Office de la Formation Professionnelle
et de la Promotion du Travail

Technicien Spécialisé

Génie Electrique

Tronc commun

Manuel de cours

Module 12

Systèmes industriels
à base d'automates programmables

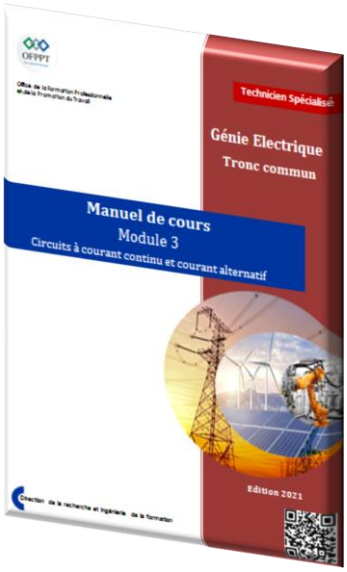
Edition 2021

Direction de la Recherche et Ingénierie de la Formation

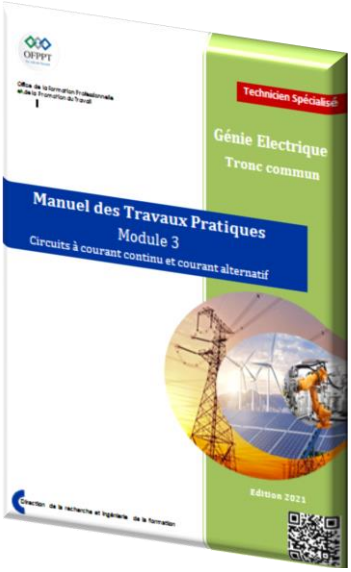
Avant-propos

Les manuels de cours, de travaux pratiques et le guide e-learning sont téléchargeables à partir de la plateforme e-learning moyennant les codes QR suivants :

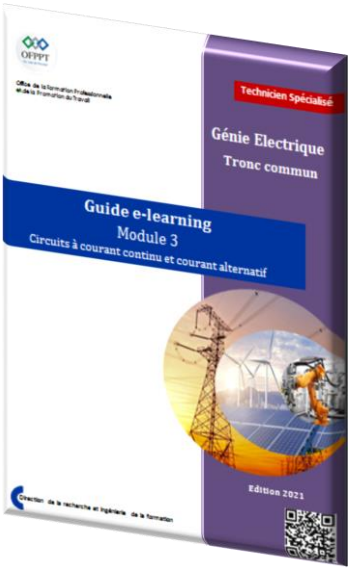
Manuel de cours



Manuel des travaux pratiques



Guide e-learning



SOMMAIRE

COMPETENCES-CIBLES ET OBJECTIFS OPERATIONNELS

Module 12 : Systèmes industriels à base d'automates programmables

Code : GETC – 12 **Durée : 75 heures**

ENONCE DE LA COMPETENCE

Installer et dépanner des systèmes industriels à base d'automates programmables

CONTEXTE DE REALISATION

- Individuellement
- À partir de :
  - Cahier de charges fonctionnelles
  - Dossier technique du système
  - Maquette virtuelle (3D)
  - Schémas, des modules fonctionnels.
- À l'aide de :
  - Automates programmables
  - Logiciels de programmation d'un automate programmable
  - Système piloté par un automate programmable

CRITÈRES GÉNÉRAUX DE PERFORMANCE

- Utilisation appropriée de l'outillage et de l'équipement
- Respect des méthodes et des conventions de programmation d'un automate.
- Qualité des travaux.
- Respect des règles de santé et de sécurité au travail.

ÉLEMENTS DE LA COMPETENCE	CRITÈRES PARTICULIERS DE PERFORMANCE
A. Interpréter l'architecture d'un système industriel géré par un automate programmable (API).	• Description correcte structurelle • Interprétation juste des consignes, des plans et des manuels. • Identification correcte des technologies. • Interprétation correcte des composants d'un système industriel gérer par API.
B. Découper le fonctionnement d'un système industriel géré par un automate programmable (API).	• Découpage fonctionnelle juste • Identification correcte des entrées/sorties des fonctions • Description fonctionnelle (temporelle) correcte des fonctions • Intégration correcte des fonctions (description globale)
C. Programmer un automate programmable industriel (API).	• Interprétation juste d'un programme API • Utilisation juste d'un logiciel de programmation • Implémentation correcte d'un programme API.
D. Diagnostiquer un système industriel géré par un automate programmable (API).	• Test et analyse correcte de fonctionnement • Identification juste des dysfonctionnements • Réparation correcte de l'anomalie • Validation juste de bon fonctionnement

Chapitre I

INTRODUCTION AUX SYSTEMES AUTOMATISES COMMANDES PAR API

1. Introduction aux systèmes automatisés

1.1 Définition d'un système de production

Un système de production est un système à caractère industriel possédant les caractéristiques suivantes :

- L'obtention de la valeur ajoutée présente un caractère reproductible.
- La valeur ajoutée peut être exprimée et quantifiée en termes économiques.

Un système de production répond au besoin d'élaborer des produits, de l'énergie ou de l'information à un coût rentable pour l'utilisateur du système. L'élaboration de la valeur ajoutée sur les matières d'œuvre est obtenue :

- Au moyen d'un ensemble de dispositifs opératifs appelés **partie opérative (PO).**
- Par l'action d'opérateurs humains et/ou de dispositifs de commande pour assurer la coordination des dispositifs opératifs.

Tout système de production possède une structure semblable au schéma suivant :

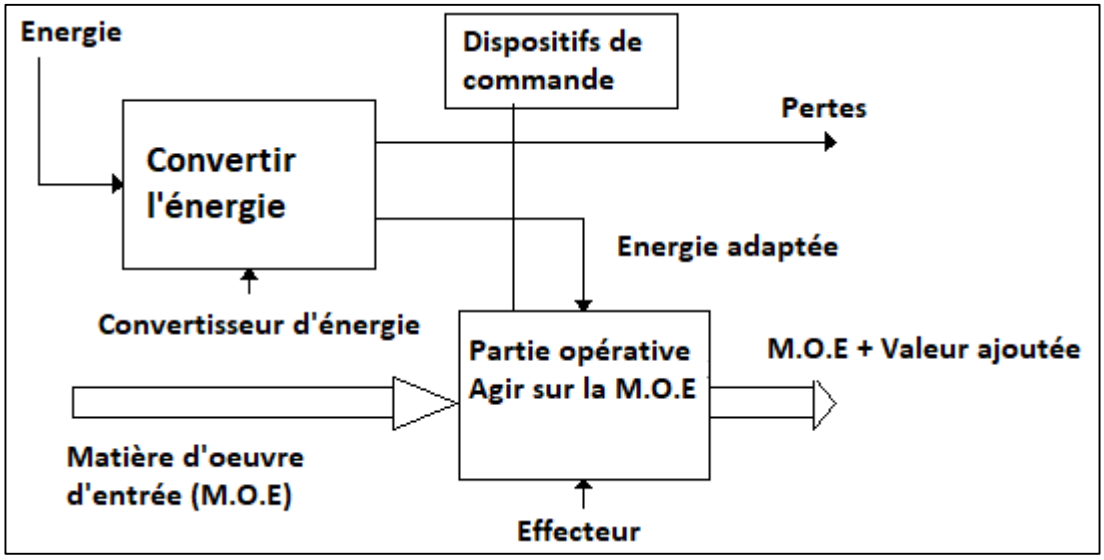


Figure 1 .1 : Structure d'un système de production

L'automatisation permet d'apporter des éléments supplémentaires à la valeur ajoutée. Ces éléments sont exprimables en termes d'objectifs:

- La recherche des coûts plus bas pour le produit par la réduction des frais de main d'œuvre, d'économie d'énergie, d'économie de la matière, etc. …
- La recherche d'une meilleure qualité du produit en limitant le facteur humain et multipliant les contrôles automatisés.
- L'amélioration de la flexibilité de la production.
- La suppression des travaux dangereux ou pénibles et l'amélioration des conditions de travail.
- La réalisation d'opérations impossibles à contrôler manuellement, par exemple des assemblages miniatures, des opérations très rapides, des coordinations complexes.

1.2 Définition d'un système automatisé de production

L'automatisation de la production consiste à transférer tout ou partie des tâches de coordination, auparavant exécutées par des opérateurs humains, dans un ensemble d'objet technique appelé **Partie Commande (PC).**

La partie commande mémorise le savoir-faire des opérateurs pour obtenir la suite des actions à effectuer sur les matières d'œuvre afin d'élaborer la valeur ajoutée. Elle exploite un ensemble d'informations prélevées sur la **partie opérative (PO)** pour élaborer la succession des ordres nécessaires pour obtenir les actions souhaités.

La Partie Commande (en abrégé P.C) :

C'est la partie qui permet de gérer, d'organiser l'enchaînement des actions, dans la suite logique le déroulement ordonné des opérations à réaliser. Elle regroupe les constituants et les composants destinés au traitement des informations (signaux) émises par les capteurs machines de la P. O. et les capteurs opérateurs de la P. R. (C'est le cerveau du système).

L'outil de description s'appelle **GRAFCET** (Graphe de Commande Étape et Transaction).

Exemple de composants de la P.C.

Constituants : automates programmables Industriel (A.P.I.), séquenceurs pneumatiques, micro-ordinateurs, etc.,

La Partie Opérative (en abrégé P.O) :

C'est la partie visible du système. Elle comporte les éléments mécaniques du mécanisme avec :

- des ***pré-actionneurs*** (distributeurs, contacteurs), lesquels reçoivent des ordres de la partie commande;
- des ***actionneurs (***vérins-moteurs) qui ont pour rôle d'exécuter ces ordres. Ils transforment l'énergie pneumatique (air comprimé), hydraulique (huile sous pression) ou électrique en énergie mécanique. Ils se présentent sous différentes formes comme :
  - **Moteurs**: hydraulique, pneumatique, électriques,
  - **vérins** : linéaires (1 ou 2 tiges) rotatifs, sans tige;
- des ***capteurs*** qui informent la partie commande de l'exécution du travail. Ils existent sous différents types comme :
  - Capteurs mécaniques, pneumatiques ou électriques;
  - Capteurs magnétiques montés sur es vérins,
  - Capteurs pneumatiques à chute de pression.

Dans un système automatisé de production, ce secteur de détection représente le service de surveillance et renseignement du mécanisme. Il contrôle, mesure, surveille et informe la PC sur l'évolution du système.

Exemple de composants de la P. O.

Actionneurs : vérins, moteurs, résistances chauffantes, etc.
Pré actionneurs : distributeurs, contacteurs, relais, etc.
Capteurs-machines : fins de course de vérins, détecteurs de position, etc.

Il est difficile, dans la pratique, d'intégrer dans une partie de commande (PC) la totalité des savoir-faire humains. En effet, l'automatisation reste souvent partielle c'est-à-dire certaines tâches restent confier à des intervenants humains. Ces tâches peuvent être classées en deux catégories: **Conduite** et **Surveillance**.

- **Conduite** : Cette catégorie regroupe les opérations de mise en marche du système, d'initialisation, de spécifications des consignes de fonctionnement, etc.
- **Surveillance** : Le modèle de fonctionnement de la partie commande (PC) (choisi par le concepteur) correspond à un ensemble de situations prévues c'est-à-dire retenues par le concepteur parmi un ensemble de situations possibles. De ce fait, il est indispensable de pouvoir faire face à des situations non prévues (non retenues pour des raisons économiques compte tenu de leur faible probabilité d'apparition). A ce niveau, seul l'opérateur est appelé à intervenir et à prendre les décisions requises par cette situation. Il assure donc une fonction de surveillance.

La Partie Relation (en abrégé P.R) :

C'est la partie qui permet le dialogue entre l'homme et la machine, elle regroupe les capteurs opérateurs et les composants de signalisation visuels et / ou sonores. Le pupitre de commande sert de support aux éléments de la P. R. Sa complexité et sa taille dépendent de l'importance du système. Il regroupe les différentes commandes nécessaires au bon fonctionnement du procédé : marche-arrêt, arrêt d'urgence, marche automatique, marche cycle/cycle...

Exemple de composants de la P.R.

Capteurs-opérateurs : boutons poussoirs, interrupteurs, commutateurs, etc.
Composants de signalisation : voyants lumineux, gyrophares, klaxon, etc.
Composants de visualisation : écrans vidéo des terminaux et des moniteurs, etc.

L'outil de description s'appelle « **GEMMA** » (Guide d'Étude des Modes de Marche set Arrêts). Ces outils graphiques (GRAFCET et GEMMA) sont utilisés également par les techniciens de maintenance, pour la recherche des pannes sur les SAP (Système Automatisé de Production).

Pendant le fonctionnement, un dialogue continu s'établit entre les trois secteurs du système, permettent ainsi le déroulement correct du cycle défini dans le cahier de charges.

Le schéma suivant détaille la structure d'un système automatisé de production :

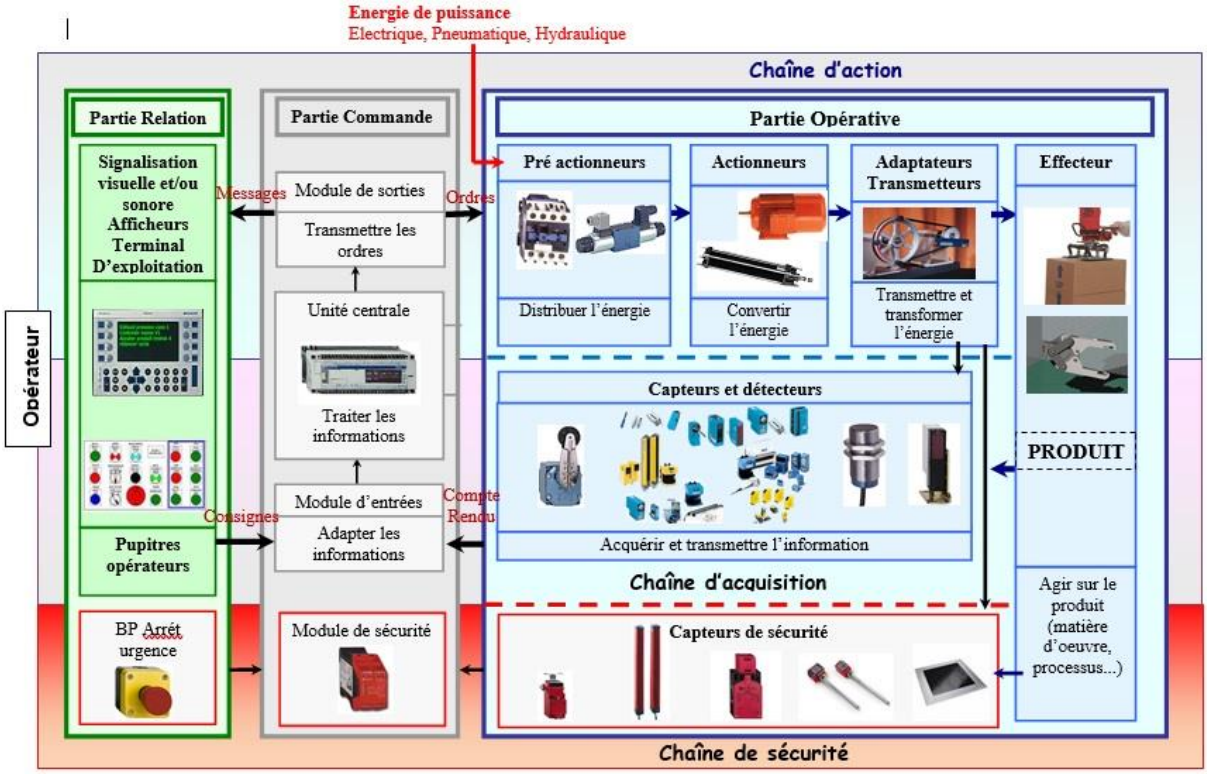


Figure 1 .2 : Schéma de principe d'un système automatisé de production

1.3 Définitions : capteur, pré actionneur, actionneur

Ces trois éléments font parties de la partie opérative, ils sont indispensables à l'automatisation d'un système de production.

a) Capteur et interface :

Un capteur est un élément de la partie opérative qui permet de recueillir des informations et de les transmettre à la partie commande. Les capteurs sont choisis en fonction des informations qui doivent être recueillies (température, son, lumière, déplacement, position).

Afin de pouvoir être traités par la partie commande, les signaux issus de capteurs placés sur le processus sont parfois conditionnés par une électronique d'interface (traitement d'image, mise en forme des signaux, amplification…)

<table>
<tr><th colspan="2">Partie opérative</th><th rowspan="2">Partie commande</th></tr>
<tr><th>Capteur</th><th>Interface</th></tr>
<tr><td> 
Valeur ohmique en fonction de la température
Capteur de température de type PT100 (100Ω à 0°C)</td><td> 
Signal électrique de type 4-20mA
Convertisseur PT100/4-20mA</td><td> 
A.P.I
(avec carte entrée analogique 4-20mA)</td></tr>
<tr><td colspan="2"> 
Signal électrique 4-20 mA en fonction de la pression
Capteur de pression de type différentiel</td><td> 
A.P.I
(avec carte entrée analogique 4-20mA)</td></tr>
</table>

b) Actionneur :

Un actionneur est un élément de la partie opérative qui est capable de produire une action physique tel qu'un déplacement, un dégagement de chaleur, une émission de lumière ou de son à partir de l'énergie qu'il a reçu.

Exemples :

- Un **moteur élec**trique, produit un mouvement de rotation à partir de l'énergie électrique reçue.
- Un **vérin pneum**atique, produit un mouvement de translation à partir de l'énergie pneumatique reçue.
- Un **thermoplongeur**, produit un dégagement de chaleur à partir de l'énergie électrique reçue.
- Une **électrovanne**, produit un mouvement à partir de l'énergie électrique reçue.

c) Pré actionneurs

Les actionneurs sont commandés par des systèmes électriques de commande intermédiaires appelés « pré actionneurs ». Ils sont indispensables pour les adaptions électriques (tension, intensité). En effet, un API ne peut mettre en œuvre un moteur de 400kw triphasé seul, il faut un pré actionneur entre l'API et ce moteur.

Ces pré actionneurs mettent en œuvre des circuits d'électronique de puissance, d'électronique du signal analogique et d'électronique numérique.

Un pré actionneur a pour fonction de transformer l'énergie issue d'une source (réseau électrique, compresseur pneumatique,…) en une énergie adaptée à l'actionneur pour un mouvement précis.

Exemples :

- **Contacteur** et **variateur de vitesse** pour commander un moteur électrique.
- **Distributeur électro**pneumatique pour commander un vérin pneumatique.
- **Gradateur de puissance** pour commander un élément chaffant.

Partie commande	Partie opérative	
	Pré actionneur	Actionneur
 ***A.P.I (avec sortie analogique 0-10 V ou 4-20mA)***	 ***Variateur de vitesse***	 ***Moteur asynchrone***
 ***A.P.I (avec sortie analogique 0-10 V ou 4-20mA)***	 ***Gradateur de puissance***	 ***Thermoplongeur***

1.4 Différents types de commande

Ils existent différents types de commandes pour les systèmes automatisés de production :

a) Système automatisé combinatoire

Ces système n'utilisent aucun mécanisme de mémorisation (ils n'ont pas de mémoire) et à une combinaison des entrées correspond une seule combinaison des sorties. La logique associée est appelée logique combinatoire. Les outils utilisés pour les concevoir sont l'algèbre de Boole, les tables de vérité, les tableaux de KARNAUGH.

Les systèmes automatisés utilisant la technique du «combinatoire» sont aujourd'hui très peu utilisés. Ils peuvent encore se concevoir sur des mécanismes simples où le nombre d'actions à effectuer est limité. Ils présentent en plus l'avantage de n'utiliser que très peu de composants (vérins, distributeurs, capteurs, cellules).

b) Système automatisé séquentiel

Ces systèmes sont les plus répandus sur le plan industriel. Le déroulement du cycle s'effectue étape par étape.

A une situation des entrées peut correspondre plusieurs situations de sortie. La sélection d'une étape ou d'une autre dépend de la situation antérieure du dispositif. La logique associée est appelée « **logique séquentielle** ». Elle peut être (**logique câblée** ou **logique programmée**).

La logique câblée

La technologie câblée consiste à raccordé des modules par des liaisons matérielles selon un schéma fourni par la description. Ces modules peuvent être électromagnétiques, électriques, pneumatiques ou fluidiques.

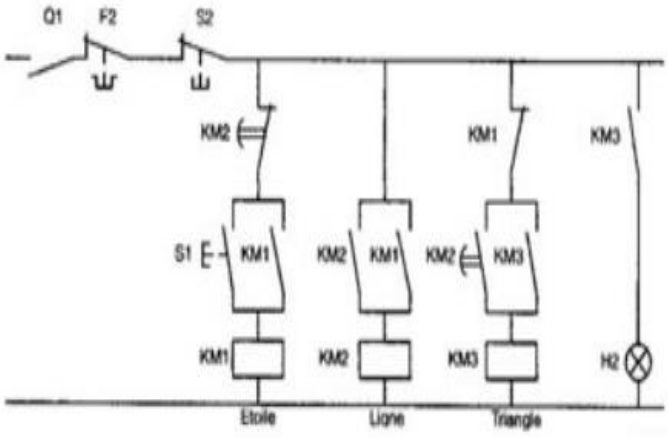
En électricité ou en électronique, les liaisons sont faites par câble électrique. En pneumatique et fluidique, il s'agit de canalisations reliant les différentes composants.

Les outils câblés sont utilisés dans l'industrie où l'on apprécie leurs qualités éprouvées; à savoir la **rapidité** et le **parallélisme**. Ils souffrent cependant d'un certain nombre de limitations parmi lesquelles nous citons:

- Leur encombrement (poids et volume),
- leur manque de souplesse vis-à-vis de la mise au point des commandes et de l'évolution de celles-ci (améliorations, nouvelles fonctions, modification, etc.): Toute modification impose la modification de câblage voire un changement de composants,
- Leur difficulté de maîtriser des problèmes complexes,
- La complexité de recherche des pannes et donc du dépannage,
- Leur coût élevé pour les systèmes complexes.

Exemple : Schéma de commande d'un moteur électrique en logique câblée

Entrées :
Km1 : Contact de KM1
S2 : BP arrêt
S1 : BP marche
Q1 : Sectionneur fermé
F2 : Contact associé au relais thermique
Sorties :
KM1 : Contacteur étoile
KM2 : Contacteur Ligne
KM2 : Contacteur étoile
H2 : voyant fin de démarrage



La logique programmée

La logique programmée correspond à une démarche séquentielle, seule une opération élémentaire est exécutée à la fois, c'est un traitement série. Le schéma électrique est transcrit en une suite d'instruction qui constitue le programme. En cas de modification des équations avec les mêmes accessoires, l'installation ne comporte aucune modification de câblage seul le jeu d'instructions est modifié.

Si un circuit est réalisé en logique programmée, il utilisera moins de composants puisque ceux-ci réalisent directement les fonctions logiques désirées. Un circuit ayant moins de composants sera habituellement moins coûteux à concevoir, réaliser et distribuer. La réduction du nombre de composants électroniques tend aussi à augmenter la fiabilité des circuits et à réduire la consommation énergétique.

La technologie programmable consiste à substituer le fonctionnement de l'automatisme par un programme chargé sur un constituant programmable c'est-à-dire des machines destinées à traiter de l'information (un API dans la plus part des cas). Leur utilisation en gestion et en calcul scientifique est connue. Alors, les applications techniques relèvent de l'informatique industrielle. En termes d'avantage, nous citons:

- Moins de câble et d'encombrement
- Fiabilité de l'automatisme.
- Facilité de modification
- Flexibilité
- Résolution des problèmes complexes.

Cependant, elle souffre de problème de **parallélisme**. Le constituant programmable peut être soit un micro-ordinateur, soit une carte électronique ou bien un automate programmable.

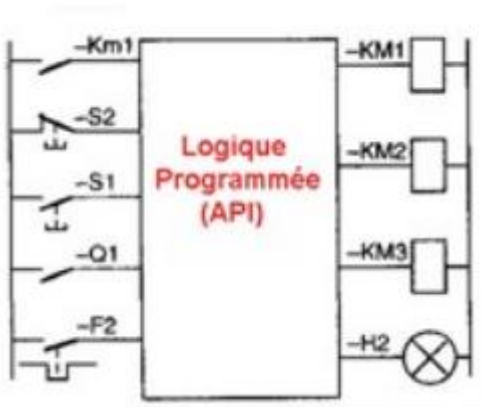
L'automate simplifie grandement le schéma de la logique câblée prenant en compte tout ce qui est extérieur à la programmation, comme les voyants. Il sert pour se substituer à une partie commande complexe qu'on programmera dans un automate.
Automatiser avec les automates pour :

- réduire les coûts d'ingénierie
- réduire les coûts de maintenance

Exemple : Schéma de commande d’un moteur électrique en logique programmée

Entrées :
- **Km1** : Contact de KM1
- **S2** : BP arrêt
- **S1** : BP marche
- **Q1 :** Sectionneur fermé
- **F2** : Contact associé au relais thermique

Sorties :
- **KM1** : Contacteur étoile
- **KM2** : Contacteur Ligne
- **KM2** : Contacteur étoile
- **H2** : voyant fin de démarrage



c) Choix de la logique de réalisation des SAP

Pour choisir la meilleure technologie pour un automatisme donné, on utilise généralement deux critères: la **faisabilité** et l'**optimisation**.

Critère 1: Faisabilité :

« La réalisation avec une technologie donnée est-elle possible ou non ? »

Critère 2: Optimisation :

« La réalisation avec une technologie donnée conduit-elle au coût global le plus bas ? »

La réponse à ces deux questions est illustrée sur le graphe suivant :

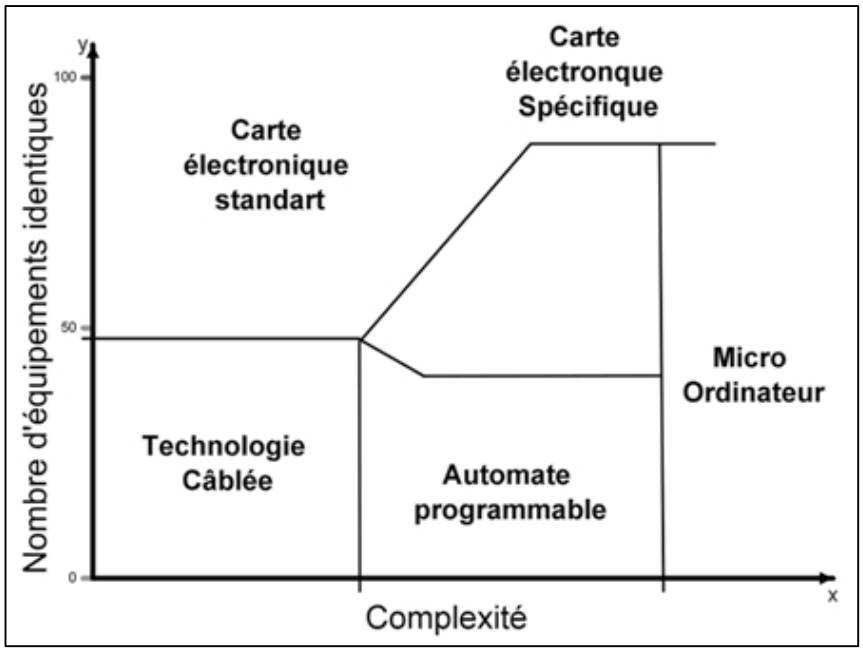


Figure 1.3 : Graphe choix PC pour SAP

Le choix de la technologie peut être résumé comme suit :

- Le micro-ordinateur est utilisé surtout dans les cas des systèmes complexes (nombre d'E/S assez grand, calculs sur les réels, etc.).
- Les cartes électriques spécifiques sont utilisées pour résoudre un problème bien défini. Elles sont appelées uniquement dans le cas où le nombre d'exemplaire est supérieur à 100 car leur coût est assez élevé.
- Les cartes électroniques standards sont utilisées dans les automatismes grand public : distributeurs, parking, etc.
- Les automates programmables sont utilisés dans les cas des systèmes complexes, flexible et évolutifs.

2. Architecture d'un système automatisé de production

2.1 Architecture centralisée

Historique de l'API :

C'est **MODICOM** qui créa en 1968, aux USA, le premier automate programmable. Son succès donna naissance à une industrie mondiale qui s'est considérablement développé depuis.

L'automate programmable représente aujourd'hui l'intelligence des machines et des procédés automatisés dans l'industrie, des infrastructures et du bâtiment.

