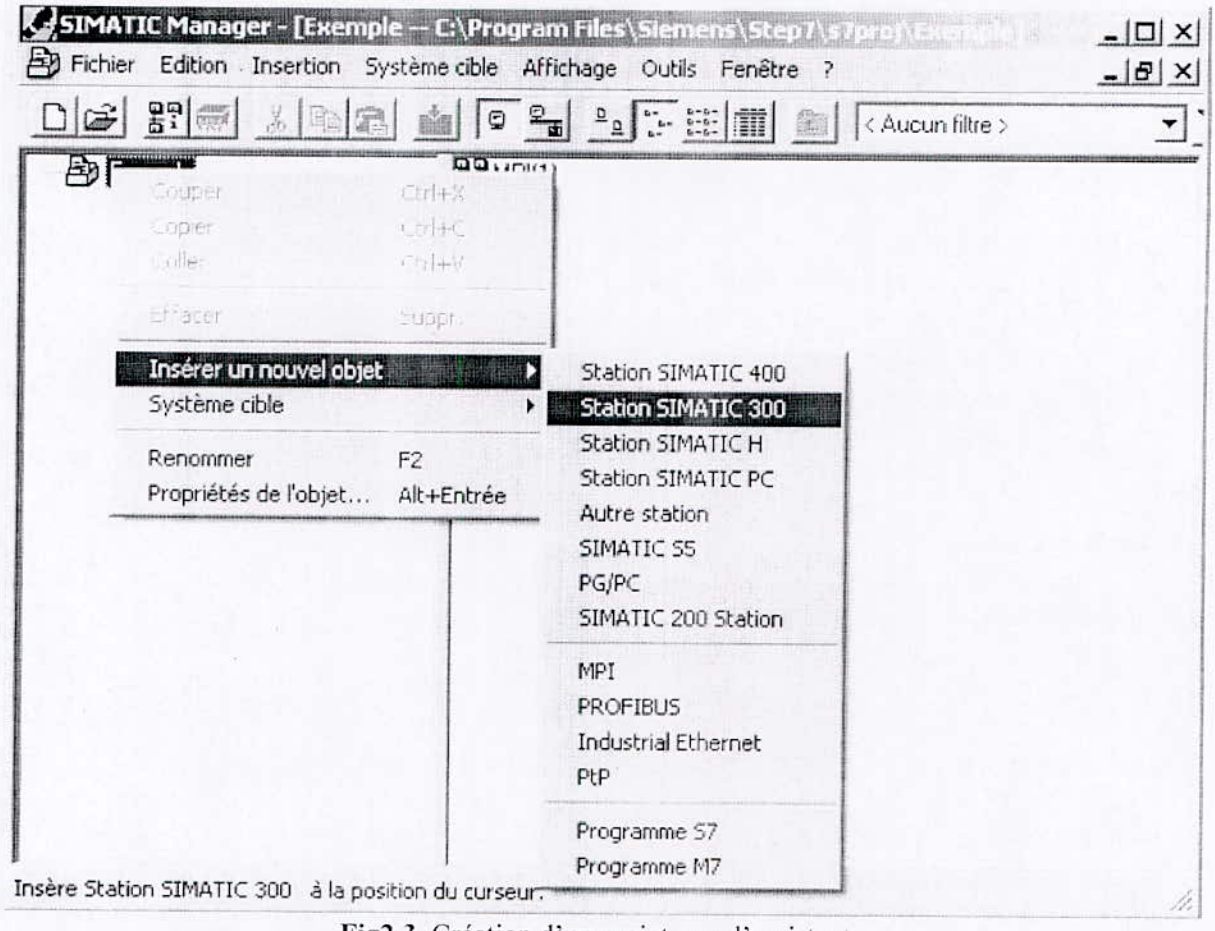


Fig2.3. Création d'un projet sans l'assistant.

8.2. Configuration matérielle :

La configuration matérielle est une étape très importante, elle permet de reproduire à l'identique le système utilisé (alimentation, CPU, modules etc.…).

Pour effectuer cette configuration, il faut aller sur l'icône Station SIMATIC300.

Sur la fenêtre de droite s'affichent deux icônes : « Matériel » et le nom de la CPU.

Il faut ouvrir l'icône matériel : la fenêtre HW Config s'ouvre.

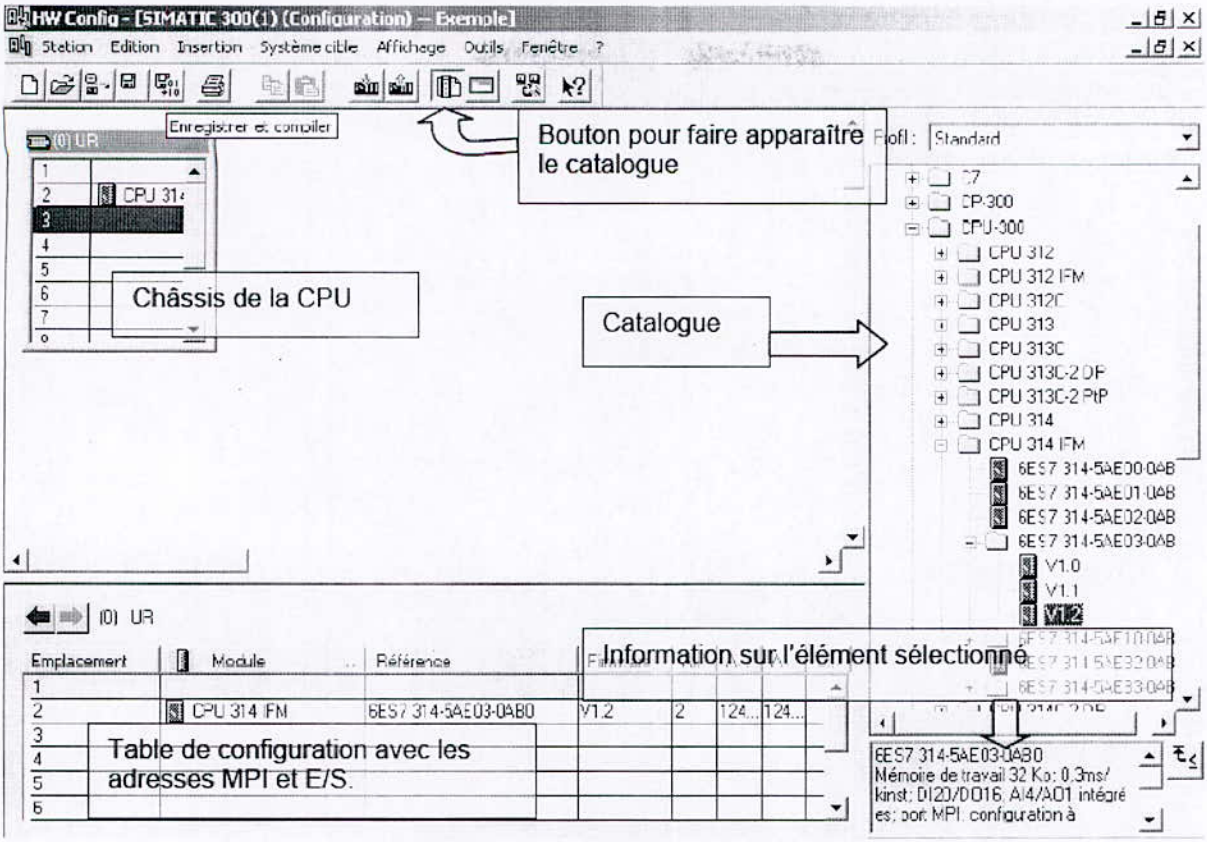


Fig2.4. Configuration du matériel.

Nous avons tout d'abord besoin d'un châssis ou RACK puis d'une alimentation. Dans le catalogue ouvrir le dossier PS-300 qui se trouve dans SIMATIC-300 puis choisir un modèle PS 307 2A.

Pour l'installer sur le châssis, sélectionner l'emplacement 1 à l'aide de la souris, puis double- cliquer sur l'alimentation.

On peut de la même manière insérer des composants sur le châssis en fonction de la configuration réelle. Dans notre cas, l'insertion de la CPU S7 314 IFM de référence 6ES7314-5EA03-0AB0 se fera dans l'emplacement 2.

Le paramétrage de la CPU, à l'aide de menu, permet de définir des caractéristique, telles que : le comportement à la mise en route, la surveillance du temps de cycle ainsi que l'activation et la désactivation des plages de rémanences et les fonctions intégrées, ces données sont enregistrées dans les blocs de données systèmes.

Le paramétrage des modules est réalisé automatiquement au démarrage de la CPU. Ainsi, le remplacement d'un module est ainsi possible sans nouveau paramétrage.

A la fin de la configuration, il suffit de cliquer sur Station>Enregistrer et compiler pour valider les changements apportés au châssis. De cette manière les changements seront pris en compte dans le reste du projet.

8.3. Définition des mnémoniques :

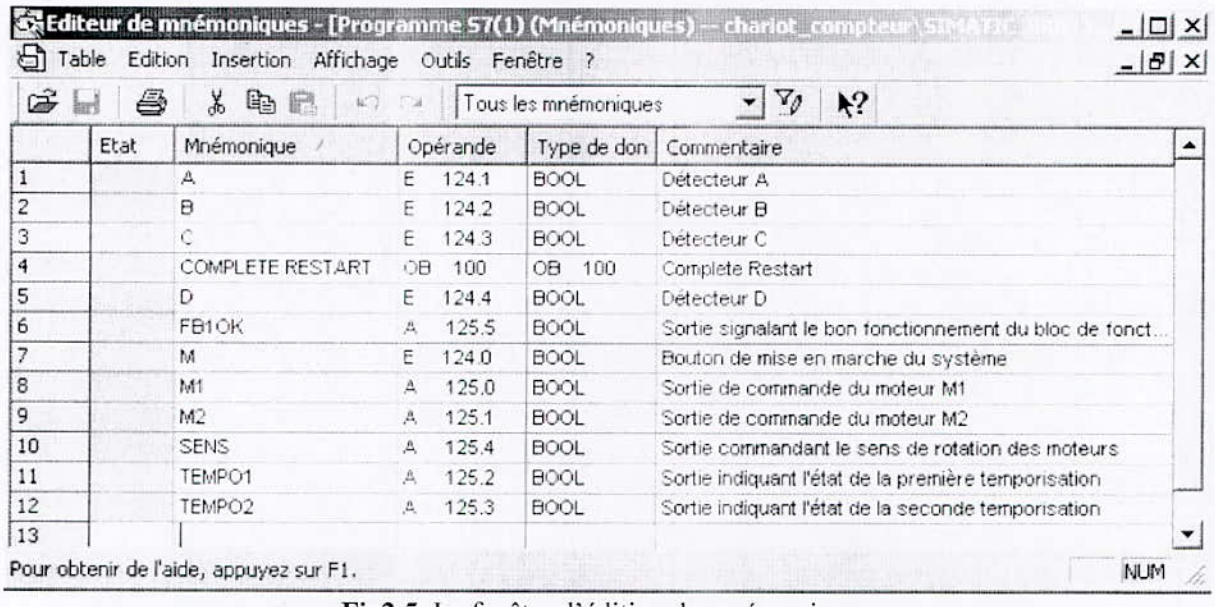
Il faut définir les variables qui vont être utilisées lors des étapes de programmation. L'utilisation de noms communs est plus aisée que la manipulation de chiffres (ex : utiliser « moteur » au lieu du bit de sortie A0.0).

Pour accéder à la table des mnémoniques : cliquer sur le dossier programme dans la fenêtre du projet, puis sur l'icône mnémoniques.

L'utilisation de cette table consiste à :

- Donner un nom à la mnémonique dans la première colonne.
- Donner la variable associée à cette mnémonique dans la seconde colonne.
- Le type de la donnée est automatiquement généré par STEP 7.
- Ecrire éventuellement un commentaire dans la colonne prévu à cet effet.

Après avoir défini toutes les mnémoniques, il suffit d'enregistrer pour que les changements soient pris en compte dans le reste du projet.



	Etat	Mnémonique	Opérande	Type de don	Commentaire
1		A	E 124.1	BOOL	Détecteur A
2		B	E 124.2	BOOL	Détecteur B
3		C	E 124.3	BOOL	Détecteur C
4		COMPLETE RESTART	OB 100	OB 100	Complete Restart
5		D	E 124.4	BOOL	Détecteur D
6		FB1OK	A 125.5	BOOL	Sortie signalant le bon fonctionnement du bloc de fonct...
7		M	E 124.0	BOOL	Bouton de mise en marche du système
8		M1	A 125.0	BOOL	Sortie de commande du moteur M1
9		M2	A 125.1	BOOL	Sortie de commande du moteur M2
10		SENS	A 125.4	BOOL	Sortie commandant le sens de rotation des moteurs
11		TEMPO1	A 125.2	BOOL	Sortie indiquant l'état de la première temporisation
12		TEMPO2	A 125.3	BOOL	Sortie indiquant l'état de la seconde temporisation
13					

Fig2.5. La fenêtre d'édition de mnémoniques.

Si on a besoin d'insérer de nouveaux objets dans le projet (ex : d'autre blocs de programme) il suffit de cliquer avec le bouton droit de la souris sur le dossier ou l'on veut ajouter l'objet puis insérer nouvel objet, et dans le menu sélectionner l'objet voulu.

9. ORGANISATION D'UN PROGRAMME UTILISATEUR :

Le logiciel de base STEP 7, permet de structurer le programme utilisateur.

Cette structuration est réalisée par la subdivision du programme en différentes parties autonomes. Il en résulte les avantages suivants :

- Ecrire des programmes importants et clairs.
- Standardiser certaines parties du programme.
- Simplifier l'organisation du programme.
- Modifier facilement le programme.
- Simplifier le test du programme, car on peut l'exécuter section par section.
- Faciliter la mise en service.

9.1. Hiérarchisation dans un projet :

Chaque programme utilisateur est structuré sous forme de projet.
Un projet représente l'ensemble des données et programmes d'une solution d'automatisation et se trouve à la tête d'une hiérarchie d'objets qui sont :

- objet projet.
- objet station.
- objet Modules programmables.
- objet Programme S7/M7.
- objet dossier Sources.
- objet dossier Blocs.

Fig2.6. L'hiérarchie d'un projet STEP 7.

9.1.1. Objet station :

Une station SIMATIC 300/400 représente une configuration matérielle S7.
Elle comporte un plusieurs modules programmable.

9.1.2. Objet modules programmables :

Représente les données de paramétrage d'un module comme les CPU et les modules fonctionnels.

9.1.3. Objet programme S7/M7 :

Un programme (S7/M7) est un dossier contenant les logiciels pour les modules programmable. Dans un programme S7/M7 figure déjà :

Une table de mnémoniques, un dossier « Blocs » et un dossier « Sources ».

9.1.4. Objet dossier sources :

Il contient les programmes source sous forme de texte. On peut saisir une partie ou tout le programme sous forme de source que l'on compilera ensuite en blocs.

Les avantages de la création de programme sous forme de source sont :

- La possibilité de programmer plusieurs blocs dans une même source.
- L'avantage d'enregistrer la source malgré la présence éventuelle d'erreurs de syntaxe, ce qui n'est pas possible lors de la création de blocs de code avec vérification de syntaxe. Cela signifie que les erreurs de syntaxe ne seront signalées que lors de la compilation de la source.

Elle peut être créée et traitée avec le choix de l'éditeur ASCII, puis importée et compilée en blocs individuels. La compilation entraîne la génération des différents blocs et leur sauvegarde dans le programme utilisateur S7.

La source doit être écrite en utilisant la syntaxe du langage de programmation « liste d'instructions (LIST) », et nécessite par conséquent l'application de règles précises.

9.1.5. Objet dossier Blocs :

Le dossier Blocs contient les blocs que l'on doit charger dans la CPU pour réaliser la tâche d'automatisation.

Il englobe les blocs de code (OB, FB, SFB, FC, SFC) qui contiennent les programmes qu'on doit charger dans la CPU, et les blocs de données (DB d'instance et DB globaux) qui contiennent les paramètres du programme.

Les blocs d'organisation (OB) :

Ils 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. On peut programmer les blocs d'organisation et déterminer ainsi le comportement de la CPU.

Le bloc d'organisation **OB1** est généré automatiquement lors de la création d'un projet, il représente le programme principal (MAIN dans le langage C). En effet, c'est le programme appelé cycliquement par le système d'exploitation.

Les autres blocs, existant dans le projet seront exécutés à leur appel par l'OB1.

Les blocs fonctionnels (FB) :

- **Le FB :**

C'est un sous programme écrit par l'utilisateur, il facilite la programmation de fonctions complexes souvent utilisées. Il exécute par l'appel d'autre bloc de code.

Un bloc de données d'instance, qui constitue sa mémoire, lui est associé. Ce dernier contient les paramètres transmis au FB ainsi que les variables statiques.

- **Le bloc fonctionnel système (SFB) :**

C'est un bloc fonctionnel intégré à la CPU S7. Les SFB font partie du système d'exploitation, par conséquent, ils ne sont pas chargés en tant que partie du programme. Comme les FB, les SFB sont des blocs avec mémoire. On doit donc également créer pour les SFB des blocs de données d'instance que l'on charge dans la CPU en tant que partie du programme.

Ils sont utilisés pour des fonctions spéciales intégrées de la CPU 314 IFM, comme ils peuvent être utilisés pour la communication via des liaisons configurées.

Les fonctions (FC) :

- **La Fac :**

Elle contient des routines pour les fonctions fréquemment utilisées, comme le renvoi d'une valeur au bloc appelant. Elle est sans mémoire et contient uniquement des variables temporaires qui sont sauvegardées dans la pile de données locales et perdues à l'achèvement de cette fonction.

Mais elle peut faire appel à des blocs de données globaux pour la sauvegarde de ses données.

- **La fonction système (SFC) :**

C'est une fonction intégrée dans la CPU S7, pré- programmée et testée. Elle est appelée à partir du programme. Comme ces fonctions font partie du système d'exploitation, elles ne sont pas chargées en tant partie du programme. Comme les FC, les SFC constituent des blocs sans mémoire.

Parmi les fonctionnalités qu'elles proposent :

Le contrôle du programme, la gestion des alarmes horaires et temporisées, la mise à jour de la mémoire image du processus, l'adressage de modules et la création de messages relatifs aux blocs.

Les blocs de données d'instance (DB d'instance) :

Associé à chaque bloc fonctionnel, il contient les paramètres effectifs et les données statiques du FB.

On peut utiliser plusieurs DB pour un même FB ; par exemple, un FB pour la commande de plusieurs moteurs, les données de chaque moteur sont sauvegardées dans différents DB.

Les blocs de données globaux (DB) :

A l'opposé des DB d'instance qui ne sont associés qu'au blocs fonctionnels, les DB globaux servent à l'enregistrement de données utilisateur pouvant être utilisées par tous les autres blocs de code.

Remarque :
A l'appel d'un FB, le DB d'instance lui correspondant est automatiquement généré une fois qu'il est inséré à son emplacement.
Mais si on veut créer un bloc de données à partir du dossier bloc, on procède comme suit :

- On clique avec le bouton droit de la souris, puis sur Insérer un nouvel objet>Bloc de données,
- Ou Dans la fenêtre SIMATIC Manager, on clique sur le menu Insertion> BlocS7>bloc de données.

Dans les deux cas, le type du bloc de données sera demandé (DB d'instance ou DB global).

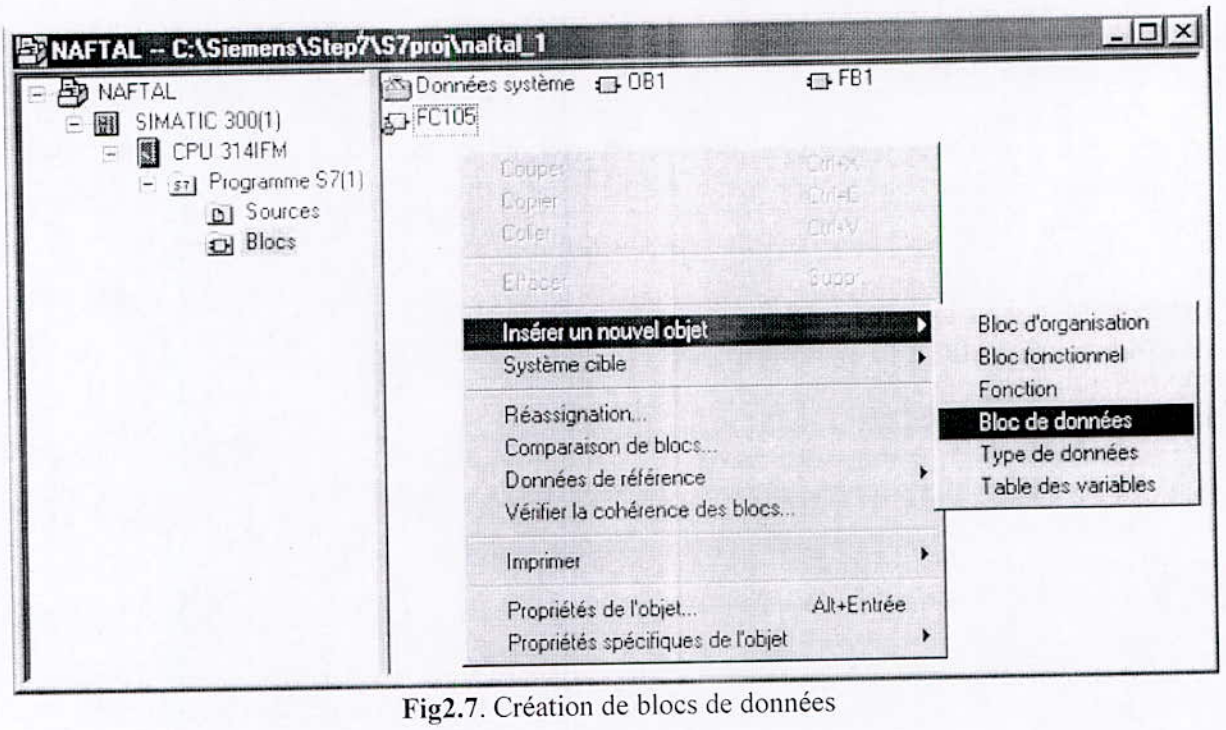


Fig2.7. Création de blocs de données

10. BLOCS D'ORGANISATION :

Ils permettent l'organisation totale du programme utilisateur, en déclenchant l'exécution conditionnelle de certaines parties du programme, ceci se fait :

- A la mise en route de la CPU.
- A des horaires précis ou cycliquement.
- A l'apparition d'erreurs.
- A l'apparition d'alarmes de processus.

10.1. Blocs d'organisation dans la CPU S7-314 IFM :

Les blocs d'organisation qui existe dans la CPU S7-314 IFM sont :

10.1.1. L'OB1 (programme cyclique) :

Comme dans toutes les CPU S7, l'OB1 est exécuté de manière cyclique.
Il peut être interrompu par tous les autres OB car il dispose de la classe de priorité la plus basse. Les événements suivants provoquent son appel par le système d'exploitation :

- La fin du traitement de la mise en route.
- La fin du traitement de son cycle précédent.

10.1.2. L'OB100 (Mise en route) :

Il contient le programme à exécuter lors d'une mise en route.

10.1.3. Les OB d'alarmes :

On appelle alarmes les événements qui déclenchent l'appel d'un OB donné.

- **OB10 (Alarme horaire) :**

A l'appel de cet OB, le programme qui s'y trouve ne s'exécute qu'une seule fois ou de manière cyclique toutes les minutes, toutes les heures, tous les jours, toutes les semaines, tous les mois, tous les ans, ou à la fin du mois.

Pour appeler l'OB10, il faut qu'il soit activé et paramétré, ceci se fait en utilisant :

- STEP : Dans HW Config, activer et paramétré l'alarme horaire en accédant
Accédant aux propriétés de la CPU.

- Le programme utilisateur : activer l'alarme en appelant le SFC30 et la
Paramétrer avec la SFC28.

- **OB20 (Alarme temporisée) :**

Cet OB est appelé après l'écoulement d'un retard paramétré. Pour ce faire, il faut procéder comme suit :

- Appeler la fonction système SFC32,
- Charger l'OB d'alarme temporisée dans la CPU comme partie du programme.

- **OB35 (Alarme cyclique) :**

L'OB est appelé à des intervalles de temps égaux qui peuvent être de 1 ms à 60000 ms. La valeur de cycle par défaut est de 100 ms, mais on peut la changer en accédant aux propriétés de la CPU dans HW config.

- **OB40 (Alarme processus) :**

Quand un module déclenche une alarme de processus, le système d'exploitation identifie de quel module il s'agit, et déclenche l'OB40, qui une fois exécuté, permet l'acquittement de cette voix particulière.

Si une autre alarme de processus sur le même module est détectée entre l'identification et l'acquittement de l'alarme en cours, elle n'est pas prise en considération.

Dans le cas de la CPU 314 IFM, les alarmes de processus peuvent être déclenchées par les fonctions intégrées, si elles sont validées lors de la configuration de ces fonctions.

10.1.4. Les OB d'erreurs :

- **OB82 (diagnostic des modules) :**

Le diagnostic signale les erreurs survenant sur les modules et permet alors l'exécution de l'OB82. Cet OB contient, dans ses variables locales, l'adresse de base logique du module erroné ainsi que des informations de diagnostic de quatre octets de long.

Si l'OB82 n'a pas été programmé, la CPU passe à l'état arrêt à la survenue d'une erreur.

- **OB80 (Erreur asynchrone) :**

Il est appelé si l'une des erreurs suivantes se produit lors de l'exécution d'un OB :

- Dépassement du temps de cycle.
- Erreur d'acquittement lors de l'exécution d'un OB.
-Saut de l'heure de déclenchement d'un OB (horloge avancée).

Si l'OB80 n'a pas été programmé, la CPU passe à l'état arrêt à la survenue d'une erreur.

- **OB81 (Erreur d'alimentation) :**

Cet OB est exécuté si un événement provoque une erreur d'alimentation ; par exemple, absence ou rétablissent de la tension de sauvegarde dans la CPU.

Si l'OB81n'a pas été programmé, la CPU ne passe pas à l'état arrêt à la survenue d'une erreur.