Evolution de l'API :

✓ Dans les années 80, les automates deviennent de plus en plus gros :
- De plus en plus d'entrées/sorties
- Un automate commande plusieurs machines (**Architecture centralisée**)

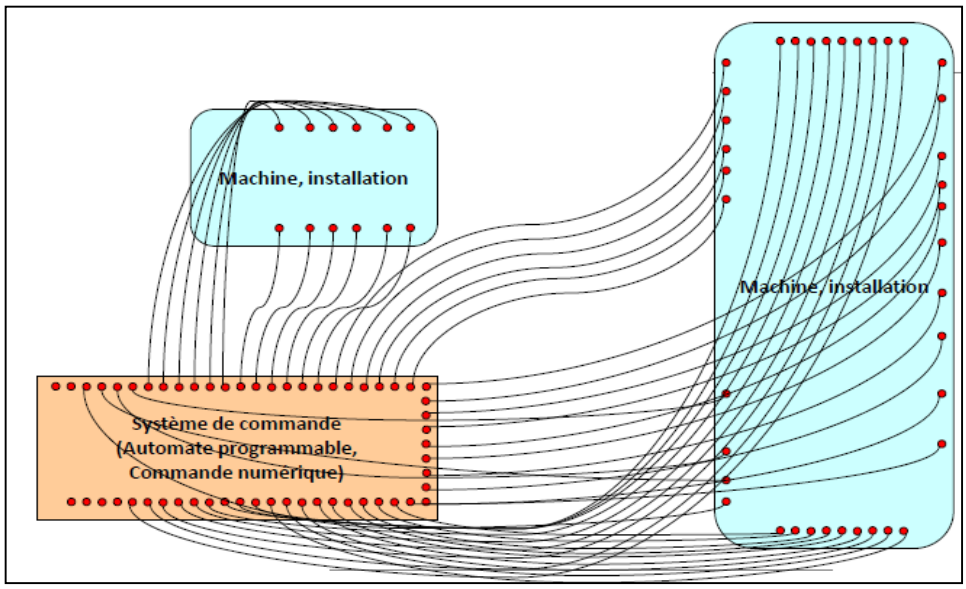


Figure 1 .4 : Schéma de principe d'une architecture centralisée

2.2 Architecture décentralisée

Dans les années 90, on utilise des automates plus petits reliés entre eux par des réseaux de communications (**Architecture décentralisée**).

Les objectifs principaux de la décentralisation sont :

- Simplifier et modulariser le câblage (Transmission de nombreuses informations par un bus de terrain et remplace les nombreuses liaisons filaires par un câble unique).
- Baisser les coûts.

Deux approches utilisées

- Décentralisation de la périphérie seulement.
- Décentralisation du contrôle complet.

Dans une architecture décentralisée, on parle souvent de **coupleur** et de **contrôleur :**

- **Coupleur** (Gère seulement des entrées sorties).
- **Contrôleur** (Permet en plus d'exécuter un programme localement).

La décentralisation de la périphérie seulement :

- Le programme est toujours hébergé par un contrôleur central.
- La périphérie décentralisée est gérée par un coupleur électronique qui traduit :
    - Les messages du bus de terrain en signaux pour les actuateurs.
    - Les signaux capteurs en messages sur le bus de terrain.

Le schéma de principe d'une architecture décentralisée est donné par la figure suivante :

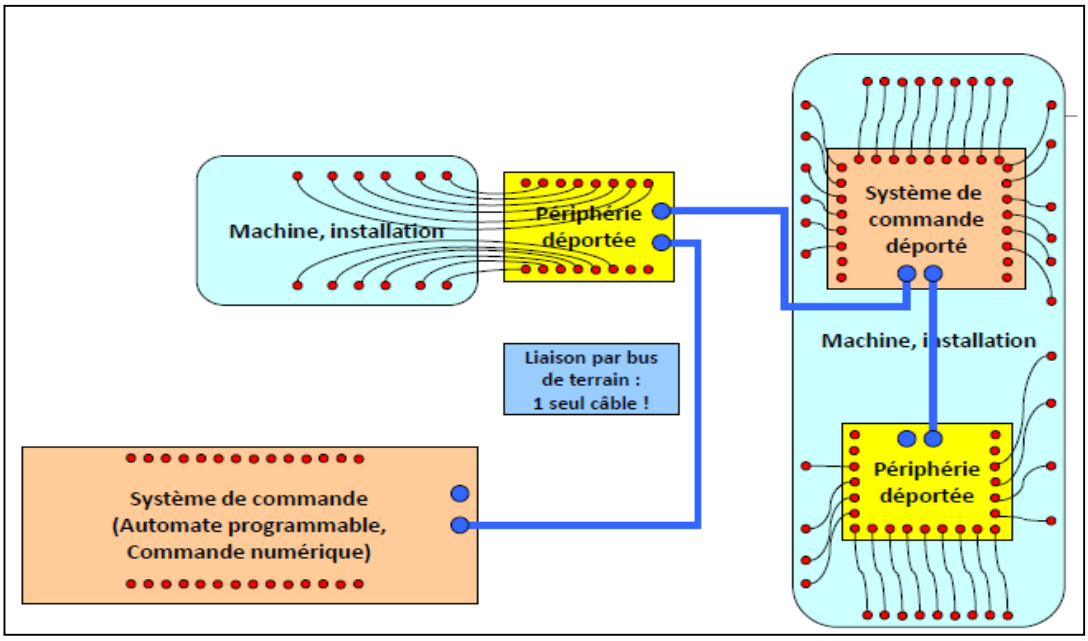


Figure 1 .5 : Schéma de principe d'une architecture décentralisée

De nos jours, les entrées/sorties sont aussi décentralisées, on parle de modules déportés. L'automate possède de plus en plus de ports de communication et de moins en moins d'entrées/sorties en local.

3. Présentation d'un automate programmable industriel

3.1 Introduction

Les API sont particulièrement bien adaptées aux problèmes de commande séquentielle et d'acquisition des données. Ils autorisent la réalisation aisée d'automatismes allant de quelques dizaines jusqu'à plusieurs milliers d'entrées/sorties.

Les API se caractérisent par :

- ✓ une technologie adaptée à l'environnement industriel et éliminant tous travaux de fileries importants,
- ✓ une programmation qui utilise le langage de l'automaticien (et non celui de l'informaticien),
- ✓ des possibilités de simulation et de visualisation qui apportent à l'utilisateur une aide efficace à la mise au point et à l'exploitation (modification aisée de l'automatisme).

Une puissance de traitement et un ensemble de cartes spécialisées permettant un développement aisé d'applications standards : communication, asservissement d'axes, régulation...

3.2 Architecture matérielle des API

Un Automate Programmable se présente sous la forme d'un ou plusieurs racks dans lesquels viennent s'enficher les différents modules fonctionnels :

- L'alimentation 110/220 VCA ou 24/48 VCC,
- L'unité centrale de traitement à base de microprocesseurs,
- Des cartes d'entrées/sorties logiques (TOR),
- Des cartes d'entrées/sorties analogiques (ANA),
- Des cartes de comptage rapide (CPT),
- Des cartes de communication (CPL),
- Des cartes dites " intelligentes " ou "métier" (réseaux, asservissement, régulation, ...).

La figure suivante, donne un exemple d'architecture matérielle d'un API.

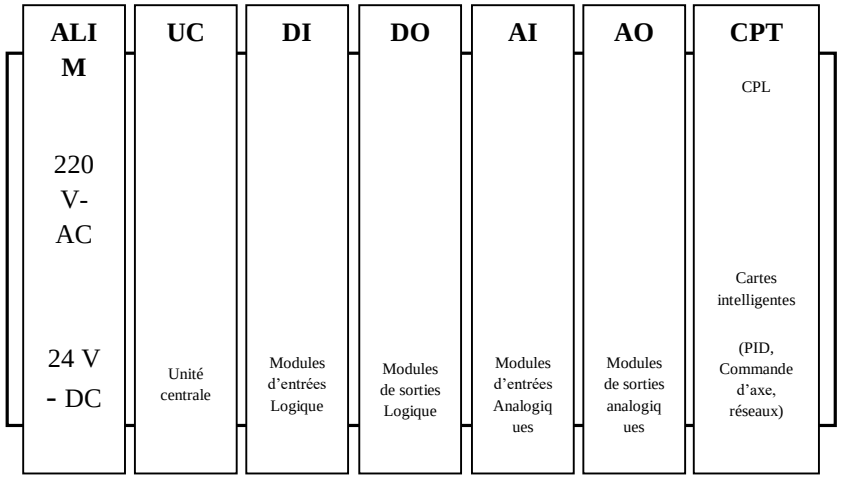


Figure 1 .6 : Exemple d'architecture matérielle d'un API

Chaque module d'entrées/sorties comporte un bornier de raccordement et un ensemble de visualisation de l'état logique de chaque voie. Cette organisation modulaire permet une grande souplesse de configuration adaptée aux besoins de l'utilisateur ainsi qu'un diagnostic et une maintenance aisés.

a) Structure interne des API

La figure suivante, présente les différents éléments de base de tout API, à savoir :

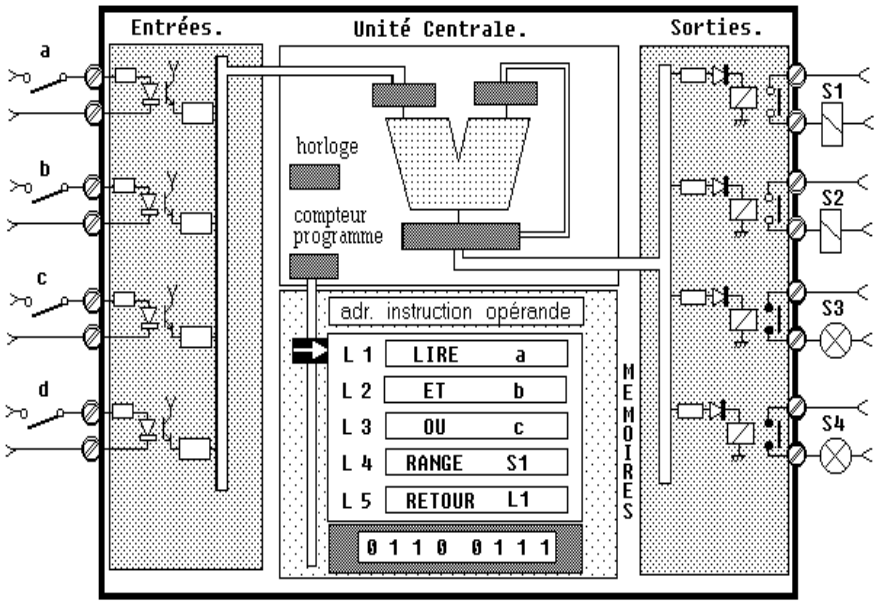


Figure 1 .7 : Eléments de base d'un API

Le PROCESSEUR :

A base de microcontrôleurs ou de microprocesseurs, il gère le fonctionnement de l'unité centrale, le processeur effectue des opérations logiques, de temporisation, de comptage ou de calcul.

Sa fonction principale consiste à rechercher en mémoire, via les BUS, chaque instruction qu'il va exécuter, ainsi que les données internes ou externes sur lesquelles porte cette instruction.

La MEMOIRE CENTRALE :

En règle générale la mémoire d'une CPU se compose :

- ✓ D'une zone de mémoire programme : La mémoire Programme, de type RAM secourue ou EEPROM, contenant l'ensemble des instructions,
  - Dans la phase d'étude et de mise au point cette mémoire doit être reprogrammable (RAM, EAPROM Electrically Alterable)
  - Dans la phase d'exploitation la mémoire doit être sauvegardée par batterie ou être non volatile (EEPROM, Flash)
- ✓ D'une zone de mémoire de données : La mémoire de données, de type RAM secourue, contenant les bits internes, les tempos, les compteurs, etc...
  - Pour sauvegarder l'état des E/S, compteurs, variables internes…
  - Attention, il peut y avoir plusieurs zones. Bien regarder les spécificités indiquées par le constructeur. Il faut connaitre le comportement de cette zone en cas de coupure de l'énergie.

L'INTERFACE CONSOLE :

Elle permet le dialogue entre l'UC et les divers périphériques, notamment la console de programmation, nécessaire à l'écriture et la mise au point des programmes.

Les MODULES D'ENTREES :

Ils permettent la prise en compte par l'UC des informations venant du processus à commander.

Les MODULES DE SORTIES :

Ils permettent d'envoyer les ordres élaborés par l'UC vers le processus à commander.

Remarque :

Les 3 premiers éléments (Processeur, Mémoire, Interface console) tiennent généralement sur une seule et même carte constituant l'Unité Centrale (UC) de l'automate programmable.
Les entrées-sorties industrielles se présentent sous forme de cartes répondant à un certain format et utilisant une certaine connectique (malheureusement très spécifique à chaque constructeur) afin de venir se connecter aux bus de l'automate programmable.

b) Cartes d'entrées logiques TOR

La figure 1-8 présente les éléments constituants un module d'entrées logiques. Les cartes d'entrées logiques se caractérisent par le nombre de voies supportées et la nature de la tension d'activation des voies. Chaque entrée logique arrive sur la carte d'entrée en passant par un étage d'adaptation, un étage d'isolation (afin d'assurer un isolement total de l'ordre de 4000 V, entre les tensions industrielles et les signaux émis vers le bus de données) et un étage d'anti rebond.

Une fois enfichée sur le châssis de l'API, chaque voie du module d'entrée possède une adresse qui sert pour la lecture de l'état logique de la voie concernée.

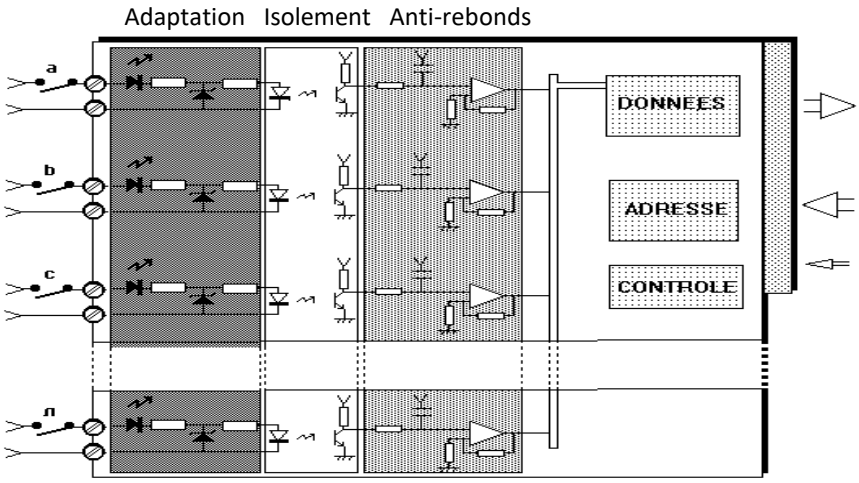
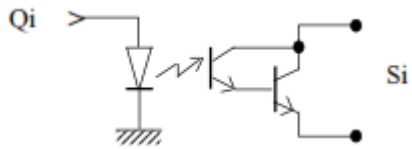


Figure 1-8: Eléments de base d'un module d'entrées logiques

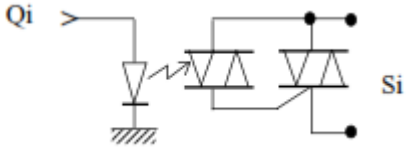
c) Cartes de sorties logiques TOR

Chaque carte comporte généralement 8, 16 ou 32 sorties logiques. L'état logique de chaque sortie est mémorisé par une bascule binaire (bistable) avant d'être envoyé en sortie via un étage d'adaptation. Cet étage de sortie permet d'une part d'assurer un isolement total (de l'ordre de 4000 V) avec les signaux industriels et d'autre part de fournir la puissance voulue en sortie.
On distingue trois types de sorties :

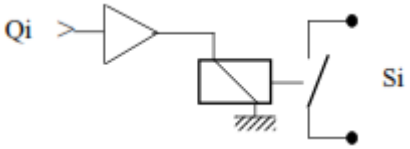
- ✓ statiques à transistor collecteur ouvert 5 à 60 V continu, 1 à 2 A, isolement par photo-coupleur,



- ✓ statiques à thyristor ou triac 110 à 220 V alternatif, 1 à 2 A, isolement par photo-coupleur,



- ✓ à relais, contact 1 à 2 A, isolement par l'entrefer, plus parfois par photo-coupleur.



La figure 1-9 donne un exemple de module de sortie TOR à relais.

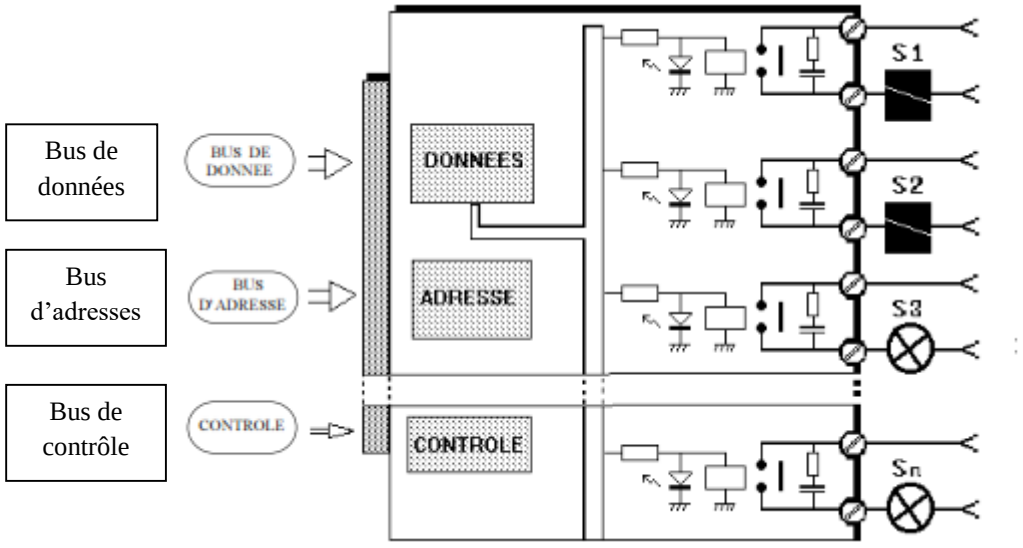


Figure 1-9: Eléments de base d'un module de sorties logiques à relais

Une fois enfichée sur le châssis de l'API, chaque voie du module de sortie possède une adresse qui sert pour l'envoi de la commande correspondante à une voie concernée.

d) Cartes d'entrées/sorties analogiques

Chaque carte comporte plusieurs entrées ou sorties analogiques, tension (0/10 V ou -10/10 V par exemple), ou courant (4 à 20 mA) dont la chaine d'acquisition peut correspondre au schéma fonctionnel donné par la figure 1-5.

La chaine d'acquisition et de traitement répond périodiquement (exemple : toutes les 100 ms) à la procédure suivante :

1°. Acquisition de la mesure via la carte d'entrées analogique. Pour cela le CAN (Convertisseur Analogique Numérique) fournit une valeur numérique codée sur 12 ou 16 bits.

2°. Le programme de traitement (algorithme de calcul) détermine sous forme numérique la valeur à fournir en sortie, en tenant compte des différents paramètres :

-La valeur de la mesure,
-La valeur de la consigne,
-Le type d'algorithme (PID,...).

3°. Emission par la carte de sortie analogique du signal de commande. Pour cela, le CNA (Convertisseur Numérique Analogique) convertit la valeur numérique de sortie sous forme analogique.

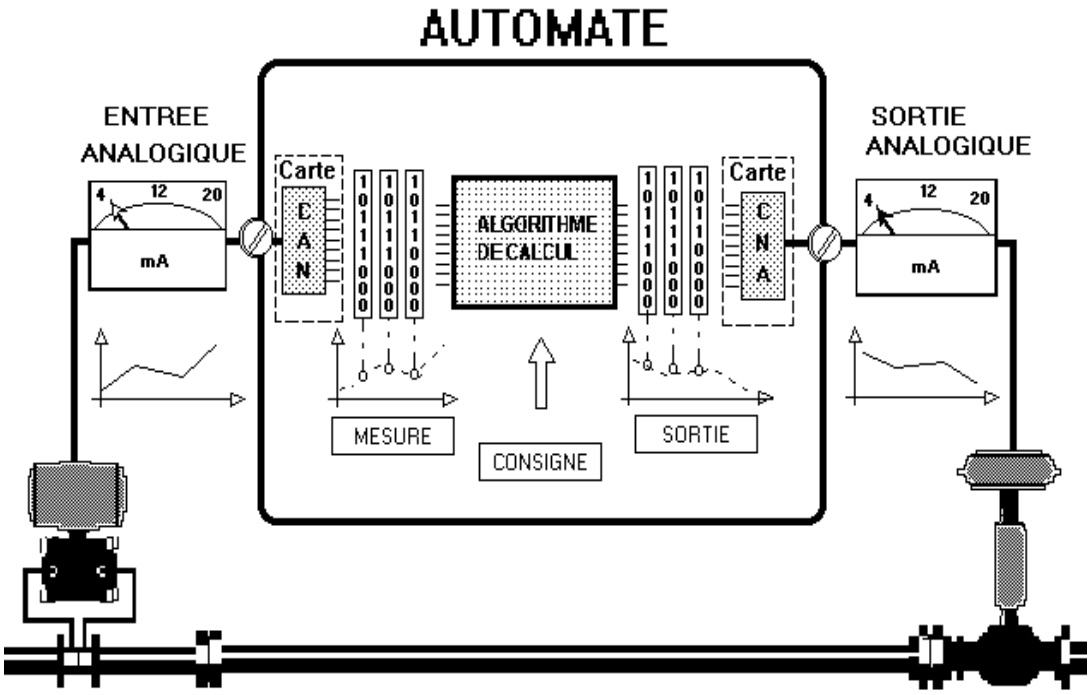


Figure 1-10: Schéma fonctionnel des modules d'entrées / sorties analogiques

e) Cartes d'entrées-sorties complexes

Toutes ces cartes dites "intelligentes" disposent en plus des adaptateurs d'entrée et de sortie, d'un microprocesseur assurant un traitement local de certaines fonctions d'automatismes.

Ceci permet, d'une part d'éviter un développement souvent fastidieux de l'application, d'autre part de réduire considérablement la place mémoire et le temps d'exécution au niveau de l'UC de l'automate programmable.

La structure fonctionnelle générale d'une carte d'entrée/sortie complexe est donnée par la figure 1-11.

La mise en œuvre d'une telle carte se résume souvent à la simple écriture dans des registres de données :

- d'une part, d'un code de fonction précisant le type et les modalités du traitement à effectuer,
- d'autre part, des paramètres d'exploitation nécessaires à ce traitement.
- Eventuellement, à la lecture des résultats ou d'état du traitement effectué.

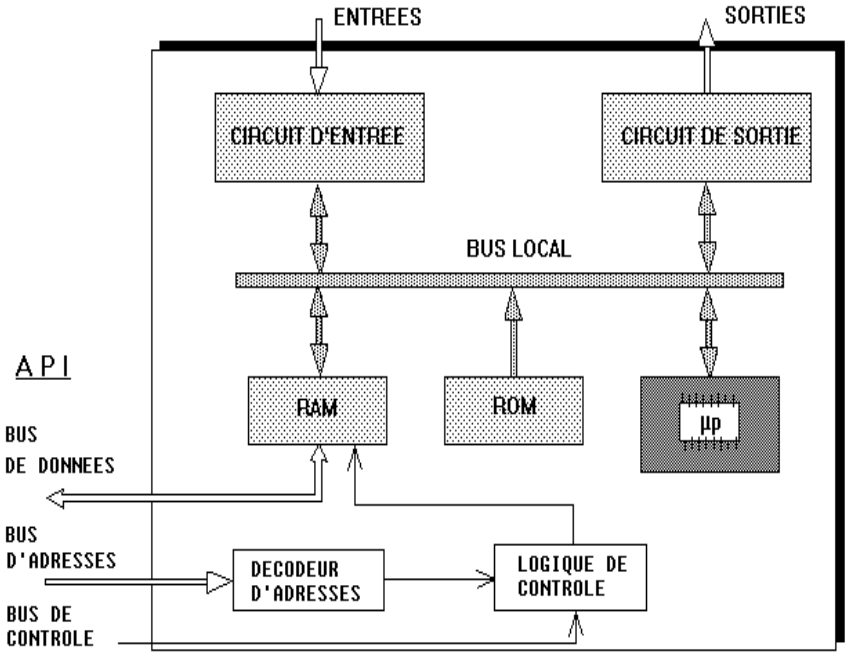


Figure 1-11: Schéma fonctionnel des modules d'entrées / sorties complexes

Exemples :

Carte de régulation PID

Entrées analogiques (4 – 20 mA)
Sorties analogiques (4 - 20 mA)
Son rôle est de réaliser une ou plusieurs boucles de régulation simples ou complexes (mixte, cascade) avec éventuellement des opérateurs associés ($\sqrt{\ }$, $\sum$,…, limites haute et basse, ...)

Carte de commande d'axes

Entrées analogiques (0 – 10 V) ou numérique provenant d'un codeur absolu,
Sorties analogiques (0 – 10 V),

Son rôle est d'asservir un axe (portique ou robot) en vitesse et en position en déterminant la consigne à envoyer à l'électronique de commande du moteur.

Modules de couplage

Entrées série asynchrone, Sorties série asynchrone,

Son rôle est l'échange d'information (bits et mots) entre divers automates et ordinateurs interconnectés en réseau local. Ces modules qui permettent de connecter l'automate à d'autres systèmes de traitement:

- Console de programmation et de test,
- Coupleurs avec la console de programmation,
- Extension de bus,
- Etendre le bus de l'automate pour des configurations multi-châssis (longueur limitée),
- Communications à distance par lignes série (téléphoniques, coaxiales, fibres optiques…)
- Décentralisation des châssis entrées / sorties industrielles sur des distances importantes (ordre du km)

3.3 Architecture logicielle des API

a) Description du programme

Un programme d'Automate Programmable est constitué d'un ensemble de **tâches cycliques**, **périodiques** ou **évènementielles**. En outre, une tâche peut éventuellement faire appel à un ou plusieurs sous programmes (S.P).

Une tâche ou un sous-programme est constituée d'une suite d'instructions symboliques ou graphiques.

Enfin, chaque instruction comporte un code opération précisant le type de traitement à effectuer et éventuellement une ou plusieurs opérandes. Ces opérandes permettent de préciser les données sur lesquelles portent les instructions et sont identifiées par un repère (adresse d'une entrée, d'une sortie ou d'une variable interne).

b) Tâches cycliques (asynchrones)

Pour ce type de tâche l'unité centrale exécute l'ensemble des instructions constituant la (ou les) tâche cyclique d'une manière **asynchrone**. Ainsi, suivant les instructions traitées, ces tâches peuvent avoir une durée d'exécution variable d'un cycle à l'autre.

c) Tâches périodiques (synchrones)

Alors que dans la majorité des applications d'automatismes, le fait que le traitement ne soit pas exécuté à une période fixe ne pose pas réellement de problèmes, il n'en va pas de même pour certaines applications.

C'est le cas par exemple en régulation, où les acquisitions et le traitement des algorithmes doivent s'effectuer avec une périodicité précise, par exemple toutes les 0,1 secondes. Dans ce cas la tâche est dite **synchrone** et dispose alors d'un niveau de priorité supérieure à celle des tâches cycliques.

d) Tâches événementielles (interruption)

Ce type de tâche s'apparente à la précédente, mais au lieu d'être exécutée périodiquement, elle est appelée suite à un **évènement particulier** :

- INTERNE:

- ✓ Passage STOP → RUN,
- ✓ Passage RUN → STOP,
- ✓ Changement d'état d'une entrée, d'une sortie ou d'une variable mémoire,

- EXTERNE :

- ✓ Reprise secteur,
- ✓ Interruption envoyée par une carte d'E/S spéciale, défaut d'un périphérique,
- ✓ etc.

Ces tâches ont généralement le niveau de priorité maximum. Elles sont surveillées par un chien de garde et mises en défaut en cas de dépassement du temps d'exécution imparti.

e) Sous programmes

Lorsqu'une tâche doit être utilisée plusieurs fois, on peut l'organiser sous forme d'un sous-programme.

Chaque fois que le processeur rencontre une instruction d'appel de sous-programme, il exécute la séquence d'instructions correspondante. A la fin du sous-programme, une instruction de retour ramène alors le processeur à la ligne suivant celle de l'appel.

En outre, un sous-programme peut lui-même faire appel à d'autres sous-programmes.

La figure 1-12 donne un exemple de traitement des sous programmes.

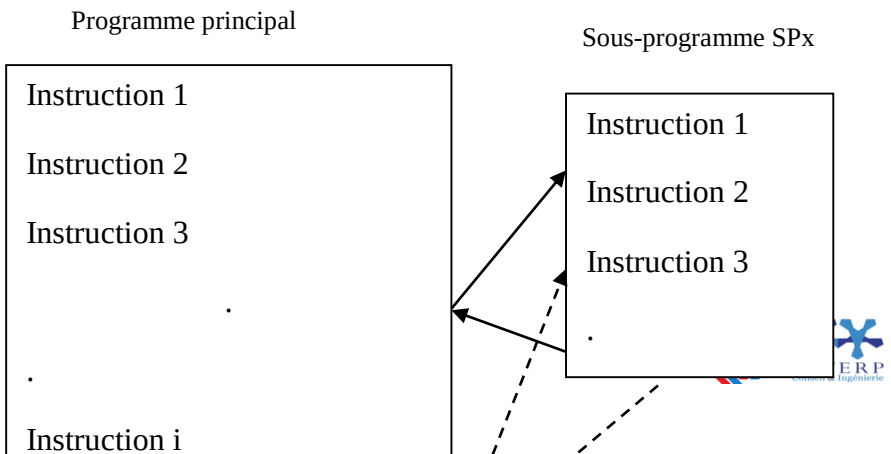


Figure 1-12: Exemple de traitement d'un sous-programme

f) Cycle automate

Un API exécute son programme de manière cyclique ou périodique de la manière suivante :

- ✓ Lecture des entrées
- ✓ Traitement du programme
- ✓ Écriture des sorties

Fonctionnement cyclique :

Les cycles s'enchaînent les uns après les autres, le temps de cycle dépend de la durée d'exécution du programme. Cette durée dépend du nombre de calculs et de la puissance de traitement de la CPU.

Fonctionnement périodique:

Il est parfois possible de fixer le temps de cycle pour certaines parties du programme. Dans le jargon informatique, on nomme ce fonctionnement par le terme: «Interruption de programme»

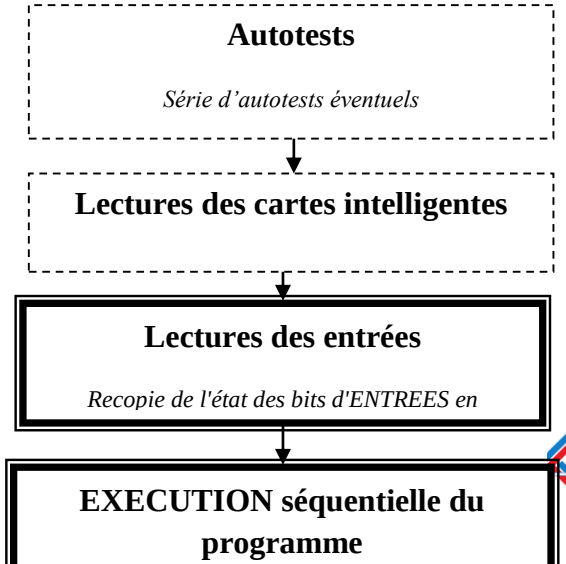


Figure 1-13: Cycle d'exécution d'un programme sur un automate

Le temps d'exécution d'un cycle est contrôlé par une surveillance du temps appelée **chien de garde.** Si le temps de cycle dépasse le chien de garde, la CPU se mettra en défaut. Sur certains modèles d'automates, le temps de surveillance de cycle est réglable.

Dans un API SIEMENS nous retrouvons également ce concept de fonctionnements cyclique et périodique. Chez SIEMENS, **ces différents fonctionnements se nomment des OB (blocs d'organisation)**.

OB1 (Main) pour le fonctionnement cyclique de l'API. D'autres OB internes sont également mis à disposition pour l'utilisateur. (OB de démarrage, d'alarmes ou d'erreurs).

4. Les langages de programmation des API

Les langages de programmation des API sont de nature diverse, relevant d'un choix à priori, en fonction des utilisateurs visés et du type de traitement à effectuer. La norme **CEI 1131-3** a classé ces langages en trois familles:

4.1 Langages littéraux

Langage IL : Liste d'instructions

Ce langage utilise des codes symboliques et peut être représenté sous forme de lignes ou sous forme d'une suite d'instructions. La figure 1-14 donne un exemple de programme écrit en langage LIST.

$$\mathbf{A4.0} = \mathrm{E0.0}.\overline{\mathrm{E0.1}} + \overline{\mathrm{E0.0}}.\mathrm{E0.1} = \mathrm{E0.0} \oplus \mathrm{E0.1}$$

```
U    E   0.0
UN   E   0.1
O
UN   E   0.0
U    E   0.1
=    A   4.0
```

Figure 1-14: Exemple de programme écrit en langage LIST

Langage ST : Littéral structuré

Le langage littéral structuré est un langage évolué de type algorithmique particulièrement adapté à la programmation de fonctions arithmétiques complexes, à la manipulation de tableaux et à la gestion de messages. C'est un langage informatique de même nature que le Pascal, il utilise les fonctions comme if ... then ... else ... (si ... alors ... sinon ...).

4.2 Langages graphiques

Langage LD : langage à contacts (Ladder)

Le langage des API d'origine américaine utilise le symbolisme classique des schémas à relais accompagné de blocs graphiques préprogrammés pour réaliser des fonctions d'automatisme (calculs, temporisations, compteurs, ...).

La figure 1-15 donne un exemple de programme écrit en langage ladder.

$$\mathbf{A4.0} = \mathrm{E0.0}.\overline{\mathrm{E0.1}} + \overline{\mathrm{E0.0}}.\mathrm{E0.1} = \mathrm{E0.0} \oplus \mathrm{E0.1}$$

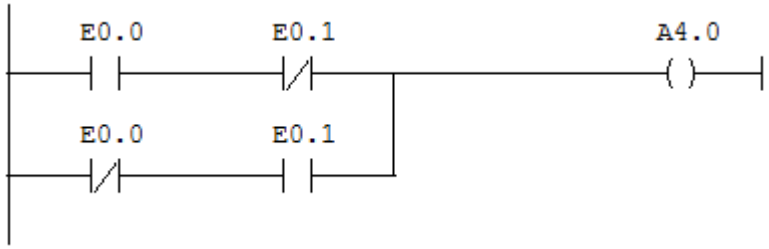


Figure 1-15: Exemple de programme écrit en langage ladder

En milieu industriel, ce type de langage permet l'adoption sans effort particulier d'un API par des utilisateurs de tous niveaux, en particulier par des techniciens d'entretien. Il facilite les opérations de maintenance et de dépannage par la parfaite correspondance avec des circuits

classiques à relais. On peut, par contre, lui reprocher sa rigidité qui pénalise l'utilisateur quant aux fonctions disponibles sur la machine.

Langage FBD : diagramme fonctionnel — logigramme

Sur certains automates, entre autre Siémens, on peut également programmer en logigramme graphique.
La figure 1-16 donne un exemple de programme écrit en langage logigramme.

$$\mathbf{A4.0} = \mathrm{E0.0}.\overline{\mathrm{E0.1}} + \overline{\mathrm{E0.0}}.\mathrm{E0.1} = \mathrm{E0.0} \oplus \mathrm{E0.1}$$

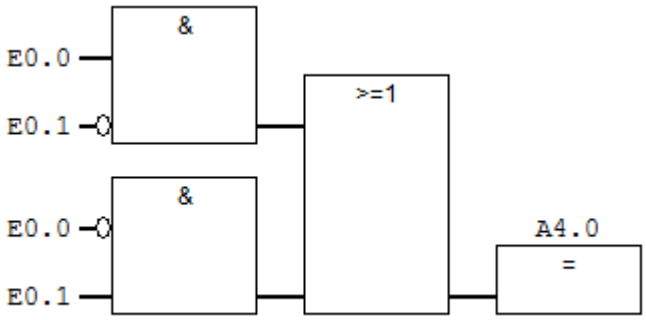


Figure 1-16: Exemple de programme écrit en langage logigramme

4.3 Langage SFC- diagramme fonctionnel en séquence

Le langage SFC (GRAFCET), est utilisé par certains constructeurs d'automates (Schneider, Siemens,…). Il permet une programmation aisée des systèmes séquentiels tout en facilitant la mise au point des programmes ainsi que le dépannage des systèmes.

La figure 1-17 donne un exemple de programme écrit en langage GRAFCET sur un API S7-300.

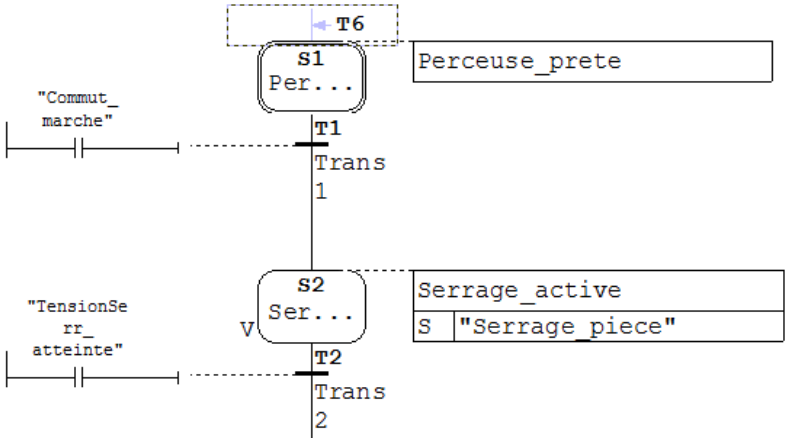


Figure 1-17: Exemple de programme écrit en langage GRAFCET

4.4 Traitements numériques

Outre tous les traitements logiques de base (opérations sur bits), les API disposent de possibilités de traitements numériques de plus en plus performantes (opérations sur mots).

Ces traitements peuvent se répartir en 3 principales familles :

Opérations mathématiques :
- Addition, soustraction,
- Multiplication, division,
- Racine carrée, ...

Opérations logiques sur mots :
- ET logique => forçage à 0 de bits,
- OU logique => forçage à 1 de bits,
- OU exclusif => inversion de bits,

Opérations diverses :

- Blocs texte => Gestion des dialogues,
- Blocs fonctionnels => Temporisation, comparaison, comptage, transfert, etc…
- Blocs fonctionnels => Asservissement d'axes,
- Blocs fonctionnels => Régulation PID,

Ces opérations sont programmées soit en langage littéral, soit en langage ladder via des blocs fonctionnels.
Les traitements numériques portent sur des données pouvant être sous plusieurs formes :

a) Nombres entiers en virgule fixe

Ces nombres sont codés sur 16 bits (en simple précision) ou 32 bits (en double précision). Le bit de poids fort définit le signe (**0 : positif**, **1 : négatif**) et les autres bits définissent la valeur absolue du nombre. Chaque donnée est comprise entre : **(- 32 768)** à (**+ 32 767**) sur 16 bits, **(- 2 147 483 648**) à **(+ 2 147 483 647**) sur 32 bits.

La représentation en virgule fixe d'un nombre entier est **encodée en complément à 2** dans le cas des nombres négatifs.

Pour représenter un nombre binaire en complément à deux, on inverse tous les bits de la valeur absolue du nombre et on ajoute au résultat obtenue + 1 :$(-A = \bar{A} + 1)$

Exemple N°1: représentation du nombre 59 sur 16 bits

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1
Bit de Signe	Combinaison binaire de la valeur absolue du nombre **Nombre représenté = + 59 = $(0000000000111011)_2$**														

Exemple N°2: représentation du nombre -59 sur 16 bits

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	1
Bit de Signe	Combinaison binaire en complément à deux de la valeur absolue du nombre **Nombre représenté = - 59= $(1111111111000101)_2$**														

Figure 1-18: Exemple de représentation d'un nombre entier à virgule fixe

L'intérêt de ce type de représentation tient au fait que lorsque on additionne ou on soustrait 2 nombres ainsi codés, on obtient directement le bon résultat, quels que soient les nombres additionnés (positifs ou négatifs).

b) Nombres réels en virgule flottante

Pour représenter un nombre réel dans une machine informatique ou un système électronique, il a été nécessaire de trouver une écriture compatible avec la taille mémoire qu'on lui accorde. On a donc privilégié la notation scientifique et l'écriture en virgule flottante.

La notation est dite flottante car elle consiste à faire glisser les nombres significatifs à droite ou à gauche de sorte que le nombre tienne dans le format.

La notation flottante comprend trois composantes:

Le **signe**, la **mantisse**, et l'**exposant** de la puissance de 2 comme indique la figure 1-19.

Bit de Signe	Exposant	Mantisse
(0 -› positif, 1 -› négatif)	exposant permettant d'obtenir une représentation normalisée du nombre	partie fractionnaire de la représentation normalisée

Figure 1-19: Représentation d'un nombre à virgule flottante

La mantisse est exprimée en binaire complémenté à deux pour les nombres négatifs.

Les nombres réels à virgule flottante sont codés sur 32 bits et permettent d'effectuer des opérations arithmétiques précises. La figure 1-20 donne un exemple de représentation d'un nombre réel à virgule flottante sur 32 bits.

Format binaire à virgule flottante sur 32 bits		
Bit de signe (1 bit)	Exposant (sur 8 bits)	Partie fractionnaire de la mantisse (23 bits)

Figure 1-20: Représentation d'un nombre à virgule flottante sur 32 bits

Pour trouver la représentation en virgule flottante d'un nombre réel, il y a donc trois étapes :

1. Convertir le nombre en binaire,
2. Déterminer la représentation normalisée du nombre,
   (Représentation normalisée = **1, mantisse × 2^{exp}**)
3. Ajouter le décalage à l'exposant trouvé et convertir ce résultat en binaire. La valeur du décalage dépend du nombre **n** de bits utilisés pour stocker l'exposant : décalage **= 2^{n-1} -1**

Ainsi, si l'exposant de la représentation normalisée vaut **exp**, la valeur stockée sera : exp **+ décalage.**

Exemple N° 3 : Conversion du nombre décimal **2,625** sur un format à virgule flottante de 32 bits.

Décimal	signe	Partie entière	Partie décimale	exposant
2,625	0	10	0,625x2 = **1**,25 0,25x2 = **0**,5 0,5x2 = **1** ⇒ **101**	2^0
2,625 =	0	10,	101	2^0
Représentation normalisée	0	1,	0101	2^1
	0	Exp + décalage = 1 + 127 = 128	Partie décimale	
codage	0	10000000	01010000000000000000000	
Représentation en virgule flottante	**0**10000000**01010000000000000000000**			

Exemple N°4 : Conversion du nombre décimal **- 4,75** sur un format à virgule flottante de 32 bits.

Décimal	signe	Partie entière	Partie décimale	exposant
- 4,75	1	100	0,75x2 = **1**,5 0, 5x2 = **1** ⇒ **11**	2^0
- 4,75 =	1	100,	11	2^0
Représentation normalisée	1	1,	0011	2^2
	1	Exp + décalage = 2 + 127 = 129	Partie décimale	
Codage binaire	1	10000001	00110000000000000000000	
Codage binaire C2			11010000000000000000000	
Représentation en virgule flottante	**1**10000001**10100000000000000000000**			

Exemple N°5 : Conversion du nombre décimal **2007** sur un format à virgule flottante de 32 bits.

1- $(2007)_{10}$ -› $(11111010111)_2$
2- $(11111010111)_2 = (1,1111010111)_2 \times 2^{10}$ (**exp = 10**)
3- Décalage = $2^{8-1}-1$ + exp = 127+10 = 137 = 10001001 ⇒
$(2007)_{10}$ =

0	**10001001**	**11110101110000000000000**
Bit de signe	exposant	mantisse

Exemple N°6 : Conversion du nombre décimal $(\mathbf{-0,28125})_{10}$

1- $(-0,28125)_{10}$-› ($-\mathbf{0}x2^{-1}+\mathbf{1}x2^{-2}+\mathbf{0}x2^{-3}+\mathbf{0}x2^{-4}+\mathbf{1}x2^{-5}$) = $(\mathbf{-0,01001})_2$
2- $(\mathbf{-0,01001})_2 = (-1,001)_2 \times 2^{-2}$ (**exp = -2**)
3- Décalage = $2^{8-1}-1$ + exp = 127+-2 = 125 = 01111101 ⇒
$(-0,28125)_{10}$ =

1	**01111101**	**11100000000000000000000**
Bit de signe	exposant	mantisse

Mantisse en complément à 2

Remarque :

Pour trouver la valeur décimale d'un nombre codé en virgule flottante, on effectue les étapes inverses.

Pour traiter les algorithmes de régulation, notamment ceux faisant appel à des calculs récursifs pour lesquels les erreurs d'arrondi se cumulent, il est nécessaire de coder les nombres avec une précision suffisante. On utilisera des nombres entiers en double précision ou des nombres en virgule flottante.

4.5 Utilisation d'un logiciel de programmation

Pour pouvoir utiliser le logiciel :

- ✓ Il faut d'abord commencer par l'installation du logiciel de programmation, pour cela on met le CD ROM dans le lecteur de CD ROM et on suit les démarches habituelles d'installation soit sous Windows ou sous un autre système d'exploitation.
- ✓ Configurer le matériel c'est à dire spécifier l'automate avec lequel on va travailler en donnant sa référence.
- ✓ On choisit le mode de programmation en choisissant l'éditeur approprie : Ladder (à contact) ; (LD) langage structurée (LS), ou Grafcet, (CHART).

- ✓ Le mode en ligne consiste à passer l'application pour que l'automate l'exécute. Pour cela le logiciel offre cette possibilité qui permet de passer du Mode local en mode connecté en appuyant sur l'icône Connecter, puis Transférer.

La méthode suivante permet de programmer, tester, mettre au point et sauvegarder un programme.

- ✓ Étape 1 : **Configuration de l'application**
  Nom de l'application, valeurs des constantes, horodateur, paramètre des compteurs, temporisateurs.

- ✓ Étape 2 : **Saisie des symboles**
  Saisie des noms de Symboles pour chaque repère utilisé dans votre programme automate(contacts, bobines…).

- ✓ Étape 3 : **Saisie du programme**
  Et validation des saisies.

- ✓ Étape 4 : **Sauvegarde régulière du programme**
  En cours de saisie –PC vers disque.

- ✓ Étape 5 : **Transfert du programme dans l'automate**

- ✓ Étape 6 : **Mise en RUN et teste du programme**

- ✓ Étape 7: **Mise au point du programme (**Éditeur de données).

- ✓ Étape 8 : **Sauvegarde du programme**
  Après mise au point – Automate → disque

- ✓ Étape 9 : **création du dossier de l'application**

5. Mise en œuvre des API

La mise en œuvre d'un automate est subordonnée à quelques règles concernant :

- ✓ l'implantation des modules
- ✓ l'alimentation de l'automate
- ✓ l'alimentation des E/S.

5.1 Implantation des modules

L'automate se présente en général sous la forme de bacs simple ou double format dans lesquels viennent s'enficher les modules (alimentation, unité centrale, cartes d'E/S...). Dans certains cas, un détrempage mécanique évite tout risque d'erreur lors de la mise en place ou l'échange.

Adressage des cartes :

Il existe deux possibilités pour l'adressage des cartes d'entrées-sorties. L'adresse d'une entrée ou d'une sortie peut dépendre étroitement de sa position physique dans un rack. Aucune intervention sur la carte n'est alors nécessaire pour affecter cette adresse à la carte, celle-ci étant pré décodée au niveau du fond de panier du rack. Tout déplacement de la carte avec son bornier entraîne une modification des adresses des E/S connectées à ce bornier. Il est alors important que le bornier reste en place après déconnexion.

Inversement, et plus rarement l'affectation de l'adresse peut être faite par intervention (matérielle ou logicielle) sur la carte. Quel que soit sa position, la carte reconnaît toujours son adresse et le connecteur de raccordement des E/S, dans ce cas, doit d'être maintenu solidaire de la carte pour pouvoir se déplacer avec elle.

De façon générale, l'adressage d'une carte E/S se fait suivant :

- ✓ la numérotation du bac
- ✓ l'emplacement géographique de la carte dans le bac.

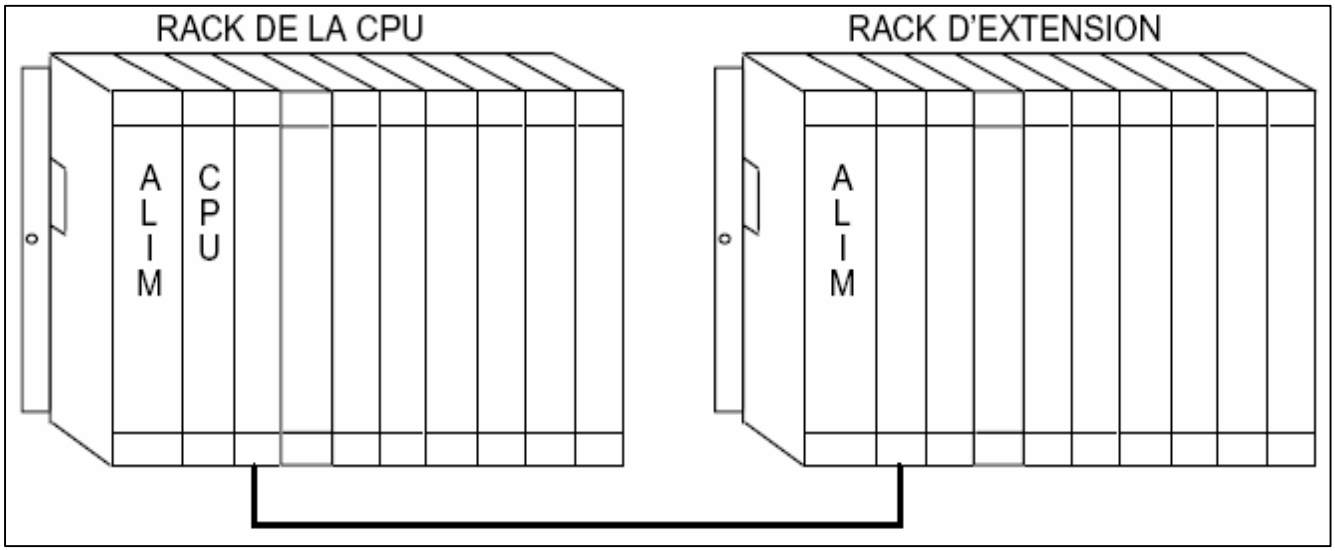


Figure 1-21: Exemple d'implantation de modules dans un API

Exemple d'adressage d'une E/S :

% I ou **% Q** N° du Rack, N° du module, N° de la voie
Suivant les constructeurs et les automates, la numérotation du premier module d'E/S peut débuter en 0 (TSX 47, 67, 87, … Allen Bradley, Siemens, ...), en 1 voir en 2 (Omron).

5.2 Raccordement de l'alimentation automate

Suivant le régime du neutre choisi, plusieurs branchements sont possibles.

a) Neutre à la Terre (Régime TT)

- ✓ le neutre de l'installation est relié à la terre.
- ✓ les masses sont reliées à la terre.

b) Mise au Neutre (Régime TN)

- ✓ le neutre de l'installation est relié à la terre
- ✓ les masses sont reliées au neutre par un conducteur de Protection (PE), ces deux derniers peuvent être distincts ou confondus.

c) Neutre isolé (Régime IT)

- ✓ le neutre n'est pas relié à la terre ou l'est volontairement par l'intermédiaire d'une impédance (1000 W à 2000 W)
- ✓ les masses sont reliées à la terre.

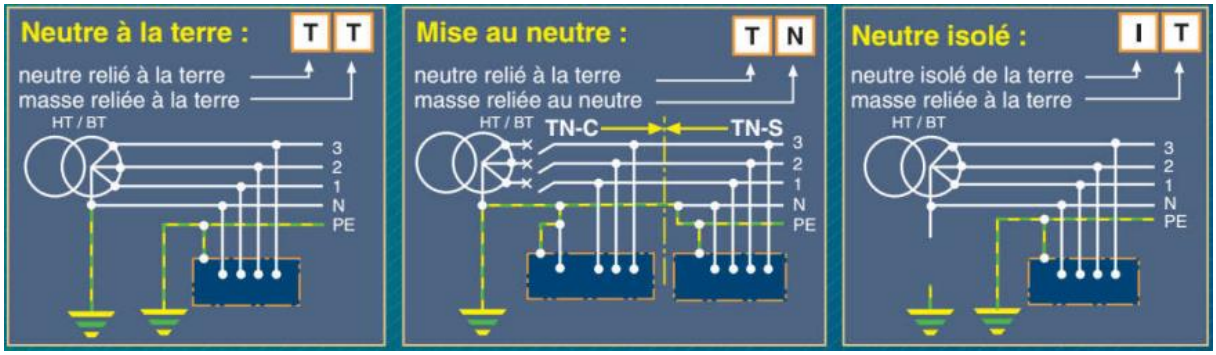


Figure 1-22: Schémas de branchement de l'alimentation d'un API selon le régime de neutre

5.3 Câblage des entrées et sorties

Le câblage des entrées-sorties TOR se fait par torons situés à proximité de la face avant, et le chemin de câble est réalisé par l'intermédiaire de guides fils ou goulottes.

Il est nécessaire de séparer les différents circuits lors des raccordements, à savoir :

- alimentation 220 VCA
- hauts niveaux d'entrées (48 à 220 V)
- hauts niveaux de sorties (48 à 220 V)
- signaux bas niveaux (5 V, +12/-12 V, 4/20 mA)

a) Raccordement des alimentations :

Il est nécessaire de respecter certaines règles, à savoir :

- Le raccordement des masses et de la terre de référence doit se faire en étoile. (dans le cas de perturbations HF un maillage des masses sera préférable).
- Le pôle négatif des alimentations continues est relié à la terre, sauf si elles assurent un isolement galvanique, et ce uniquement dans le cas des E/S logiques.
- Dans le cas d'alimentation par transformateur, il est nécessaire de relier un conducteur du secondaire à la terre, sauf dans le cas d'isolation galvanique. (protection du personnel).

<u>Schéma de principe</u>

La masse de référence (M) est la masse de l'armoire ou bien la masse de l'embase du châssis principal.

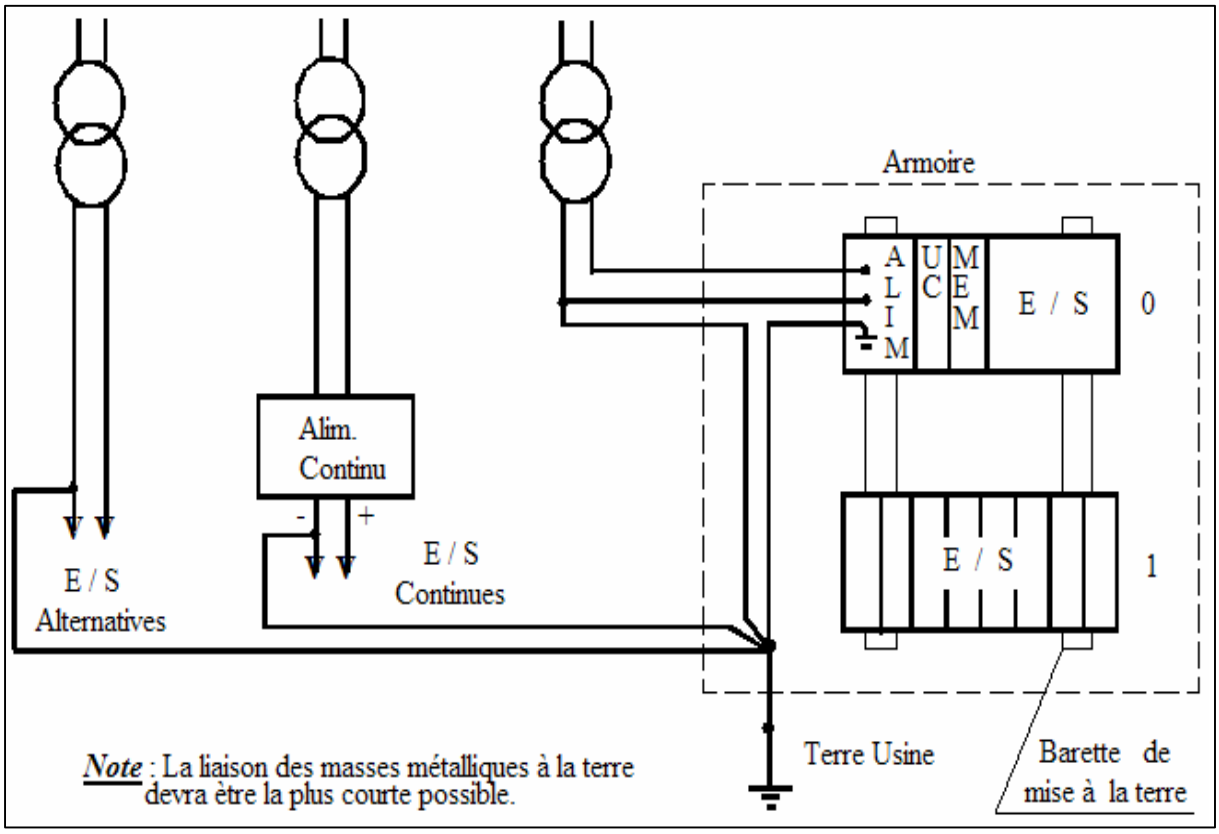


Figure 1-23: Schéma de principe de raccordement des alimentations d'une armoire

Exemple de commande :

Le schéma ci-dessous réalise la mise sous tension en cascade des alimentations suivant l'ordre :

1. Mise sous tension du module automate et des entrées.
2. Mise sous tension des sorties ou des commandes manuelles de sécurité.