- **OB85 (Erreur d'exécution de programme) :**

Cet OB est exécuté si l'un des événements suivants se produit :

- Absence d'OB dans la CPU lors de son appel.
- Erreur lors de l'accès du système d'exploitation à un bloc.
- Erreur d'accès à la périphérie lors de l'actualisation de la mémoire image Des entrées.
- Erreur d'accès à la périphérie lors du transfert de la mémoire image aux Modules de sorties.

Si l'OB n'a pas été programmé, la CPU passe à l'état arrêt à la survenue d'une erreur.

- **OB87 (Erreur de communication) :**

Le programme de cet OB est exécuté, si une erreur de communication se produit. Par exemple, l'état d'ensemble des données globales ne peut pas être écrit dans le bloc de données.

Si l'OB87 n'a pas été programmé, la CPU passe à l'état arrêt si une erreur survient.

- **OB121 (Erreur de programmation) :**

L'apparition d'erreur lors du traitement du programme utilisateur, déclenche l'appel de ce bloc. Cette erreur peut être :

- Une erreur de conversion BCD.
- Un numéro de temporisation erroné.
- Une erreur de numéro de compteur.
- Une erreur d'écriture à l'accès au DB.
- Une erreur de numéro de bloc à l'ouverture d'un DB.
- Une erreur de numéro de bloc à l'appel d'une FC.
- Une erreur de bloc à l'appel d'un FB.
- Un DB non chargé.
- Un FC non chargé.
- Un FB non chargé.

Si l'OB121 n'a pas été programmé, la CPU passe à l'état arrêt à la survenue d'une erreur.

- **OB122 (Erreur d'accès à la périphérie) :**

Cet OB se déclenche à l'apparition d'erreur d'accès à la périphérie en lecture ou en écriture.

Si l'OB122 n'a pas été programmé, la CPU passe à l'état arrêt à la survenue d'une erreur.

Remarque

Les erreurs résolues par l'appel des deux blocs OB121 et OB122 peuvent être masquées, démasquées ou lues, par les fonctions systèmes suivantes :

- La SFC36 : masque certains codes d'erreurs.
- La SFC37 : démasque les codes d'erreurs qui ont été masquées par la SFC36.
- La SFC38 : lit le registre d'erreur.

10.2. Classes de priorité des blocs d'organisation :

Les blocs d'organisation définissent l'ordre 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.

OB	Classes de priorité
OB1	1
OB10	2
OB20	3
OB35	12
OB40	16
OB80	26
OB81	26
OB82	26
OB85	26
OB100	27
OB121	Selon l'OB responsable de l'interruption
OB122	

Tableau 2.1. Classes de priorité des blocs d'organisation

Remarque :

STEP7 permet la modification des classes de priorité pour les alarmes horaires, alarmes temporisées, alarmes cycliques de processus. Mais cette propriété n'est pas possible dans les CPU S7 300.

11. DEROULEMENT D'UN PROGRAMME :

11.1. Les programmes existants dans la CPU :

Dans une CPU, s'exécutent deux programmes ; le système d'exploitation et le programme utilisateur.

11.1.1 Le système d'exploitation :

Il regroupe toutes les fonctions et procédures qui ne sont pas liées à la tâche de programmation, ces fonctions et procédures sont :

- La mise en route.
- L'actualisation de la mémoire image des entrées MIE et l'émission de la mémoire image des sorties MIS.
- L'appel du programme utilisateur.
- L'enregistrement des alarmes et l'appel des OB d'alarme.
- La détection et le traitement des erreurs.
- La gestion des zones de mémoires.
- La communication avec les différents partenaires de communication.

La modification des paramètres par défaut du système d'exploitation permet d'influer sur le comportement de la CPU dans des cas précis. Par exemple, le changement des classes de priorités des blocs d'organisation dans les CPU S 400.

11.1.2. Le programme utilisateur :

Il est créé par l'utilisateur puis chargé dans la CPU, il contient toutes les fonctions nécessaires au traitement de la tâche d'automatisation. il doit :

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

Le déroulement d'un programme commence par une mise en route qui n'est exécutée qu'une seule fois, suivie de l'exécution cyclique du programme utilisateur.

11.2. La mise en route :

Lors de la mise en route la CPU se comporte comme suit :

- Le programme contenu dans l'OB de mise en route est exécuté, dans ce programme on peut définir des initialisations utiles pour le programme utilisateurs.
- Aucun traitement de programme déclenché par horloge n'est possible.
- Les temporisations sont mises à jour.
- Le compteur d'heure de fonctionnement est exécuté.
- Les sorties TOR des modules de signaux sont verrouillées, mais peuvent être mises à 1 par accès direct.

En plus de la mise sous tension, les causes de la mise en route peuvent être :

- Le changement de position du commutateur de mode de fonctionnement de STOP à RUN ou à RUN/P.
- A la demande d'une fonction de communication depuis la console de programmation (PG) ou par l'appel des blocs fonctionnels de communication.

Il existe 3 types de mise en route :

Démarrage à chaud, démarrage à froid et le redémarrage qui correspondent respectivement aux OB 100, OB 102, OB 101.

La seule mise en route disponible dans la CPU 314 IFM est le démarrage à chaud.

11.2.1. Démarrage à chaud :

Quand la CPU fonctionne sans pile de sauvegarde, un effacement général est automatiquement effectué à la mise sous tension, puis un démarrage à chaud est exécuté Le programme utilisateur doit se charger de nouveau.

Au démarrage à chaud, les démarches faites par la CPU dans l'ordre sont :

- Effacer la pile des interruptions et la pile des blocs.
- Effacer les mémentos, temporisation et compteurs non rémanents,
- Effacer la mémoire image des sorties,
- Effacer les sorties des modules de signaux,
- Rejeter les alarmes de processus,
- Rejeter les alarmes temporisées,
- Rejeter les alarmes de diagnostic,
- Actualiser la liste d'état système (SZL),
- Exploiter les paramètres des modules et les transmettre aux modules ou bien leur transmettre les valeurs par défaut,
- Traiter l'OB de mise en route concerné,
- Actualiser la mémoire image des entrées,
- Valider les sorties TOR (débloquer les sorties TOR) après passage à l'état de fonctionnement « marche ».

11.2.2. Démarrage à froid et redémarrage :

Au démarrage à chaud, l'effacement des mémentos, des temporisations et des compteurs concernent uniquement les zones non rémanentes, alors que le démarrage à froid les efface tous.

Le redémarrage n'efface aucune zone de mémoire ; la pile des interruptions, la pile des blocs, les mémentos, les temporisations, les compteurs.

L'effacement de la mémoire image des sorties et les sorties des modules de signaux est paramétrable. La propriété qui distingue le redémarrage des deux autres types de mise en route est le traitement du cycle restant (partie du programme utilisateur n'ayant pas pu être exécutée en raison d'une mise hors tension).

11.3. Exécution cyclique :

A la fin de la mise en route, le programme utilisateur s'exécutera de manière cyclique dans l'OB comme suit :

- Ecriture de la mémoire image des sorties MIS dans les modules de sorties.
- Lecture des entrées et mise à jour de la mémoire image des entrées.
- Traitement du programme utilisateur.

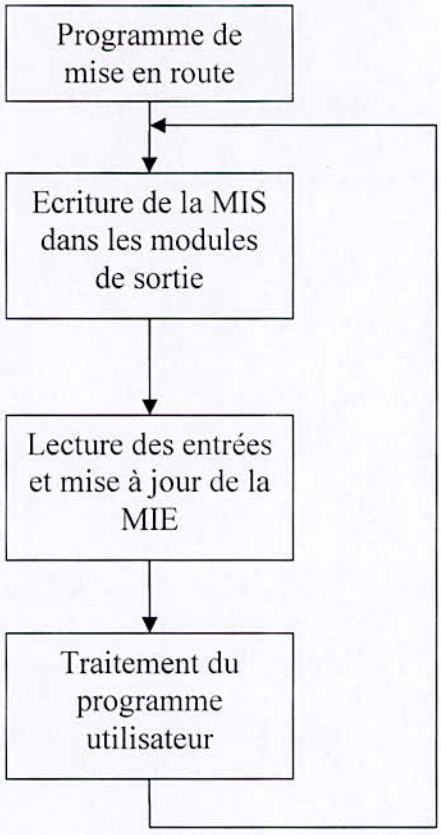


Fig2.8. Déroulement d'un programme utilisateur.

12. LES LANGAGES DE PROGRAMME :

12.1. Le CONT :

C'est un langage dont la logique est inspirée des réseaux électriques, ce qui en fait un langage facile pour les habitués des montages à base de relais.

Le langage à contact, est aussi appelé CONT ou LADDER. C'est un langage entièrement graphique, bien adapté au traitement logique simple de type combinatoire. Il utilise les symboles graphiques comme des contacts à ouverture ou à fermeture et des bobines. Ainsi un programme écrit en langage à contacts ne se présente pas sous la forme d'une liste d'instructions, mais comme une liste de schémas développés classique.

Des blocs d'opérations logiques ou arithmétiques pré-programmés peuvent être insérés dans les réseaux de contacts.

Les opérations combinatoires sur bits utilise deux chiffres : 1 et 0. Pour les contacts et les bobines, 1 signifie activé ou excité et 0 signifie désactivé ou désexcité.

Les opérations de combinaison sur bits évaluent les états de signal 1 et 0. et les combinent selon la logique booléenne. Le résultat de ces combinaisons est égal à 1 ou 0. Il s'agit du résultat logique (RLG).

Les éléments principaux d'un réseau en langage à contacts sont :

- ⊢ Barre d'alimentation (à gauche).
- ⊣ Barre représentant la masse (à droite).
- ⊣⊢ Contact associé à une variable.
- —( ) Bobine de sortie.

12.1.1. Barre d'alimentations et liaisons :

Un diagramme CONT est limité à sa gauche par des barres d'alimentation et à sa droite par une barre représentant la masse.

Les différents composants sont reliés entre eux par des arcs de liaisons horizontaux ou verticaux. Chaque segment de liaison peut prendre les états TRUE ou FALSE (1 ou 0).
L'état d'une liaison est propagé à sa droite.

Toute liaison horizontale connectée à une barre d'alimentation à gauche est active.
On peut avoir des liaisons multiples en combinant plusieurs arcs de liaisons horizontaux à un arc de liaison vertical.

Avec ces quelques règles on peut déjà écrire un réseau basic :

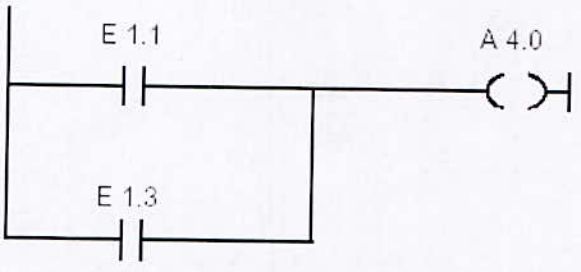


Fig2.9. « OU » logique avec CONT.
A4.0 n'est mise à 1 que si : E1.1 **ou** E1.3 sont à 1.

12.1.2. Les contacts :

Les principaux contacts qui peuvent être utilisés dans un diagramme sont :

- contact à fermeture ⊣⊢
- contact à ouverture ⊣/⊢

Remarques :

- Le nom de la variable associée est inscrit au –dessus du symbole graphique.
- Les contacts ne représentent pas ce qui est relié physiquement à l'automate, mais sont une interrogation à 1 ou à 0 de l'opérande associé.

- Un contact représente une liaison entre l'état d'un arc et la variable booléenne associée à ce contact.

En effet l'état de la liaison à droite du contact n'est que le résultat du ET logique entre l'état de l'arc de gauche et :
- L'état de la variable associée au contact pour un contact à fermeture.
- L'inverse de l'état de la variable associée au contact pour un contact à ouverture.

12.1.3. Les bobines :

Les bobines standard sont une affectation du résultat logique à la variable associée.

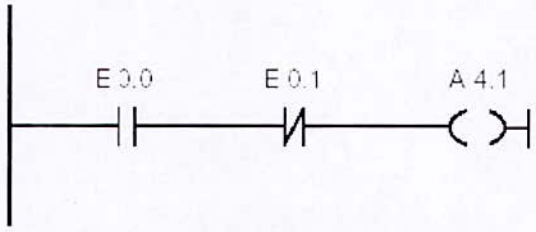


Fig2.10. ET logique avec CONT.

A4.1 n'est mise à 1 que si : E0.0 **et non** (E0.1) sont à 1 simultanément.

Remarque :
Les opérations CONT possibles avec STEP 7 sont citées en annexe.

12.2. Le LOG :

Le langage LOG est un langage graphique qui utilise des boites logiques d'algèbre de BOOL. La base de ce langage est la logique binaire, mais on peut aussi faire des opérations plus complexes telles que les opérations mathématiques à l'aide de blocs.

Les instructions LOG peuvent être sous quatre formes :

- sous forme d'éléments : comme une inversion logique.
- sous forme de boite avec opérande : comme une affectation.
- sous forme de boite avec opérande et valeur : comme une temporisation.

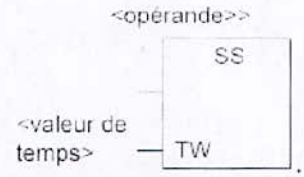


Fig2.11. Boite avec opérande et valeur.

- sous forme de boite avec paramètre : comme diviser un nombre réel.

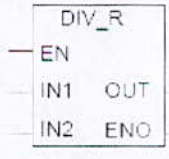


Fig2.12. Boite avec paramètre.

Remarque :

- Pour l'activation d'une boite, il faut que EN soit à l'état haut.
- Si la boite s'exécute sans erreur alors ENO est mise à 1, dans le cas contraire ENO est à 0.

Adressage :

Il n'y a que deux types d'adressage possibles avec LOG :

- Adressage immédiat : donne une constante comme opérande.
- Adressage direct : donne une adresse comme opérande.

Exemple :

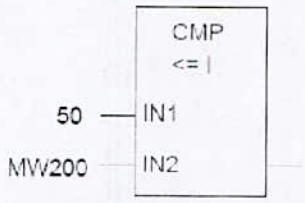


Fig2.13. Adressage immédiat et direct avec LOG.

50 est la valeur effective du premier paramètre, c'est un adressage immédiat.
MW200 est la zone mémoire où est stockée la valeur du second paramètre : c'est un adressage direct.

12.3. Le LIST :

12.3.1. Présentation :

Le langage LIST figure parmi les langages de base du logiciel STEP7, sa syntaxe est similaire à celle de l'assembleur.

C'est le langage le plus proche du langage machine MC7, des CPU S7, ce qui lui donne l'avantage d'être le langage le plus adapté pour la programmation avec optimisation d'espace mémoire et de temps d'exécution.

Il dispose d'un jeu d'instruction très important permettant la création de programmes utilisateur complets.

Tout programme écrit en CONT ou en LOG peut être réécrit en LIST.

12.3.2. Structure et éléments de LIST :

Dans le langage LIST, l'éditeur de programme divise l'espace de programmation en deux parties ; partie programme et partie commentaire, ces deux parties sont séparées par *II.*

12.3.3. Structure d'une instruction :

Une instruction LIST est constituée soit :

- En une opération seulement,
- En une opération + opérande.

L'opérande peut indiquer :

- une constante.
- un bit du mot d'état.
- une mnémonique.
- un bloc de données.
- un identificateur d'opérande d'une zone mémoire précise + une adresse.

Exemple :

Type d'opérande		**Exemple**	**Explication**
Constante	Entier	L 25	Charger l'entier 25
	Temporisation	L S5T#12s	Charger 12 secondes comme valeur initiale de la temporisation dans l'accumulateur 1
Adresse dans le mot d'état	Bit dans le mot d'état	U BIE	Interroger l'état du bit RB du mot d'état
Mnémonique	Nom symbolique	L vitesse	Charger l'octet, le mot ou le double mot dont le mnémonique est vitesse
Bloc de Données	Donnée du bloc de données.	U DB4.DBX5.1	Interroger l'état du bit 5.1 du bloc de donnée 4

Tableau 2.2. Quelques types d'opérandes.

Choix du GRAFCET :

Lorsque certaines spécifications sont exprimées en langage courant, il y a un risque permanent d'incompréhension. Certains mots sont peu précis, mal définis ou possèdent plusieurs sens, ce qui nous amènent à dire que Le langage courant est mal adapté pour décrire précisément les systèmes séquentiels

Le GRAFCET fut donc créé pour représenter de façon symbolique et graphique le fonctionnement d'un automatisme. Cela permet une meilleure compréhension de l'automatisme par tous les intervenants.

Afin de mieux éclaircir notre travail, on utilise le GRAFCET comme étant le langage de programmation principal de ce projet, et cela par l'utilisation de l'application de STEP7 destinée à ce type de langage (graphique) qui est le S7-Graph.

13. Avantages De L'utilisation Du S7- Graph

- Avec le langage de programmation S7-GRAPH, l'étendue des fonctions de STEP7 va être élargie à une programmation graphique pour l'enchaînement des commandes.
- Avec S7-GRAPH, vous pouvez programmer l'enchaînement des commandes clairement et rapidement.
- Le processus sera pour cela décomposé en unités et représenté graphiquement. Dans les unités, les actions à accomplir seront définies.
- Les conditions de la prochaine étape (réceptivités) peuvent être programmées en CONT ou en LOG.
- Le langage de programmation S7-GRAPH correspond à la norme DIN EN 61131-3 (IEC 61131-3) définissant le langage SFC « Sequential Function Chart ».

14. Enchainement Des Commandes

Un Graph-7 est une succession d'états obligatoires, dont l'enchaînement dépend de la validation des transitions entre chaque état.

La série d'étapes peut être programmée d'une manière particulière, avec par exemple, des sauts, des boucles et des branchements.

On peut programmer ce déroulement de commande avec le S7-GRAPH, puisque cet enchaînement peut être représenté graphiquement très facilement et très rapidement.

Il y a deux catégories de Graph-7 :

14.1 Commande temporelle

Avec une commande temporelle, les transitions sont uniquement dépendantes du temps. Pour produire une transition on peut utiliser par exemple une minuterie, un compteur, une temporisation ou une succession de cames en régime constant.

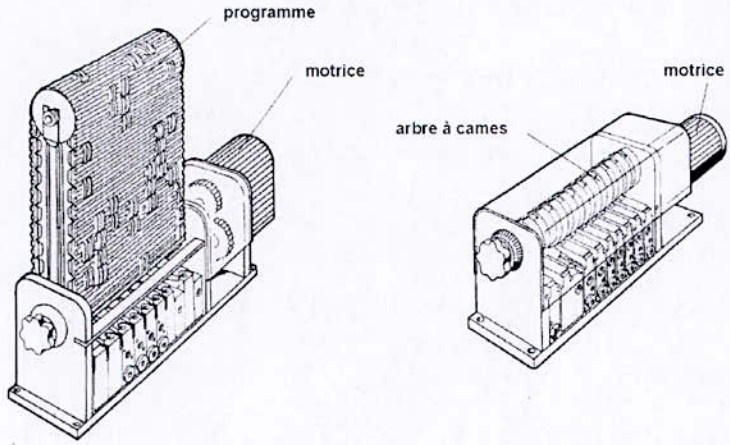


Fig2.14. Exemple de commande temporelle.

14.2 Commande dépendant des événements

Dans un Graph-7 dépendant du procédé, les transitions sont uniquement dépendantes du signal du dispositif commandé. Pour produire ce signal, des générateurs de signaux tels que des capteurs, des commutateurs, des boutons ou des capteurs sont utilisés. Les signaux produits peuvent aussi être liés à des fonctions de temps.

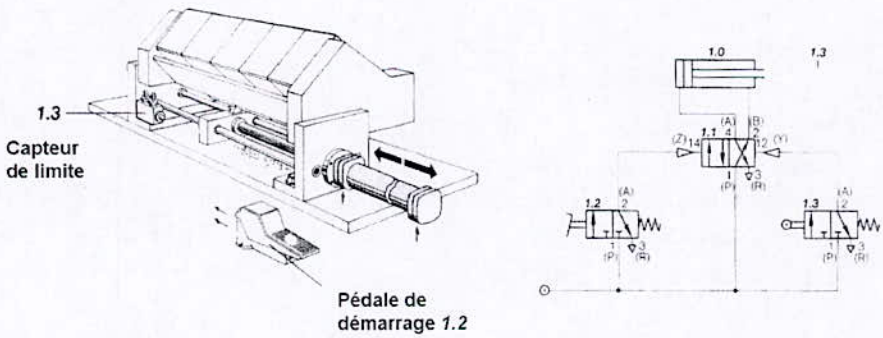


Fig2.15. Exemple de commande dépendant des évènements.

Système de découpe

Après l'action sur la pédale de démarrage l'arbre du vérin sort. Le capteur de limite détecte la position extrême de l'arbre du vérin. La rentrée après détection s'effectue automatiquement.

15. DIFFERENTES REPRESENTATIONS DES DIAGRAMMES D'ETAT

La coordination des actions de commande peut être produite à travers des représentations adéquates. Même avec des problèmes exigeants, les liens entre les éléments se montrent encore rapides et sûres. En outre, une représentation de diagrammes d'état permet une compréhension pointue dans un cadre élargi.

Différentes formes de représentation des diagrammes d'état

-Description des tâches de commande

Le déroulement des commandes est décrit sous forme de texte.

-Mise en forme chronologique

De brèves phrases représentent les actions.

-Représentation en tableau

Un tableau représente le déroulement pas à pas.

-Représentation concise

A travers une représentation simplifiée des actions le déroulement peut être représenté rapidement et simplement.

-Diagramme d'état

A l'aide d'un diagramme pas à pas et temporel, le déroulement des commandes est représenté graphiquement pour fournir une meilleure vue d'ensemble des liens.

-Graph-7

Une représentation orientée processus des tâches de commande. Le Graph-7 remplace ou complète la description textuelle et représente les tâches de commande avec leurs caractéristiques essentielles et leurs applications respectives. S7-GRAPH est un langage de programmation qui correspond fondamentalement à un Graph-7.

A partir d'un exemple de programme, nous allons commenter les différentes représentations.

15.1 Descriptions des tâches de commande

On souhaite créer une commande pour une perceuse automatique. La barre va être découpée par un fonctionnement commun de l'unité de travail et de la lame.

Le déplacement est produit par le vérin d'avancement (vérin B), qui est entraîné par les mouvements de va et vient de la pince de serrage pneumatique (vérin A).

Si la barre se trouve bloquée contre un arrêt, elle est retenue par le tendeur (vérin C).

Ensuite le découpage peut commencer (vérin D) et en même temps se produit l'ouverture de la pince de serrage (vérin A). Si la pince de serrage (vérin A) est ouverte, alors le retour (vérin B) s'effectue jusqu'en position initiale.

Quand le découpage est terminé (vérin D) et que l'unité de travail a atteint sa position initiale, le tendeur (vérin C) s'ouvre et un nouveau cycle de travail peut commencer.

Le démarrage est déclenché en appuyant sur la pédale de démarrage, si tous les vérins se trouvent dans leurs positions finales.

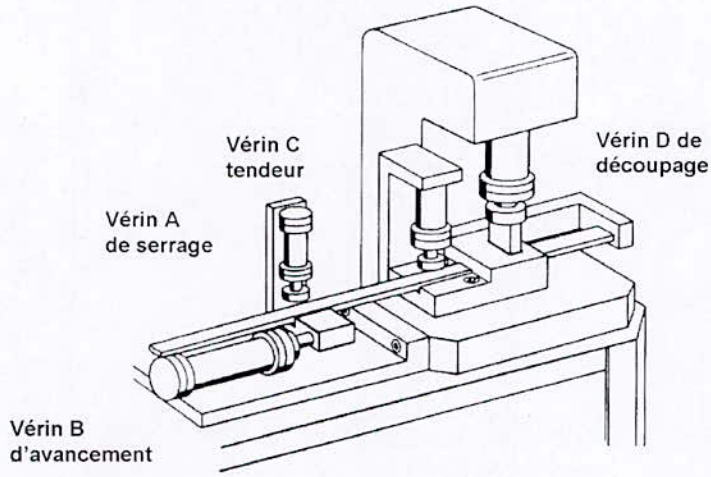


Fig2.16. Figure représentative de une perceuse automatique.

15.2 Mise en forme chronologique

Le piston du vérin A descend et applique une pression,
le piston du vérin B se déplace et décale la barre jusqu'à la butée,
le piston du vérin C descend et tend la barre dans la direction de la perceuse,
le piston du vérin A remonte (la pince de serrage est alors ouverte) et le vérin D descend (Découpage),
le piston du vérin B revient (l'unité de poussée revient à la position initiale) et le piston du vérin D remonte,
le piston du vérin C remonte et relâche la tension.

15.3 Représentation en tableau

Etape	Piston du vérin A	Piston du vérin B	Piston du vérin C	Piston du vérin D
1	Sortie	-	-	-
2	-	Sortie	-	-
3	-	-	Sortie	-
4	Rentrée	-	-	Sortie
5	-	Rentrée	-	Rentrée
6	-	-	Rentrée	-

Tableau 2.3. Représentation de la mise en forme chronologique.

15.4 Représentation concise

Pour le déroulement des actions, il est souvent insignifiant de préciser quelle tâche va avec quel mouvement. Donc un déroulement des actions simplifié peut aussi être employé pour différentes commandes. Lors d'un grand projet le déroulement des actions devrait d'abord être décrit de manière concise, puisque celui-ci donne un aperçu rapide des mouvements.

Dans la représentation concise, à un mouvement est attribué une désignation.

Convention pour la sortie du piston du vérin : +

Convention pour la rentrée du piston du vérin : -

Pour les moteurs, M+ désigne le moteur tournant dans le sens conventionnel, M- dans le sens inverse et M* l'arrêt. Les mouvements parallèles vont être décrits en représentation concise les uns au-dessous des autres. Voilà, pour notre exemple, la manière de représenter : **A- B- C- D-, A+ B+ C+ D+**

15.5 Diagramme d'état

15.5.1 Diagramme pas à pas

Ici le déroulement d'une action va être représenté. Il est dépendant à chaque étape (Changement d'état de n'importe quelle unité de travail) du chemin suivi.

S'il s'agit d'une commande avec plusieurs éléments de travail, alors ceux-ci doivent être représentés de la même manière et les uns sous les autres. La liaison entre ces éléments est assurée par les étapes.