Circuit de puissance

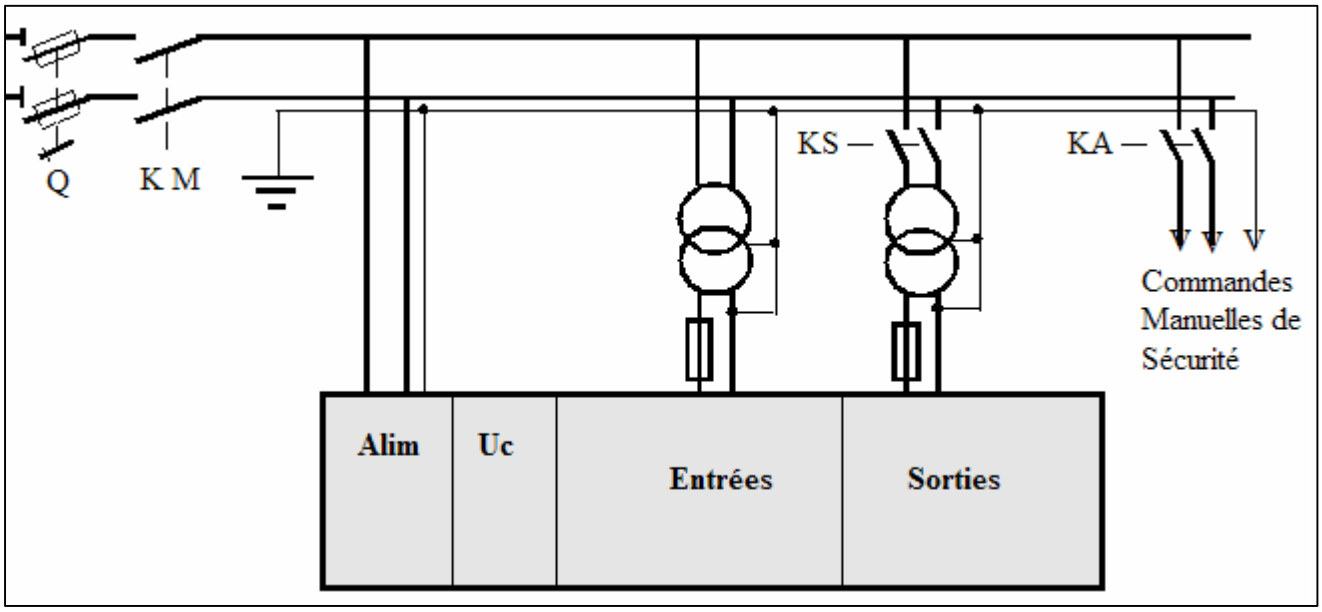


Figure 1-24: Schéma de principe de la mise sous tension du module automate et des entrées

Circuit de commande

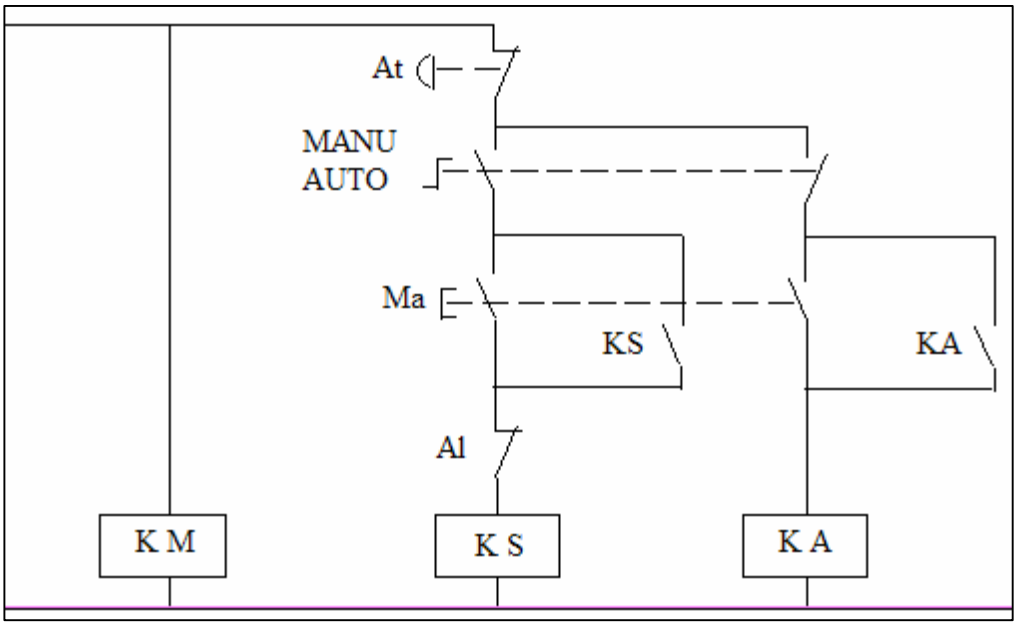
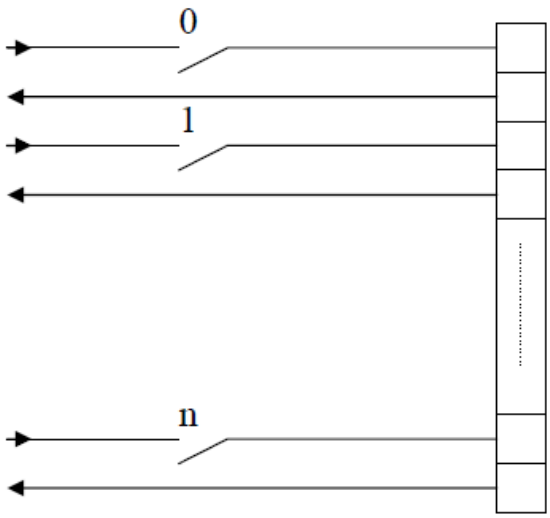
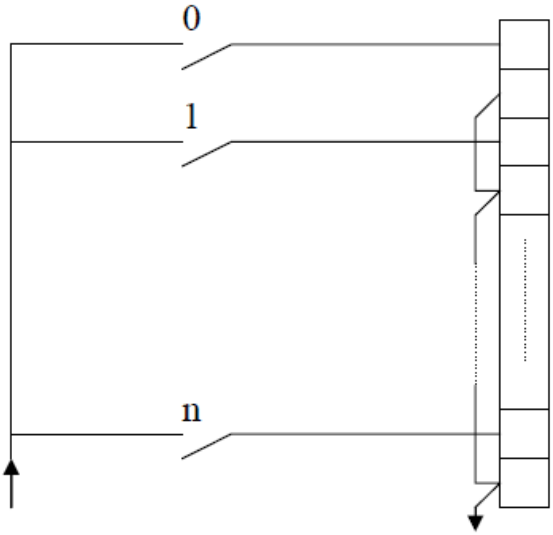


Figure 1-25: Schéma de principe de la mise sous tension des sorties ou des commandes manuelles de sécurité.

a) Raccordement des entrées/sorties :

Une carte comporte 8 ou 16 voies d'entrées ou de sorties ; deux types de raccordement sont possibles :

Voies indépendantes	Voies communes
	
Les voies ne disposent d'aucun point commun entre elles et doivent être alimentées séparément (capteurs particuliers ou alimentés par des sources de tension différentes).	Chaque voie est raccordée au bornier par un seul fil. L'alimentation est commune par groupe de 4, 8, 6 voies. La liaison des communs est réalisée par shunt interne ou ponts entre bornes de l'extérieur.

Entrées avec contrôle de ligne

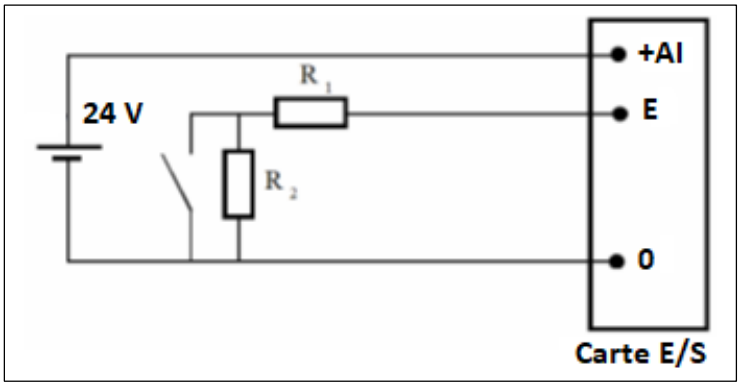
Ces modules d'entrées permettent de recevoir des détecteurs de position 3 fils de type NAMUR. Hormis les états 0 et 1 un contrôle de ligne permet de détecter le court-circuit et la coupure.

ETAT 1 : **I1 = U/R1**

ETAT 0 : **I0 = U/(R1+R2)**

Court - circuit : **I max**

Coupure : **I = 0**



b) Raccordement des entrées/sorties spéciales

E/S logique

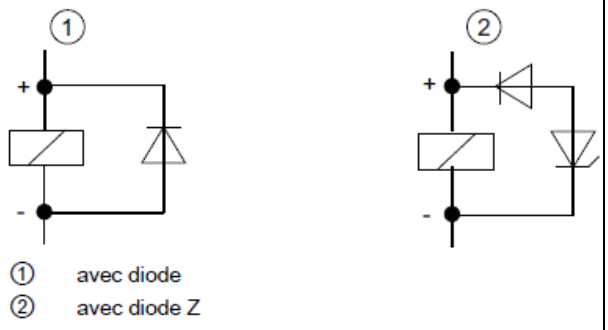
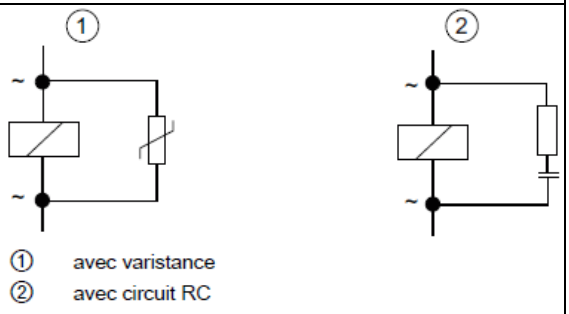
Les entrées rapides, les liaisons utilisées pour la communication (RS232, boucle de courant...) seront réalisées par des câbles torsadés blindés dont le blindage sera relié à la terre des deux côtés.

E/S analogique

Les entrées sorties analogiques (mesures, commande de régulation,...) seront réalisées par des câbles torsadés blindés dont le blindage sera relié à la terre d'un seul côté.

Protection des sorties

En général les voies de sortie tout ou rien sont protégées d'origine contre les surtensions. Dans le cas où un contact est inséré entre la voie de sortie et une charge selfique, une protection externe devra être prévue.

Protection des sorties en courant continu	**Courant alternatif**
	
Les bobines commandées par courant continu sont montées comme sur la figure suivante avec des diodes et diodes Z. Le circuit avec diodes/diodes Z a les propriétés suivantes : ● Les surtensions de coupure peuvent être totalement évitées. La diode Z a la tension de coupure la plus élevée. ● Temporisation de coupure élevée (6 à 9 fois plus élevée que dans circuit de protection). La diode Z coupe plus vite que le circuit de diodes.	Les bobines commandées par courant alternatif sont montées comme sur la figure suivante avec des varistances et des circuits RC. Le circuit avec varistance a les propriétés suivantes : ● L'amplitude de la surtension de coupure est limitée, mais non atténuée. ● La pente de la surtension reste identique. ● La temporisation de coupure est faible. Le circuit avec circuits RC a les propriétés suivantes : ● L'amplitude et la pente de la surtension de coupure diminuent. ● La temporisation de coupure est faible.

5.4 Les étapes à suivre pour raccorder un automate

Pour raccorder un automate, il est recommandé de suivre :

- Les spécifications du fabricant
- La technique de raccordement
- De vérifier si les modules sont dans leurs embases respectives. Vérifier le type, le

numéro du modèle et le diagramme de câblage. Vérifier l'emplacement des embases dans le document pour l'assignation des adresses d'E/S.

- De localiser le paquet de fils correspondant à chaque module et le diriger à travers le conduit à l'emplacement du module.
- Identifier chacun des fils dans le paquet et s'assurer qu'ils correspondent à ce module en particulier.
- En commençant avec le premier module, repérer le fil dans le paquée qui se branche à la borne la plus basse. Au point où le fil arrive à la même hauteur que le point de terminaison, plie le fil à angle droit vers la borne.
- De couper le fil pour qu'il dépasse de 6 mm du côté de la vis de la borne. Dégainer l'isolant du fil à approximativement 9 mm
- Insérer le fil sous la plaque de la borne et serrer la vis.
- Si deux modules ou plus utilisent la même source d'alimentation, on peut utiliser du cavalier «jumpers » pour le câblage de la source d'alimentation d'un module à l'autre.
- Si le câble blindé est utilisé, en brancher seulement un bout à la mise à la terre, préférablement au châssis. Ce branchement évitera toutes boucles possibles de retour de masse. L'autre bout doit être coupé et non branché.
- De répéter la procédure de câblage pour chaque fil du paquet jusqu'à ce que le câblage du module soit complété. Après que tous les fils aient été branchés, tirer doucement sur chacun pour s'assurer d'avoir un bon branchement.
- De répéter la procédure de câblage jusqu'à ce que tous les modules soient terminés.

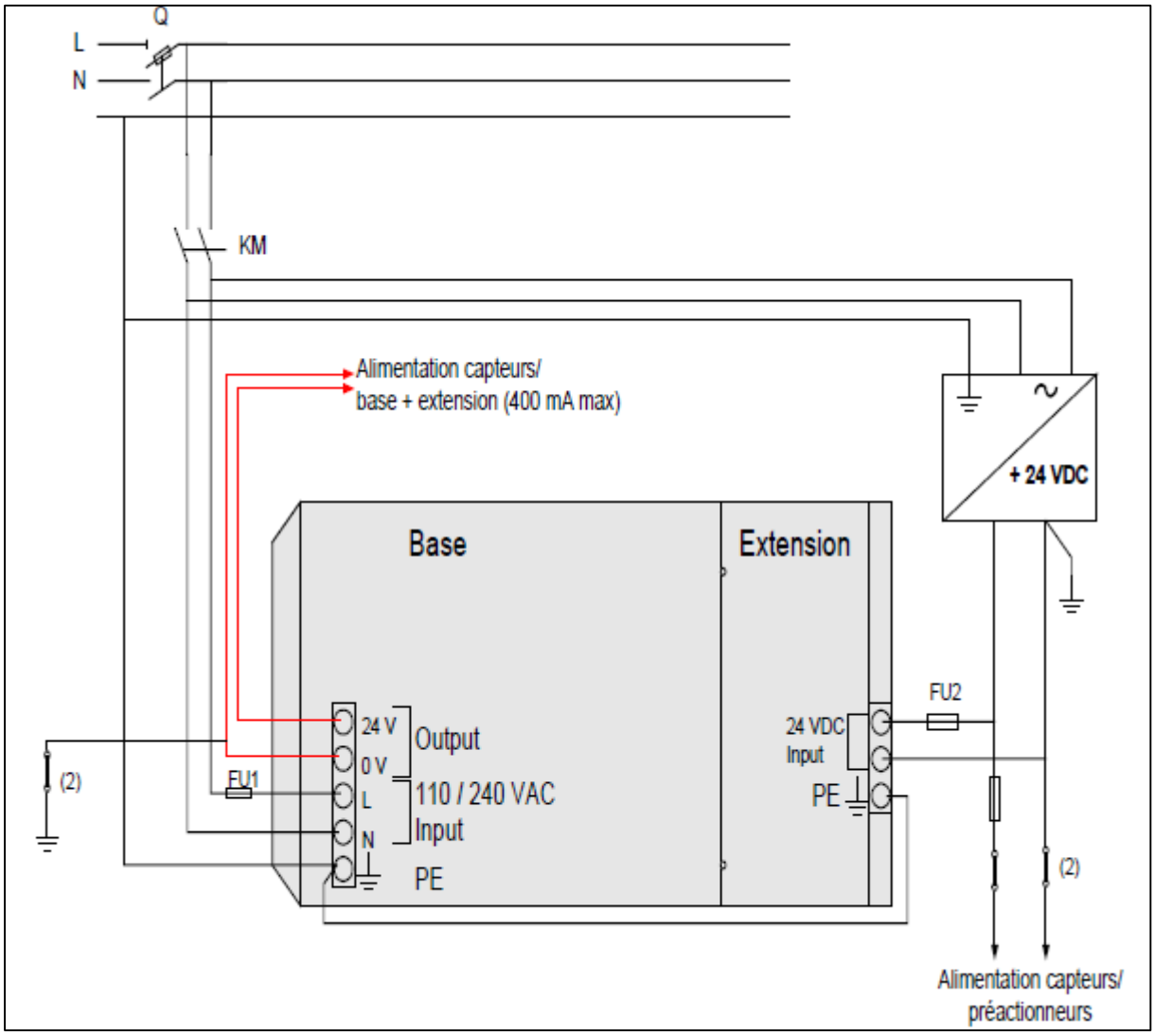


Figure 1-26: Schéma de raccordement d'un automate

5.5 Comment câbler un automate programmable?

En fonction de l'architecture matérielle de l'automate, le câblage peut différer. Les constructeurs d'automates proposent dans leurs catalogues deux catégories d'automates :

- ✓ les **automates compacts** (qui ont une alimentation et des entrées/sorties intégrées)
- ✓ les **automates modulaires** qui nécessitent des modules séparées pour assurer leur alimentation ou pour ajouter des entrées/sorties supplémentaires.

Lors du câblage d'un automate, les entrées/sorties de l'automate doivent toujours être alimentées. On rencontre souvent pour ce qui est des entrées digitales du **24 VDC** (d'autres tensions d'alimentation alternatives ou continues peuvent être utilisées en fonction du modèle de l'automate).

Pour les sorties digitales, on rencontre souvent du **220-240 VAC**, il faut cependant noter que l'on peut aussi avoir une tension continue (DC) comme tension de sortie. Aussi les sorties digitales peuvent être de plusieurs types : **transistors, relais ou triacs**.

Les sorties relais ou contact sec sont les plus flexibles vu qu'elles peuvent actionner à la fois des **actionneurs DC et AC**. Les sorties de type relais sont néanmoins plus lentes (temps de commutation plus longue, environ 10 ms) et sont généralement aussi plus chers. Cependant, les sorties de type relais sont plus puissantes et moins sensibles aux variations et pics de tensions.

Pour les **sorties de type transistor et triac :**

- ✓ Les **sorties transistor** ne peuvent activer que des actionneurs DC. Ils utilisent des transistors NPN ou PNP avec des courants pouvant généralement aller jusqu'à 1 A. Leur temps de réponse est très rapide (inférieur à 1 ms).
- ✓ Les **sorties triac** ne peuvent activer que des sorties AC et sont très adaptés aux actionneurs nécessitant moins de 1 A.

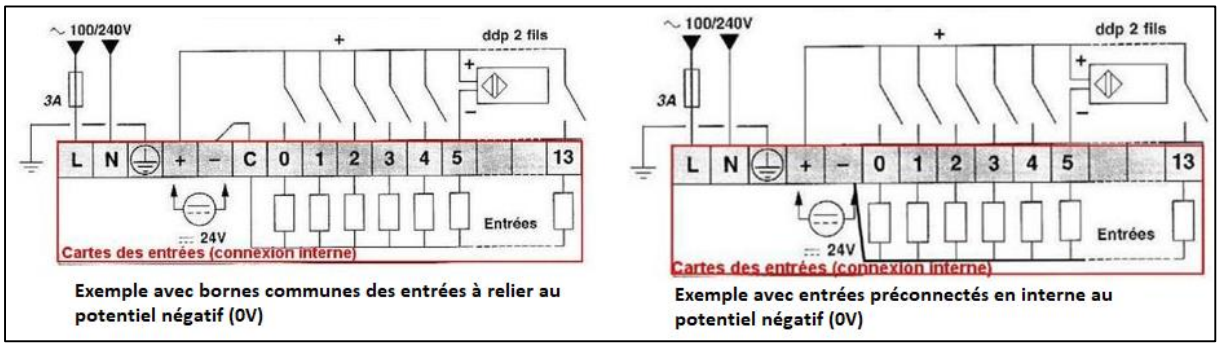
Pour ce qui est des E/S sorties analogiques, on aura souvent des signaux **0-10V, 4-20mA** ou encore **0-20mA**.

Branchement des Entrées TOR

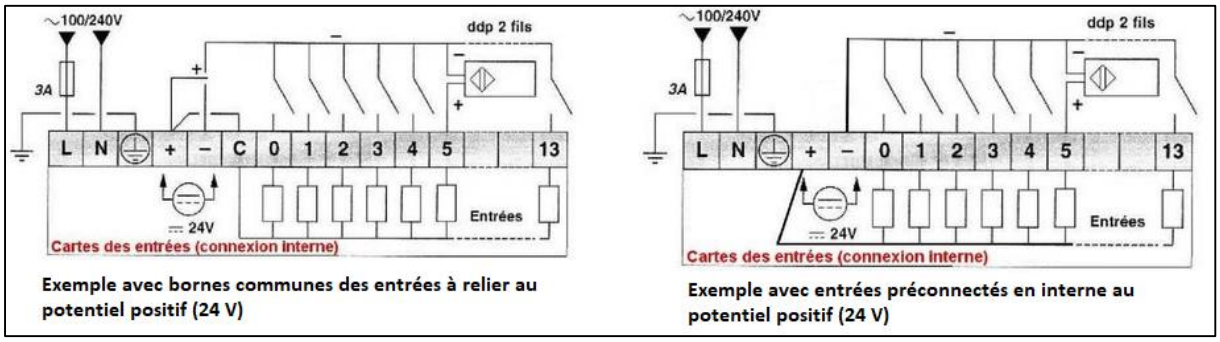
Le principe de raccordement consiste à envoyer un signal électrique vers l'entrée choisie sur l'automate dès que l'information est présente.

L'alimentation électrique peut être fournie par l'automate (en général 24V continu) ou par une source extérieure. Un automate programmable peut être à **logique positive** (Le commun interne des entrées est relié au 0V) ou **logique négative** (le commun interne des entrées est relié au 24V).

Exemple de câblage des entrées TOR en logique positive :



Exemple de câblage des entrées TOR en logique négative :

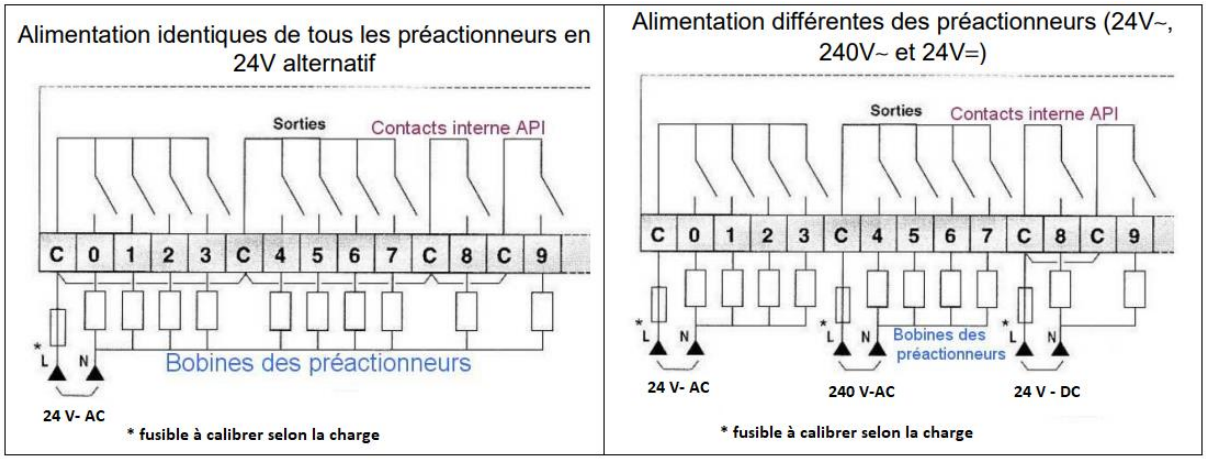


Branchement des sorties

Le principe de raccordement consiste à envoyer un signal électrique vers le préactionneur connecté à la sortie choisie de l'automate dès que l'ordre est émis.

L'alimentation électrique est fournie par une source extérieure à l'automate programmable

Exemple de câblage des sorties TOR:



6 Présentation des structures de contrôle de type API

6.1 Structure de contrôle pour les petites applications industrielles

Pour les **petites applications industrielles simples**, les constructeurs proposent des petits automates (relais intelligent compact).

Constructeur	Modèle	Caractéristiques
Allen Bradley	**MicroLogix/Micro800**	- Module programmable compact avec possibilité d'extension - Programmation par soft: Connected Components Workbench - Langages: Ladder/logigramme/ST

		
Siemens	**LOGO** 	- Module programmable compact avec possibilité d'extension - Programmation par soft: LOGO! Soft Comfort - Langage: Ladder/logigramme
Schneider	**ZELIO LOGIC** 	- Module programmable compact avec possibilité d'extension - Programmation par soft: Zélio Soft - Langages: Ladder/logigramme

6.2 Structure de contrôle pour les applications industrielles d'entrées de gamme

Pour les **petites applications industrielles d'entrées de gamme**, les constructeurs proposent des automates compacts (avec possibilités d'extension). Ils sont en général dédiés à la petite machine.

Constructeur	**Modèle**	**Caractéristiques**

Allen Bradley	**Compact Logix** 	- API compact avec E/S intégrés - Possibilité d'extension de cartes - Communication industrielle: Ethernet IP DeviceNet - Programmation par soft : RSLogix 5000 - Langages: Ladder/logigramme/ST/SFC
Siemens	**S7 1200** 	- API compact avec E/S intégrés - Possibilité d'extension de cartes - Communication industrielle: Ethernet IP-Profinet-Profibus - Programmation par soft : Step7 TIA-PORTAL - Langages: Ladder/logigramme/ST
	MODICOM: M221/M241/M251/M258	- API compact avec E/S intégrés - Possibilité d'extension de cartes

Schneider		- Communication industrielle: Ethernet IP-Modbus-Canopen - Programmation par soft : Somachine - Langages: Ladder/logigramme/ST/SFC

L'architecture de l'API - S71200 est donnée par la figure suivante :

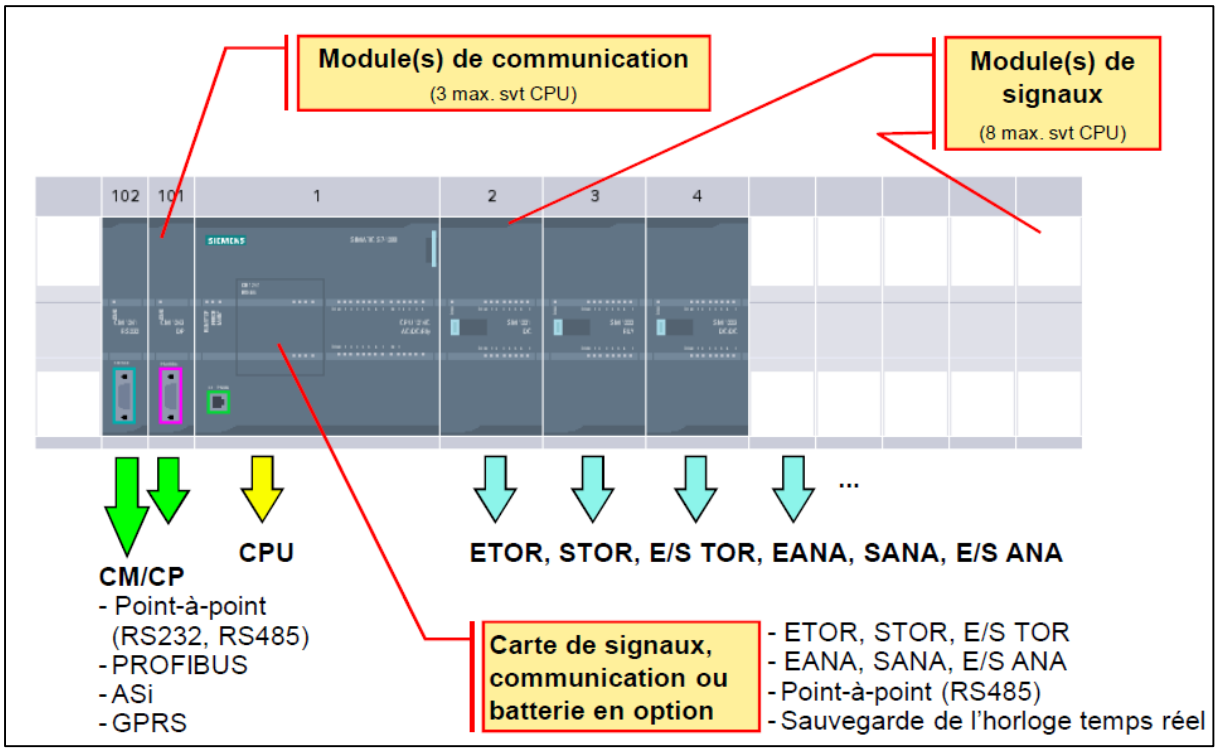


Figure 1.27 : Architecture de l'API S7-1200

6.3 Structure de contrôle pour les applications industrielles de haut de gamme

Pour les **applications industrielles de haut de gamme et les applications process**. les constructeurs proposent des automates compacts (avec possibilités d'extension). Ils sont en général dédiés à la grosse machine ou l'automatisation de process industriels de grande envergure.