Avec un diagramme pas à pas la distance entre chaque étape est toujours la même. De plus, les lignes de signaux peuvent être reportées dans le diagramme pas à pas.

Voilà, pour notre exemple, la manière de représenter le diagramme pas à pas.

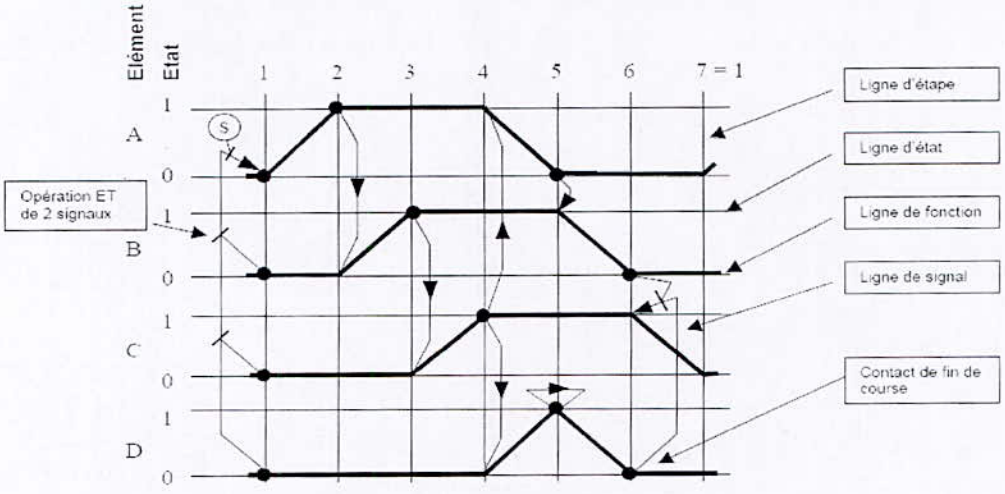


Fig2.17. Représentation du diagramme pas à pas.

15.5.2 Diagramme temporel

Le diagramme temporel est identique au diagramme pas à pas mais en rajoutant les aspects temporels des mouvements. On trouve sur la bordure inférieure du diagramme la durée des actions. La distance de la ligne d'étape change suivant la durée d'exécution nécessaire. En revanche, le nombre d'étape et la catégorie des actions restent inchangés.

16. ETUDE DE LA PERCEUSE AUTOMATIQUE

Afin d'étudier la perceuse automatique, on doit d'abord assigner les variables conditionnelles et les éléments de travail du Graph-7.

16.1 Variables conditionnelles

S0 Bouton de démarrage

S1 a0 Capteur de fin de course : Fin de la sortie du piston du vérin A

S2 a1 Capteur de fin de course : Fin de la rentrée du piston du vérin A

S3 b0 Capteur de fin de course : Fin de la sortie du piston du vérin B

S4 b1 Capteur de fin de course : Fin de la rentrée du piston du vérin B

S5 c0 Capteur de fin de course : Fin de la sortie du piston du vérin C

S6 c1 Capteur de fin de course : Fin de la rentrée du piston du vérin C

S7 d0 Capteur de fin de course : Fin de la sortie du piston du vérin D

S8 d1 Capteur de fin de course : Fin de la rentrée du piston du vérin D

16.2 Attribution des actions

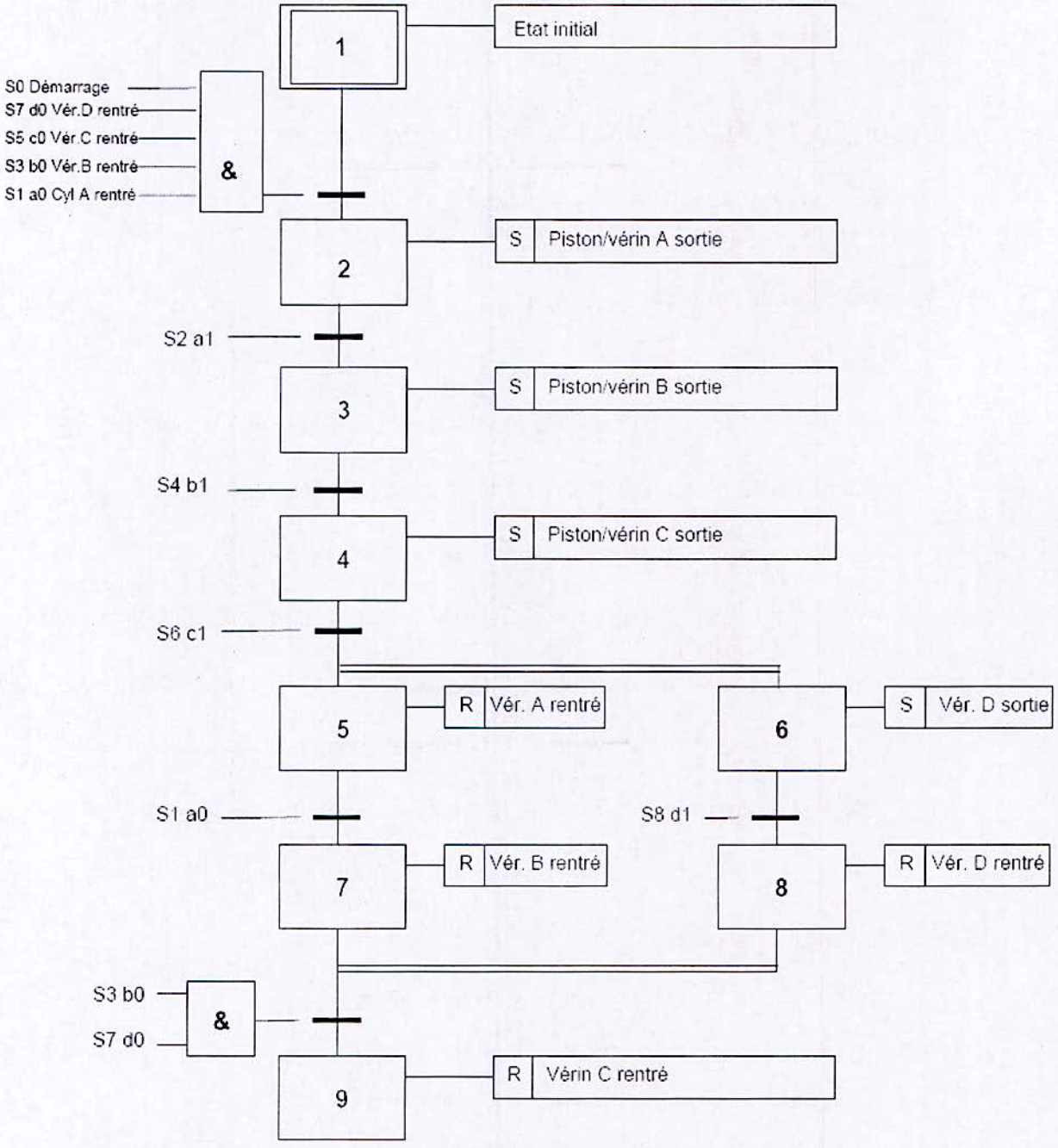
Y1 Action : Sortie/rentrée du piston du vérin A

Y2 Action : Sortie/rentrée du piston du vérin B

Y3 Action : Sortie/rentrée du piston du vérin C

Y4 Action : Sortie/rentrée du piston du vérin D

Grafcet de la perceuse automatique



17. CREATION D'UN PROGRAMME S7-GRAPH

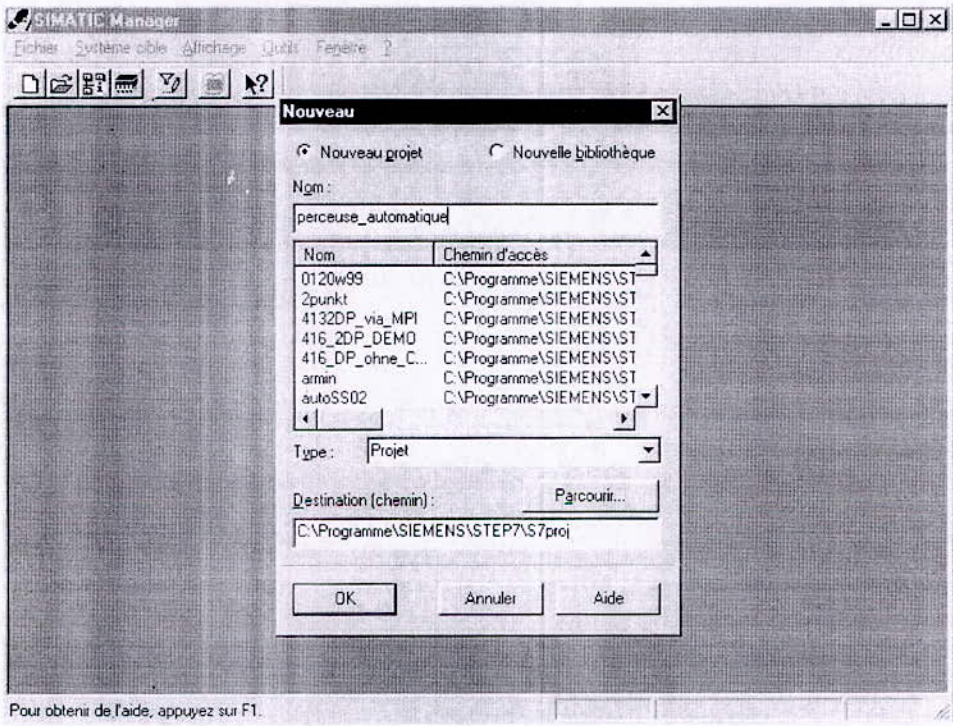
A partir du Graph-7, on peut maintenant créer un programme S7-GRAPH exécutable.

17.1 Démarrer SIMATIC- Manager et créer un nouveau projet

1. Cliquer sur l'icône **Nouveau**.

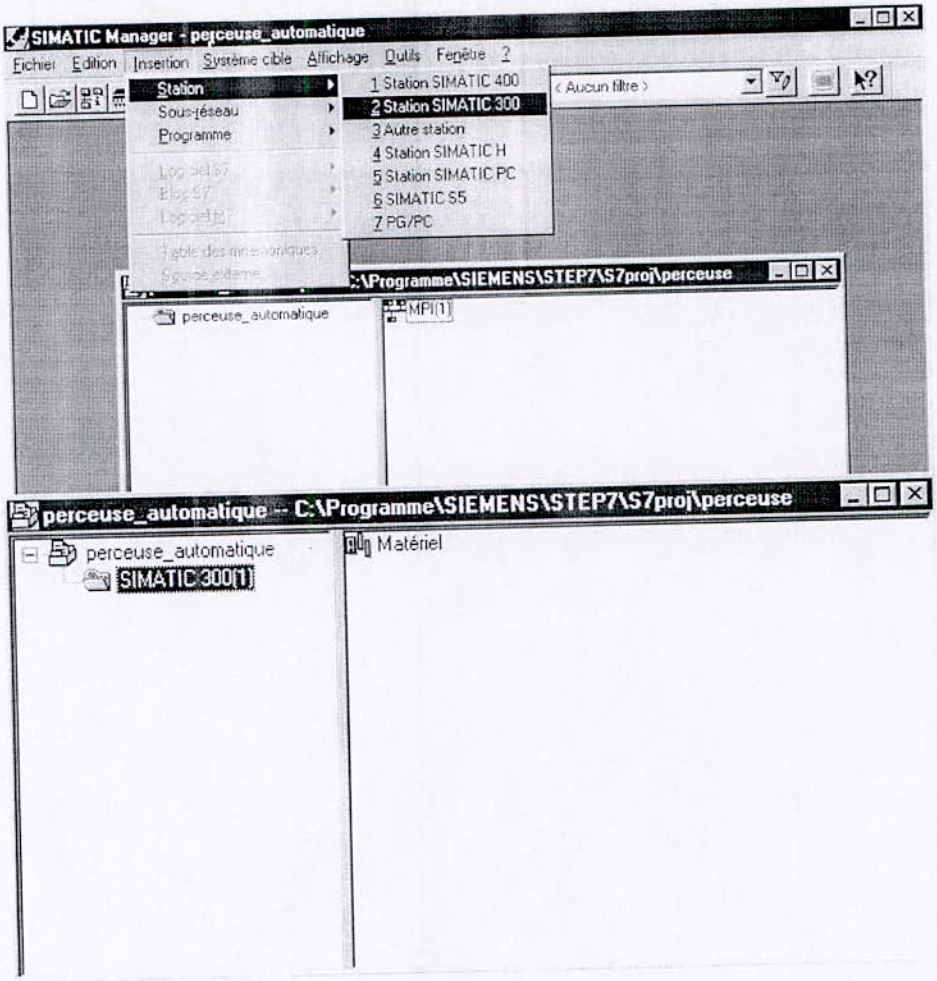
2. Saisir le nom du projet.

3. Cliquer sur **OK**.



17.2 Insérer la station SIMATIC 300 et ouvrir la configuration matérielle

1. Sélectionner le projet **perceuse_automatique**.
2. Cliquer sur **Insertion**.
3. Choisir **Station**.
4. Cliquer sur **Station SIMATIC 300**.
5. Choisir **SIMATIC 300(1)**.
6. Double-cliquer sur **Matériel**.



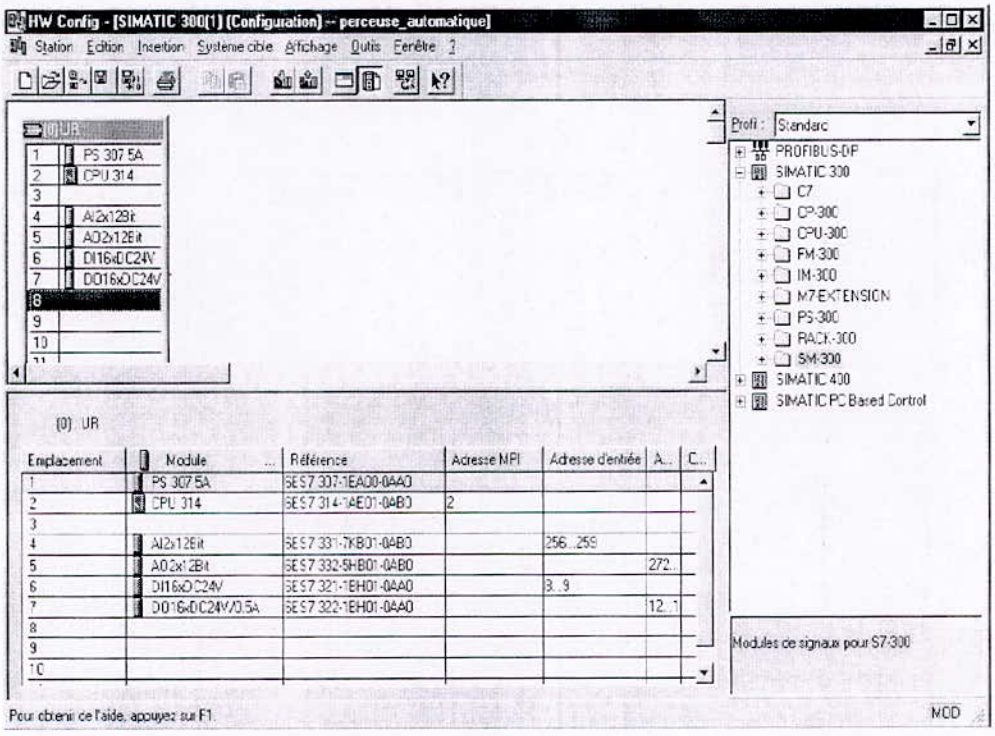
17.3 Configurer le matériel et transmettre à l'automate

1. Entrer les composants de l'installation.

2. **Enregistrer** et **Compiler**.

3. **Charger dans le module** le matériel.

4. Fermer la fenêtre.



Remarque

La configuration présentée ici est un exemple.
A chaque automate correspond une configuration matérielle propre.

17.4 Créer la table des mnémoniques

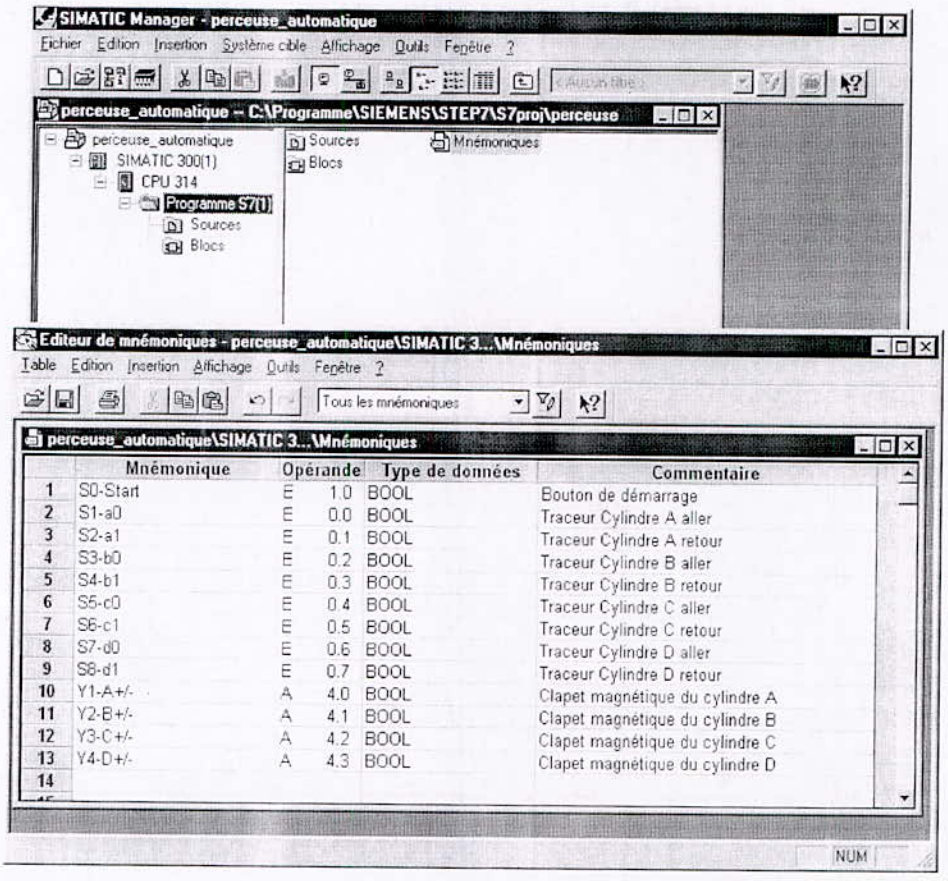
1. Développer l'arborescence et cliquer sur **Programme S7 (1)**.

2. Double-cliquer sur **Mnémoniques**.

3. Saisir la table des mnémoniques.

4. Enregistrer la table des mnémoniques.

5. Fermer la fenêtre.



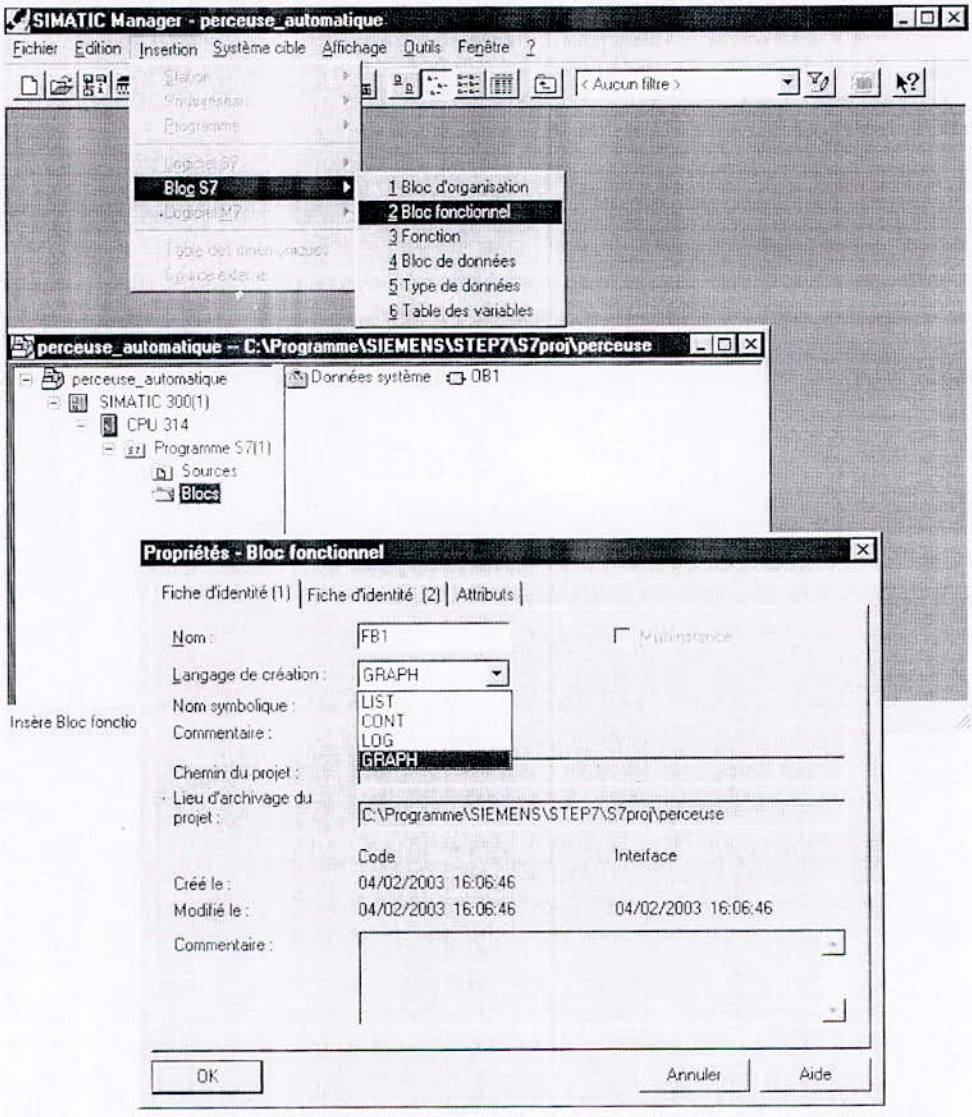
	Mnémonique	Opérande	Type de données	Commentaire
1	S0-Start	E 1.0	BOOL	Bouton de démarrage
2	S1-a0	E 0.0	BOOL	Traceur Cylindre A aller
3	S2-a1	E 0.1	BOOL	Traceur Cylindre A retour
4	S3-b0	E 0.2	BOOL	Traceur Cylindre B aller
5	S4-b1	E 0.3	BOOL	Traceur Cylindre B retour
6	S5-c0	E 0.4	BOOL	Traceur Cylindre C aller
7	S6-c1	E 0.5	BOOL	Traceur Cylindre C retour
8	S7-d0	E 0.6	BOOL	Traceur Cylindre D aller
9	S8-d1	E 0.7	BOOL	Traceur Cylindre D retour
10	Y1-A+/-	A 4.0	BOOL	Clapet magnétique du cylindre A
11	Y2-B+/-	A 4.1	BOOL	Clapet magnétique du cylindre B
12	Y3-C+/-	A 4.2	BOOL	Clapet magnétique du cylindre C
13	Y4-D+/-	A 4.3	BOOL	Clapet magnétique du cylindre D
14				

Remarque

L'adressage des opérandes dépend de chaque automate.

17.5 Insérer le bloc fonctionnel du S7-GRAPH

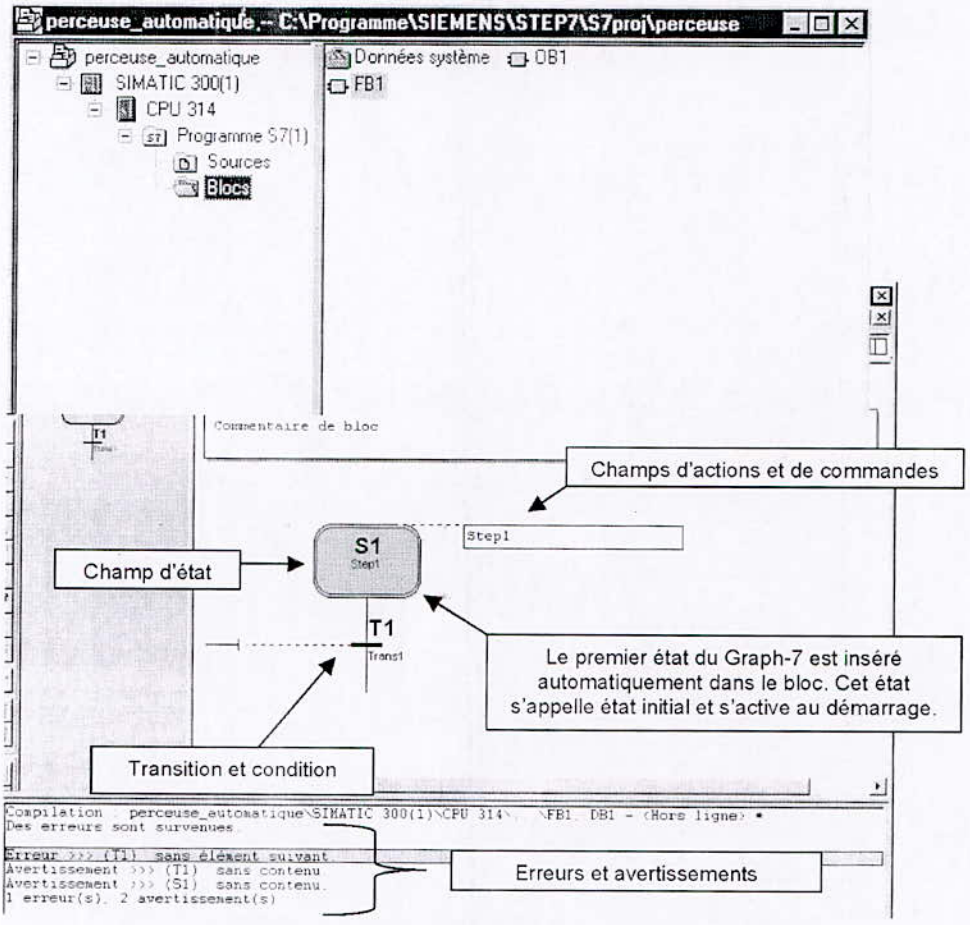
1. Ouvrir l'arborescence et cliquer sur bloc.
2. Cliquer sur **Insertion**.
3. Choisir **Bloc S7**.
4. Cliquer sur **Bloc fonctionnel**.
5. Choisir **GRAPH** en tant que langage de création.
6. Cliquer sur **OK**.



17.6 Ouvrir le S7-GRAPH et saisir l'enchaînement

1. Cliquer sur **Bloc**.

2. Double cliquer sur **FB1. Le programme S7 GRAPH s'ouvre.**



17.7 Le principe du langage Graph-7

Le langage Graph-7 est constitué par une suite d'états qui sont activés selon des conditions de transitions dans un ordre défini.

Le démarrage du Graph-7 est toujours constitué d'un ou plusieurs états initiaux qui se trouvent à des emplacements quelconques. Tant que l'action d'un état est en cours, cet état est actif. Quand plusieurs états s'exécutent simultanément, toutes ces étapes sont actives.

Un état est quitté, si tous les évènements sont exécutés, et si la condition de transition pour passer à l'étape suivante est remplie.

Le prochain état, qui suit cette transition, devient actif.

A la fin du Graph-7, il y a un saut vers une étape quelconque de ce Graph-7 ou vers un autre Graph-7 FB. C'est pour cette raison que le déroulement du Graph-7 peut être cyclique. A la fin du Graph-7 on peut aussi mettre une étape finale ou puits. Le déroulement se termine alors par cet état final.

17.8 Etape active

Une étape active est une étape, dont les actions sont en cours d'exécution.
Une étape est active

- si la condition de la transition en amont est remplie ou
- s'il s'agit de l'étape initiale au démarrage du Graph-7 ou
- si elle est appelée par une autre action active.

17.9 Elément du Graph-7

	Etape et transition		Ouvrir Branche ET
	Insérer un saut		Fermer Branche ET
	Puits (fin de graphe)		Ouvrir Branche OU
	Insertion d'étapes		Fermer Branche OU

17.10 Créer le Graph-7 selon l'enchaînement choisi

17.10.1 Première étape

1. Double-cliquer et saisir le **commentaire de bloc** et la **désignation de l'étape**

2. Cliquer sur l'**entrée de la transition**.

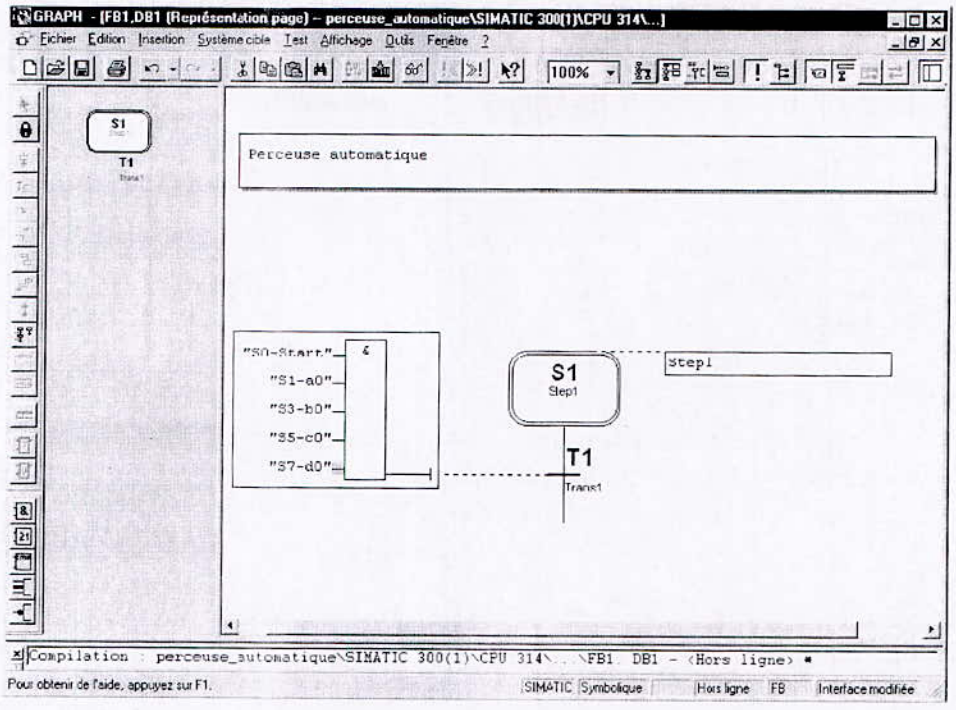
3. Insérer une **boîte ET**.

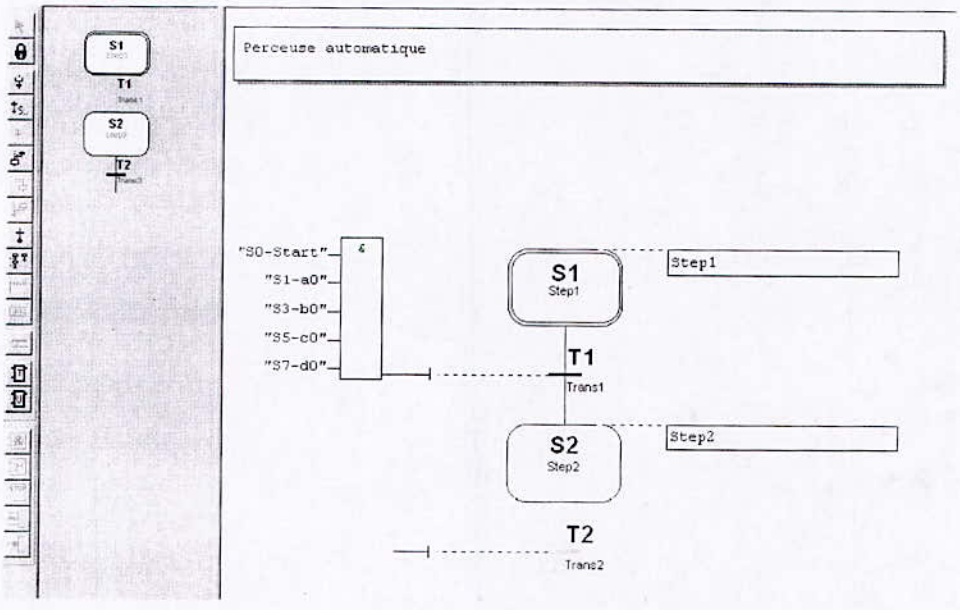
4. Ajouter des **entrées binaires**.

5. Saisir les opérandes de la boîte ET.

6. Cliquer sur la **transition T1**.

7. Insérer une **étape + transition**. **La deuxième étape est insérée.**





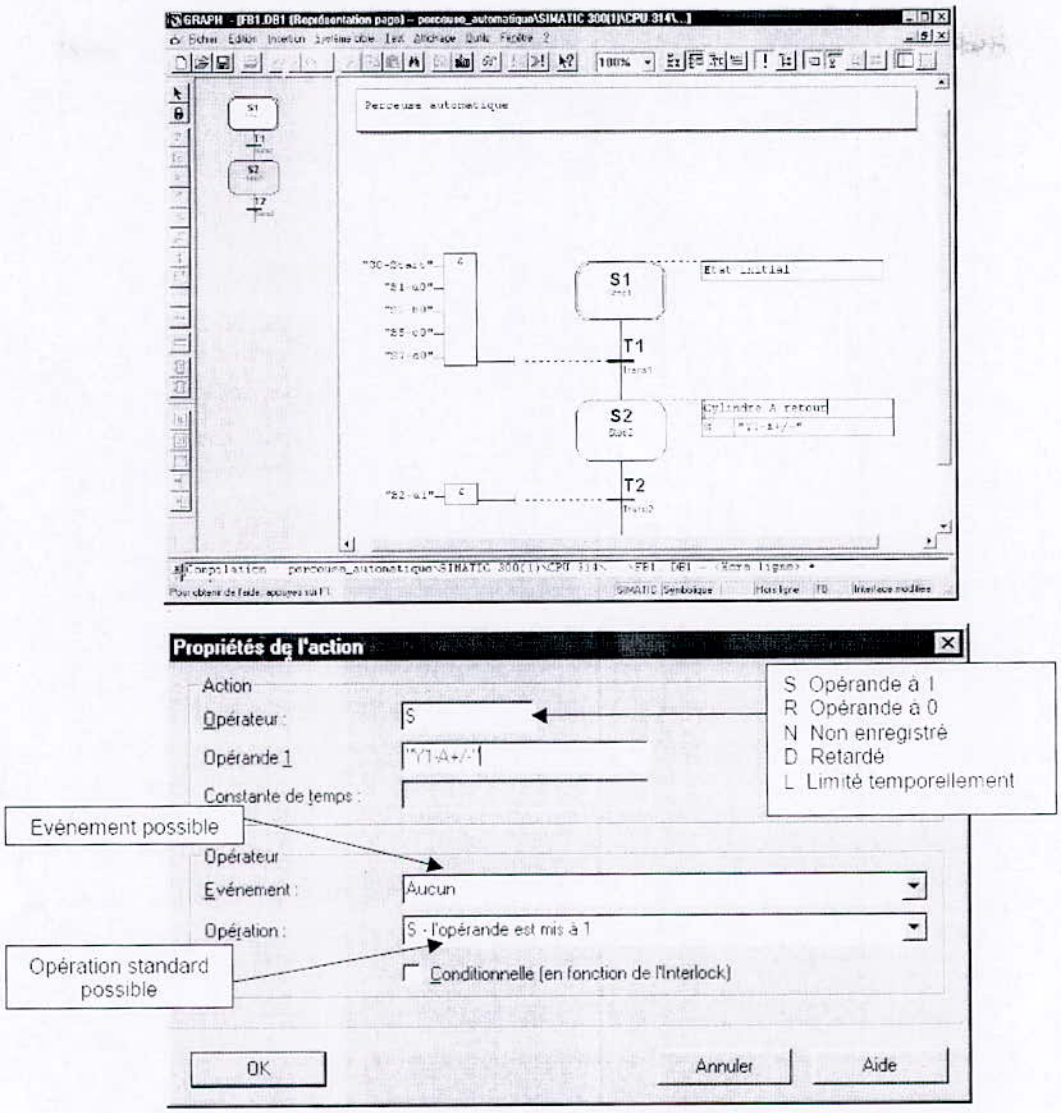
17.10.2 Deuxième étape

1. Cliquer sur le **commentaire** et saisir le nom de l'étape.

2. Insérer une **action**.

3. Saisir les actions à exécuter en double-cliquant ou avec les bouton droit de la souris en choisissant **propriétés de l'objet**.

4. Saisir la transition.



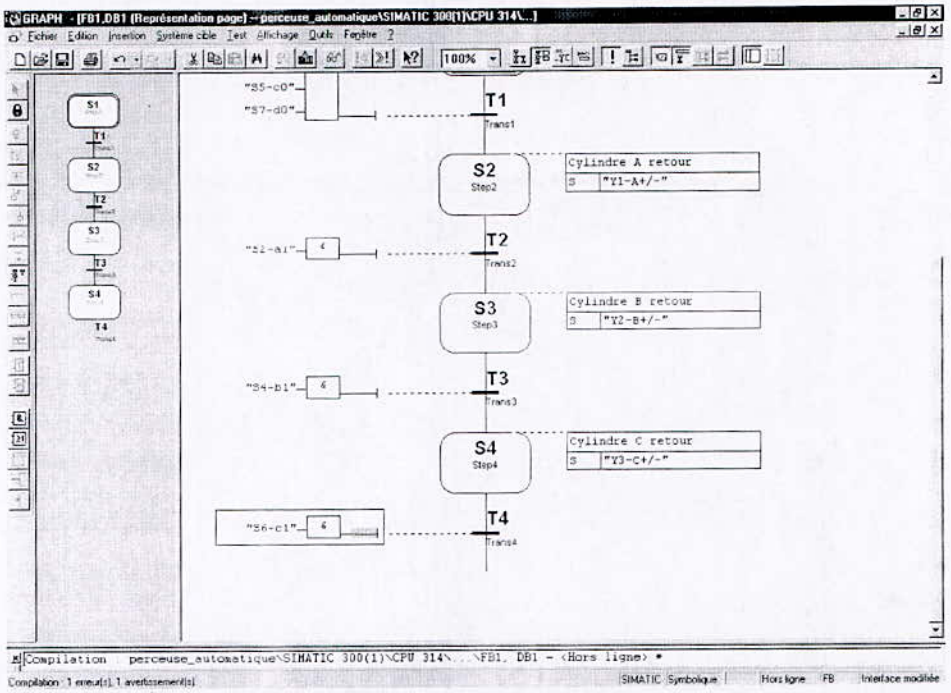
17.10.3 Troisième et quatrième étapes

1. Cliquer sur la **transition T2**.

2. Cliquer deux fois sur **étape + transition** pour insérer l'étape S3 et S4.

3. Entrer les commentaires et les actions.

4. Entrer les transitions.



Les étapes suivantes vont être insérées avec un branchement.

Il existe deux catégories de branchements

La branche OU,  va être insérée après l'étape choisie et commence avec une transition. Toutes les étapes de la branche OU vont être traitées seulement si la condition de transition est
remplie.
Le branchement peut se terminer soit par une transition à gauche, soit par un puits.

 Terminer la branche OU.

 Insérer un puits.

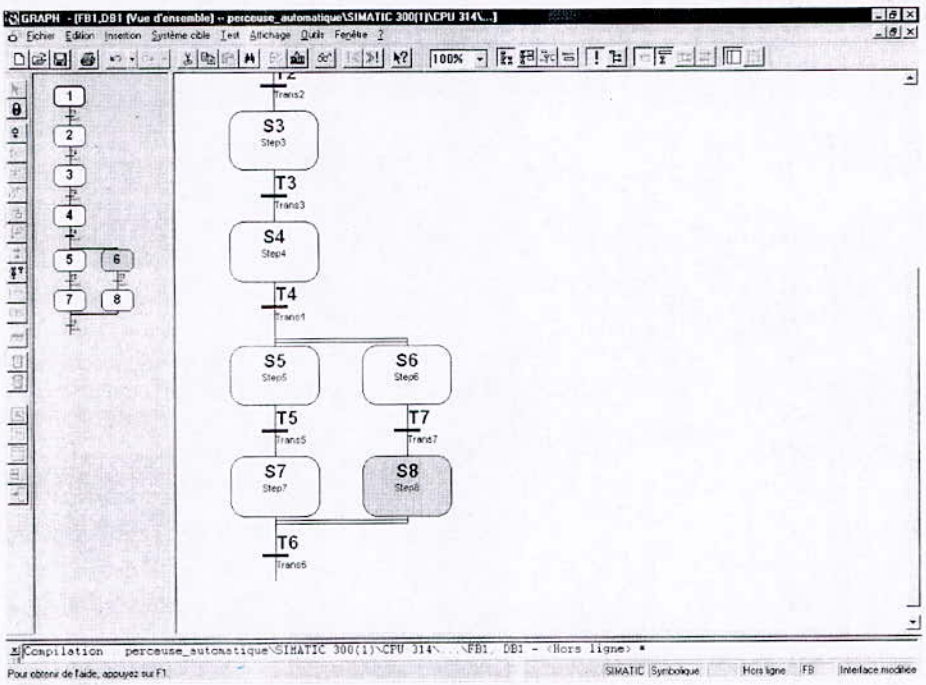
 La branche ET, va être insérée après la transition choisie et commence avec un état. Toutes les étapes de la branche ET vont être traitées puisqu'elles s'exécutent en parallèle du branchement principal.

Le branchement se termine par une étape à gauche.

 Terminer la branche ET.

17.10.4 Insérer un branchement

Pour insérer des branchements, il est préférable de travailler en mode vue d'ensemble.



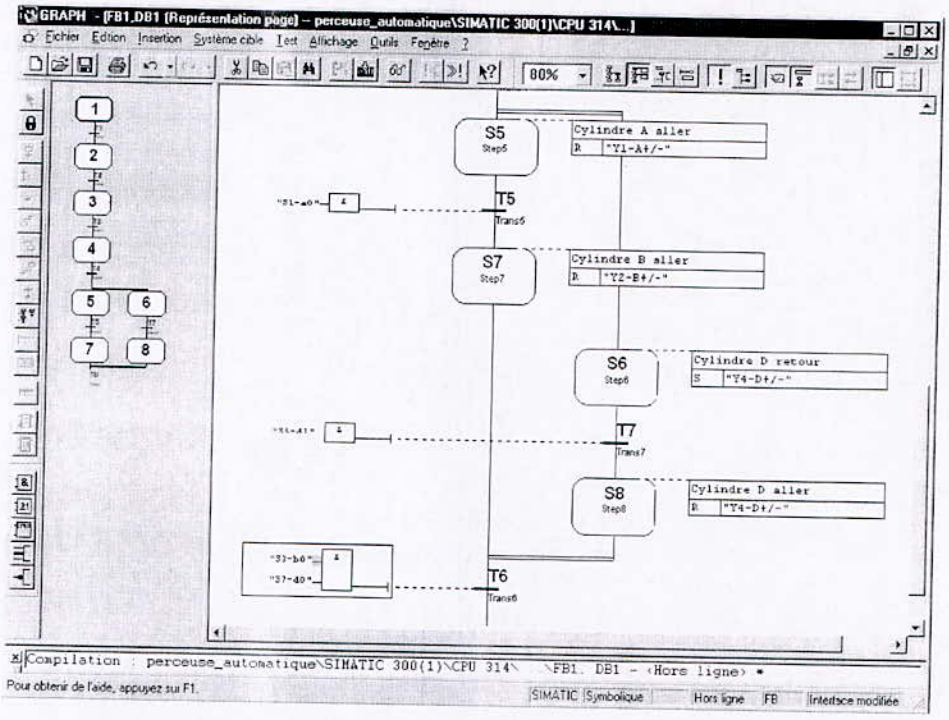
Pour créer une branche ET, procéder comme suit.

1. Cliquer sur la **transition T4**.
2. Insérer **étape + transition** (L'étape S5 et la transition T5 sont insérées).
3. Cliquer sur la **transition T4**.
4. Insérer une **branche ET** (L'étape S6 est insérée).
5. Cliquer sur la **transition T5**.
6. Insérer **étape + transition** (L'étape S7 et la transition T6 sont insérées).
7. Cliquer sur l'**étape S6**.
8. Insérer **étape + transition** (L'étape S8 et la transition T7 sont insérées).
9. Cliquer sur l'**étape S8**..
10. Cliquer sur fermeture de la **branche ET**.
11. Cliquer sur l'**étape S7**.

Pour saisir les actions et les transitions, vous devez vous mettre en représentation page.

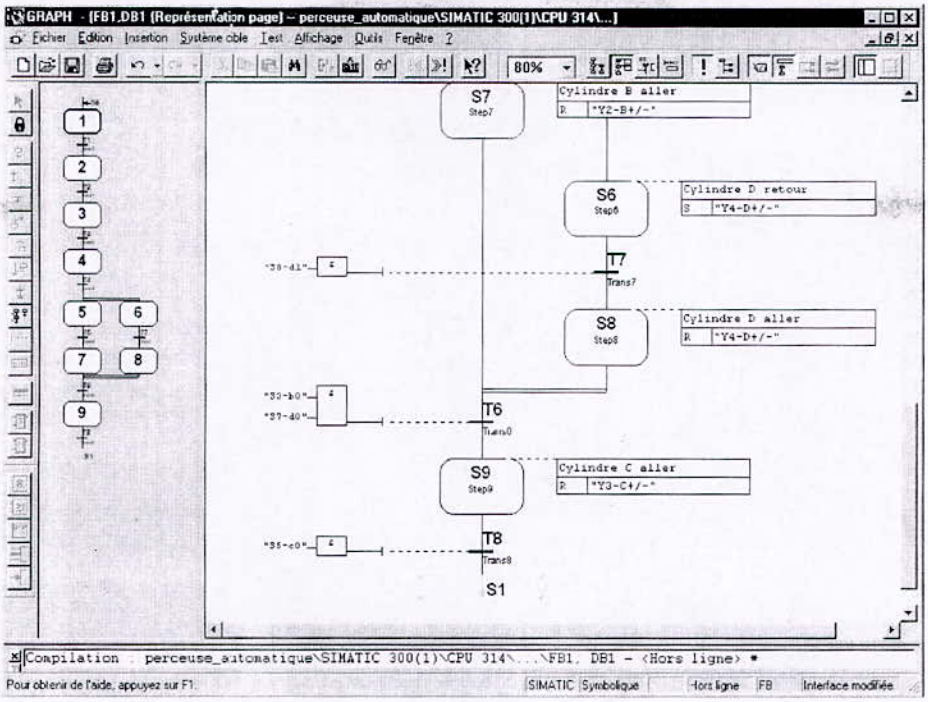
17.10.5 Insérer les actions et les transitions de l'étape 5 à 8 et insérer la dernière étape

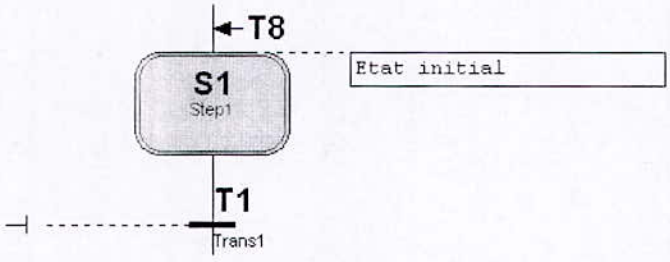
1. Passer en **représentation page**.
2. Saisir les actions.
3. Entrer les transitions.
4. Cliquer sur la **transition T6**.
5. Insérer **étape + transition**.



17.10.6 Traiter la dernière étape et insérer un retour vers l'étape initiale

1. Saisir le commentaire et l'action.
2. Saisir la transition.
3. Cliquer sur la **transition T8.**
4. Insérer un saut.
5. Saisir S1 sur le saut ou cliquer sur l'**étape S1** afin d'insérer automatiquement le saut.

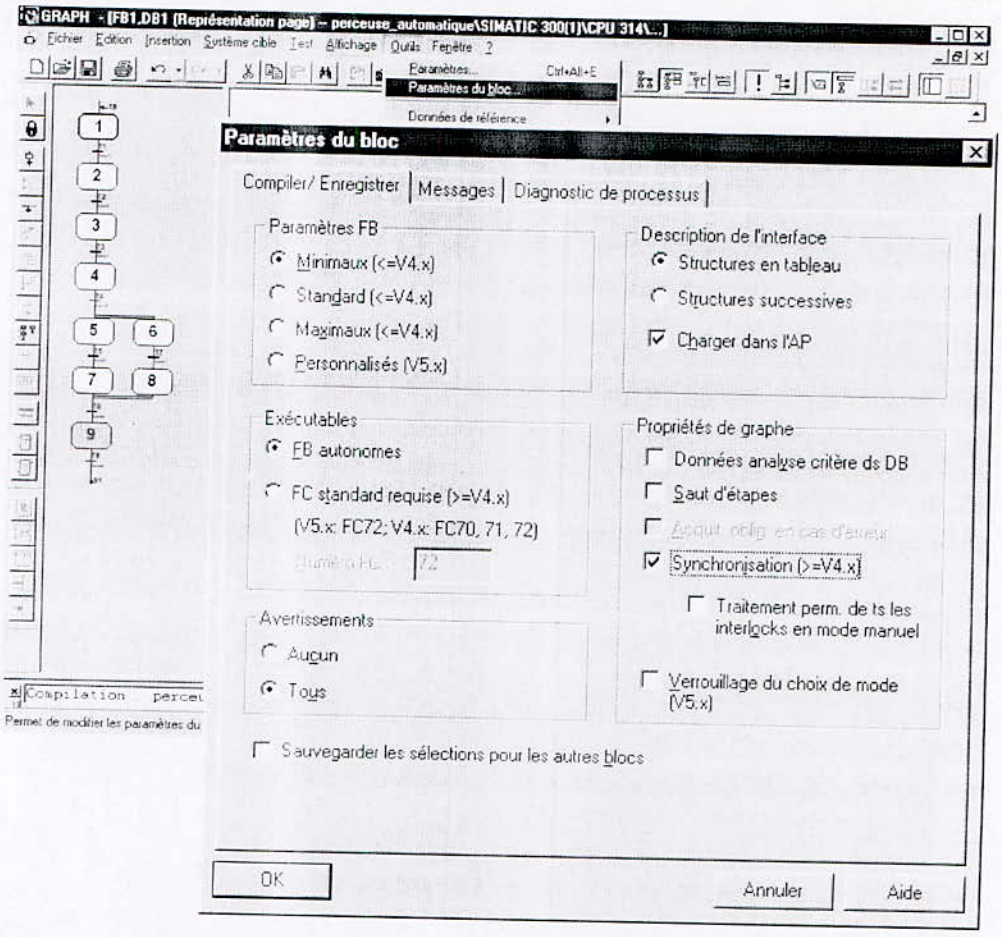




17.10.7 Configurer les paramètres du bloc et enregistrer le bloc fini

Avant l'enregistrement du bloc vous devez modifier ses propriétés.

1. Cliquer sur **Outils**.
2. Cliquer sur **paramètres du bloc**.
3. Mettre les paramètres FB à **Minimaux**.
4. Mettre exécutable à **FB autonome** afin que le FC standard soit relié au bloc fonctionnel.
5. Cocher **Synchronisation**.
6. Cliquer sur **OK**.
7. Enregistrer le bloc.
8. Fermer S7-GRAPH.



Remarque

S'il y a des erreurs dans le bloc alors celui-ci ne peut pas être enregistré. Un bloc erroné peut être généré seulement en tant que source. Avec cette sauvegarde, le bloc de données correspondant et le SFC64 sont produits. Les deux seront copiés dans le répertoire du bloc.