Constructeur	Modèle	Caractéristiques
Allen Bradley	**Control Logix 5580** 	- API standard avec modules E/S - Communication industrielle: EthernetIP-ControlNet-DeviceNet Programmation soft: - Studio 5000 Logix Designer - Langages: Ladder/logigramme/ST/SFC
Siemens	**S71500 / S7300 / S7400**  S7-300   S7-1500 S7-400	- S71500/s7300/s7400 - API standard avec modules E/S - Communication industrielle: EthernetIP-Profinet-Profibus - Programmation par soft : Step7 TIA-PORTAL - Langages: Ladder/logigramme/ST/ LIST/SFC
Schneider	**MODICOM:** **M340 /M580/ Prenium / Quantum**  M340  Premium  M580  Quantum	- API compact avec E/S intégrés - Possibilité d'extension de cartes - Communication industrielle: Ethernet IP-Modbus-Canopen - Programmation par soft : - UNITY PRO - Langages: Ladder/logigramme/ST/SFC

L'architecture de l'API - S71500 est donnée par la figure suivante :

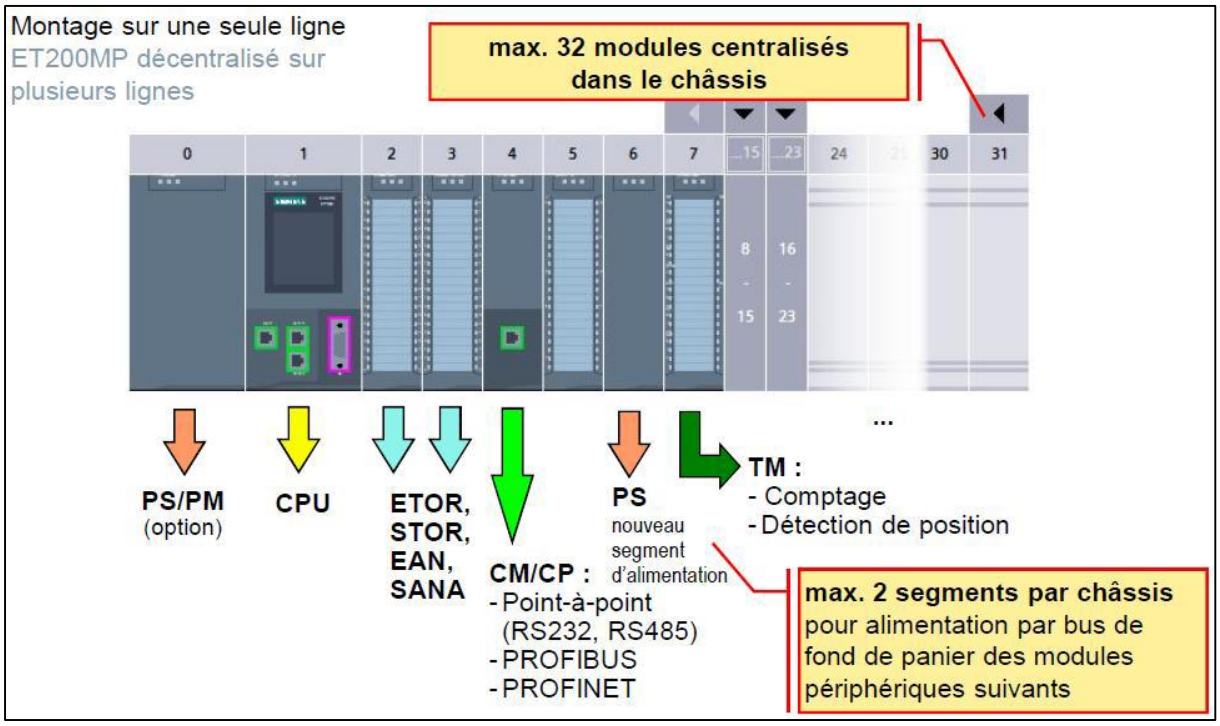
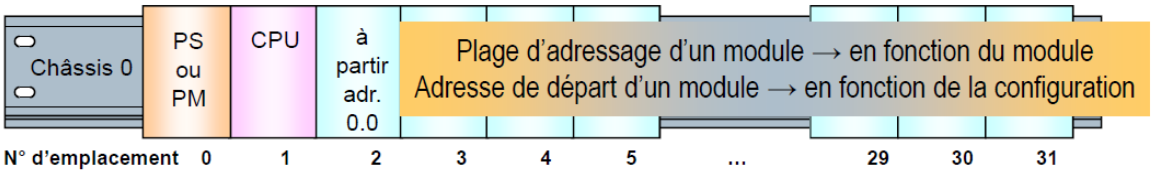


Figure 1.28 : Architecture de l'API S7-1500

Le plan d'adressage des modules



Les adresses par défaut des modules périphériques :

- ✓ Adressage en fonction de l'emplacement
- ✓ Adressage à partir de l'adresse = 0
- ✓ Les adresses sont attribuées en continu dans l'ordre dans lequel les modules périphériques ont été configurés

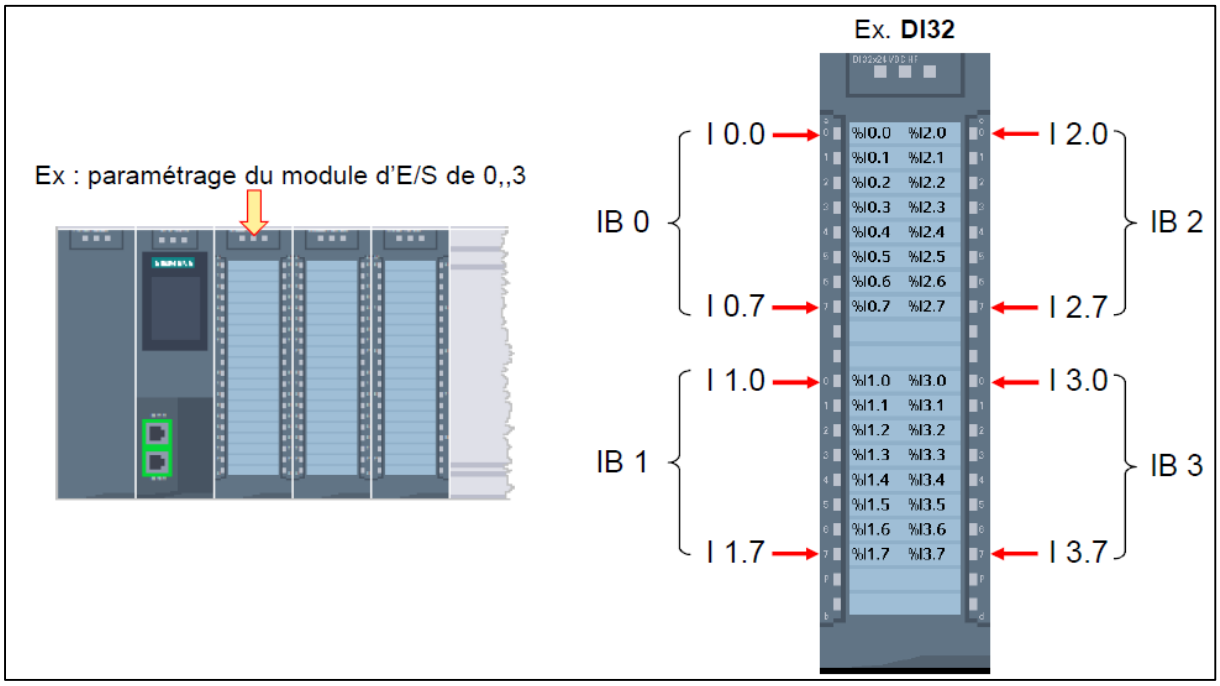


Figure 1.29 : Exemple d'adressage d'un module d'entrée logique DI32 de l'API S7-1500

Les menus de l'afficheur de l'API S7-1500 sont donnés par la figure suivante :

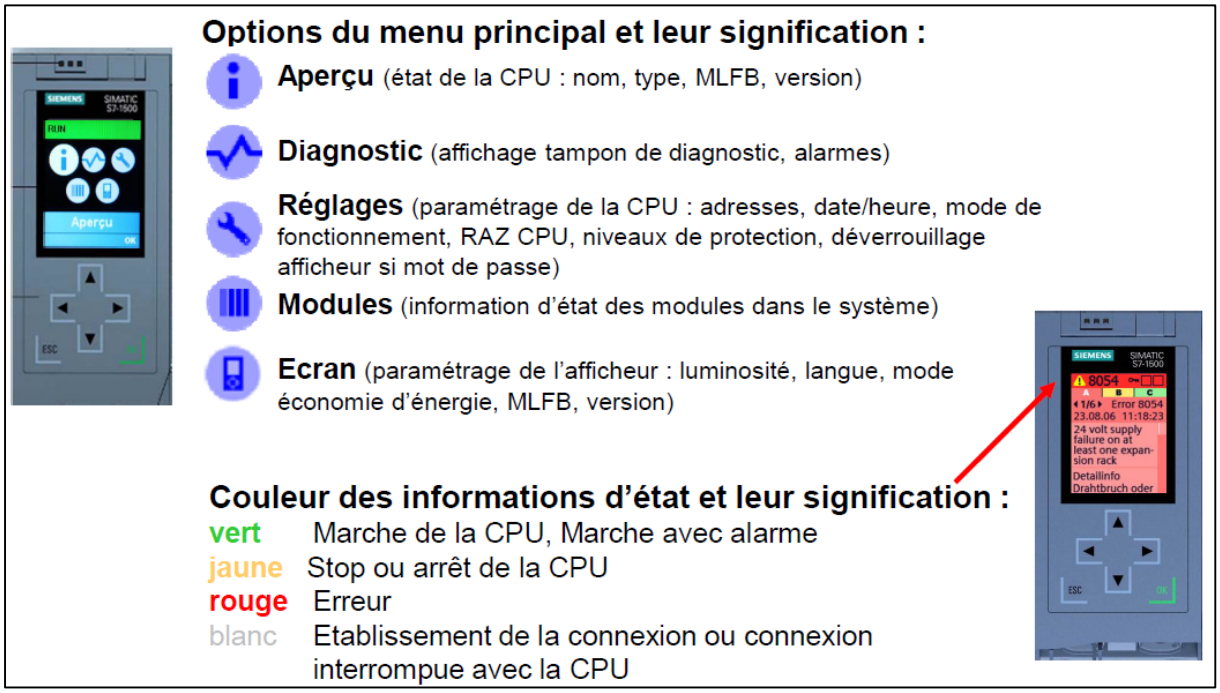


Figure 1.30 : Options du menu principal et leur signification de l'API S7-1500

6.4 Périphérie décentralisée

Pour les applications industrielles de haut de gamme et les applications process, les automaticiens intègrent souvent des entrées/sorties déportées que l'on nomme « **périphérie décentralisée** ». Les échanges avec les CPU se font par réseau industriel.

Constructeur	Modèle	Caractéristiques
Allen Bradley	✓ **CompactBlock LDX I/O** ✓ **Modules POINT I/O** ✓ **Modules d'E/S FLEX 5000** ✓ **Modules d'E/S FLEX 1794** 	- Tout type d'E/S - Différents indices de protection - Différentes interfaces réseaux - E/S de sécurité - Modèles ATEX
Siemens	**En armoire (IP20)** ✓ **ET200SP** ✓ **ET200MP (format S71500)** ✓ **ET200M (format S7300)** ✓ **ET200ISP (ATEX IP30)** **Hors armoire:** ✓ **ET200pro (IP65/67)** ✓ **Etc.**  	- Tout type d'E/S - Différents indices de protection - Différentes interfaces réseaux - E/S de sécurité - Modèles ATEX
	API logic contrôler : ✓ **OTB**	- Tout type d'E/S - Différents indices de protection - Différentes interfaces réseaux

Schneider	✓ **TM5 IP20** ✓ **TM7 IP67** **API M340** ✓ **M580 Quantum:** ✓ **X80** ✓ **Advantys STB IP20** ✓ **Advantys ETB IP67** 	- E/S de sécurité - Modèles ATEX

SIMATIC ET 200SP est un système de périphérie décentralisée modulable permettant de coupler les signaux du processus à un automate de niveau supérieur via un bus de terrain.

Domaine d'utilisation :
Multifonctionnel, le système de périphérie décentralisée SIMATIC ET 200SP convient à l'utilisation dans différents domaines d'application :

- Structure modulable
- Modèles avec CPU
- Modules d'interface PROFINET IO ou PROFIBUS DP
- Applications pour la technique de sécurité
- Large éventail de modules de périphérie (64 modules max) (31 départs-moteurs)
- Indice de protection IP 20
- Montage encastré dans armoire électrique (rail DIN)

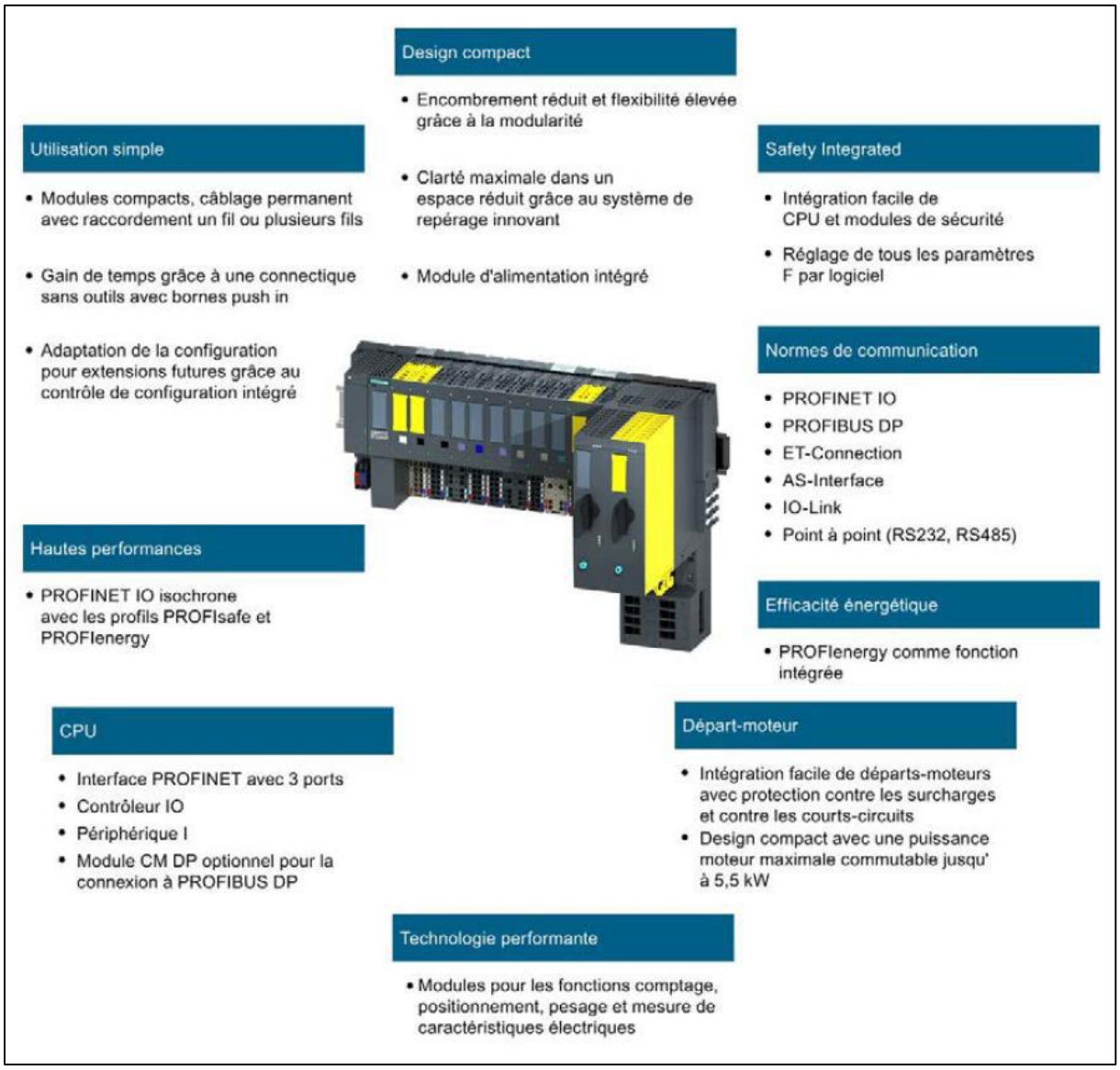


Figure 1.31 : Caractéristiques techniques du module déporté ET 200SP

7 Présentation des modules d'entrée/ sorties logiques

Dans cette partie, nous allons présenter les différents modules d'entrées /sorties proposées par les constructeurs. Nous illustrerons cette partie avec une architecture SIEMENS s71500.

7.1 Cartes d'entrées TOR

Les cartes d'entrées TOR sont souvent alimentés en 24V continu.
Deux câblages possibles: Logique positive ou logique négative
Les entrées sont généralement isolées électriquement de l'unité centrale et peu sensibles aux perturbations (filtrage –temps de retard)

Lorsqu'une entrée TOR est active (état d'un capteur, état d'un BP, etc.) une LED s'allume sur la CPU.
Le schéma de câblage du Module d'entrées TOR à transistors **Module 32DI (6ES7 521 1BL00**

0AB0) est donné par la figure suivante :

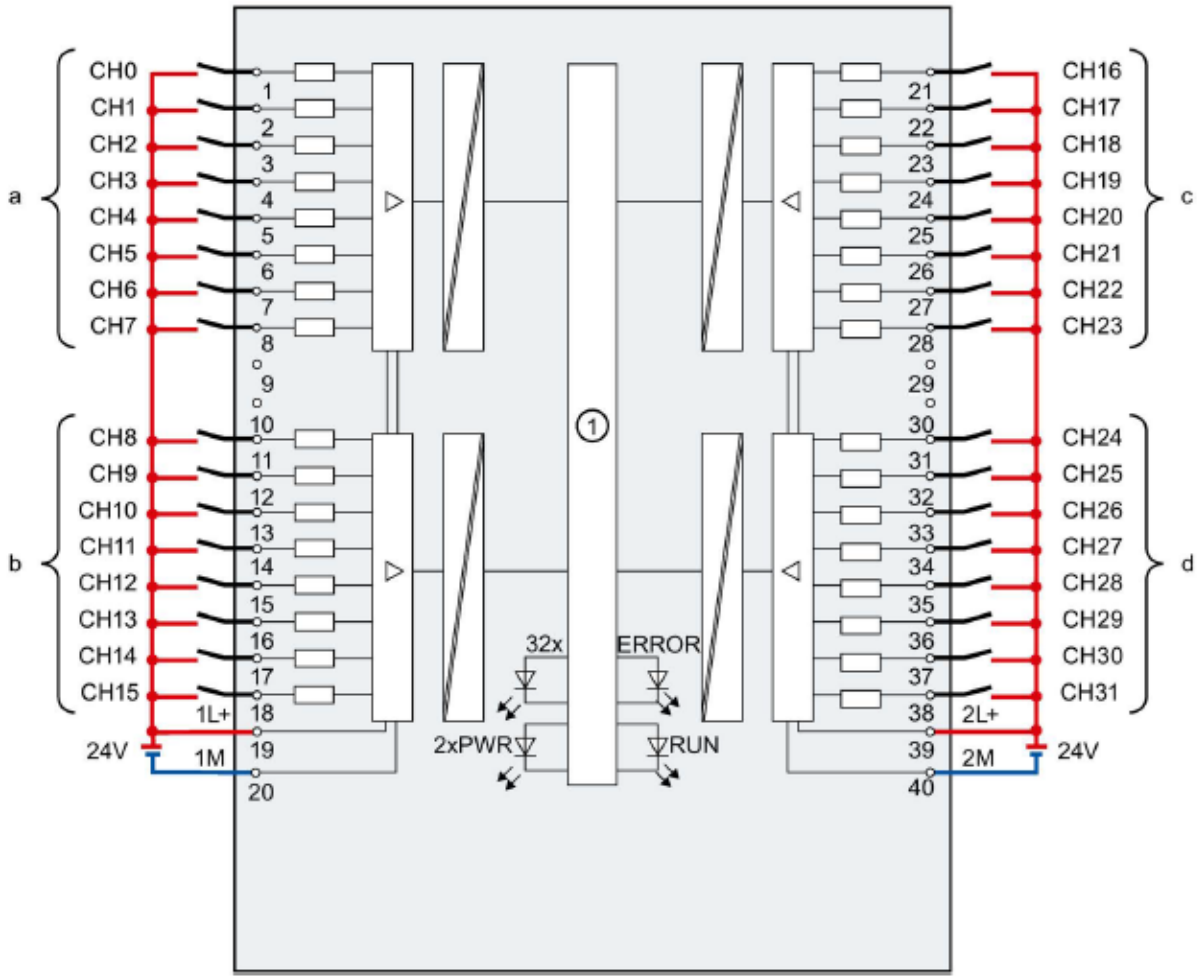


Figure 1.32 : Schéma de câblage d'un module d'entrée logique - Module 32DI (6ES7 521 1BL00 0AB0)

7.2 Cartes de sorties TOR

Les cartes de sorties tout ou rien permettent de raccorder à l'automate, les différents pré-actionneurs tels que :

- Distributeurs, vannes, contacteurs, voyants, etc.
- Les tensions de sorties usuelles sont de 5 volts en continu ou de 24, 48, 110, 220 volts en continu ou en alternatif.
- Les courants vont de quelques milliampères à quelques ampères.
- Ces cartes possèdent le plus souvent des transistors ou des relais et parfois des triacs.
- L'état de chaque sortie est visualisé par une diode électroluminescente, lorsque la sortie est active dans le programme.

Ce sont les caractéristiques de pré-actionneurs qui nous permettront de choisir entre un module de sorties à transistors ou un module de sorties à relais.

- Exemple de carte à transistor: 24V/0,5A par voie
- Exemple de carte à relais : contact sec tenues électriques: 230V/5A

Ainsi pour l'électrovanne, on choisira une carte à relais et pour le contacteur les deux modèles de cartes peuvent convenir. (Une carte à relais coûte plus chère qu'une carte à transistors).

Le schéma de câblage du Module de sorties TOR à transistors Modèle à 32 sorties (**Module 32DQ -6ES7 522 1BL10 -0AA0**) est donné par la figure suivante :

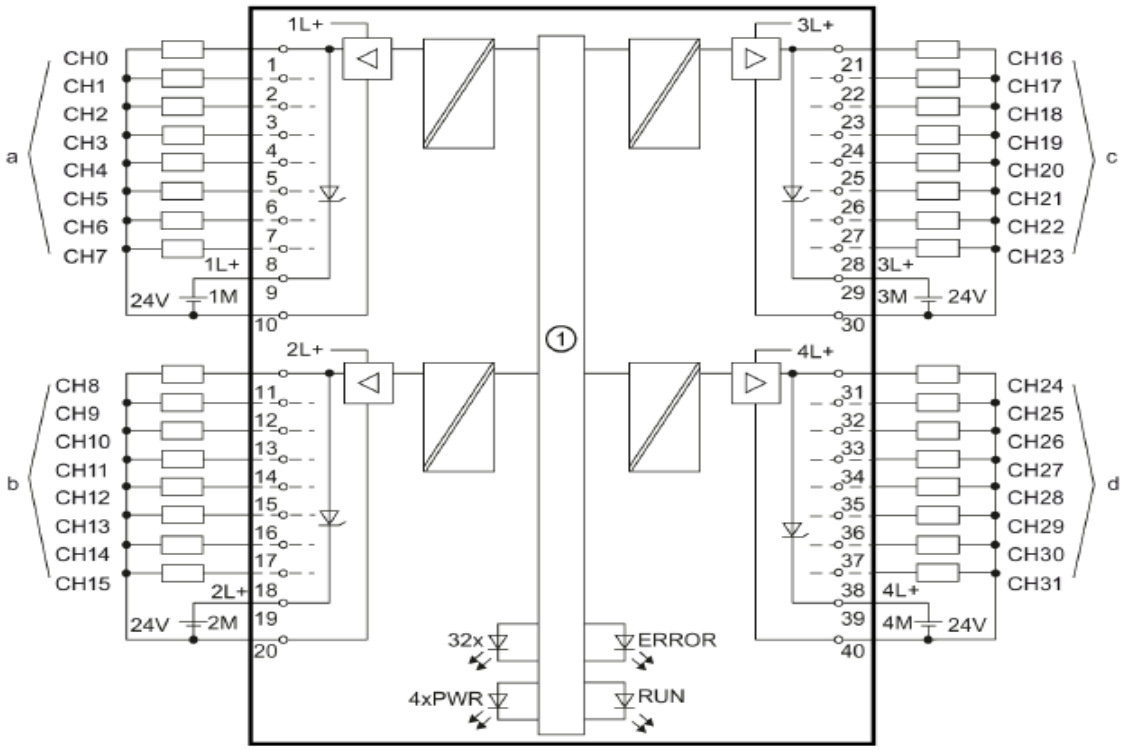


Figure 1.33 : Schéma de câblage d'un module de sortie logique à transistors - Module 32DQ (6ES7 522 1BL10 -0AA0)

Le schéma de câblage du Module de sorties TOR à relais Modèle à 8 sorties (**Module 8DQ à relais (6ES7 522 5HF00 0AB0)** est donné par la figure suivante :

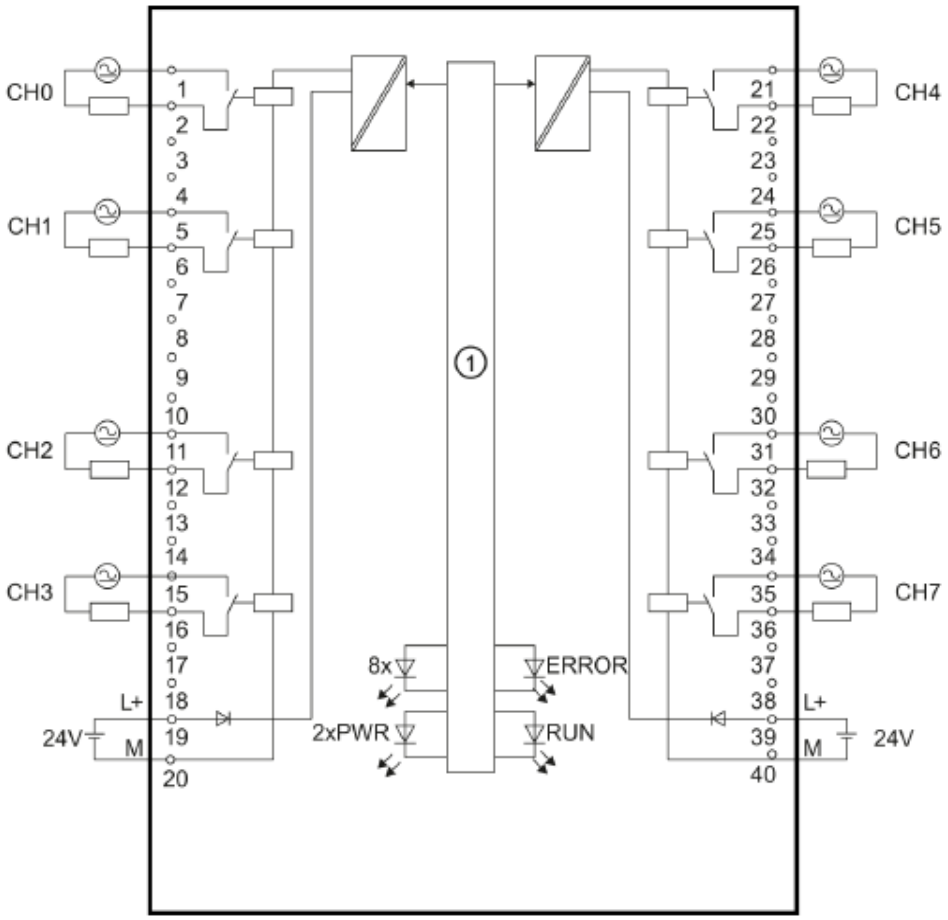


Figure 1.34 : Schéma de câblage d'un module de sortie logique à relais – (Module 8DQ à relais (6ES7 522 5HF00 0AB0)

8 Présentation des modules d'entrée/ sorties analogiques

8.1 Introduction

Les automates ne peuvent traiter des valeurs analogiques que sous forme de configurations binaires. Des capteurs de mesure raccordables au module analogique acquièrent des grandeurs physiques, par ex. pression ou température.

Cette valeur analogique est mesurée sous forme de courant, tension ou résistance par le module d'entrées analogiques.

Afin que la CPU puisse traiter la valeur de courant ou de tension acquise, un convertisseur analogique-numérique (CAN) intégré au module d'entrées analogiques la convertit en un nombre entier de 16 bits.

Un CNA (conversion numérique analogique) permet la conversion d'un signal numérique en signal analogique (Sortie ANA sur API).

Leur résolution dépend du nombre de bits. Pour un CAN ou CNA à N bits, le nombre d'états possibles en sortie est 2^N, ce qui permet d'exprimer des signaux numériques de 0 à 2^N-1 en code binaire naturel.

Le format de variable pour une grandeur analogique reste le nombre entier sur un format de 16 bits (IW…pour les entrées et QW pour les sorties), une valeur signée correspond donc à : 2^{16}-1 = 65535 soit une plage signée entre -32768 à 32767.