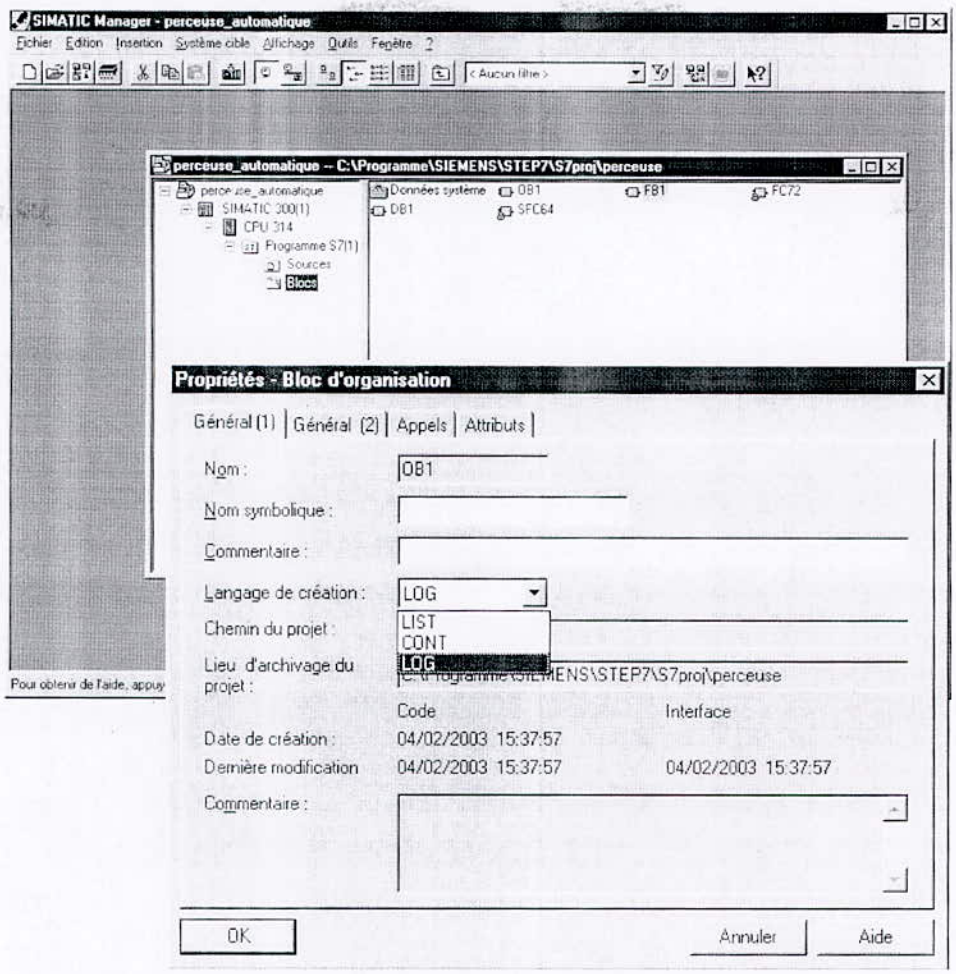
17.11 Ajuster les propriétés du bloc d'organisation et ouvrir OB1

1. Cliquer sur **Blocs**.

2. Double-cliquer sur **OB1**.

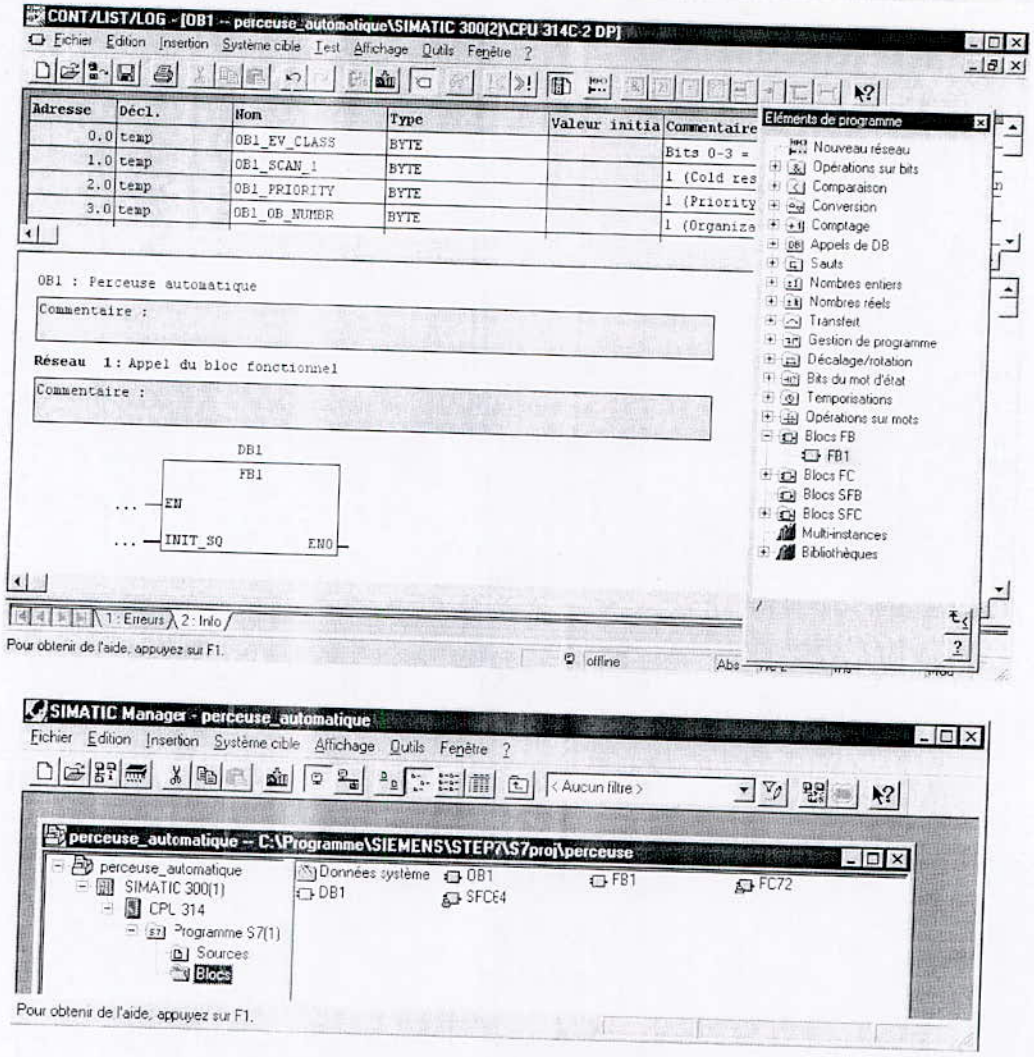
3. Choisir **LOG** en tant que langage de création dans les propriétés du bloc d'organisation.

4. Cliquer sur **OK**.



17.12 Traiter le bloc d'organisation OB1 et charger les blocs dans le module

1. Saisir la rubrique de blocs et la rubrique de réseau.
2. Cliquer dans le champ de saisie.
3. Ouvrir **Eléments de programme**.
4. Insérer **FB1** en double-cliquant.
5. Entrer DB1.
6. Enregistrer le bloc OB1.
7. Fermer CONT/LIST/LOG.
8. Cliquer sur **blocs** et **charger dans le module**.



Après le transfert du bloc dans la CPU, le programme peut être testé.

CONCLUSION

Pour situer l'objectif du présent travail, nous avons commencé par l'historique des API, ensuite nous avons exposé, et d'une façon approfondie, les deux côtés : matériel et logiciel de l'API.

Le chapitre suivant sera consacré au choix de l'API pour le processus de chromage au sein de l'unité BCR.

CHAPITRE -3-

LA CPU S7-314 IFM
ET SES FONCTIONS INTEGREES

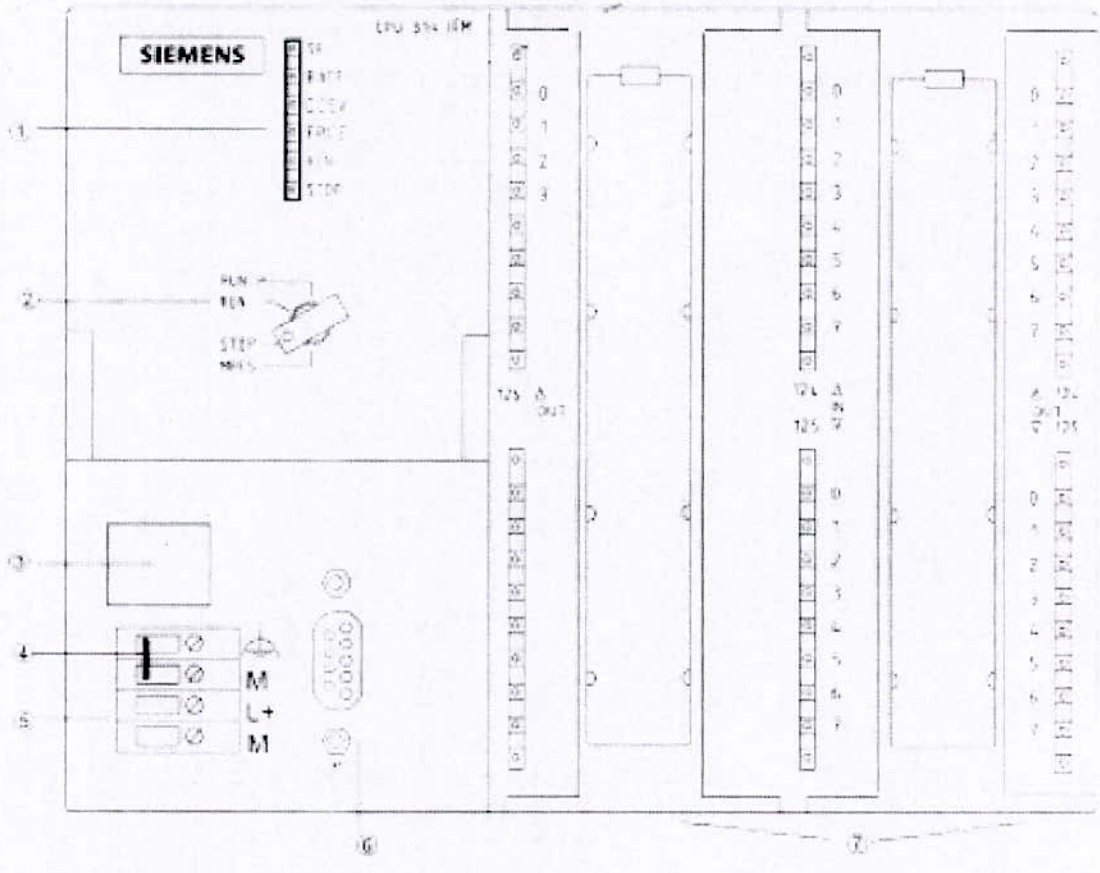
INTRODUCTION

Après avoir décrit le processus et les API d'une manière générale, nous allons au cours de ce chapitre choisir pour notre application un API et motiver ce choix.

Nous allons décrire toutes les fonctionnalités correspondantes à ce type d'API avec tous les détails que nous jugeons utiles.

1. PRESENTATION DE LA CPU

SIEMENS propose plusieurs gammes de produits qui peuvent être utilisés dans se projet. On a opté pour l'utilisation d'un automate programmable appartenant à la série S7-300 (S7-314 IFM). Sa caractéristique principale est l'intégration de modules comportant entre autres des fonctions intégrées.



1 LED de visualisation d'état et de défaut
2 Commutateur de mode de fonctionnement
3 Logement pour pile de sauvegarde ou accumulateur
4 Cavalier (amovible)
5 Bornes pour la tension d'alimentation et la terre fonctionnelle
6 Interface multipoint MPI des CPU
7 Entrées/sorties intégrées

Fig3.1. Vue générale de la CPU S7-314 IFM.

Les CPU de la série S7-300 se composent des éléments suivants :

1.1. LED de visualisation d'état et de défaut :

(Rouge) SF Défaut de matériel ou de logiciel.
(Rouge) BATF Défaillance de la pile.
(Verte) 5V cc L'alimentation 5V cc est correcte .
(Jaune) FRCE Le forçage permanent est actif.
(Verte) RUN CPU en RUN.
(Jaune) STOP CPU en STOP ou ATTENTE ou en démarrage.

Fig3.2. LED de visualisation d'état de la CPU

1.2. Commutateur de mode de fonctionnement :

Le changement de mode se fait à l'aide d'une clé :

Position	Signification	Explication
RUN –P	Mode de fonctionnement RUN-PROGRAM	La CPU traite le programme utilisateur. Le programme peut être modifié. Dans cette position la Clé ne peut pas être retirée.
RUN	Mode de fonctionnement RUN	La CPU traite le programme utilisateur. Le programme ne peut être modifié qu'avec légitimation par mot de passe la Clé peut être retirée.
Position	Signification	Explication
STOP	Mode de fonctionnement	La CPU ne traite aucun programme utilisateur.
	STOP	La Clé peut être retirée.
MRES	Effacement Général	Position instable du commutateur, pour effectuer l'effacement général il faut respecter un ordre particulier de commutation

Tableau 3.1. Le commutateur de mode de fonctionnement de l'API

1.3. Pile de sauvegarde ou accumulateur :

L'utilisation de l'accumulateur ou de la pile de sauvegarde est nécessaire pour l'horloge temps réel.

La pile de sauvegarde est aussi utilisée pour :

- la sauvegarde du programme utilisateur s'il n'est pas enregistré dans la mémoire morte.
- pour étendre la zone rémanente de données.

L'accumulateur est rechargé à chaque mise sous tension de la CPU. Son autonomie est de quelques jours voir quelques semaines, au maximum. La pile de sauvegarde n'est pas rechargeable mais son autonomie peut aller jusqu'à une année.

1.4. Carte mémoire :

La CPU S7-314 IFM utilisée n'est pas dotée d'une carte mémoire mais il faut savoir que la plupart des CPU en possèdent, son rôle est de sauvegarder le programme utilisateur, le système d'exploitation et les paramètres qui déterminent le comportement de la CPU et des modules en cas de coupure de courant.

1.5. Interface MPI : (interface multipoint)

L'interface MPI est l'interface de la CPU utilisée pour la console de programmation (PG), le pupitre opérateur (OP) ou pour la communication au sein d'un sous réseau MPI.

La vitesse de transmission typique est de 187,5 k bauds.

2. CARACTERISTIQUES TECHNIQUES DE LA CPU :

La CPU S7-314 IFM possède des entrés/sorties intégrées, leur câblage se fait par le biais de connecteurs frontaux 20 ou 40 points.

2.1. Tableaux techniques :

Les tableaux ci-dessous résument les principales caractéristiques de la CPU S7-314 IFM

Mémoires	
Mémoire de travail intégrée uniquement	32ko
Mémoire de chargement intégrée	48Ko de RAM 48Ko de FEPROM
Impossibilité d'extension pour la mémoire de travail ainsi que la mémoire de chargement	
Mémentos	
Nombre	2048 bits
Rémanence : réglable par défaut	de MB 0 à MB 143 de MB 0 à MB 15
Mémentos de cadence	Un octet de mémento
Blocs de données	
Nombre	Maximum 127 (DB 0 réservé)
Taille	Maximum 8 Ko
Rémanence : réglable par défaut	Maximum 2 DB, 144 octets de données. Pas de rémanence

Blocs	
Blocs d'organisation (OB)	13
Taille	Maximum 8 Ko
Profondeur d'imbrication : Par classe de priorité Supplémentaire à l'intérieur d'un OB d'erreur	 8 4
Blocs fonctionnels (FB)	128
Taille	Maximum 8 Ko
Fonctions (FC)	128
Taille	Maximum 8Ko
Temporisations /compteurs	
Compteurs S7 Rémanence par défaut Rémanence réglable Plage de comptage	64 Z0 àZ7 Z0 à Z63 0 à 999
Temporisations S7 Rémanence par défaut Rémanence réglable Plage de temps	128 Aucune temporisation rémanente T0 à T7 10 ms à 9990 s

Tableau 3.2. Principales caractéristiques de la CPU S7-314 IFM

Zones d'adresses (entrées/sorties)	
Numériques Numériques intégrées Spéciales	0.0 à 123.7/0.0 à 123.7 124.0 à 125.7/124.0 à 125.7 126.0 à 126.3/124.0 et 124.1
Analogiques Analogiques intégrées	256 à 751/256 à 751 128 à 135/128 à 129
Mémoire image (non réglable)	128 octets/128 octets
Sauvegarde	
Avec pile	Toutes les données
Sans pile	144 octets

Tableau 3.3. Zone de mémoire et de périphérie

Fonction de test et de diagnostic	
Etat/forçage de variables	Oui
Variable	Entrées, sorties, DB, temporisations, compteurs, mémentos.
Nombre Etat de variables Forçage de variables	 Maximum 30 Maximum 14
Forçage permanent Variables Nombre	Oui Entrées, sorties Max 10
Nombre de points d'arrêt	2
Tampon de diagnostic Nombre d'entrées (non réglables)	Oui 100

Tableau 3.4. Fonctions et test de diagnostique.

Interface de communication MPI	
Vitesse de transmission	19.2 , 187.5 k bauds

Tableau 3.5. Interface de communication.

Tensions, Courants	
Tension d'alimentation Plage admissible	24V cc 20,4 à 28,8 V
Consommation (en marche à vide)	Typique 1.0 A

Tableau 3.6. Tensions et courants.

Fonctions intégrées	
Compteur	1 ou 2 selon la configuration utilisateur
Fréquencemètre	Maximum 10 KHz
Positionnement	1 voie

Tableau 3.7. Fonctions intégrées.

3. LES REGISTRES DE LA CPU :

3.1. Le mot d'état :

15	14	.....					8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	RB	BI1	BI0	DEB	DM	OU	ETAT	RLG	/PI

C'est un registre composé de 9 bits qui nous renseignent sur l'état de la CPU à chaque instant.

3.1.1. Première interrogation /PI :

Le fonctionnement de ce bit est le suivant :

- L'état de /PI est interrogé au même moment que l'état de l'opérande en cours,
- Si /PI est à 0, la CPU exécute la séquence comme étant une nouvelle, et met le bit /PI à 1, seul le résultat de l'interrogation de l'opérande sera mémorisé dans le RLG,
- Tant que /PI est à 1, le résultat de l'interrogation de l'opérande en cours est comparé, selon l'opération combinatoire effectuée, à celui mémorisé précédemment dans le RLG ,
- La fin d'une séquence ou une instruction de saut conditionnel remettent le bit /PI à 0.

3.1.2. Le bit du résultat logique RLG :

Il contient le résultat d'une opération combinatoire sur bits, ou le résultat d'une comparaison.

Dans une séquence combinatoire, le résultat d'une interrogation est toujours combiné avec le RLG, suivant la règle booléenne établie, à condition que /PI soit à 1.

Si ce dernier est à 0, c'est le contenu de l'opérande en cours qui lui est affecté.

3.1.3. Le bit d'état ETAT :

Contient la valeur du bit en accès, il est utilisé uniquement pour les opérations combinatoires ayant accès à la mémoire. Pour les opérations n'ayant pas accès aux mémoires, ce bit est à 1 et n'a pas de signification.

3.1.4. Le bit OU :

Ce bit est utilisé lors de l'utilisation de l'opération ET avant OU,

Le RLG d'une séquence interne est transféré vers ce bit, pour pouvoir enregistrer le nouveau résultat dans le bit RLG.

3.1.5. Le bit de débordement DEB :

Il est mis à 1 par une opération arithmétique, une opération de conversion ou une opération de comparaison de nombres à virgule flottante lorsqu'il y a débordement.

3.1.6. Le bit de débordement mémorisé DM :

Il est mis à 1 au même moment que DEB, et il le reste après la correction de l'erreur, il indique donc si une erreur s'est produite dans l'une des opérations exécutées précédemment.

L'opération SPS le remet à 0.

3.1.7. Les bits indicateurs BI1 et BI0 :

Ils nous informent sur les résultats des opérations suivantes, avec ou sans débordement :

- Le résultat d'une opération arithmétique.
- Le résultat d'une opération de comparaison.
- Le résultat d'une opération combinatoire sur mots.
- Les bits décalés par une opération de rotation ou de décalage.

3.1.8. Le bit du résultat binaire RB :

Il constitue un lien entre le traitement d'opérations combinatoires sur bits et sur mots. En effet, il permet d'utiliser le résultat d'une opération sur mots, comme étant un résultat binaire, et l'intégrer à une séquence combinatoire sur bits.

Il correspond aussi à la sortie de validation ENO les FB et les FC.

3.2. Accumulateur1 et Accumulateur 2 :

Registres sur 32 bits, qui permettent de traiter des octets, des mots ou des doubles mots. Ils sont utilisés pour le chargement des opérandes. Le résultat d'une opération se trouve toujours dans l'accumulateur 1.

3.3. Registres d'adresses : AR1 et AR2 :

Deux registres sur 32 bits, renferment les adresses des opérandes en cours d'utilisation.

3.4. Pile des parenthèses :

Octet de mémoire utilisé pour les combinaisons d'expressions entre parenthèses, on peut avoir jusqu'à 7 niveaux de parenthèses, appelés « entrées », chaque entrée englobe les bits du mot d'état suivants ; RLG, RB, OU.

L'opération fermer parenthèse ")" ferme l'expression entre parenthèses et extrait une entrée de la pile, puis définit le nouveau RLG qui est le résultat de la combinaison du RLG en cours avec celui mis dans la pile des parenthèses.

4. LES FONCTION INTEGREES DU S7-314 IFM :

Les automates programmables S7-314 IFM, ont un module intégré qui comporte plusieurs fonctions intéressantes. Ces fonctions permettent de contrôler des systèmes à des coûts très appréciables. Les autres solutions possibles pour gérer une tâche d'automatisation sont :

- Programmer des blocs pour la tâche.
- Mettre en place des modules spécialisés.

Le tableau suivant montre les 3 possibilités citées ainsi que les différents critères de sélection :

Critère de sélection	**Programme utilisateur**	**Fonctions intégrées**	**Modules de fonction**
Liaison directe avec les E/S	Non	Oui	Oui
Impact sur le temps de cycle	Oui	Minime	Non
Taux de couverture des applications	Faible	Moyen (50%)	Elevé (95%)
performance au niveau du temps de réaction	Faible	Moyen	Elevé
Traitement des défauts de processus	Non	Limité	Oui

Tableau 3.8. Critères de sélections pour la résolution d'un problème d'automatisation à base d'automates programmables.

Les fonctions intégrées à la CPU S7-314 IFM sont :

- Fonction fréquencemètre.
- Fonction compteur (1 compteur pour comptage/décomptage)
- Fonction compteur A/B (2 compteurs A et B pour comptage/décomptage).
- Fonction positionnement.

Les modules spécialisés offrent un ajout sur temps de cycle nul, mais il faut noter que le prix d'une solution avec des modules spécialisés est nettement supérieur à celui d'une solution avec CPU dotée de fonctions intégrées. Il est donc préférable d'utiliser une CPU IFM, si les performances d'un module spécialisé ne sont pas nécessaires.

Il existe 5 entrées spéciales sur une CPU S7-314 IFM. Les configurations possibles sont :
- 4 entrées pour les alarmes processus. (Entrées TOR)
- 4 entrées TOR pour la fonction intégrée compteur.

- 4 entrées TOR pour la fonction intégrée compteurs A/B.
- 1 entrée TOR pour la fonction intégrée fréquencemètre et 3 entrées TOR standard.
- 3 entrées TOR pour la fonction intégrée positionnement et 1 entrée TOR standard.

Les entrées spéciales qui ne sont pas utilisées pour une fonction intégrée ou pour une alarme de processus peuvent être utilisées comme entrées TOR standard.

Remarques :

La fréquence maximum tolérée par ces fonctions intégrées est de 10kHz. Lorsque cette valeur est dépassée :

- Le fonctionnement correct de la CPU n'est plus garanti.
- Le temps de cycle est allongé. (Sachant que le dépassement du temps de cycle paramétré pour le chien de garde fait passer la CPU en mode STOP)
- Le temps de réaction à l'alarme processus est allongé.
- La communication peut être perturbée et même coupée.

Toutes les fonctions peuvent déclencher des alarmes processus. Ces alarmes appellent l'OB40.

Activation et paramétrage des fonctions intégrées :

L'activation des fonctions intégrées se fait par le biais de STEP 7.

Pour choisir un mode de fonctionnement, il faut ouvrir HW CONFIG de STEP 7. Après avoir effectué la configuration matérielle, il faut ouvrir les propriétés de la CPU, puis dans l'onglet *fonction intégrée* choisir la fonction à activer.

Un bouton *paramètre* permet une configuration plus approfondie des fonctions intégrées.

Dans les paragraphes suivants, les configurations des fonctions intégrées seront toujours relatives aux fenêtres citées précédemment.

4.1. Fonction intégrée fréquencemètre :

4.1.1. Description de la fonction intégrée fréquencemètre :

L'entrée pour la fonction fréquencemètre est à l'adresse E126.0

Cette fonction permet de mesurer une fréquence allant jusqu'à 10kHz.

La fréquence est donnée en comptant le nombre de fronts montants du signal de mesure sur une durée donnée.

Il existe 2 principes de mesure :
- Principe 1 : utilisé pour des durées de mesure de 0.1s, 1s ou 10s. La fréquence est donnée selon la formule suivante :

$$\text{Fréquence} = \frac{\text{nombre de fronts montants}}{\text{durée de mesure}}$$

- Principe 2 : utilisé pour des durées d'actualisation de 1ms, 2ms ou 4ms. La fréquence est donnée en calculant le temps écoulé entre deux fronts montants sur l'entrée de mesure.

La durée de mesure est configurable avec STEP 7.

Une nouvelle mesure est lancée dès qu'un résultat est trouvé, ainsi la fréquence est actualisée en permanence.

La fonction intégrée fréquencemètre comporte deux comparateurs qui permettent de vérifier que la plage de travail admise est bien respectée :

- Le comparateur de limite supérieure réagit dès que la fréquence dépasse un seuil U_LIMIT. Dans ce cas, le bit STATUS_U du SFB 30 se met à 1.

- Le comparateur de limite inférieure réagit dès que la fréquence est en dessous d'un seuil L_LIMIT. Dans ce cas, le bit STATUS_L du SFB 30 se met à 1.
- Le dépassement vers le haut ou vers le bas d'une limite peut, si la configuration à été faite, déclencher une alarme de processus. (la gestion des alarmes sera décrite plus tard)

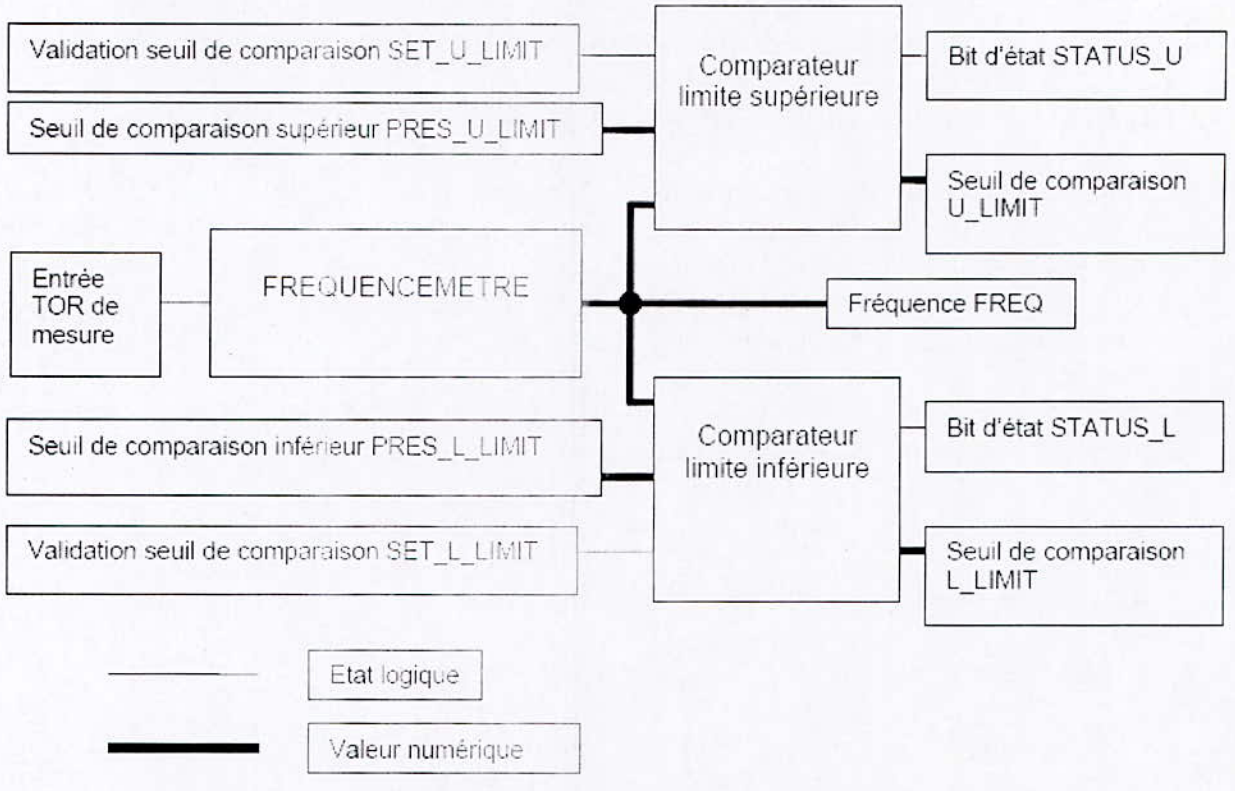


Fig3.3. Représentation synoptique de la fonction intégrée Fréquencemètre.

4.1.2. Description du bloc fonctionnel système SFB 30 :

Le bloc affecté à la fonction intégrée fréquencemètre est le SFB 30.
Sa représentation en CONT est la suivante :

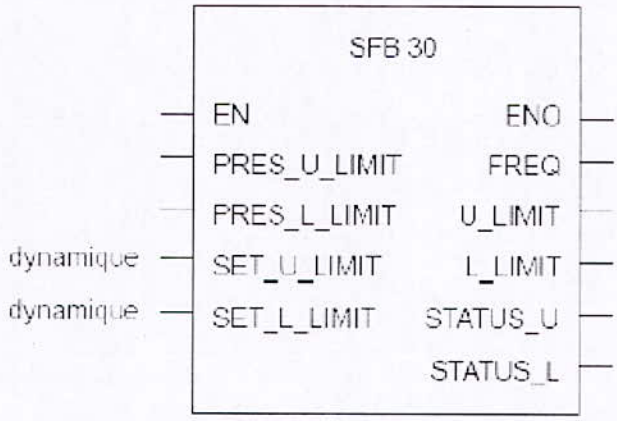


Fig3.4. représentation du SFB 30 avec CONT.

Les tableaux suivants décrivent les paramètres d'entrée et de sortie du bloc :

Paramètre d'entrée	**Description**
EN	Entrée d'activation du bloc (valeur booléenne) **Type de donnée : BOOL**
PRES_U_LIMIT	Ce paramètre permet de définir le nouveau seuil de comparaison PRES_U_LIMIT qui sera validé par un front montant sur le paramètre SET_U_LIMIT. **Type de donnée : DINT**
PRES_L_LIMIT	Ce paramètre permet de définir le nouveau seuil de comparaison PRES_L_LIMIT qui sera validé par un front montant sur le paramètre SET_L_LIMIT. **Type de donnée : DINT**
SET_U_LIMIT	Valide le seuil de comparaison PRES_U_LIMIT s'il y a un front montant. Au même moment, le bit d'état STATUS_U est mis à 0 ou à 1, selon le nouveau seuil de comparaison. **Type de donnée : BOOL**
SET_L_LIMIT	Valide le seuil de comparaison PRES_L_LIMIT s'il y a un front montant. Au même moment, le bit d'état STATUS_L est mis à 0 ou à 1, selon le nouveau seuil de comparaison. **Type de donnée : BOOL**

Tableau 3.9. Paramètres d'entrée du SFB 30.

Paramètre de sortie	Description
ENO	Est validée à 1 si aucun défaut de traitement n'a été détecté sur le bloc SFB30. Si ENO = 0, le SFB n'a pas été traité ou a été traité de façon insatisfaisante. **Type de donnée : BOOL**
FREQ	Donne la fréquence mesurée en mHz. **Type de donnée : DINT**
U_LIMIT	Donne le seuil de comparaison supérieur utilisé pour le cycle en cours. **Type de donnée : DINT**
L_LIMIT	Donne le seuil de comparaison inférieur utilisé pour le cycle en cours. **Type de donnée : DINT**
STATUS_U	Se met à 1 si le seuil de comparaison supérieur à été dépassé vers le haut. Dans tous les autres cas il est à 0. **Type de donnée : BOOL**
STATUS_L	Se met à 1 si le seuil de comparaison inférieur a été dépassé vers le bas. Dans tous les autres cas il est à 0. **Type de donnée : BOOL**

Tableau 3.10. Paramètres sorties du SFB 30.

4.1.3. Description du bloc de données associé au SFB30 :

Ce bloc système doit être lié à un DB d'instance. Ce DB est par défaut le DB 62, mais il peut être configuré avec STEP 7.

La structure de ce bloc de données est la suivante :

Opérande	**Mnémonique**	**Signification**
DBD 0	PRES_U_LIMIT	Seuil de comparaison (nouveau) pour limite supérieure
DBD 4	PRES_L_LIMIT	Seuil de comparaison (nouveau) pour limite inférieure
DBX 8.0	SET_U_LIMIT	Validation du seuil de comparaison pour limite sup.
DBX 8.1	SET_L_LIMIT	Validation du seuil de comparaison pour limite inf.
DBD 10	FREQ	Fréquence
DBD 14	U_LIMIT	Seuil de comparaison (actuel) pour limite supérieure
DBD 18	L_LIMIT	Seuil de comparaison (actuel) pour limite inférieure
DBX 22.0	STATUS_U	Bit d'état Limite supérieure
DBX 22.1	STATUS_L	Bit d'état Limite inférieure

Tableau 3.11. Structure du DB 62.

Les données pour la fonction intégrée fréquencemètre occupent 24 octets dans le DB d'instance en commençant par l'adresse 0.

4.2. Fonction intégrée compteur :

4.2.1. Description de la fonction intégrée compteur :

La fonction intégrée compteur permet de faire un comptage et un décomptage à partir de signaux d'une fréquence allant jusqu'à 10kHz.

Ce comptage (décomptage) peut être soit sensible aux fronts montants soit aux fronts descendants en configurant ce paramètre à l'aide de STEP 7.

Les entrées/sorties utilisées pour la fonction intégrée compteur sont :

Adresse	Fonction
E126.0	Entrée TOR comptage
E126.1	Entrée TOR décomptage
E126.2	Entrée TOR sens
E126.3	Entrée TOR Marche/Arrêt
A124.0	Sortie TOR A
A124.1	Sortie TOR B

Tableau 3.12. Entrées/Sorties associées à la fonction intégrées compteur.

L'entrée TOR « sens » permet d'inverser le comptage et le décomptage, si l'entrée E126.2 est à 0 alors l'entrée TOR de comptage effectue une décrémentation et l'entrée TOR de décomptage une incrémentation.

Le schéma de la page suivante décrit l'architecture de la fonction intégrée compteur.

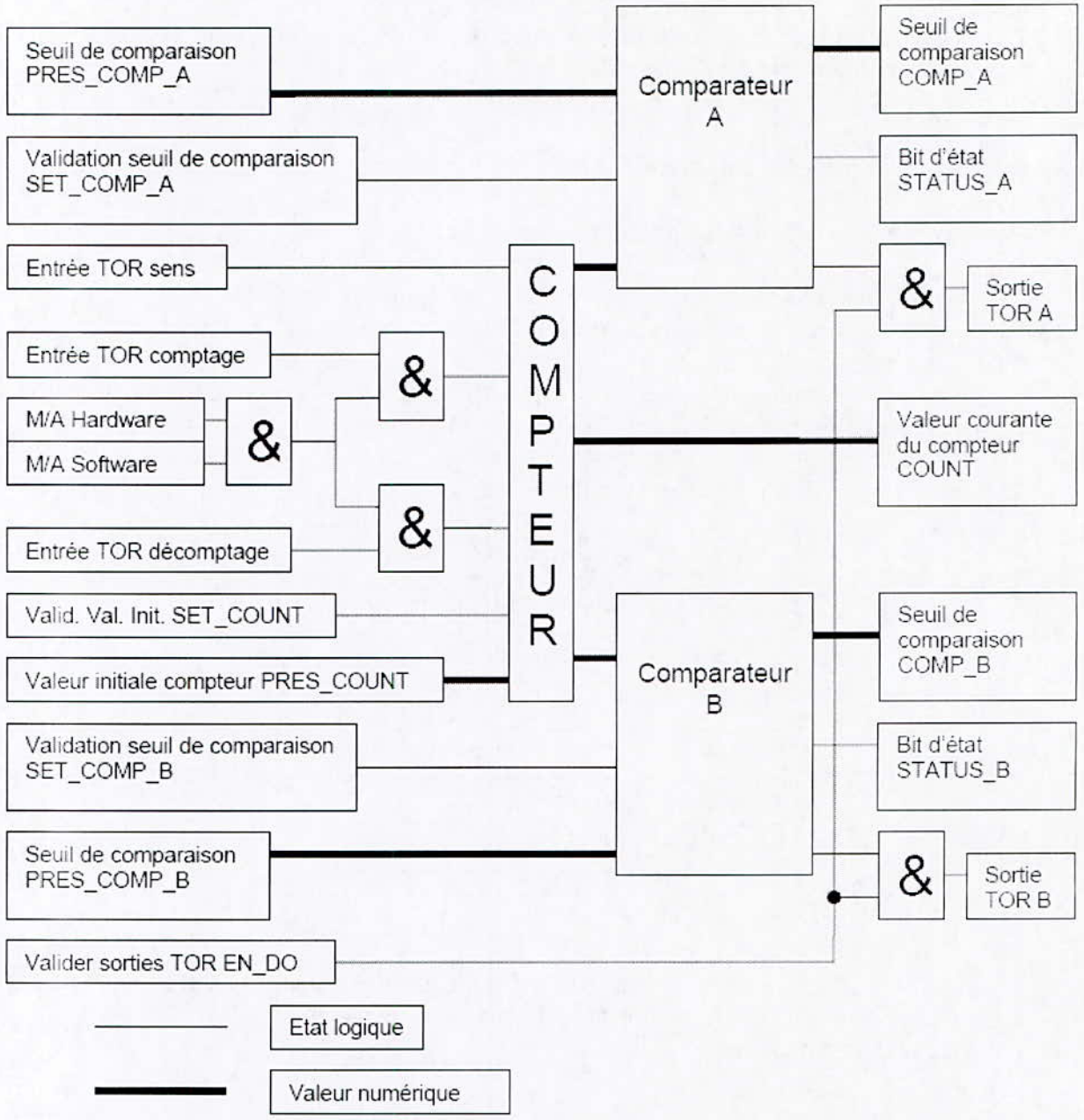


Fig3.5. Représentation synoptique de la fonction intégrée compteur.

4.2.2. Fonctionnement des comparateurs A et B :

Ces deux comparateurs sont configurables. Les bits d'état se mettent à 1 si le compteur indique une valeur supérieure ou égale au seuil de comparaison.

Ainsi la réaction des sorties TOR A (respectivement TOR B) peut être configurée comme suit :

- Aucun changement quel que soit l'état du compteur.
- Mise à 1 si le compteur atteint le seuil de comparaison PRES_COMP_A (respectivement PRES_COMP_B) par le bas.
- Mise à 1 si le compteur dépasse le seuil de comparaison PRES_COMP_A (respectivement PRES_COMP_B) vers le bas.

Une alarme processus ainsi qu'une initialisation du compteur peuvent être activées dans la même fenêtre.

4.2.3. Description du bloc fonctionnel système SFB 29 :

Le bloc fonctionnel affecté à la fonction intégrée compteur est le SFB 29. Sa représentation graphique est la suivante :

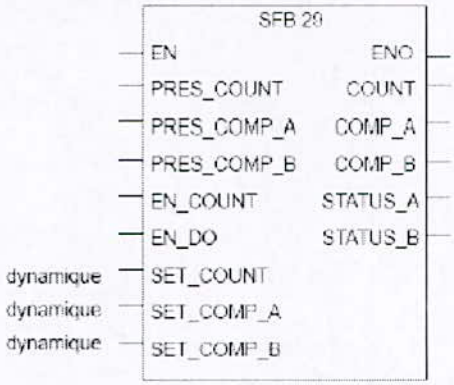


Fig3.6. Représentation du SFB 29 avec CONT.

Les tableaux suivants définissent les entrées et les sorties du bloc SFB 29 :

Entrée	**Fonction**
EN	Entrée d'activation du bloc. **Type de donnée : BOOL**
PRES_COUNT	Permet de définir une valeur initiale de comptage qui sera prise en compte soit lors de la présence d'un front montant sur SET_COUNT soit par un évènement de comptage **Type de donnée : DINT**
PRES_COMP_A	Permet de définir un nouveau seuil de comparaison pour le comparateur A. Ce seuil est pris en compte s'il y a un front montant sur l'entrée SET_COMP_A ou s'il y a un évènement de comptage. **Type de donnée : DINT**
PRES_COMP_B	Permet de définir un nouveau seuil de comparaison pour le comparateur B. Ce seuil est pris en compte s'il y a un front montant sur l'entrée SET_COMP_B ou s'il y a un évènement de comptage. **Type de donnée : DINT**
EN_COUNT	Permet d'activer le compteur. Si ce paramètre est à 0 le compteur ne fonctionne pas. Il faut aussi savoir que ce bit est combiné avec l'entrée matérielle avec un ET logique, ce qui fait que le compteur fonctionne si et seulement si les deux paramètres sont à 1. **Type de donnée : BOOL**
EN_DO	Active les sorties TOR A et B. **Type de donnée : BOOL**
SET_COUNT	Initialise le compteur avec la valeur PRES_COUNT **Type de donnée : BOOL**
SET_COMP_A	Valide le seuil de comparaison du comparateur A s'il y a un front montant. **Type de donnée : BOOL**
SET_COMP_B	Valide le seuil de comparaison du comparateur B s'il y a un front montant. **Type de donnée : BOOL**

Tableau 3.13. Paramètres d'entrée du SFB 29.

Sortie	Fonction
ENO	Se met à 1 si aucun défaut de traitement n'a été constaté dans le bloc. **Type de donnée : BOOL**
COUNT	Fournit la valeur courante du compteur. • Un dépassement de la valeur Max. vers le haut : le compteur continu à partir de la valeur minimale admise. • Un dépassement de la valeur Min. vers le bas : le compteur continu à partir de la valeur maximale admise. **Type de donnée : DINT**
COMP_A	Fournit le seuil de comparaison actuel pour le comparateur A. **Type de donnée : DINT**
COMP_B	Fournit le seuil de comparaison actuel pour le comparateur B. **Type de donnée : DINT**
STATUS_A	Si COUNT ≥COMP_A alors STATUS_A=1 Si COUNT< COMP_A alors STATUS_A=0 **Type de donnée : BOOL**
STATUS_B	Si COUNT ≥COMP_B alors STATUS_B=1 Si COUNT < COMP_B alors STATUS_B=0 **Type de donnée : BOOL**

Tableau 3.14. Paramètres de sortie du SFB 29.

Remarque :
DINT est un nombre entier double, il est donc codé sur 32 bits. Les valeurs admises sont donc de ce qui donne une plage de –2147483648 à 2147483647. 1 2 à 2 31 31 − + −

4.2.4. Description du bloc de données associé au SFB 29 :

Ce bloc système doit être lié à un DB d'instance. Ce DB est par défaut le DB 63, mais il peut être configuré avec STEP 7.

La structure de ce bloc de données est la suivante :

Opérande	Mnémonique	Signification
DBD 0	PRES_COUNT	Valeur initiale du compteur
DBD 4	PRES_COMP_A	Seuil de comparaison COMP_A (nouveau)
DBD 8	PRES_COMP_B	Seuil de comparaison COMP_B (nouveau)
DBX 12.0	EN_COUNT	Marche / Arrêt hardware
DBX 12.1	EN_DO	Validation des sorties TOR
DBX 12.2	SET_COUNT	Initialisation du compteur
DBX 12.3	SET_COMP_A	Validation du seuil de comparaison COMP_A
DBX 12.4	SET_COMP_B	Validation du seuil de comparaison COMP_B
DBD 14	COUNT	Valeur courante du compteur
DBD 18	COMP_A	Seuil de comparaison COMP_A (actuel)
DBD 22	COMP_B	Seuil de comparaison COMP_B (actuel)
DBX 26.0	STATUS_A	Bit d'état A
DBX 26.1	STATUS_B	Bit d'état B

Tableau 3.15. Structure du DB 63.

4.3. Fonction intégrée compteur A/B :

4.3.1. Description de la fonction intégrée compteur A/B :

Tableau représentant les broches pour l'utilisation de la fonction intégrée compteur A/B.

Adresse	Fonction
E 126.0	Compteur A : Comptage (comptage/décomptage)
E 126.1	Compteur A : Décomptage (Sens)
E 126.2	Compteur B : Comptage (comptage/décomptage)
E 126.3	Compteur B : Décomptage (Sens)
A 124.0	Sortie TOR Compteur A
A 124.1	Sortie TOR Compteur B

Tableau 3.16. Entrées/Sorties associées à la fonction intégrée Compteur A/B.

Cette fonction comporte en fait deux compteurs A et B indépendants. Chaque compteur comporte deux entrées et une sortie. Il y a deux configurations possibles pour chaque compteur:

- Une entrée pour le comptage et une autre pour le décomptage.
- Une entrée pour le comptage/décomptage et une autre pour le sens.

Les entrées ne sont sensibles qu'aux fronts montants. Contrairement à la fonction compteur précédente, celle-ci ne possède pas d'entrée d'activation matérielle. La seule façon d'activer ces compteurs est l'entrée EN_COUNT.

Le schéma suivant montre la structure de cette fonction :

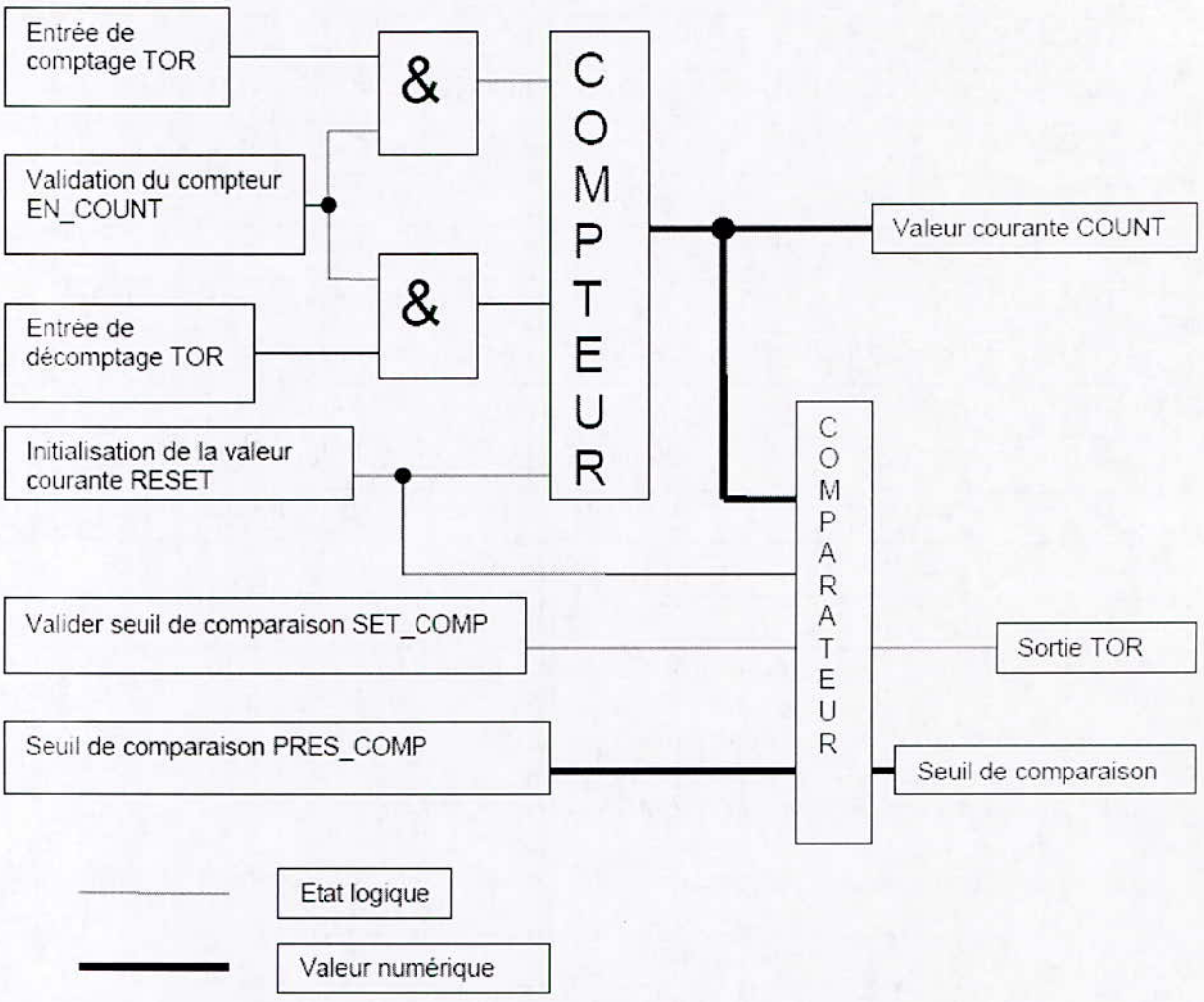


Fig3.5. Représentation synoptique de la fonction intégrée compteur A/B (pour un seul compteur)

4.3.2. Le bloc fonctionnel système associé à cette fonction est le SFB 38 :

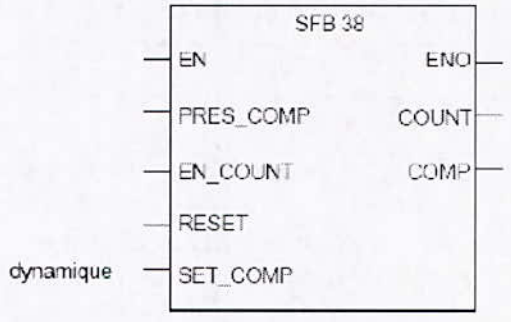


Fig3.6. Représentation du SFB 38 avec CONT.

Les tableaux suivants décrivent les entrées et sorties de ce bloc :

Entrée	Fonction
EN	Entrée d'activation du bloc. **Type de donnée : BOOL**
PRES_COMP	Permet de définir le nouveau seuil de comparaison qui sera pris en compte s'il a un front montant sur l'entrée SET_COMP **Type de donnée : DINT**
EN_COUNT	Permet d'activer le compteur. **Type de donnée : BOOL**
RESET	Le compteur est prêt à travailler si RESET=0 Si RESET=1 : • Le compteur est réinitialisé avec la valeur qui a été paramétrée dans STEP 7. • La sortie TOR associée est mise à 0 est n'est plus soumise à la fonction intégrée. **Type de donnée : BOOL**
SET_COMP	Valide le seuil de comparaison s'il y a un front montant. **Type de donnée : BOOL**

Tableau 3.17. Paramètres d'entrées du SFB 38.

Sortie	Fonction
ENO	Se met à 1 si le bloc a été traité sans erreur **Type de donnée : BOOL**
COUNT	Fournit la valeur courante du compteur. • Un dépassement de la valeur Max. vers le haut : le compteur continu à partir de la valeur minimale admise. • Un dépassement de la valeur Min. vers le bas : le compteur continu à partir de la valeur maximale admise. **Type de donnée : DINT**
COMP	Fournit le seuil de comparaison actuel. **Type de donnée : DINT**

Tableau 3.18. Paramètres de sorties du SFB 38.

Remarque :
Il faut noter que la valeur de RESET du compteur est configurée à l'aide de STEP 7 et non pas dans le SFB 38.

On peut configurer des actions dans le cas d'un évènement compteur. Ces évènements sont :

- Valeur du comparateur atteinte par le bas.
- Valeur du comparateur quittée vers le bas.

Les actions possibles sont :

Paramètre	Etats possibles
Sortie TOR	- Non affectée. - Activée. - Désactivée. - Bascule.
Alarme processus	- Activée. - Désactivée
Initialiser compteur	- Activée. - Désactivée
Initialiser comparateur	- Activée. - Désactivée

Tableau 3.19. Actions possibles en cas d'un évènement compteur.

4.3.3. Description des blocs de données associés au SFB 38 :

Les DB d'instance des compteurs sont par défaut :

- DB 60 pour le compteur A.
- DB 61 pour le compteur B.