Le schéma de principe de traitement des entrées /sorties analogiques par API est donné par la figure suivante :

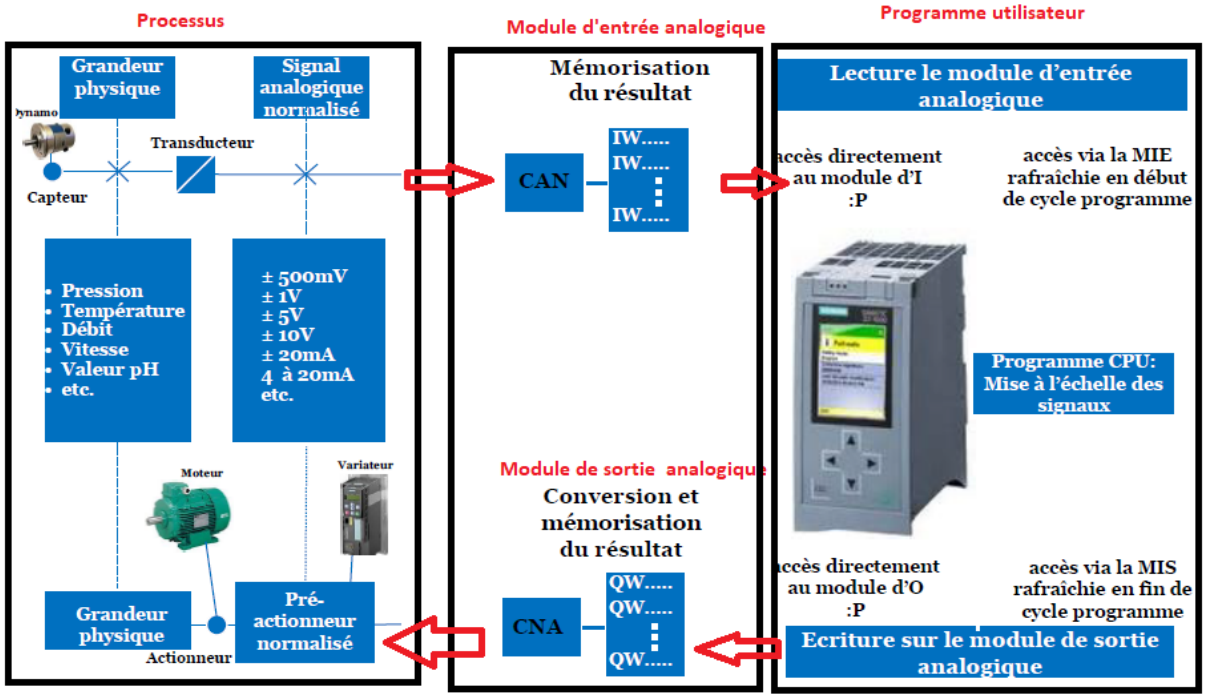


Figure 1.35 : Schéma de principe de traitement des entrées/sorties analogiques

8.2 Cartes d'entrées analogiques

On distingue principalement 3 types de signaux d'entrée analogiques : **les signaux type tension, courant et résistance**.

- Les signaux de type tension

Les signaux de type tension sont générés par certains types de capteurs analogiques. Les plages d'opération les plus courantes sont le 0-5 VDC et le 0-10 VDC

- Les signaux de type courant

Le signal 4-20 mA est devenu le signal de courant standard dans l'industrie. Dès fois on peut aussi rencontrer du 0-20 mA.

- Les signaux de type résistance

Les signaux de type résistance sont dans la plupart du temps utilisés avec les capteurs de température comme les thermocouples.

Pour une carte d'entrée analogique, le type de mesure est paramétrable pour chaque voie avec la suite logicielle, associé bien entendu au câblage adéquat.

La figure suivante, montre le schéma de câblage d'un module analogique à 8 entrées (Module 8AI (6ES7 531 7PF00 0AB0)

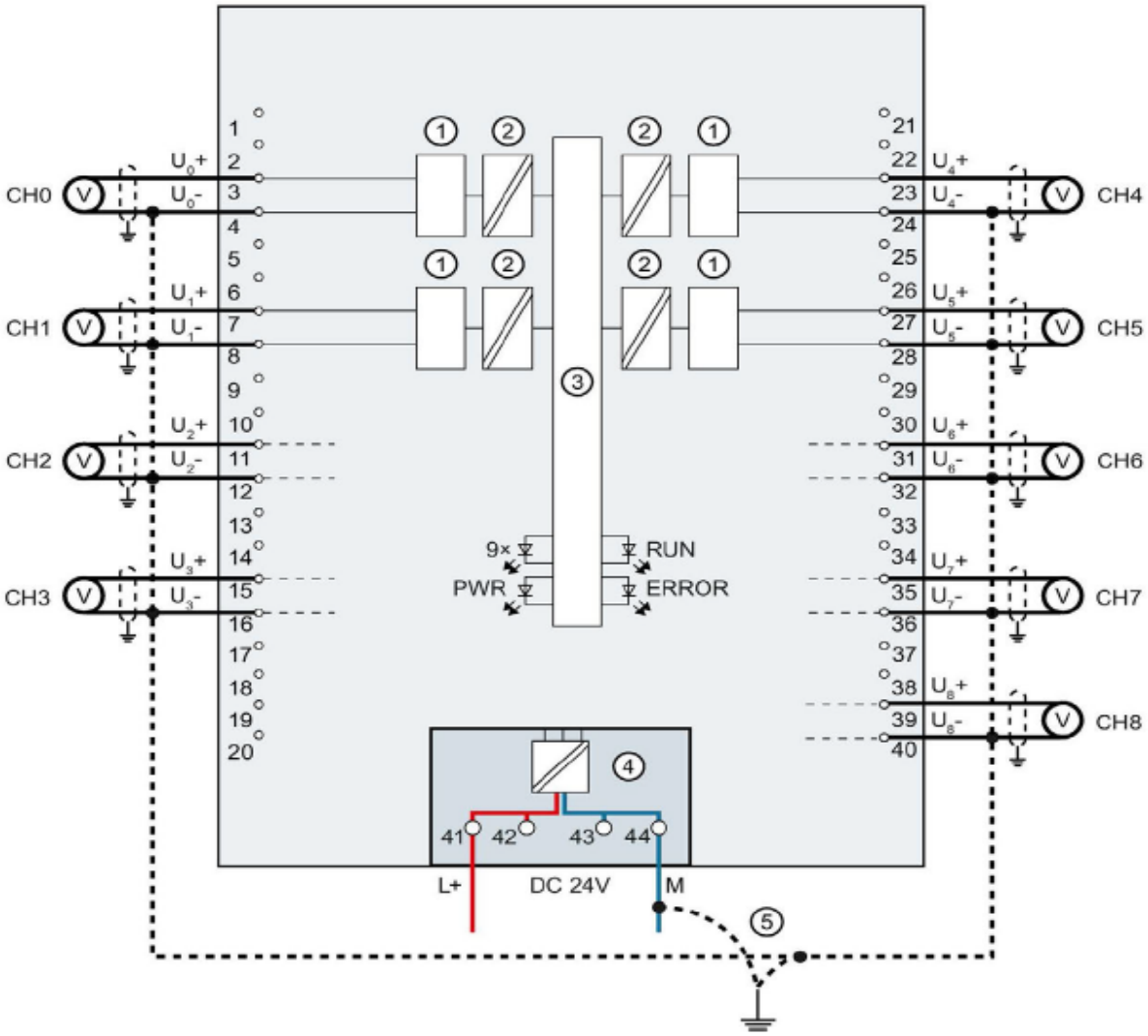


Figure 1.36 : Schéma de câblage d'un module d'entrée analogique - Module 8AI (6ES7 531 7PF00 0AB0)

Ce module possède les caractéristiques techniques suivantes :

- ✓ 9 entrées analogiques avec séparation galvanique
- ✓ Types de mesure paramétrables pour chaque voie :
  - Tension
  - Résistance
  - Thermomètre à résistance (RTD)
  - Thermocouple (TC)

8.3 Cartes de sorties analogiques

A la différence des entrées analogiques, les sorties analogiques sont des signaux variables générés par l'automate (module de sortie analogique) et qui agissent sur des pré-actionneurs analogiques. Par exemple une sortie analogique peut permettre de contrôler un variateur de vitesse ou encore une électrovanne proportionnelle.

Comme pour les entrées analogiques, les types de signaux les plus fréquents sont les types de signaux de type tension, courant et résistance qui sont paramétrable pour chaque voie avec la suite logicielle, associé bien entendu au câblage adéquat.

La figure suivante, montre le schéma de câblage d'un module analogique à 4 sorties (Module 4AQ (6ES7 532 5ND00 0AB0)

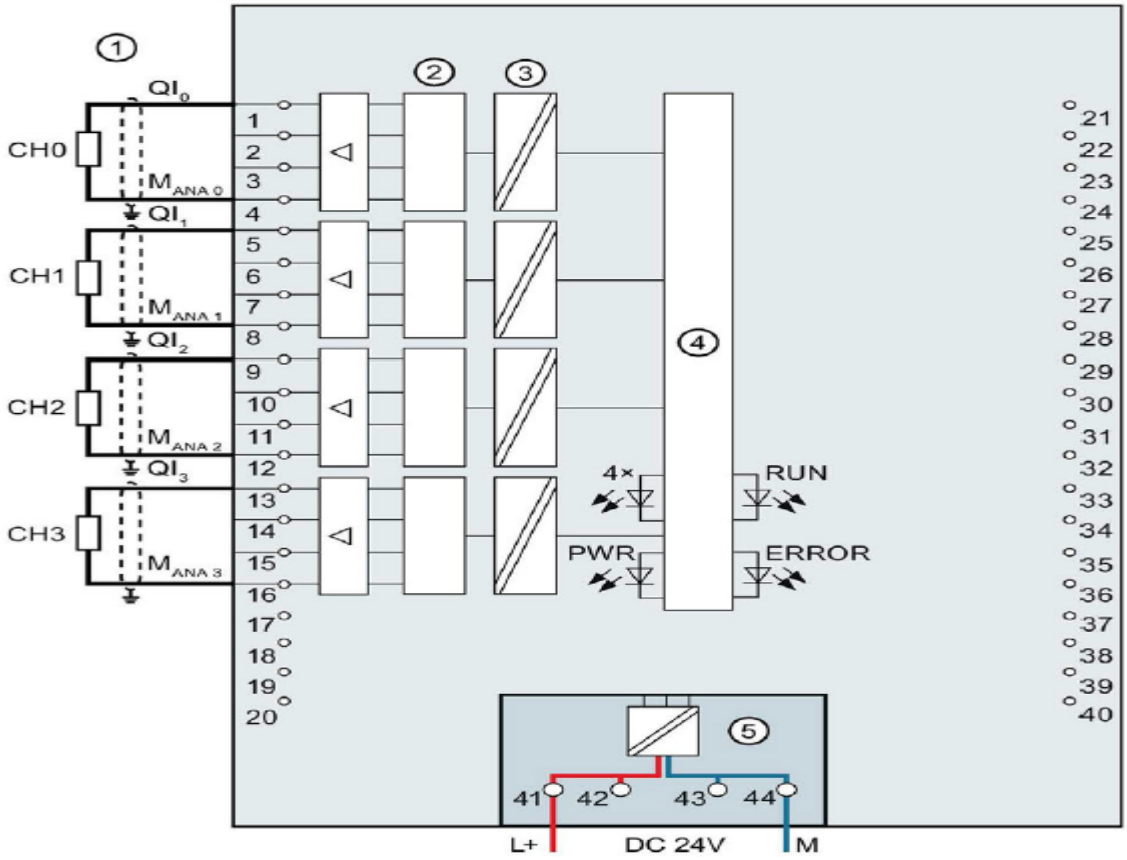


Figure 1.37 : Schéma de câblage d'un module de sortie analogique - Module 4AQ (6ES7 532 5ND00 0AB0)

Ce module possède les caractéristiques techniques suivantes :

- ✓ sorties analogiques avec séparation galvanique
  - Sortie de tension sélectionnable voie par voie
  - Sortie de courant sélectionnable voie par voie

8.4 Cartes spécifiques

Dans l'automatisme industriel, il existe un certains nombres de capteurs et de pré-actionneurs spécifiques à certaines applications.

Parmi ces applications, on retrouve souvent comme:

- ✓ La mesure de température (exemples: PT100, PT1000, Thermocouple, etc.)
- ✓ Le comptage rapide (exemples : codeur incrémental, capteur inductif, etc.)
- ✓ La mesure de positions (exemples : capteur ohmique à câble, codeur absolu, etc.
- ✓ La commande d'axe (exemple: commande PTO pour asservissement en position, etc.)
- ✓ Les systèmes de pesage (exemple: bascule de pesage avec jauges de contrainte, etc.)
- ✓ Les systèmes sans contact (exemples: RFID, système d'identification optique, etc.)
- ✓ Etc…

a) Module d'entrées analogiques pour la mesure de température

Ce module d'entrées analogiques est polyvalent, (entrée ANA en tension), il permet également de mesurer des températures via des capteurs de type PT100, PT1000 (technologies 2 fils, 3 fils et 4 fils) et également des thermocouples.

La configuration s'effectue avec l'outil logiciel.
La figure suivante, montre le schéma de câblage du module 8AI (6ES7 531 7PF00 0AB0).

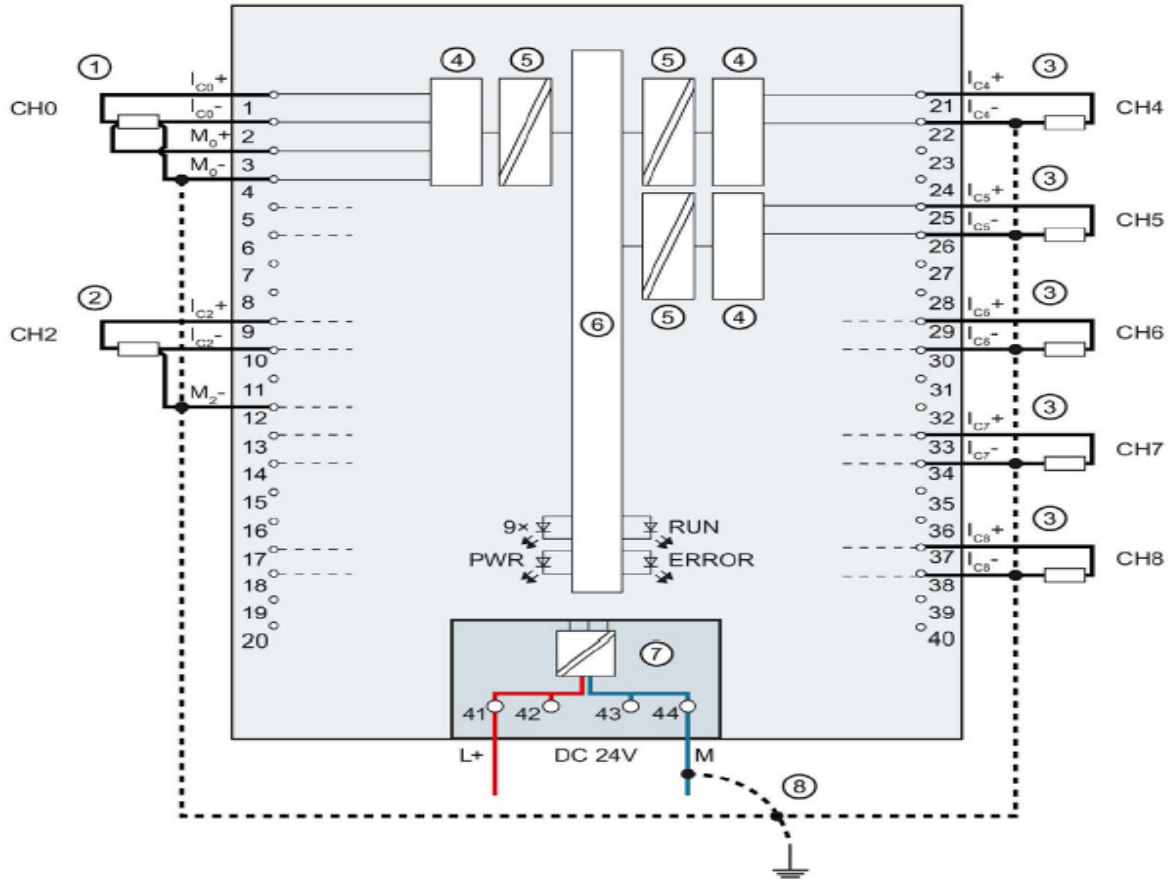


Figure 1.38 : Schéma de câblage d'un module 8AI (6ES7 531 7PF00 0AB0).

b) Module de comptage rapide

Le comptage consiste à acquérir et totaliser des événements. Les compteurs du module technologique acquièrent des signaux de capteur et des impulsions et les évaluent de manière appropriée. **(Regarder la fréquence max des impulsions en fonction de la carte)**

Le sens de comptage peut être prédéfini par des signaux de capteur ou d'impulsion appropriés.
Les entrées TOR permettent de commander les processus de comptage. Vous pouvez commuter les sorties TOR exactement aux valeurs de comptage définies indépendamment du programme utilisateur.

La figure suivante, montre le schéma de câblage du module TM Count 2x24V (6ES7 550 1AA00 0AB0).

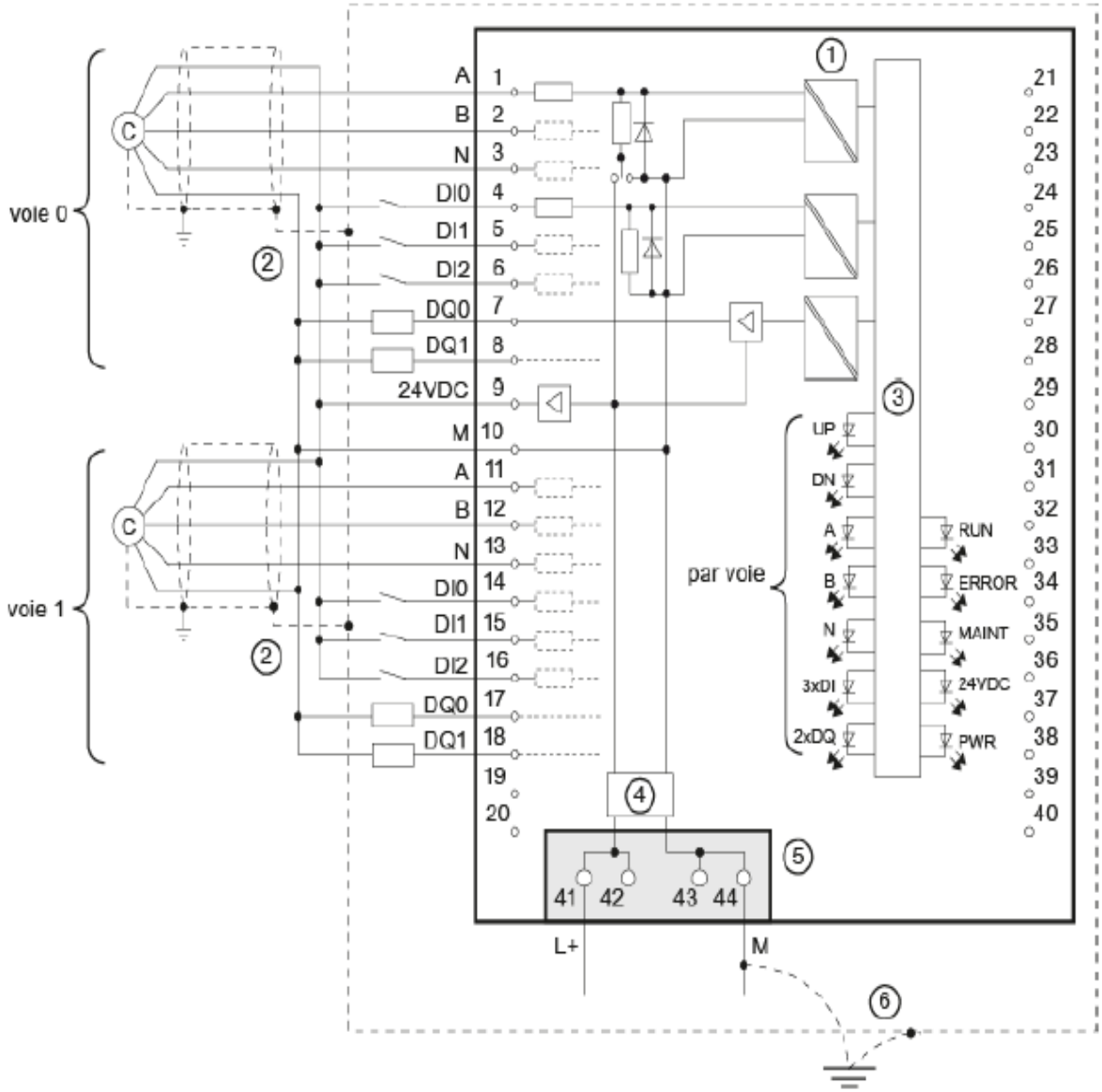


Figure 1.39 : Schéma de câblage du module TM Count 2x24V (6ES7 550 1AA00 0AB0).

c) Module de commande pour entrainements PTO pour moteur pas à pas

Pulse Train Output est une interface simple et universelle entre une commande et un entraînement.
PTO est prise en charge par de nombreux entraînements pas à pas et servomoteurs et qui est également désignée comme une interface impulsions-direction.

Pulse-Train-Output (PTO) est une technique de modulation dédiée à l'asservissement de position d'axes ou en contrôle de vitesse. La fréquence est réglable et le rapport cyclique fixe à 50%.

La figure suivante, montre le schéma de câblage du module TM PTO 4 (6ES7 553 1AA00 0AB0).

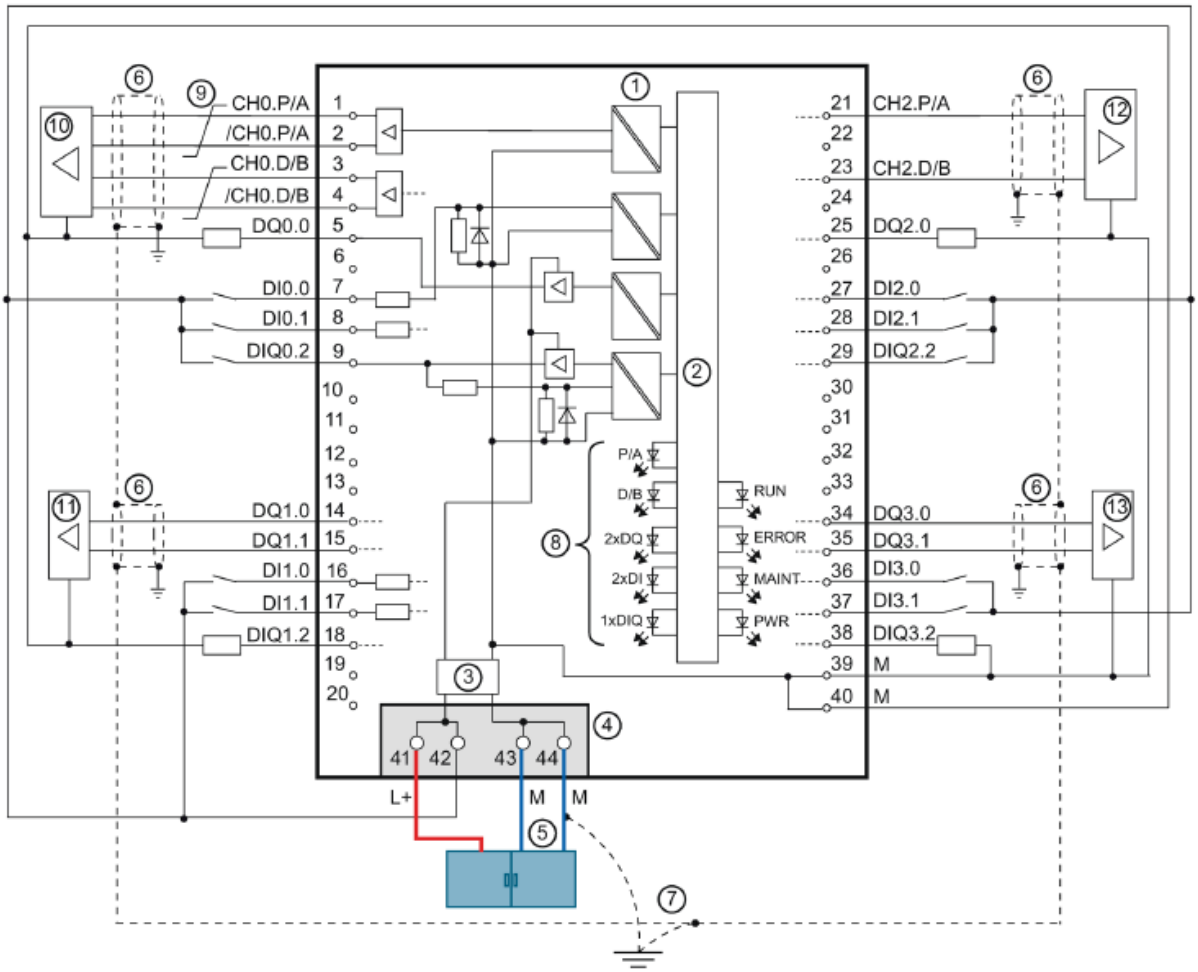


Figure 1.40 : Schéma de câblage du module TM PTO 4 (6ES7 553 1AA00 0AB0).

d) Module de pesage

La tâche primaire de l'électronique de pesage consiste aux mesures et enregistrements de valeurs de poids avec grande précision.

Grâce à l'intégration dans les solutions d'automatisme, il est possible de traiter le poids directement dans l'API.

La figure suivante, montre le schéma de câblage du module TM SIWAREX WP521 ST (7MH4 980 1AA01). Le raccordement de pesons à jauge extensiométrique en montage 4 ou 6 fils (par voie)

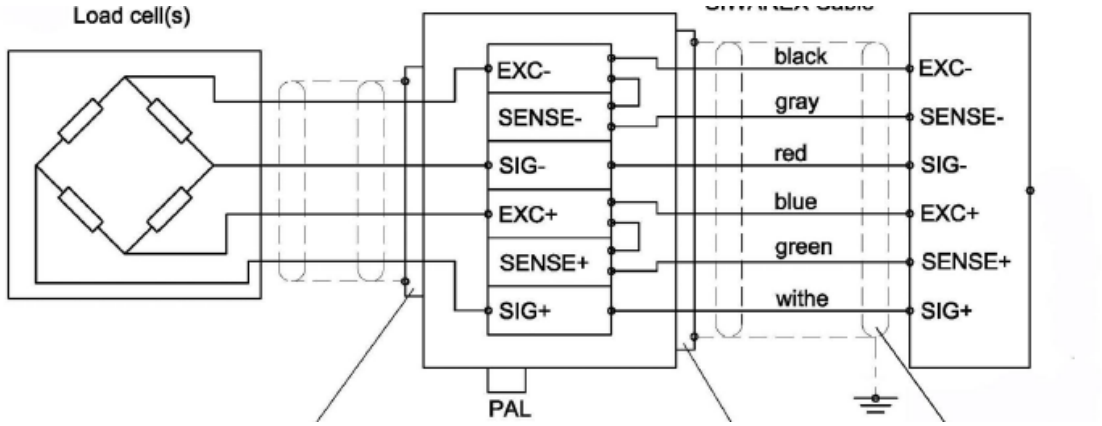


Figure 1.41 : Schéma de câblage du module TM SIWAREX WP521 ST (7MH4 980 1AA01).

Chapitre II

OUTILS DE DESCRIPTION FONCTIONNELLE DES SYSTEMES AUTOMATISES

1. Introduction

Pour un procédé donné, on doit tenir compte :

- Des contraintes industrielles (rentabilité, sécurité),
- De la nature et des types de capteurs donc des signaux délivrés par les capteurs,
- De la nature des actionneurs donc des signaux de commande acceptés par les actionneurs.

L'établissement du cahier de charges est la partie la plus importante surtout pour la disposition des capteurs. Elle détermine aussi :

- Le choix des interfaces d'E/S,
- Le nombre et le type d'E/S,
- L'utilité ou pas d'un bus de terrain.

L'établissement des séquences de l'automatisme est ce que l'on appelle le **GRAFCET de fonctionnement normal** ou aussi GRAFCET de niveau 1. C'est le diagramme des actions à réaliser en fonction des informations reçues.

L'analyse opérationnelle et technologique prend en compte les contraintes imposées par les capteurs et les actionneurs, ainsi que les modes de fonctionnement de l'automatisme (marche, arrêt, manuel, automatique, sécurité). C'est ce qu'on appelle le **GRAFCET de conduite** ou encore GRAFCET niveau 2 qui représente l'automatisme complet à réaliser. A ce stade, en fonction des problèmes rencontrés, on peut être obligé de revenir au cahier des charges et/ou au GRAFCET de niveau 1.