La structure de ces DB d'instance est :

Opérande	Mnémonique	Signification
DBD 0	PRES_COMP	Seuil de comparaison (nouveau)
DBD 4.0	EN_COUNT	Validation du compteur
DBD 4.1	RESET	Initialisation du compteur
DBD 4.2	SET_COMP	Positionnement du comparateur
DBD 6	COUNT	Valeur courante du compteur
DBD 10	COMP	Seuil de comparaison (actuel)

Tableau 3.20. Structure des DB60 et DB61

4.4. Fonction intégrée Positionnement :

4.4.1. Description de la fonction intégrée positionnement :

La fonction intégrée Positionnement de la CPU 314 IFM permet de commander le positionnement d'axes en liaison avec un programme utilisateur.

Les entrées utilisées pour cette fonction sont :

Sortie	Adresse	Rôle
Entrée TOR piste A	E 126.0	Raccordement d'un codeur incrémental pour la saisie du
Entrée TOR piste B	E 126.1	déplacement
Entrée TOR contact point de référence	E 126.2	Raccordement d'un contact du point de référence pour la synchronisation

Tableau 3.21. Entrées associées à la fonction intégrée positionnement.

La position est prélevée à l'aide d'un codeur incrémental sans signaux inversés 24V. Ce codeur délivre deux trains d'impulsions A et B déphasés de 90°, ce qui permet de connaître la vitesse et le sens de rotation.

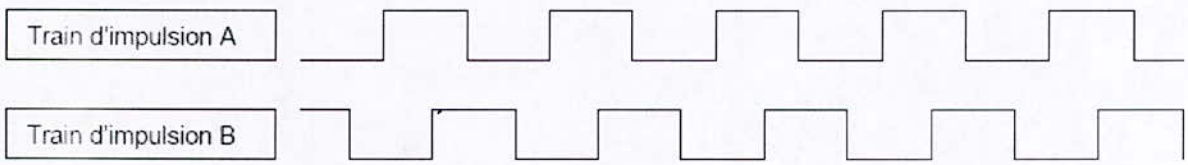


Fig3.7. Signal délivré par un codeur sans signaux inversés

Le traitement du signal délivré par le codeur est dit en mode simple. Cela veut dire que seuls les fronts montants seront pris en considération.

La fonction intégrée positionnement donne la possibilité de brancher un signal de synchronisation de valeur initiale. Ce signal est donné par la plupart des codeurs et permet de re-calibrer la position initiale à chaque passage par un point de référence donné.

Ce point de référence est matérialisé par un détecteur de position qui délivre un signal de 24V lorsque sa position est atteinte par le moteur.

4.4.2. Types de connexions possibles avec un moteur :

L'automate programmable S7-314 IFM ne peut pas être raccordé directement à un moteur, il est indispensable d'utiliser une interface de puissance pour la commande en position.

Ces interfaces peuvent être de deux types :

- Montage à contacts : permet de commander un moteur à deux vitesses.
- Variateur de fréquence : permet de commander un moteur synchrone ou asynchrone.

Commande d'un moteur à deux vitesses :

Pour ce faire, 4 sorties TOR sont utilisées :

Sortie	Adresse	Rôle
Sortie TOR petite vitesse	A124.0	Sorties pour le choix de la vitesse du moteur
Sortie TOR grande vitesse	A124.1	
Sortie TOR marche avant	A124.2	Sorties pour le choix du sens de rotation
Sortie TOR marche arrière	A124.3	

Tableau 3.22. Sorties associées à la commande d'un moteur à deux vitesses.

La figure suivante montre le profil de vitesse que l'on peut avoir pour un moteur à deux vitesses :

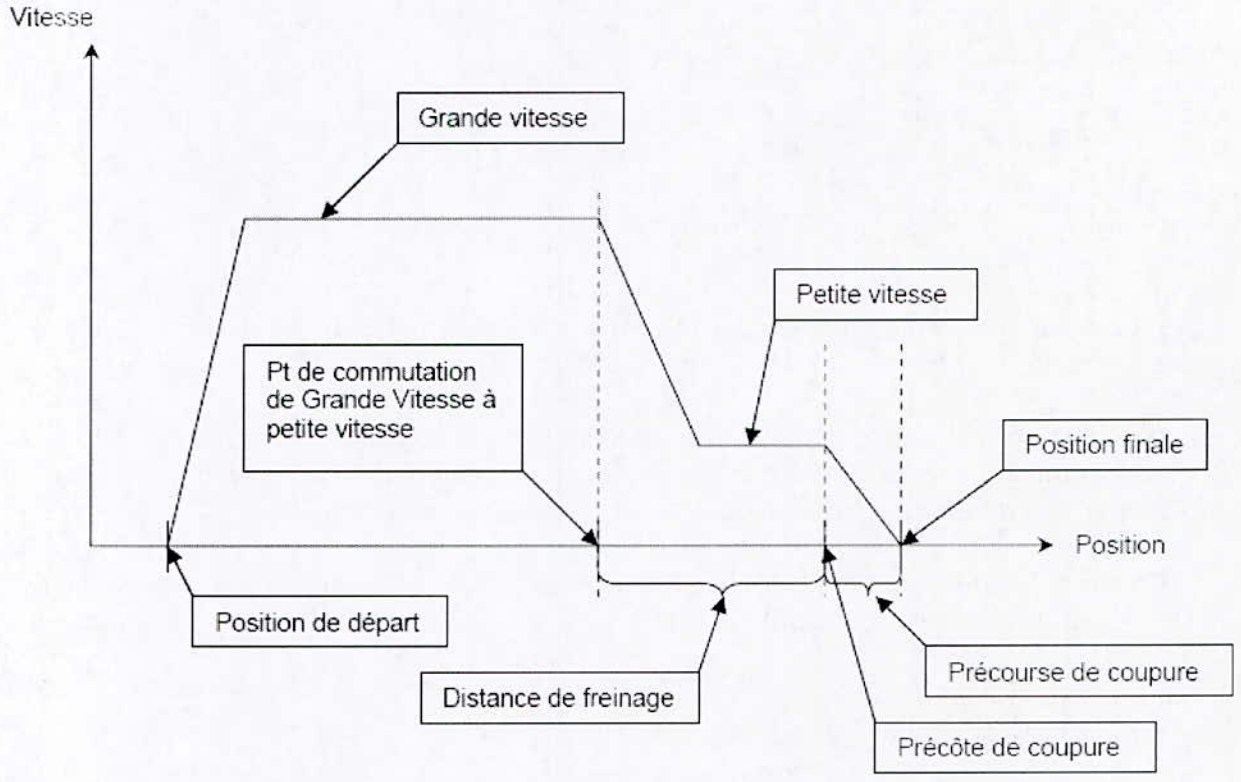


Fig3.8. Profil de vitesse d'un moteur à deux vitesses.

Le principe est de faire démarrer le moteur avec la grande vitesse, puis lors de la phase de freinage, on commute sur la petite vitesse. Juste avant d'arriver à la position finale, on coupe la petite vitesse et on laisse le moteur s'arrêter jusqu'à atteindre la position désirée.

La figure suivante montre le comportement des sorties TOR lors de la commande :

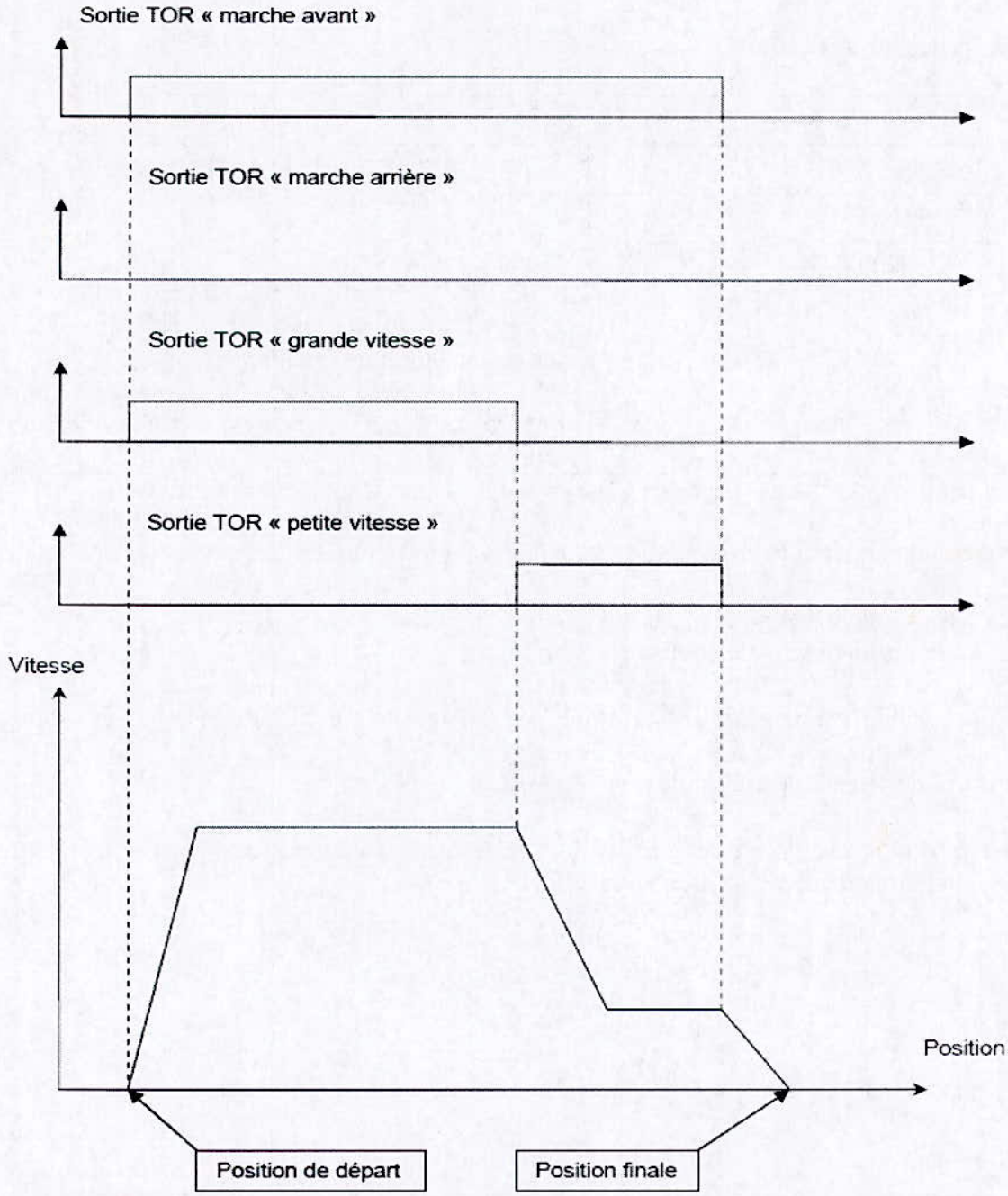


Fig3.9. Chronogramme des sorties TOR pour la fonction intégrée positionnement.
(Positionnement en marche avant)

Commande par convertisseur de fréquence :

Cette méthode est utilisée pour actionner des moteurs synchrones ou asynchrones.

Les sorties utilisées sont :

Sortie	Adresse	Rôle
Sortie TOR marche arrière	A124.2	Utilisé si le convertisseur ne gère que les signaux analogiques positifs.
Sortie TOR marche avant	A124.3	
Sortie analogique vitesse	PAW 128	Donne la valeur de la vitesse : - *f* Délivre une sortie de 0-20mA ou de 0-10V dans le cas d'un convertisseur qui ne gère que des signaux analogiques positifs. - *f* Délivre une sortie de ±20mA ou ±10V pour un convertisseur qui gère des signaux positifs et négatifs.

Tableau 3.23. Sorties associées à la commande d'un moteur par variateur de vitesse.

La commande par convertisseur de fréquence a comme contraintes principales :

- La vitesse maximale admise par le système.
- L'accélération maximale admise par le système.

La commande donnée fait en sorte que ces valeurs ne soient jamais dépassées tout en optimisant le temps.

Le profil de vitesse utilisé est de type trapézoïdal.

La figure qui suit montre les profils de vitesse et d'accélération utilisés par la fonction intégrée positionnement de la CPU S7-314 IFM:

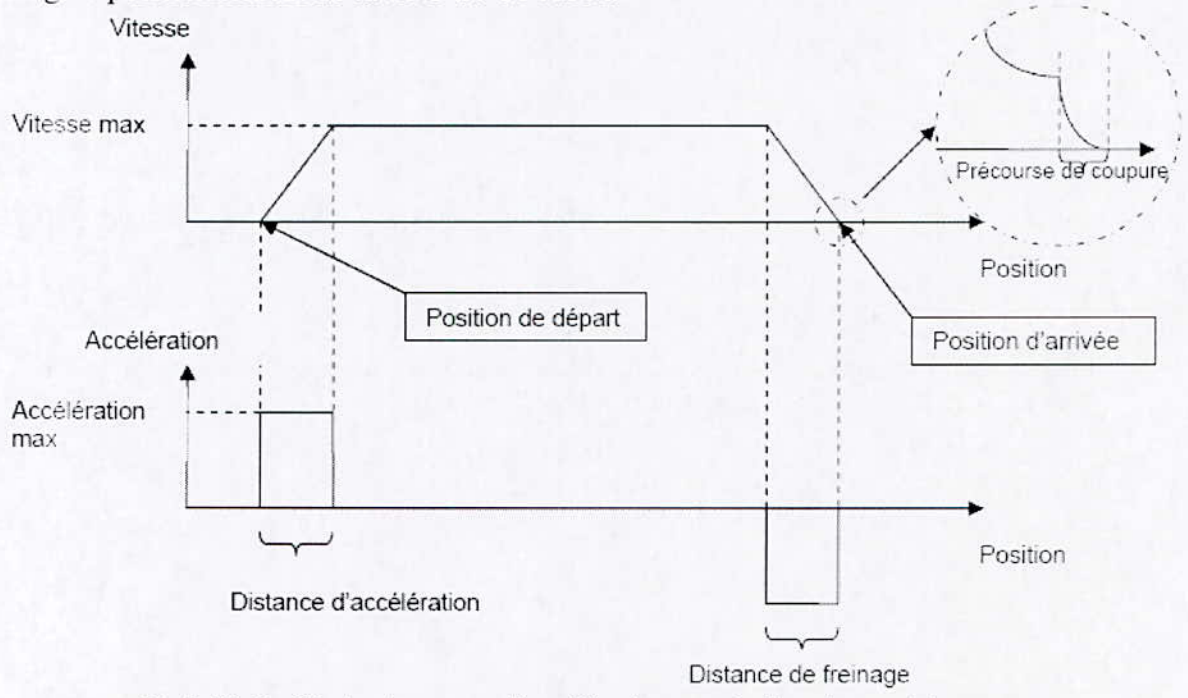


Fig3.10. Profils de vitesses et d'accélération pour la fonction positionnement.

La vitesse maximale est définie dans le programme utilisateur.

La distance d'accélération et la distance de freinage sont à paramétrer avec *STEP 7*.

On remarque qu'il existe une précourse de coupure pour l'arrêt du moteur, ce paramètre est donné dans le programme utilisateur.

Valeurs sur les sorties :

Comme indiqué précédemment, deux configurations sont possibles pour la commande :

• 1 sortie analogique (pour la valeur de la vitesse) qui délivre 0-20mA ou 0-10V
Avec deux sorties TOR (1 pour marche arrière et 1 pour marche avant).

• 1 sorties analogique (pour la valeur de la vitesse et le sens) qui délivre ±20mA ou ±10V.
Il faut savoir que la sortie analogique est donnée sous forme d'échelons. Ces échelons sont au nombre de 10 et ont la même largeur, mais possèdent des hauteurs différentes.

La hauteur des échelons est prédéfinie par la fonction intégrée positionnement.

La largeur est calculée de la manière suivante :

$$\text{Largeur de l'échelon} = \frac{\text{Distance d'accélération/freinage}}{10}$$

Cette valeur est tronquée de sa partie réelle pour être sûr que la distance d'accélération/de freinage parcourue effectivement ne soit jamais supérieure à la distance paramétrée.

La valeur analogique délivrée par la sortie se calcule de la manière suivante :

$$V_{Max} = \frac{10V}{256} \times (256 - BREAK) \qquad \text{ou} \qquad I_{Max} = \frac{20mA}{256} \times (256 - BREAK)$$

La valeur de BREAK est donnée dans le SFB 39.

CONCLUSION

Nous avons vu, dans le présent chapitre, une description détaillée de l'API S7-300 (S7-314 IFM) produit par la firme SIMENS. Nous avons aussi décrit toutes les fonctions qui y sont intégrées et qui, comme nous l'avons vu, jouent un rôle primordial dans le fonctionnement de l'API. Dans le chapitre suivant, nous entamerons une tape très importante dans notre travail et qui concerne toute l'analyse qui a été faite pour élaborer un programme (Grafcet) de commande des différentes composantes du processus de chromage.

CHAPITRE -4-

COMMANDE PAR API DU PROCESSUS DE CHROMAGE

INTRODUCTION

Au cours de ce chapitre nous allons voir un exemple de programmation des API pour l'élaboration d'un programme de commande du processus de chromage qui a été déjà décrit dans les chapitres précédents. La programmation par GRAFCET consiste en le remplacement des différentes fonctionnalités et composants câblés par des objets fictifs simplement manipulables et ajustables.

1. REALISATION DU GRAFCET DE LA CHAINE

Les processus industriels sont, dans la plupart des cas, des systèmes séquentiels. On peut alors les traduire sous forme de GRAFCET ou de réseaux de Pétri.

Chaque événement de ces processus peut être représenté sous forme d'étape, transition ou actions, donc peut être programmé avec S7Graph.

Avant d'écrire le programme, il est tout d'abord nécessaire de créer un projet, de configurer le matériel, définir la table des mnémoniques et insérer les blocs nécessaires pour la mise en marche du projet

> **Remarque**
> Si on veut travailler avec le S7Graph les FB (Blocs Fonctionnels) insérés doivent être de type GRAPH.
> Dans le STEP 7 Les OB ne peuvent pas être de type GRAPH, il sont de type CONT, LOG, ou LIST.

Dans l'exemple cité dans ce chapitre, la configuration matérielle de base est :

- CPU S7-314 IFM 6ES7 314-5AE83-0A0B version matérielle 1.2
- Alimentation PS 307 10A 6ES7 307-1KA00-0AA0
- Coupleurs pour 1 châssis d'extension 6ES7 365-0BA81-0AA0
- 3 Modules de 32 entrées TOR, 24 volts ES7 321-1BL80-0AA0
- 2 Module de 32 sorties TOR 6ES7 322-1EL00-0AA0

La table des mnémonique a été défini à partir des entrées, sorties, mémentos (variables internes ou momentanées), tempos (variables temporelles), ..., qui sont définies par les capteurs, les actionneurs, ..., déjà existants dans l'installation initiales de la chaîne.

Pour mieux gérer le système global, on a préféré de le subdiviser en plusieurs sous-systèmes. Ceci permet de mieux tester et déboguer les programmes.

Ce choix se traduit par la réalisation de chaque partie du système soit par une subroutine soit par une fonction.

Figure illustrant la structuration du système global :

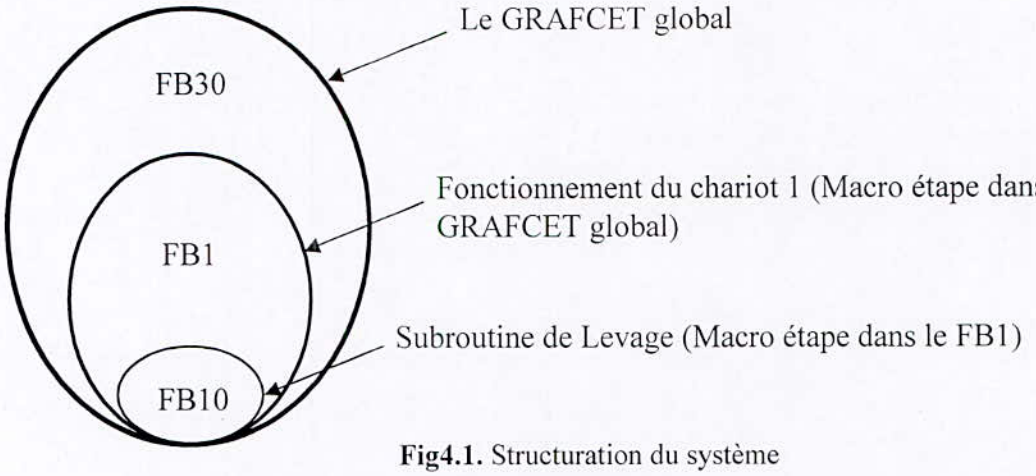


Fig4.1. Structuration du système

Pour réaliser un GRAFCET, il est préférable d'utiliser une subroutine qui, à la différence d'une fonction, possède des données statiques contenues dans le bloc de données (DB) associé. Ceci permet de sauvegarder des variables pour une utilisation dans le cycle qui suit.

Afin d'optimiser le programme, on crée différents blocs (OB : blocs d'organisations, FB : blocs fonctionnels,...).

Le tableau suivant nous présente les différents blocs crées pour le projet de chromage :

Nom du bloc	Langage de création	Type	Commentaire	DB associé
Blocs d'organisations				
OB1	CONT	Bloc d'organisation	Cycle d'exécution	-
OB100	CONT	Bloc d'organisation	Complete Restart (démarrage)	-
Blocs fonctionnels				
FB1	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot 1	DB1
FB2	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot 2	DB2
FB3	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot 3	DB3
FB4	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot 4	DB4
FB5	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot 5	DB5
FB6	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot 6	DB6
FB7	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot de chargement 1	DB7
FB8	GRAPH	Bloc fonctionnel	Fonctionnement du Chariot de chargement 2	DB8
FB9	GRAPH	Bloc fonctionnel	Fonctionnement du bain transversal	DB9
FB30	GRAPH	Bloc fonctionnel	Programme global	DB30
Subroutine				
FB10	GRAPH	Bloc fonctionnel	Levage	DB10
FB11	GRAPH	Bloc fonctionnel	Abaissement	DB11
FB12	GRAPH	Bloc fonctionnel	Translation vers l'avant	DB12
FB20	GRAPH	Bloc fonctionnel	Translation vers l'arrière	DB20

Fonctions				
FC1	CONT	Fonction	Fonction de sécurité de la chaîne	-
Fonctions Systèmes				
FC72	LIST	Fonction		-
SFC17	LIST	SFC		-
SFC18	LIST	SFC		-
SFC52	LIST	SFC		-
SFC64	LIST	SFC		-

Tableau 4.1. Les Blocs du projet chromage

Remarque :
Lors de la mise en marche de l'automate, les valeurs contenues dans la RAM peuvent être quelconques (1 ou 0) selon l'historique de la CPU (démarrage à chaud, … etc).
Il est donc nécessaire d'initialiser Le GRAFCET en utilisant l'OB100 qui est le bloc d'organisation exécuté lors de la mise en RUN de la CPU.

2. LE GRAFCET DU PROJET CHROMAGE :

2.1. Le GRAFCET global :

Le GRAFCET global du projet est représenté par le bloc fonctionnel FB30, qui est un bloc programmé par S7Graph et qui contient des macros étapes représentant l'appel de tout les blocs fonctionnels du projets tel que le FB1 (bloc fonctionnel pour le chariot1), le FB2 (bloc fonctionnel du chariot 2), ….

Le bloc FB30 est composé de :

- Etape initiale pour initialiser les mnémoniques nécessaires pour le fonctionnement des autres blocs du projet.
- Une branche **ou** pour que les blocs **FB1**, **FB2**, **FB3**, **FB4**, **FB5**, **FB6**, **FB7**, **FB8**, **FB9** puissent marcher en parallèle (momentanément).
- Etapes **S2**, **S3**, **S4**, **S5**, **S6**, **S7**, **S8**, **S9**, **S10** représentent, respectivement, l'appel des blocs fonctionnels FB1, FB2, FB3, FB4, FB5, FB6, FB7, FB8, FB9

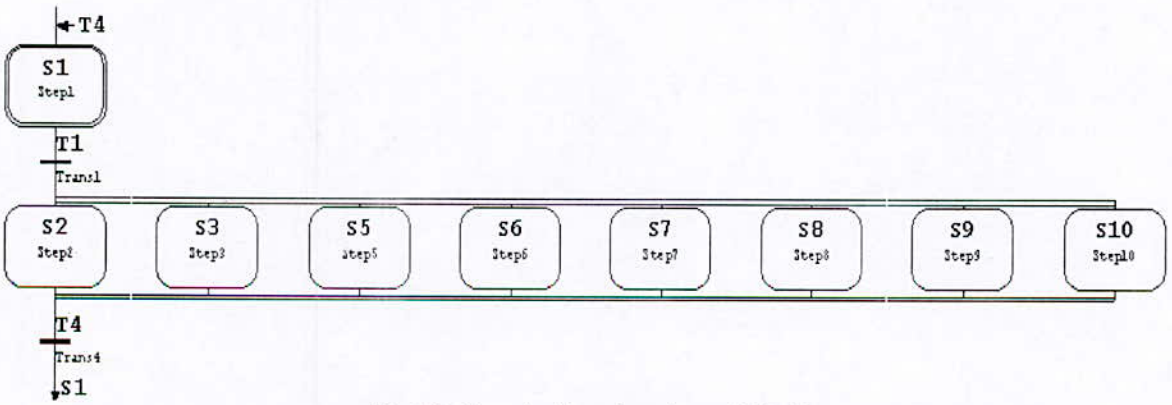


Fig4.2. Vue du bloc fonctionnel FB30

2.2. Exemple de programmation d'un GRAFCET du projet :

2.2.1. GRAFCET du chariot 1:

Diagramme pas à pas du chariot 1 :

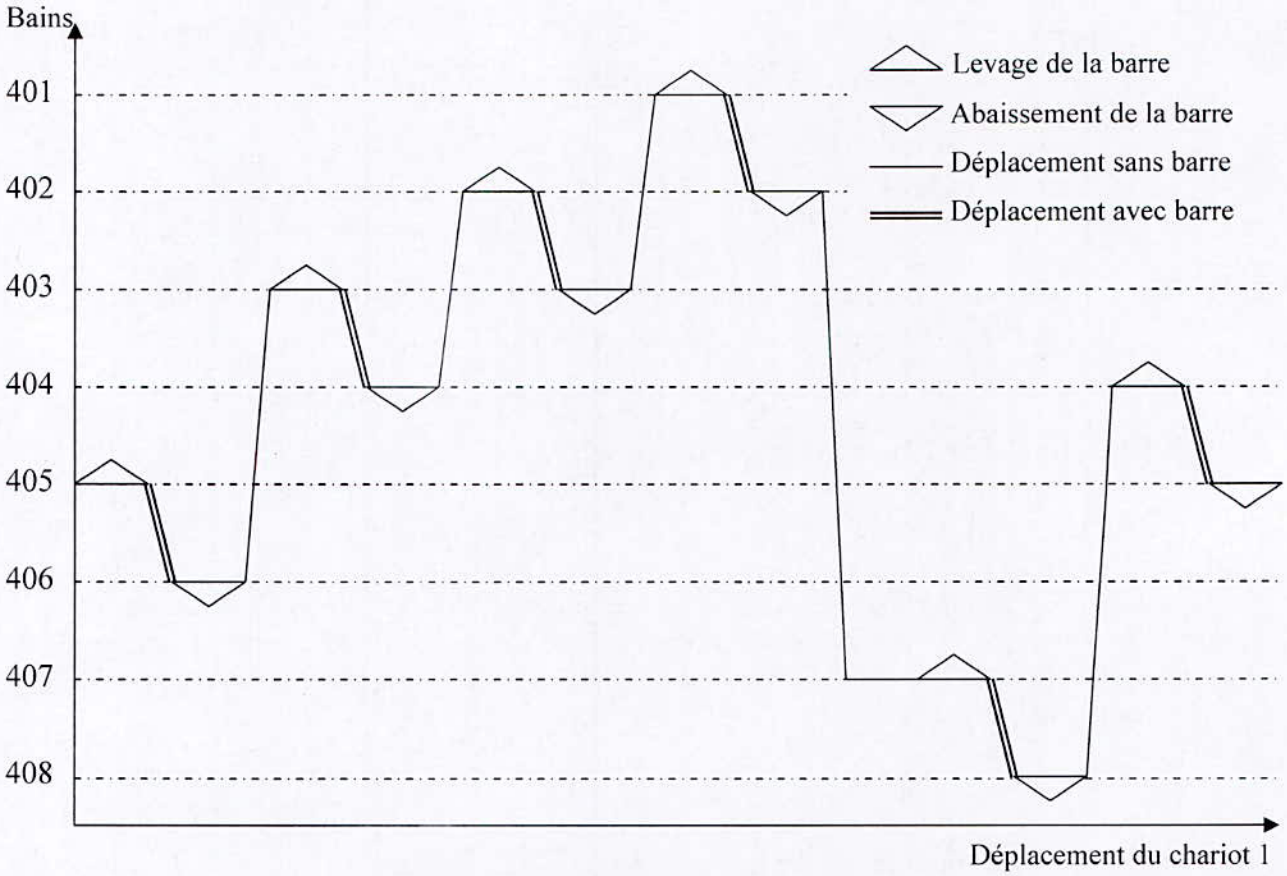


Fig4.3. Diagramme pas à pas du chariot 1

Remarque :
Pour mieux illustrer le GRAFCET du chariot 1 (bloc fonctionnel FB1), Il sera présenté dans une annexe avec les mnémoniques utilisées pour son fonctionnement.

Dans la suite seront présentées les deux subroutines FB10 et FB12 (respectivement : subroutine du levage et subroutine de translation vers l'avant).

2.2.2. GRAFCET de levage (FB10) :

Définition des mnémoniques utilisées pour la subroutine de levage :

Désignation	Valeur initiale	Fonction
		Entrées
1b3a	0	Commande manuelle du chariot (levage)
1b9	0	Limite d'abaissement
1b10	0	Limite de levage
1b11	0	Détection de la vitesse lente en abaissement
1b12	0	Détection de la vitesse lente en levage
1b17	0	Indique le Stationnement du chariot
		Sorties
Frein_lev	1	Frein du moteur transversal
Montée_lente	0	Activer la vitesse lente en levage
Montée_rapide	0	Activer la vitesse rapide en levage
		Mémentos
Levage	0	Bit à 1 indique que le chariot est en levage
Abaissement	0	Bit à 1 indique que le chariot est en abaissement
Translation_av	0	Bit à 1 indique que le chariot est en mouvement de translation vers l'avant
Translation_ar	0	Bit à 1 indique que le chariot est en mouvement de translation vers l'arrière
Prés_aprés_lev	0	Bit à 1 indique que le chariot est prés pour la translation
0c1	1	Arrêt d'urgence de la chaîne
Arrêt_chariot	1	Arrêt d'urgence du chariot

Tableau 4.2. Mnémoniques utilisées pour le FB10

Etape initiale :
C'est une étape qui comporte trois actions :

- Déclenchement du frein du moteur de levage.
- Chariot en levage avec vitesse lente.
- Mise à 1 du mémento « levage ».

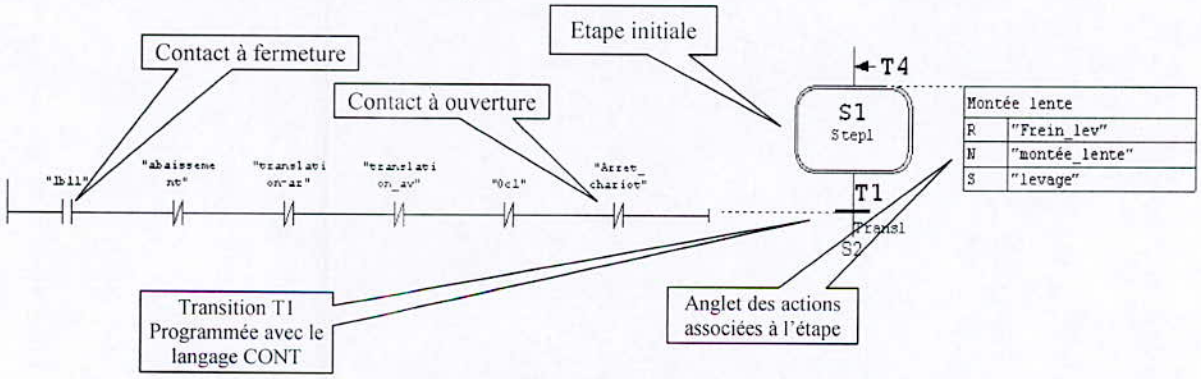


Fig4.4. Etape initiale du bloc FB10

Etape S2 :

Elle est activée après l'activation de l'étape initiale si transition T1 est vérifiée.
Ces actions :

- Déclenchement du frein du moteur de levage.
- Chariot en levage avec vitesse rapide.

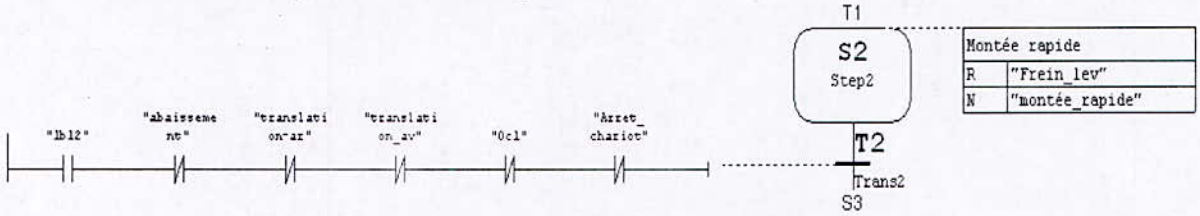


Fig4.5. Etape S2 du bloc FB10

Etape S3 :

Elle est activée après l'activation de l'étape S2 si transition T2 est vérifiée.
Ces actions :

- Déclenchement du frein du moteur de levage.
- Chariot en levage avec vitesse lente.

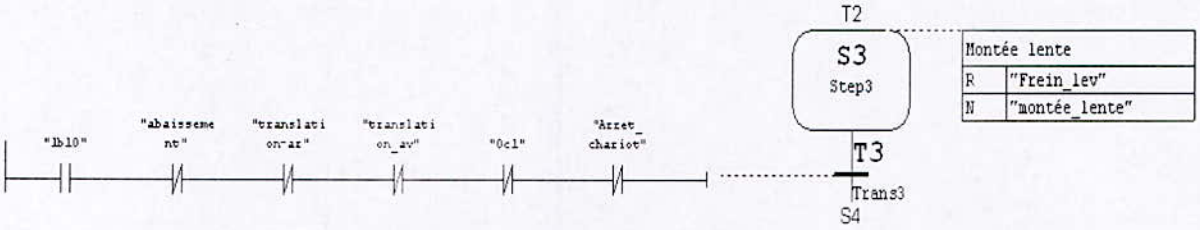


Fig4.6. Etape S3 du bloc FB10

Etape S4 :

Elle est activée après l'activation de l'étape S3 si transition T3 est vérifiée.
Ces actions :

- Il n'y a pas de déclenchement du frein du moteur de levage, donc freinage du chariot (stationnement).
- Mise à 0 du mémento « levage ».
- Mise à 1 du mémento « prés_aprés_lev ».

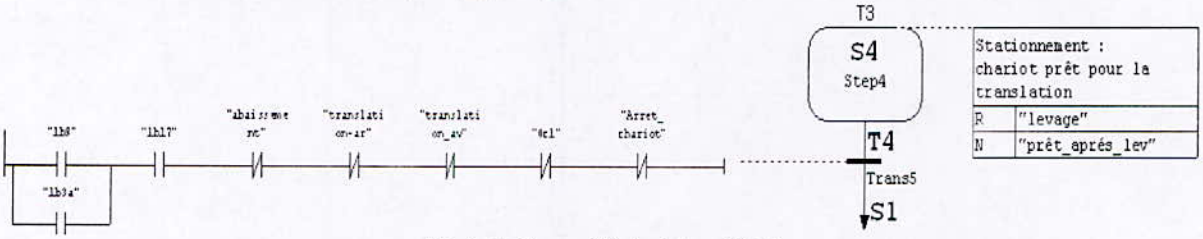


Fig4.7. Etape S4 du bloc FB10

Remarque
L'étape initiale peut être réactivé après l'activation de l'étape S4 si la transition T4 est vérifiée

2.2.3. GRAFCET de translation vers l'avant (FB12) :

A cause de l'absence des capteurs de positions pour chaque bain, on a utilisé une méthode de comptage.

Quand le chariot est en état de marche transversale, on compte les impulsions provoquées par les boues de fer placés au dessus des bains, avec les deux détecteurs de proximité « 1b17 » pour le stationnement et le « 1b0 » pour la vitesse lente vers l'avant (respectivement le « 1b1 » pour la vitesse lente vers l'arrière), et ceci comme suit :

- Le chariot est en état de marche avant lente (respectivement marche arrière lente).
- La première impulsion détectée par le « 1b0 » (respectivement le « 1b1 ») : aucun changement.
- La première impulsion détectée par le « 1b17 » : mise à 1 de la sortie « avance_rapide » (respectivement de la sortie « arrière_rapide »), le chariot avance à vitesse rapide (respectivement recule à vitesse rapide).
- La dernière impulsion détectée par le « 1b0 » (respectivement le « 1b1 ») : mise à 1 de la sortie « avance_lente » (respectivement de la sortie « arrière_lente »), le chariot avance à vitesse lente (respectivement recule à vitesse lente).
- La dernière impulsion détectée par le « 1b17 » : freinage du chariot.

On doit savoir aussi que le nombre d'impulsions détectées varie selon le nombre de bains parcourus.

Définition des mnémoniques utilisées pour le fonctionnement de la subroutine de levage :

Désignation	Type	Valeur initiale	Fonction
Entrées			
1b3c	BOOL	0	Commande manuelle du chariot (translation vers l'avant)
1b0	BOOL	0	Détection de la vitesse lente en avancement
1b9	BOOL	0	Limite d'abaissement
1b10	BOOL	0	Limite de levage
1b17	BOOL	0	Indique le Stationnement du chariot
Sorties			
Frein_translation	BOOL	1	Frein du moteur transversal
Avance_lente	BOOL	0	Activer de la vitesse lente
Avance_rapide	BOOL	0	Activer de la vitesse rapide

Mémentos			
Levage	BOOL	0	Bit à 1 indique que le chariot est en levage
Abaissement	BOOL	0	Bit à 1 indique que le chariot est en abaissement
Translation_av	BOOL	0	Bit à 1 indique que le chariot est en mouvement de translation vers l'avant
Translation_ar	BOOL	0	Bit à 1 indique que le chariot est en mouvement de translation vers l'arrière
Prés_aprés_avance	BOOL	0	Bit à 1 indique que le chariot est prés pour la translation
nbre_bain_av	WORD	-	Nombre de bains parcourus en avancement
0c1	BOOL	1	Arrêt d'urgence de la chaîne
Arrêt_chariot	BOOL	1	Arrêt d'urgence du chariot
Compteur			
Z1	WORD	nbr_bain_av	Compteur des bains parcourus en avancement

Tableau 4.3. Mnémoniques utilisées par le FB12

Etape initiale :
Ces actions :

- Mise à 1 du mémento « translation_av ».
- Déclenchement du frein du moteur de translation.
- Chariot en translation avec vitesse lente.
- Initialisation du compteur Z1

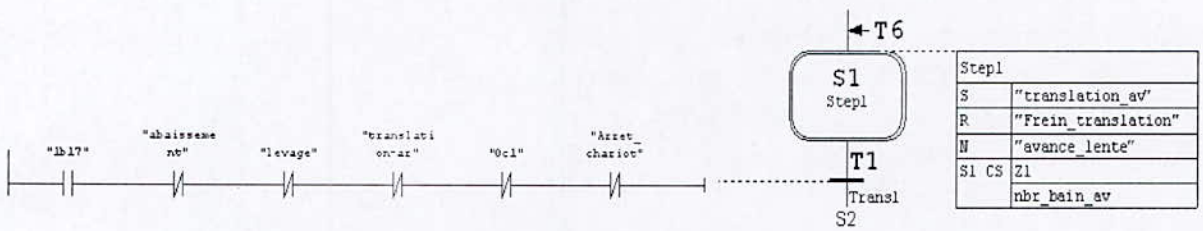


Fig4.8. Etape initiale du bloc FB12

Etape S2 :

Elle est activée après l'activation de l'étape initiale si la transition T1 ou la transition T3 sont vérifiées.

Elle est désactivée si transition T2 est vérifiée

Ces actions :

- Déclenchement du frein du moteur de translation.
- Chariot en translation avec vitesse rapide.

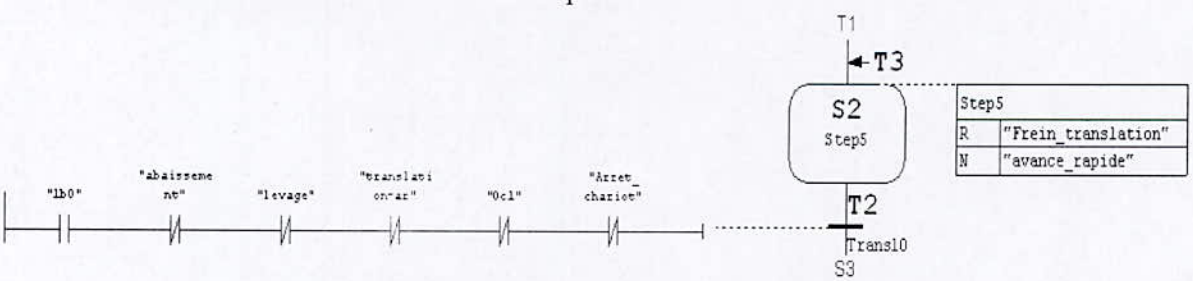


Fig4.9. Etape S2 du bloc FB12

Etape S3 :

Elle est activée après l'activation de l'étape S2 si la transition T2 est vérifiée.
Elle est désactivée si la transition T3 ou la transition T4 sont vérifiées.
Ces actions :

- Déclenchement du frein du moteur de translation.
- Chariot en translation avec vitesse rapide.
- Décrémentation du compteur Z1 à chaque activation de cette étape.

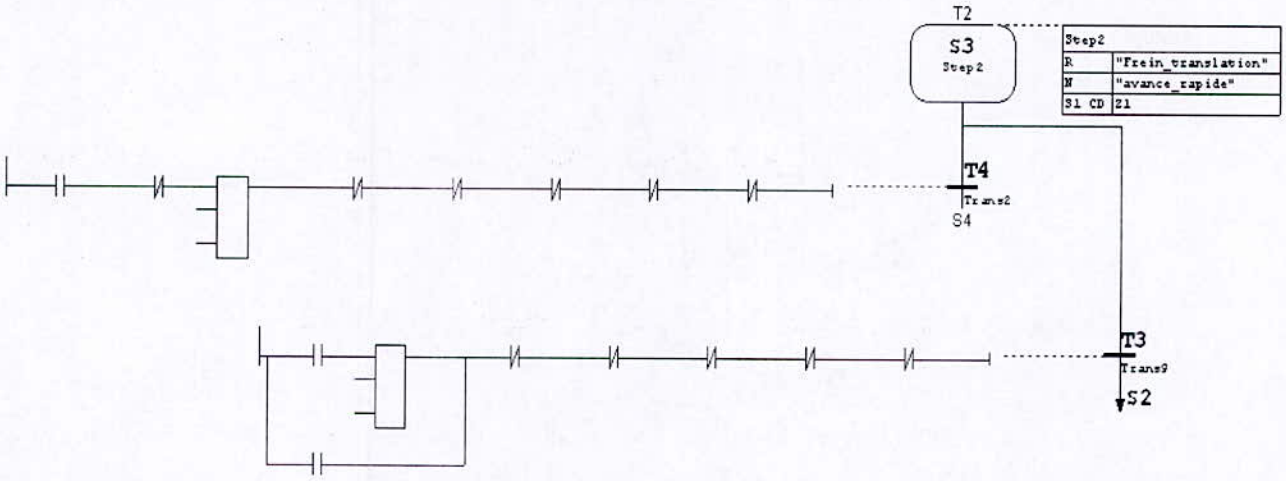


Fig4.10. Etape S3 du bloc FB12

Etape S4 :

Elle est activée après l'activation de l'étape S3 si la transition T3 ou la transition T4 sont vérifiées.
Elle est désactivée si la transition T5est vérifiée.
Ces actions :

- Déclenchement du frein du moteur de translation.
- Chariot en translation avec vitesse lente.

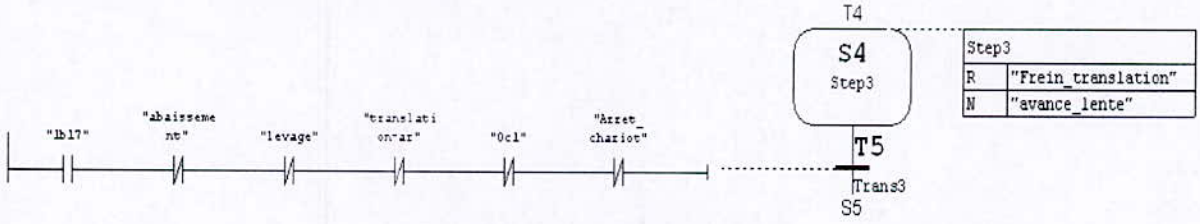


Fig4.11. Etape S4 du bloc FB12

Etape S5 :

Elle est activée après l'activation de l'étape S4 si la transition T5 est vérifiée.
Elle est désactivée si la transition T6 est vérifiée.

Ces actions :

- Il n'y a pas de déclenchement du frein du moteur de translation, donc freinage du chariot (stationnement).
- Mise à 0 du mémento « translation-av ».
- Mise à 1 du mémento « prés_aprés_avant ».

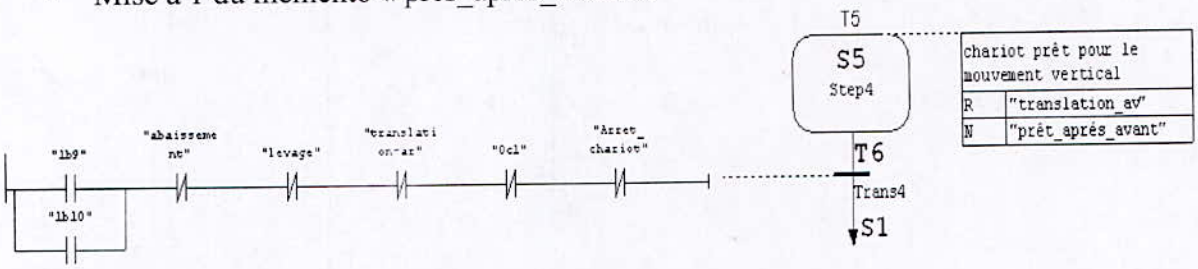


Fig4.12. Etape S5 du bloc FB12

CONCLUSION

Nous avons consacré ce chapitre à l'analyse et l'élaboration d'un programme pour le contrôle des différentes opérations de chromage en adoptant comme langage de programmation, le GRAFCET. L'analyse a concerné surtout, les entrées, les informations provenant des différents détecteurs et capteurs implantés partout où l'on veut s'informer du processus et des actions à déclencher pour en modifier l'état.

Nous venons, par conséquent, d'achever l'étape la plus importante du travail, et il ne reste qu'à évaluer le fonctionnement par simulation où donc sur le terrain.

CONCLUSION GENERALE

Au cours de notre étude, et en vue de remplacer le dispositif de contrôle de la chaîne de chromage au sein de l'unité ORSIM du groupe BCR, nous avons procédé par une étude descriptive du processus et de la chaîne ainsi que tous les accessoires complémentaires.

Nous avons jugé intéressant et commode de faire appel à la technologie programmée (Automate programmable Industriel), et ce pour des raisons de :

- Complexité de l'installation existante (logique électrique câblée),
- Diagnostic des pannes et entretien relativement difficiles,
- Problème de l'inflexibilité (installation de contrôle figée),
- Efficacité minime et autres problèmes économiques tels que : taux de productivité, coûts d'entretien...etc.

Ensuite, dans le deuxième chapitre, l'étude théorique sur les API a été établie. Nous avons décrit leurs architectures comme : le CPU et les modules d'entrées/sorties ... en terme matériel ; l'interface de programmation ...en terme logiciel.

Dans le troisième chapitre, nous avons présenté une étude assez détaillée sur le SIMATIC S7-314 IFM de SIEMENS, qui a été notre choix pour l'application visée ainsi que les raisons et les motifs du choix qui se manifestent essentiellement dans :

- Performances techniques (fréquence de traitement, type de signaux traités ... etc.),
- Capacité mémoire assez suffisante.
- Possibilité d'extension en terme du nombre élevé des entrées et des sorties du processus, exigé par certaines applications. Laquelle nous a permis d'intégrer trois autres modules de 32 entrées TOR et deux modules de 32 sorties TOR.
- Possibilité d'intégrer des modules spéciaux (modules d'entrées ou de sorties analogiques, régulateur de température, de pression, ...etc.).
- Disposition du CPU 314 IFM de fonctions intégrés telles que : le fréquencemètre, les compteurs, ...etc.

Quant au quatrième chapitre, il a été consacré à la programmation graphique dite **GRAFCET** du processus de chromage décoratif. Nous avons présenté un exemple de programme englobant le contrôle du premier chariot (en annexe), le levage et la translation sous forme de procédures (subroutines).

Enfin, on pourrait dire que remplacer les dispositifs classiques de contrôle par les Automates Programmables Industriels révèle comme étant la solution qui subviendrait aux besoins du monde industriel future voire même actuel qui ne cesse de trop exigé pour ce qui est de qualité et quantité des produits.

BIBLIOGRAPHIE

[1]. G. MICHEL, « Les A.P.I. Architecture et application des automates programmables industriels », Edition DUNOD ,1987.
[2]. A. SIMON, « Automate Programmable Industriel, Niveau 1 », Edition L'ELAN, 1991.
[3]. M. BERTRAND, «Automates programmables industriels », Techniques de l'ingénieur, Vol. S 8 015.
[4]. P. JARGOT, « Langages de programmation pour API. Norme IEC 1131-3 », Techniques de l'ingénieur , Vol. S 8 030.
[5]. D. DUPONT, D. DUBOIS, « Réalisation technologique du GRAFCET », Techniques de l'ingénieur , Vol. S 8 032.

Manuels :

[6]. SCHNEIDER ELECTRIC, « Automates Nano et Plate-forme d'automatisme Micro »,1999.
[7]. SIEMENS, « Mise en route STEP7 V5.2.Getting Started », Réf. 6ES7810-4CA06-8CA0,SIMATIC, 2002.
[8]. SIEMENS, « Programmer avec STEP7 V5.2 », Réf. 6ES7810-4CA06-8CA0, SIMATIC, 2002.
[9]. SIEMENS , «Langage CONT pour SIMATIC S7-300/400 », Réf. 6ES7810-4CA06-8CR0, SIMATIC,2002.
[10]. SIEMENS , «Langage LOG pour SIMATIC S7-300/400 », Réf. 6ES7810- 4CA06-8CR0, SIMATIC,2002.
[11]. SIEMENS, «Langage LIST pour SIMATIC S7-300/400 », Réf. 6ES7810-4CA06-8CR0, SIMATIC, 2002.
[12]. SIEMENS, «Automate programmable S7-300, Fonctions intégrées, CPU 312 IFM/314IFM », Réf. EWA 4NEB 710 6058-03a Edition 2, SIMATIC, 2000.
[13]. SIEMENS, « Liste des opérations S7-300, CPU 312 IFM, 314 IFM, 313, 314, 315,315-2 DP, 316-2 DP, 318-2 », Réf. 6ES7 398-8AA03-8CN0 Edition 2, SIMATIC,2000.
[14]. SIEMENS, « Automate programmable S7-300, Installation et configuration - caractéristiques techniques des CPU», Réf. 6ES7 398-8AA03-8CA0 Edition 2, SIMATIC,1999.
[15]. SIEMENS, « Automate programmable S7-300 Caractéristiques des CPU, CPU 312IFM–318-2 DP », Réf. 6ES7398-8FA10-8CA0, SIMATIC, 2001.
[16]. SIEMENS, «Automate programmable S7-300, Getting Started, Premières étapes pour montage et mise en service », Réf. A5E00069341 01, SIMATIC, 2000.