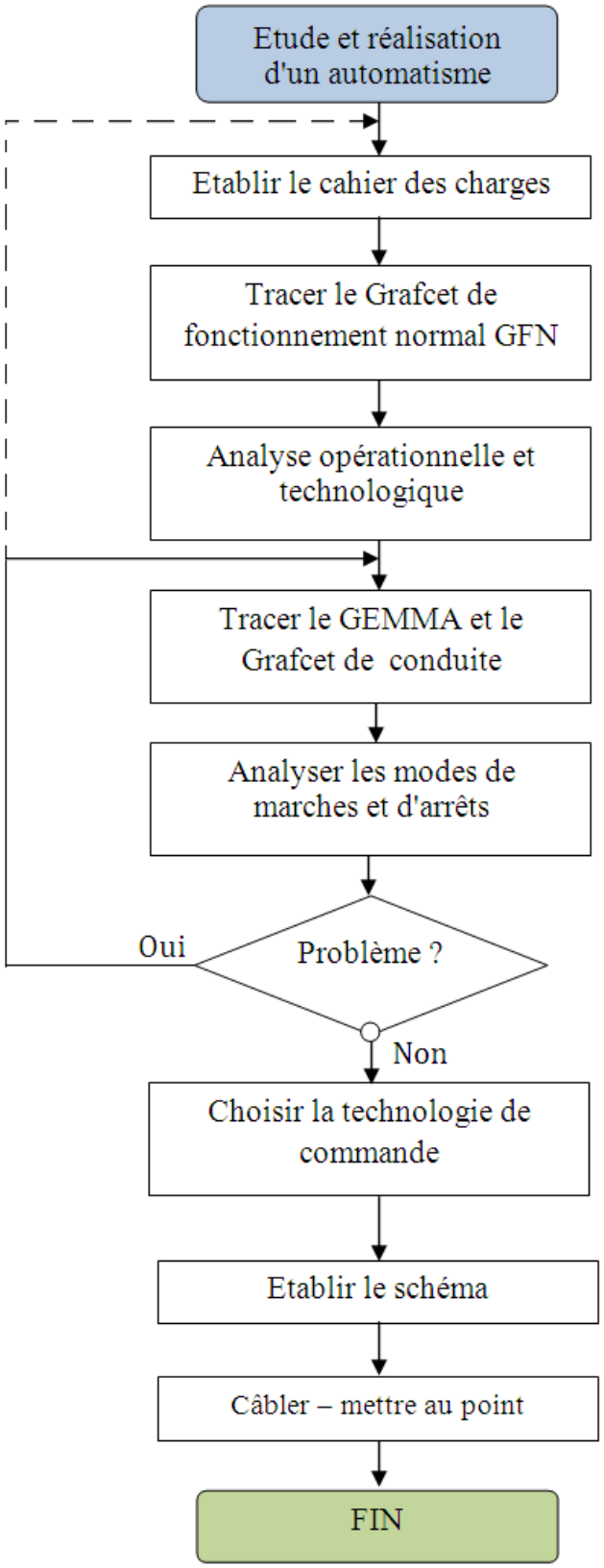


Figure 2-1: Organigramme de Conception d'un automatisme séquentiel

2. Le GRAFCET

2.1 Définition :

Le **GRAFCET** (**GRA**phe **F**onctionnel de **C**ommande par **E**tapes et **T**ransitions) ou **SFC** (**S**equential **F**onction **C**hart) est un outil **graphique** qui décrit les différents comportements de l'évolution d'un **automatisme** et établit une correspondance à caractère séquentiel et combinatoire entre :

- Les **ENTREES**, c'est-à-dire les transferts d'informations de la Partie Opérative vers la Partie Commande,
- Les **SORTIES**, transferts d'informations de la Partie Commande vers la Partie Opérative.

C'est un outil graphique puissant, directement exploitable, car c'est aussi un langage pour la plupart des **API** existants sur le marché. Lorsque le mot **GRAFCET** (en lettre capitale) est utilisé, il fait référence à l'outil de **modélisation**. Lorsque le mot **grafcet** est écrit en minuscule, il fait alors référence à **un modèle** obtenu à l'aide des règles du **GRAFCET**.

Le **GRAFCET** comprend :

- des **étapes** associées à des actions ;
- des **transitions** associées à des **réceptivités** ;
- des **liaisons orientées** reliant étapes et transitions.

La description du comportement attendu d'un automatisme peut se représenter par un **GRAFCET** d'un certain « **niveau** ». La caractérisation du «**niveau**» du GRAFCET nécessite de prendre en compte trois dimensions :

- **Le point de vue**, caractérisant le point de vue selon lequel un observateur s'implique dans le fonctionnement du système pour en donner une description. On distingue trois points de vue :
    - Un point de vue système,
    - Un point de vue Partie Opérative,
    - Un point de vue Partie Commande.

- **La spécifications**, caractérisant la nature des spécifications techniques auxquelles doit satisfaire la Partie Commande. On distingue trois groupes de spécifications :
    - Spécifications fonctionnelles,
    - Spécifications technologiques,
    - Spécifications opérationnelles.

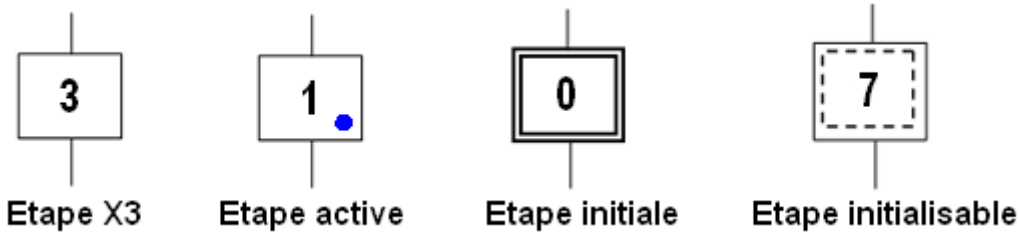
- **La finesse**, caractérisant le niveau de détail dans la description du fonctionnement, d'un niveau global (ou macro-représentation) jusqu'au niveau de détail complet où toutes les actions et informations élémentaires sont prises en compte.

2.2 Les concepts de base du GRAFCET

1 - Etape

Une **étape** symbolise un état ou une partie de l'état du système automatisé. L'étape possède deux états possibles : **active** représentée par un jeton dans l'étape ou **inactive**. L'étape i, représentée par un carré repéré numériquement, possède ainsi une variable d'état, appelée variable d'étape **X**i. Cette variable est une variable booléenne valant **1** si l'étape est active, **0** sinon.

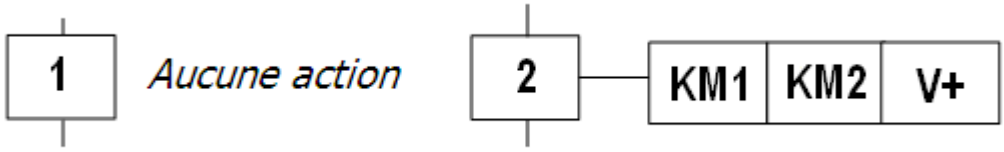
La situation initiale d'un système automatisé est indiquée par une étape dite **étape initiale** et représentée par un carré double.



Remarque : ***Dans un** grafcet **il doit y avoir au moins une étape initiale.***

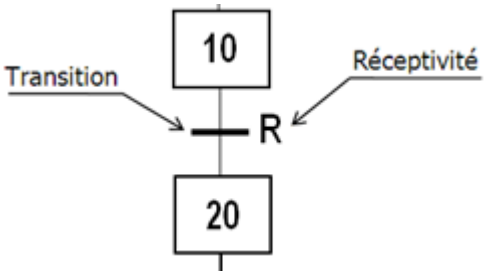
2 - Actions associées aux étapes

A chaque étape est associée une **action** ou plusieurs, c'est à dire un ordre vers la partie opérative ou vers d'autres grafcets. Mais on peut rencontrer aussi une même action associée à plusieurs étapes ou une étape **vide** (*sans action*).



3 - Transition

Une transition indique la possibilité d'évolution qui existe entre deux étapes et donc la succession de deux activités dans la partie opérative. Lors de son franchissement, elle va permettre l'évolution du système. A chaque transition est associée une condition logique appelée réceptivité qui exprime la condition nécessaire pour passer d'une étape à une autre.



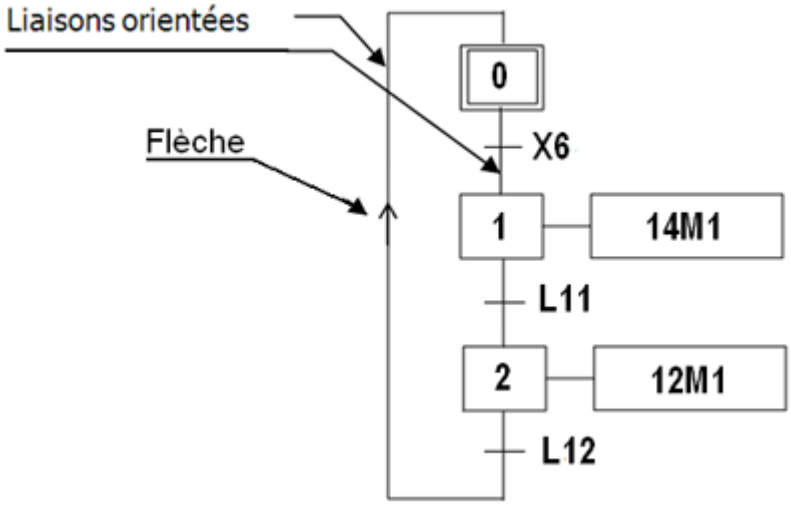
La réceptivité qui est une information d'entrée qui est fournie par :

- l'opérateur : pupitre de commande,
- la partie opérative : états des capteurs,
- du temps, d'un comptage ou toute opération logique, arithmétique...
- du grafcets : d'autres grafcet pour la liaison entre grafcets ou de l'état courant des étapes du grafcet (les Xi),
- d'autres systèmes : dialogue entre systèmes,
- .....

Remarque:
Si la réceptivité n'est pas précisée, alors cela signifie qu'elle est toujours vraie. (=1)

4 - Liaisons orientées

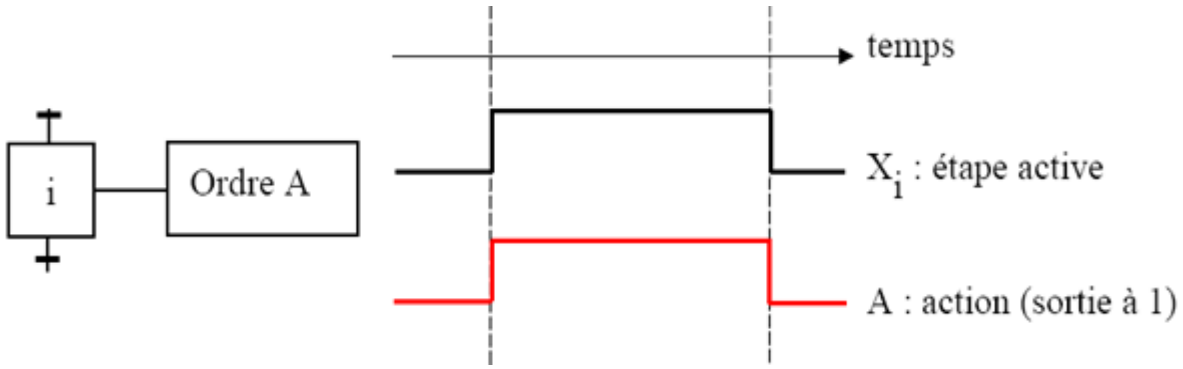
Elles sont de simples traits verticaux qui relient les étapes aux transitions et les transitions aux étapes. Elles sont normalement orientées de haut vers le bas. Une flèche est nécessaire dans le cas contraire.



2.3 Classification des actions associées aux étapes

L'action associée à l'étape peut être de 3 types : **continue**, **conditionnelle** ou **mémorisée**. Les actions peuvent être classées en fonction de leur durée par rapport à celle de l'étape.

Actions continues :



L'exécution de l'action se poursuit tant que l'étape à laquelle elle est associée reste active.

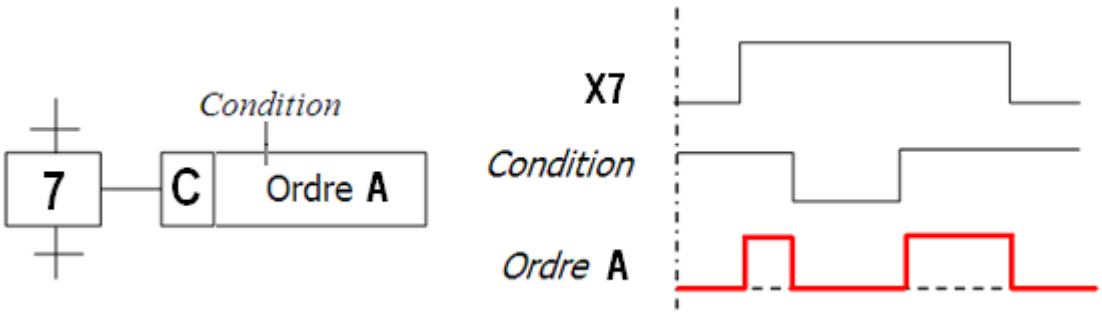
Actions conditionnelles:

C'est une action non mémorisée dont l'exécution est soumise à une condition logique.

Une action **conditionnelle** n'est exécutée que si l'étape associée est active et si la condition associée est vraie. Elles peuvent être décomposées en 3 cas particuliers:

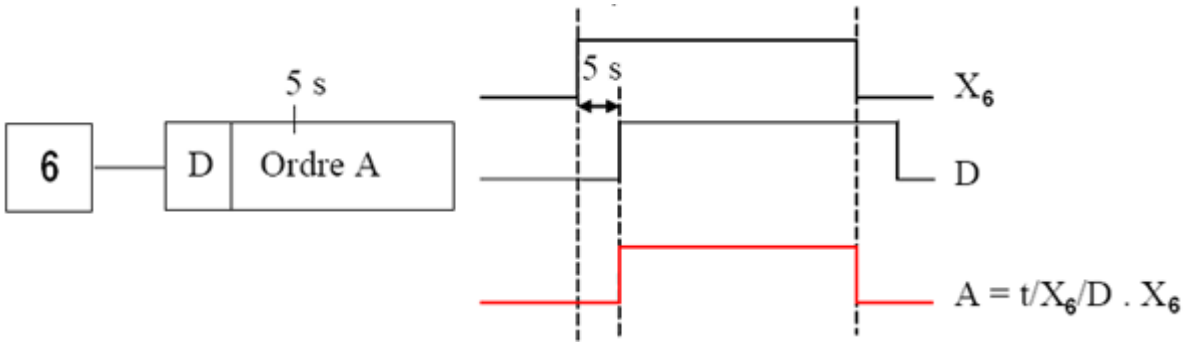
.

Action conditionnelle simple : *Type C*



- Action retardée : *Type D (delay)*

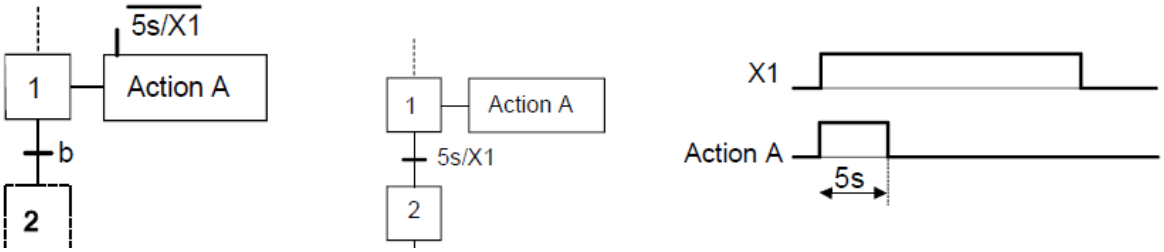
C'est une action toujours associée à une étape mais qui est vrai à l'issu d'un temps défini par l'utilisateur.
Le temps intervient dans cet ordre conditionnel comme condition logique. L'indication du temps s'effectue par la notation générale " t / xi / q " dans laquelle "xi" indique l'étape prise comme origine du temps et "q" est la durée du retard.



Exemple : "t /x6/ 5s" : prendra la valeur logique 1, 5s après la dernière activation de l'étape 6.

- Action de durée limitée:

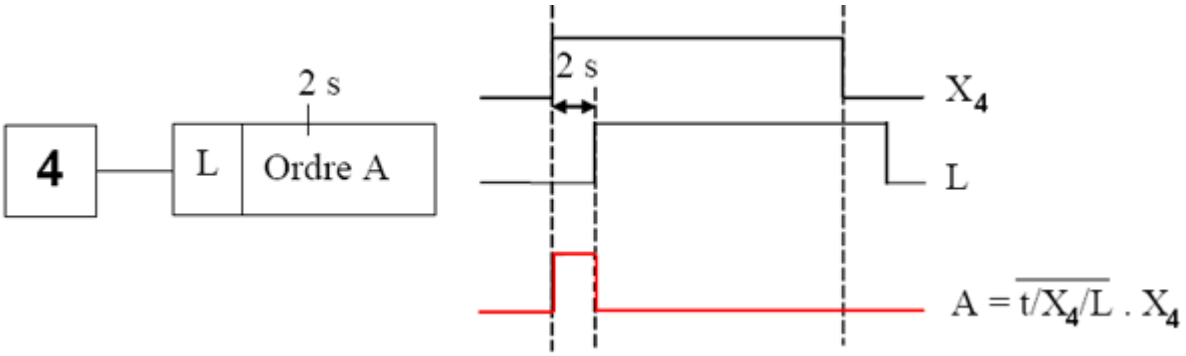
L'action limitée dans le temps est une action continue dont la condition d'assignation n'est vraie que pendant une durée t1 spécifiée depuis l'activation de l'étape à laquelle elle est associée.



- Action impulsionnelle

C'est une action de durée très petite dont la valeur est sans importance mais suffisante à priori pour obtenir l'effet souhaité.

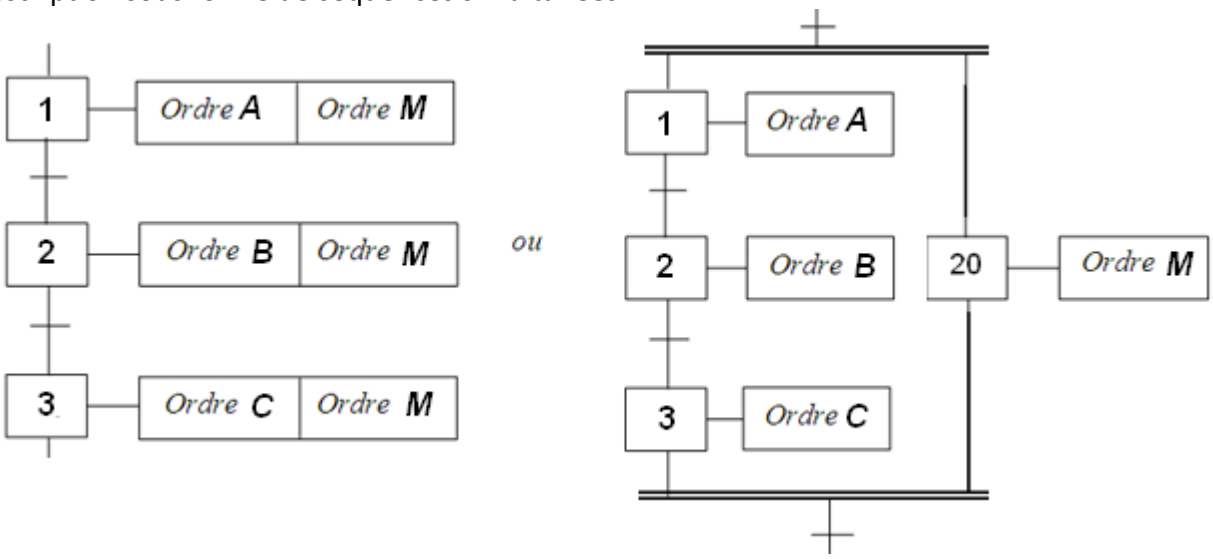
L'ordre est émis dès l'activation de l'étape à laquelle il est associé ; mais la durée de cet ordre sera limitée à une valeur spécifiée.



L'ordre "A" est limité à 2s après l'activation de l'étape 4.

Action maintenue sur plusieurs étapes:

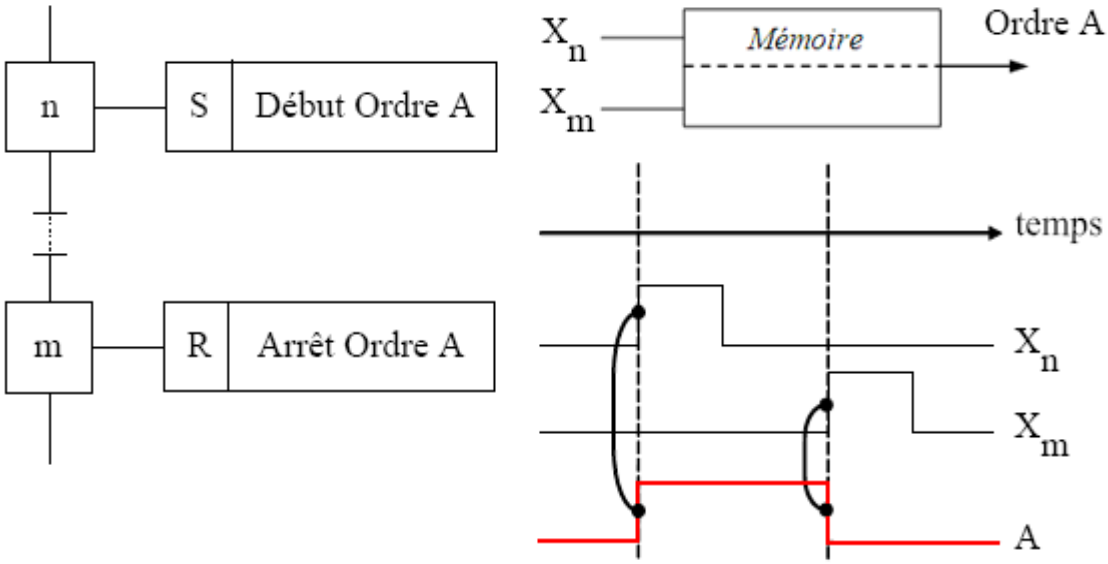
Afin de maintenir la continuité d'une action sur plusieurs étapes, il est possible de répéter l'ordre continu relatif à cette action, dans toutes les étapes concernées ou d'utiliser une description sous forme de séquences simultanées.



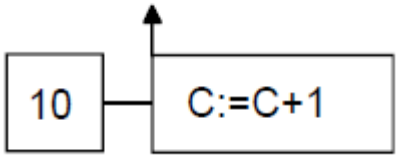
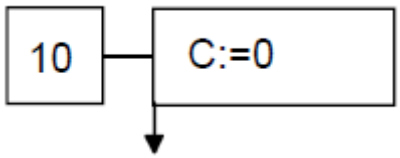
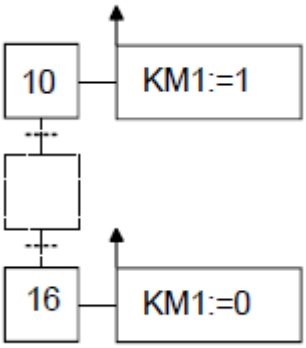
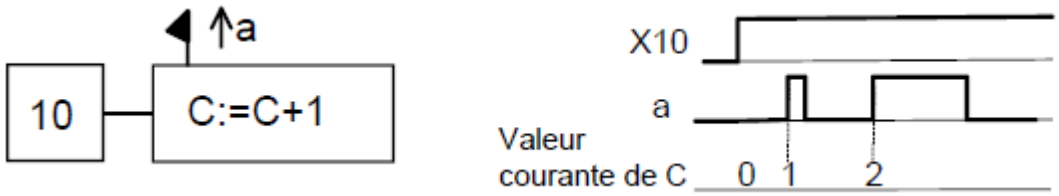
Action mémorisée :

Elle est associée à deux étapes, une étape de déclenchement : **opérateur S** comme «**set**», une étape d'arrêt : **opérateur R** comme «**reset**». L'action mémorisée est exécutée dès que l'opérateur S est activé et reste active jusqu'à l'opérateur R.

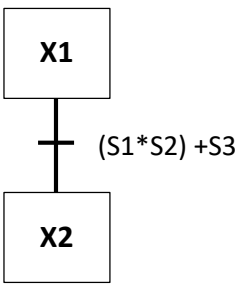
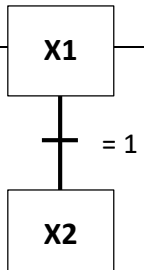
Le maintien d'un ordre, sur la durée d'activation de plusieurs étapes consécutives, peut également être obtenu par la mémorisation de l'action, obtenue par l'utilisation d'une fonction auxiliaire appelée fonction mémoire.

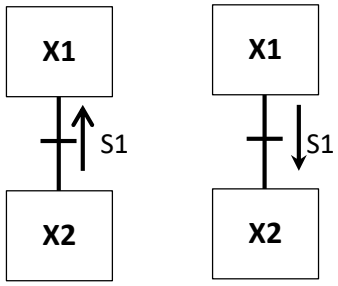
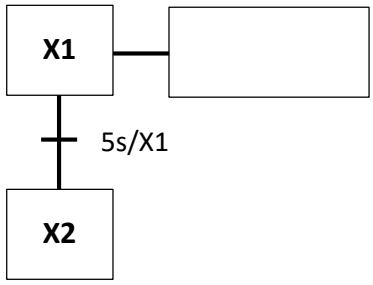
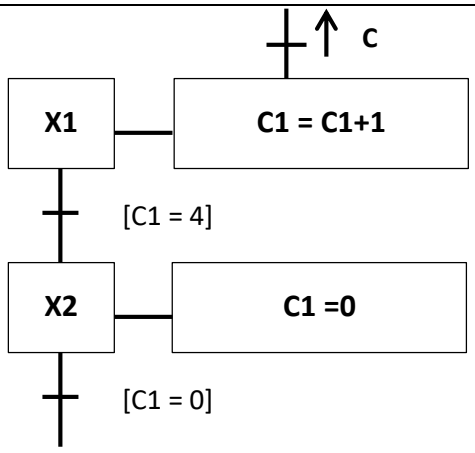


Action à l'activation et à la désactivation

Une action à l'activation est une action mémorisée lors de l'activation de l'étape liée à cette action.  Incrémentation du compteur C à l'activation de l'étape 10	**Une action à la désactivation** est une action mémorisée lors de la désactivation de l'étape liée à cette action.  Mise à 0 du compteur C à la désactivation de l'étape 10.
Exemple 1:  KM1=1 dès l'activation de l'étape 10 et reste à 1 jusqu'à l'activation de l'étape 16.	
Exemple 2:  Incrémentation du compteur C sur le front montant de « a », l'étape 10 étant active.	

2.4 Classification des transitions :

Réceptivité (équation logique):	 	La transition X1 –X2 est franchie si l'équation (S1 * S2) + S3 est vraie.
	 	

Réceptivité (toujours vraie) :		La transition X1 –X2 est franchie dès que le l'étape X1 est active.
Réceptivité (événements fronts):	cas N°1 cas N°2 	La transition X1 –X2 est franchie: • lors d'un **front montant** sur S1 (cas N°1), • ou lors d'un **front descendant** sur S1 (cas N°2).
Réceptivité (comparaison d'un temps associé à une temporisation):		La transition X1 –X2 est franchie lorsque la temporisation, démarrée à l'étape X1 est écoulée, soit au bout de 5s. **Remarque:** **la temporisation doit être également programmée en action.**
Réceptivité (comparaison d'une valeur associée à du comptage):		La transition X1-X2 est franchie si le compteur C1 a atteint la valeur 4. **Remarque:** le compteur est incrémenté de 1 sur l'étape X1 à condition d'avoir le front montant C.

2.5 Règles d'évolution d'un GRAFCET

Règle N°1 : Condition initiale

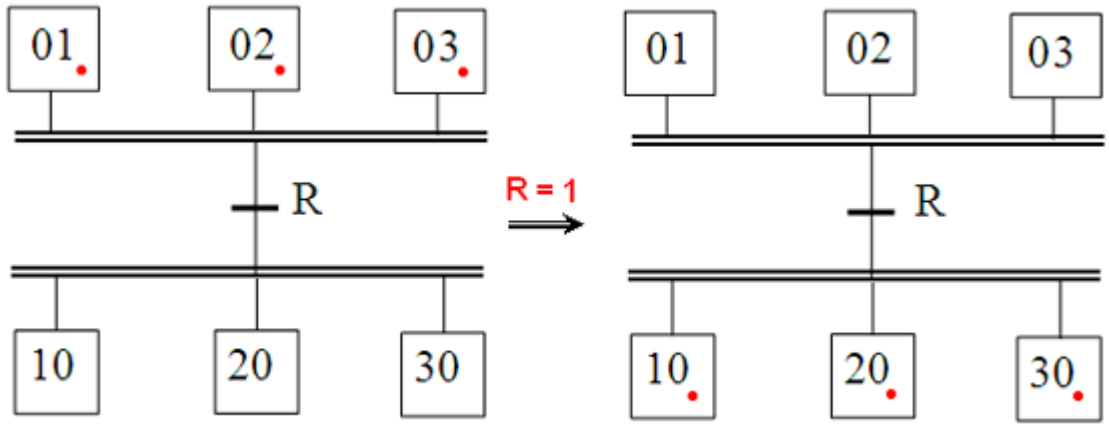
A l'instant initial, seules les étapes initiales sont actives.

Règle N°2 : Franchissement d'une transition.

Pour qu'une transition soit validée, il faut que toutes ses étapes amont (immédiatement précédentes reliées à cette transition) soient actives. Le franchissement d'une transition se produit lorsque la transition est validée, **ET seulement si** la réceptivité associée est **vraie**.

Règle N°3 : Evolution des étapes actives

Le franchissement d'une transition entraîne obligatoirement l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.



Règle N°4 : Franchissement simultané

Toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies.

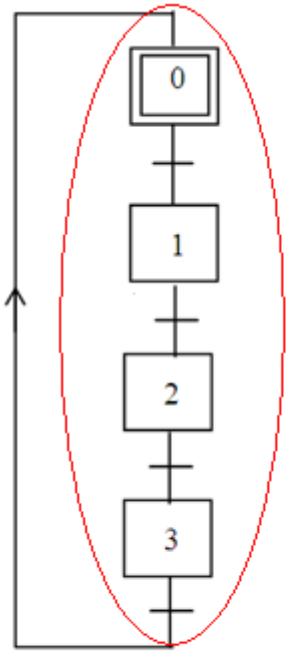
Règle N°5 : Conflit d'activation

Si une étape doit être simultanément désactivée par le franchissement d'une transition aval, et activée par le franchissement d'une transition amont, alors elle reste active. On évite ainsi des commandes transitoires (néfastes à la partie opérative).

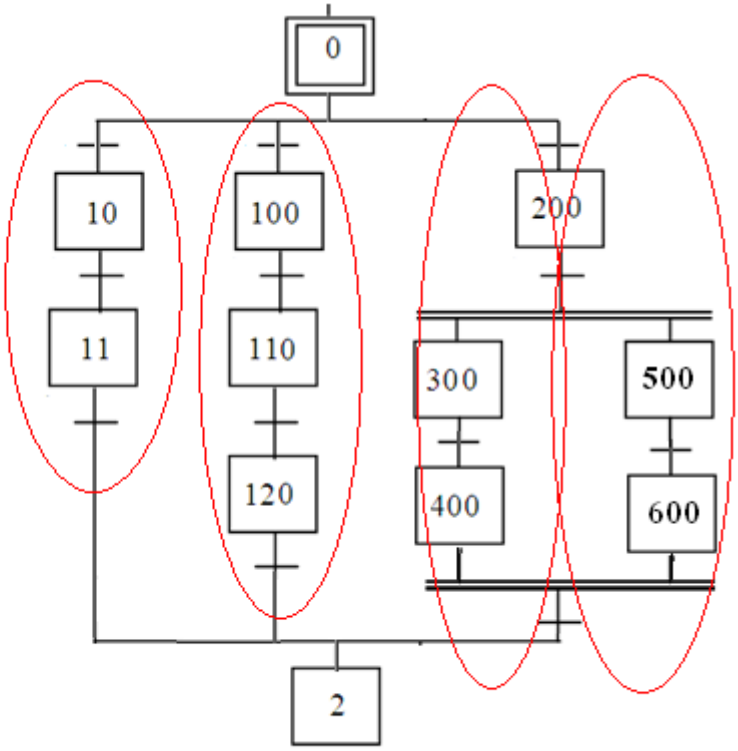
2.6 Les structures de base

Notion de Séquence :

Une séquence, dans un Grafcet, est une suite d'étapes à exécuter l'une après l'autre.
Autrement dit chaque étape ne possède qu'une seule transition AVAL et une seule transition AMONT.



Grafcet à séquence unique.

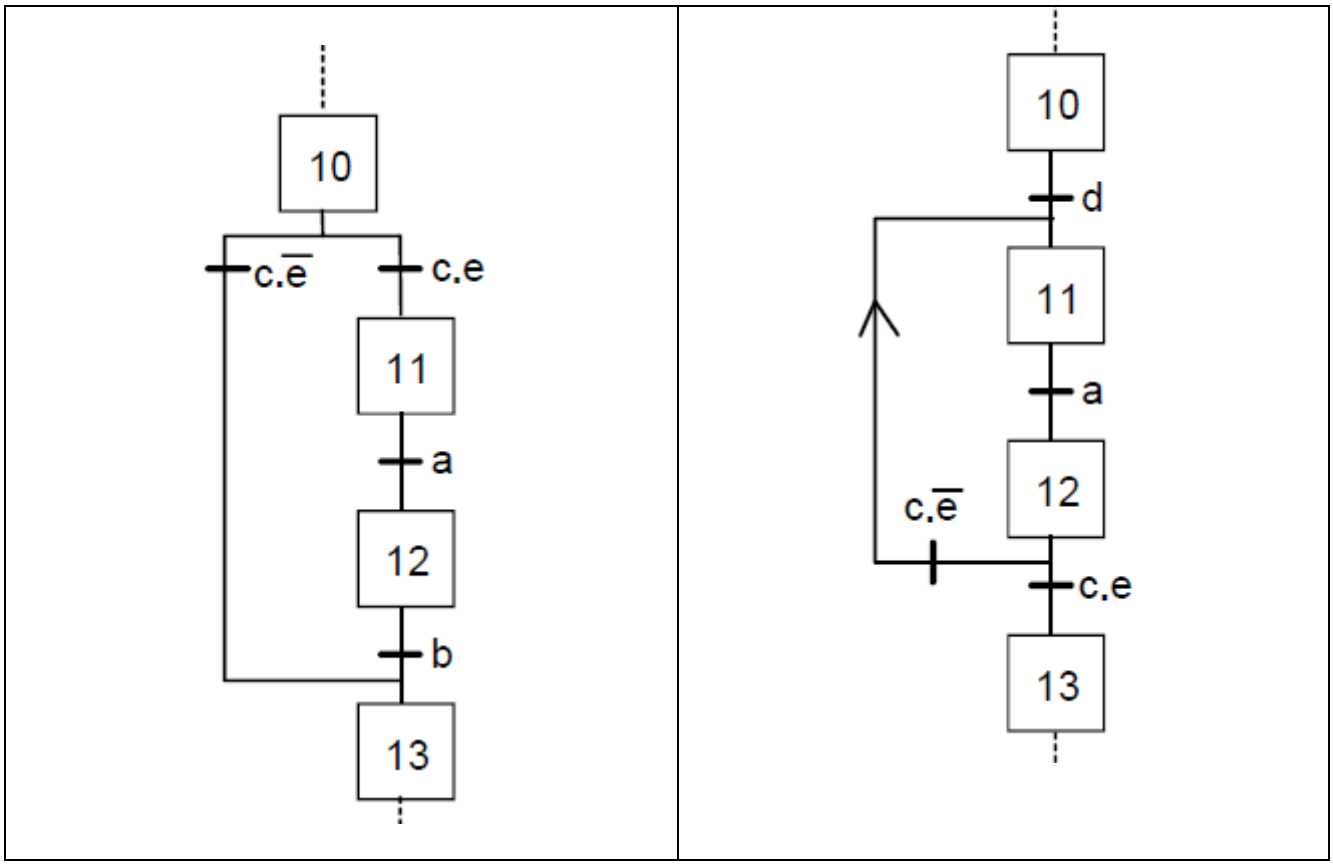


Grafcet à plusieurs séquences.

Saut d'étapes et reprise de séquence

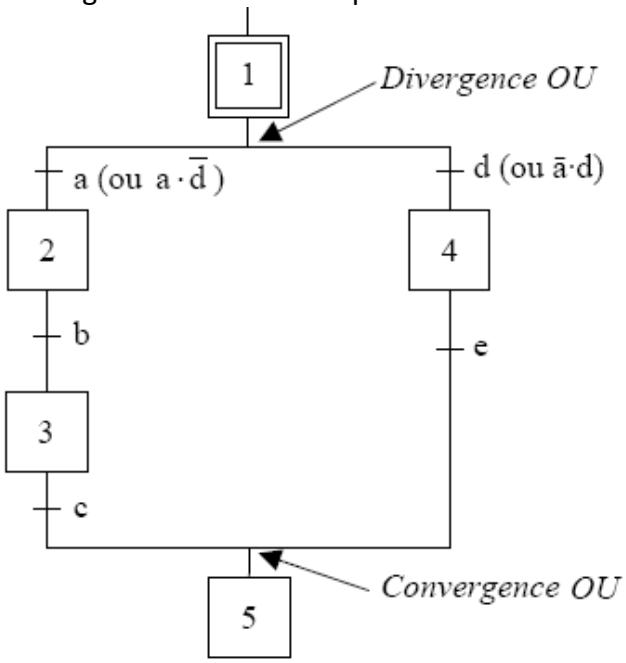
Le saut d'étapes permet de sauter une ou plusieurs étapes lorsque les actions associées sont inutiles à réaliser, La reprise de séquence (ou boucle) permet de reprendre, une ou plusieurs fois, une séquence tant qu'une condition n'est pas obtenue.

Saut d'étapes	***Reprise de séquence***
Le saut d'étapes permet de sauter une ou plusieurs étapes lorsque les actions associées à ces étapes deviennent inutiles.	La reprise de séquence permet de recommencer plusieurs fois la même séquence tant qu'une condition n'est pas obtenue.



Aiguillage entre deux ou plusieurs séquences (Divergence en OU)

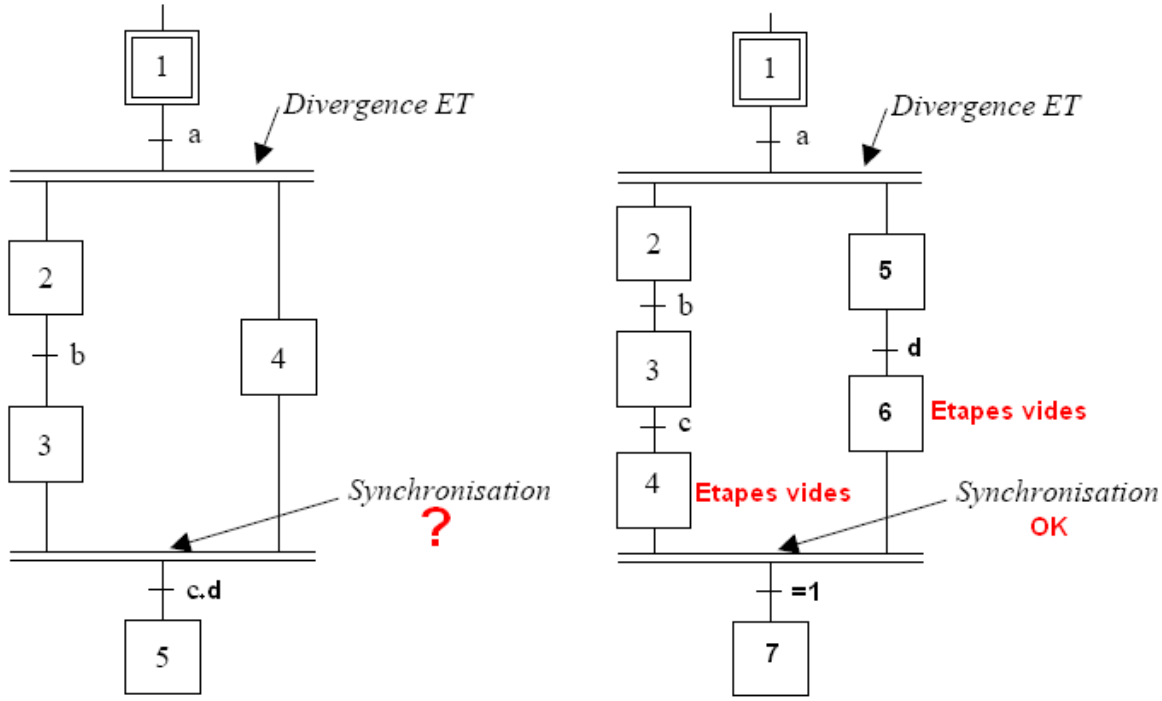
On dit qu'il y a **Aiguillage** ou **divergence en OU** lorsque le grafcet se décompose en deux ou plusieurs séquences selon un choix conditionnel. Comme la divergence en OU on rencontre aussi la convergence en OU. On dit qu'il y a convergence en OU, lorsque deux ou plusieurs séquences du grafcet converge vers une seule séquence.



Si les deux conditions a et d sont à 1 simultanément, les étapes 2 et 4 vont devenir actives simultanément, situation non voulue par le concepteur. Donc elles doivent être des conditions **exclusives**

Parallélisme entre deux ou plusieurs séquences (ou séquences simultanées ou divergence convergence en ET) :

Au contraire de l'aiguillage où ne peut se dérouler qu'une seule activité à la fois, On dit qu'on se trouve en présence d'un parallélisme structurel, si plusieurs activités indépendantes pouvant se dérouler en parallèle. Le début (d'une divergence en ET) et (la fin d'une convergence en ET) d'un parallélisme structurel sont représentés par deux traits parallèles.



La synchronisation permet d'attendre la fin de plusieurs activités se déroulant en parallèle, pour continuer par une seule.

2.7 Grafcet(s) hiérarchisés :

Les Systèmes Automatisés de production sont de plus en plus complexes, afin de simplifier l'étude, la mise en œuvre et la maintenance du système, il est nécessaire de **structurer la partie commande et la partie opérative.**

L'objectif essentiel de la structuration est de permettre une approche progressive du fonctionnement d'un système automatisé, tant au niveau de l'analyse qu'au niveau de la représentation.

Dans l'analyse structurée, **le grafcet global est décomposé en modules**, chacun de ces modules correspond à une fonction du système (Sécurité, modes de marche, etc.) ou à une sous partie de la Partie Opérative (Poste 1, Poste 2, Poste 3, etc)

La structuration est:

- soit Hiérarchique (**GRAFCET Maître, GRAFCET Esclave**)
- soit sans hiérarchie (communication entre 2 postes).

Les commandes de **forçage** et **figeage** de grafcet, sont des moyens supplémentaires qui permettent de **préciser la hiérarchie des différents grafcets.**

Les principaux GRAFCETS que l'on retrouve souvent :

- **GRAFCET de surveillance (de sécurité):** Ce GRAFCET décrit l'ensemble des procédures de sécurité du système, c'est le GRAFCET hiérarchiquement le plus important. L'arrêt d'urgence et les procédures de mise en route sont décrits dans ce GRAFCET.

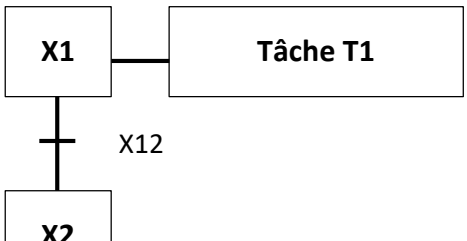
- **GRAFCET de conduite (ou GRAFCET des Modes de Marches) :** Ce GRAFCET décrit l'ensemble des procédures de Marches (auto, Cycle/Cycle, Manuel,...) et des arrêts normaux.

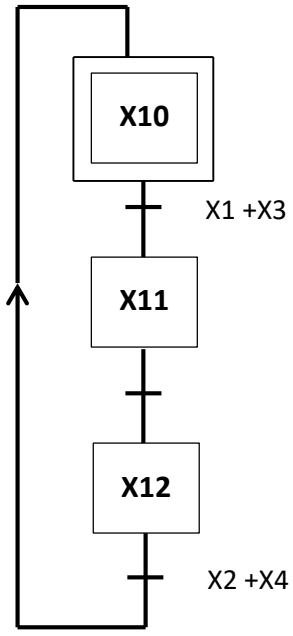
- **GRAFCET de maintenance**: Précise les procédures d'intervention de l'opérateur et de réglage de la partie opérative.

- **GRAFCET de Production**: Ce GRAFCET est le niveau de description du fonctionnement normal de l'automatisme. Ce GRAFCET est en général décomposé en plusieurs taches représentant les différentes fonctions de l'automatisme.

Comme évoqué précédemment, les GRAFCET(s) hiérarchisés forment une structure **de type maître/esclave** dans laquelle le **GRAFCET maître** donne des ordres à un ou plusieurs **GRAFCET esclaves.**

On parle alors de **GRAFCET de tâches** ou de sous-programmes GRAFCET.
Les GRAFCET esclaves renvoient un accusé d'exécution en fin de tâche.





GRAFCET Maître **GRAFCET de tâche T1**

Ordre de figeage ou de forçage

- Il faut définir des graphes de niveau hiérarchique différents
- Un graphe peut figer ou forcer un autre dans une situation donnée si il est de niveau hiérarchique supérieur.

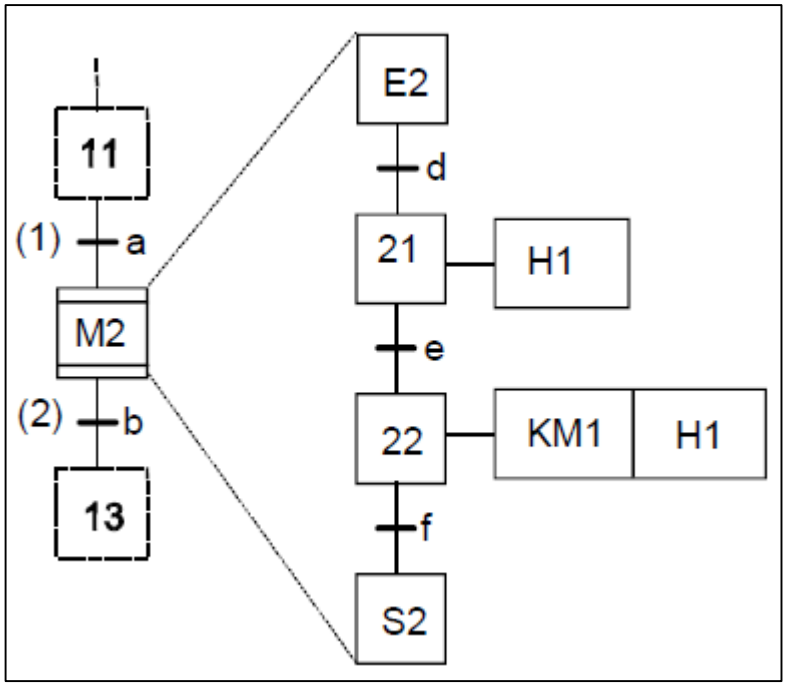
Forçage	Figeage	Figeage
Grafcet G3 X31 — G2 {20} Grafcet G2 X31 → X20	Grafcet G3 X31 — F/G2 :(*) Grafcet G2 X31 → (*)G2 X20	Grafcet G3 X31 — F/G2 :(*(21)) Grafcet G2 X31 → (*) X21
A l'étape X31 du GRAFCET G3, il y a forçage du GRAFCET G2 à l'étape X20. Plus d'évolution sur G2 tant que X31=1	L'activation de l'étape X31 du GRAFCET G3 fige le GRAFCET G2 dans sa situation courante. Plus d'évolution sur G2 tant que X31=1	L'activation de l'étape X31 du GRAFCET G3 figera le GRAFCET G2 sur l'étape X21. par rapport au forçage, le grafcet G2 continue d'évoluer mais se boquera sur l'étape X21

Structuration par macro-étapes

Avec la notion de macro-représentation, on se donne le moyen de reporter à plus tard ou sur une autre page la description détaillée de certaines séquences.

La macro-étape est la représentation unique d'un ensemble d'étapes et de transitions nommé expansion de macro-étape.

Exemple d'une macro étape M2 représentée avec son expansion :

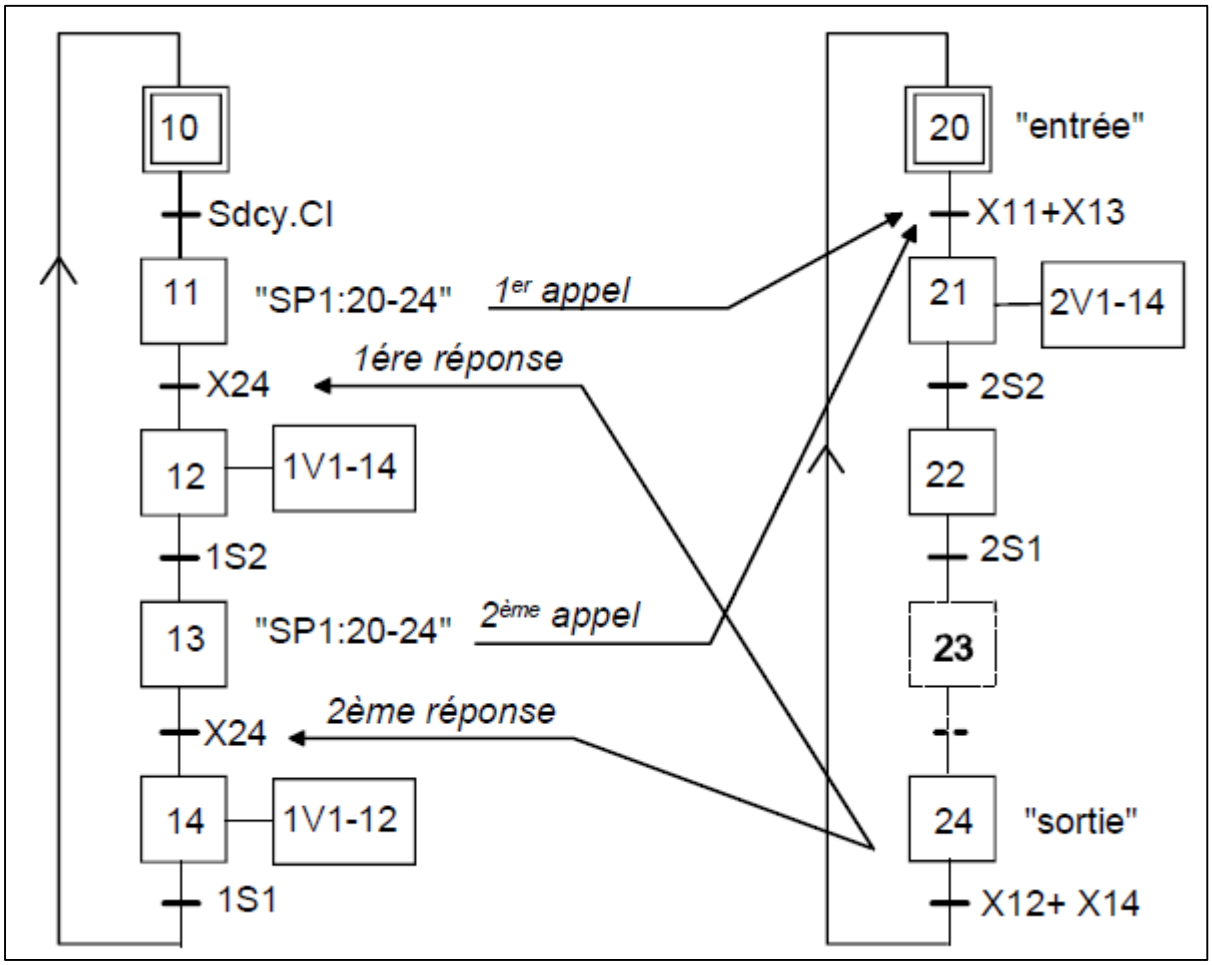


L'expansion de la macro-étape commence par une seule étape d'entrée et se termine par une seule étape de sortie, étapes qui représentent les seuls liens possibles avec le GRAFCET auquel elle appartient.

- Le franchissement de la transition (1) active l'étape E2.
- La transition (2) ne sera validée que lorsque l'étape S2 sera active.
- Le franchissement de la transition (2) désactive l'étape S2.

Structuration par GRAFCET de tâches et/ou sous-programme

La norme EN 60848 ne fait pas référence à ces notions et ne définit donc pas de symboles graphiques pour le GRAFCET de gestion des tâches. Nous pouvons continuer à utiliser la structuration par GRAFCET de sous-programme (s) en indiquant, entre guillemets (et pas dans un rectangle d'action), le nom du sous-programme appelé.

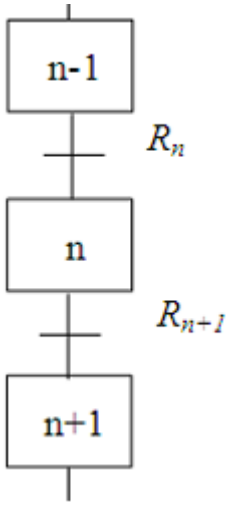


2.8 Mise en équation d'un grafcet :

Règle générale :

Pour qu'une étape soit activée il faut que :

- L'étape immédiatement précédente soit active ;
- La réceptivité immédiatement précédente soit vraie ;
- L'étape immédiatement suivante soit non active ;
- Après activation l'étape mémorise son état.



L'équation d'activation de l'étape de rang n est donnée par :

$$\mathbf{Xn} = (\mathbf{Xn} - \mathbf{1}.\mathbf{Rn} + \mathbf{Xn}).\overline{\mathbf{Xn} - \mathbf{1}}$$

3. LE GEMMA

3.1 Introduction

Le **GEMMA** (**G**uide d'**E**tude des **M**odes de **M**arche et d'**A**rrêt) est un **guide graphique** élaboré par l'Agence pour le Développement de la Productique Appliquée (**ADEPA)** en avril 1981, Il constitue une méthode d'approche des Modes de Marches et d'Arrêts des systèmes, c'est un outil d'aide complémentaire au **GRAFCET** qui permet d'exprimer de façon claire et complète les besoins en modes de marche et d'arrêt d'un système automatisé.

Le GEMMA est un document (outil-méthode) **structuré** prêt à être rempli par son utilisateur afin de suivre une approche guidée et systématique. Il précise les procédures à mettre en œuvre après analyse d'une anomalie ou un défaut de fonctionnement.

Un GEMMA est établi pour chaque machine lors de sa conception, puis utilisé tout au long de sa vie : réalisation, mise au point, maintenance, modifications, réglages… Dans ses principes et dans sa mise en œuvre, le GEMMA doit donc être connu de toutes les personnes concernées par les automatismes, depuis leur conception jusqu'à leur exploitation.

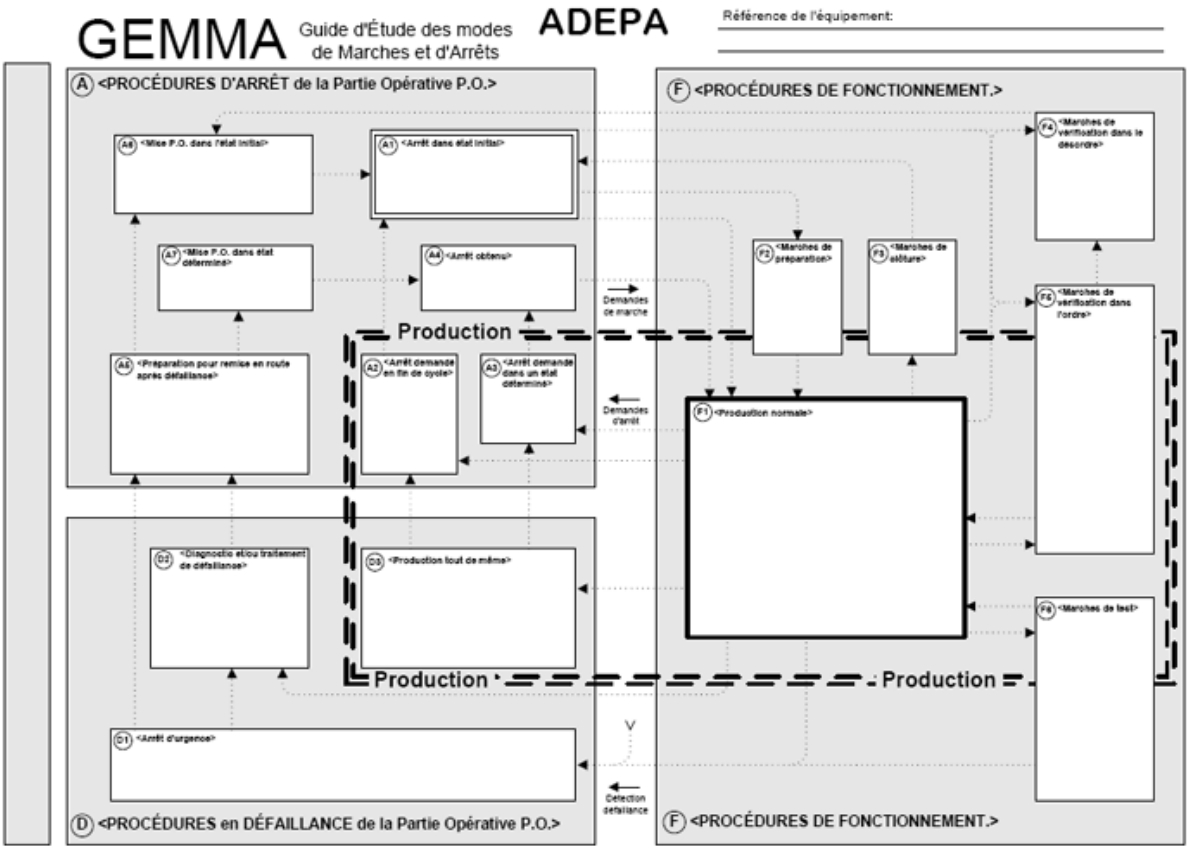


Figure 2-2 : Document GEMMA vide

3.2 Les concepts de base :

Comme le montre le schéma ci-dessous, le GEMMA définit l'état dans lequel se trouve la partie commande du système automatisé. Dans un premier temps, on peut dire que le GEMMA est divisée en deux zones :

- Partie commande hors énergie
- Partie commande sous énergie

Dans la zone **PC hors énergie**, la partie commande n'est pas alimentée en énergie. Dans cette partie il n'y a pas de modes traités par la partie commande. Seules des actions dites actions réflexes ou externes peuvent se réaliser.

Pour la zone **Partie commande sous énergie** et selon les besoins du système automatisé à étudier on choisit d'utiliser certains modes de marches et d'arrêts. En effet les modes de marches et d'arrêts ne peuvent être perçus et traités qu'en présence d'une partie commande en ordre de marche. A chacun de ces modes correspond un "**rectangle état**" disposé sur le graphisme selon une structure précise.

Une distinction supplémentaire est faite dans la zone **PC sous énergie**. On distingue la zone de production et la zone hors production par un double encadrement en pointillés de la zone de production. Les **Modes de Marches et d'Arrêts** à l'intérieur des pointillés "**Production**", correspondent à des états pour lesquels la machine produit.

Chaque famille de procédures correspond à une zone du Gemma. On distingue trois familles:

- Les **procédures de fonctionnement** regroupant les états **F**. On ne produit pas forcément dans tous les états **F**, Les modes préparatoires à la production, de réglages ou, de tests, peuvent faire partie de cette famille.
- Les **procédures d'arrêt** regroupant les états **A**, tous les modes conduisant ou traduisant à un état d'arrêt, arrêts normaux et procédures de remise en route.
- Les **procédures de défaillance**, regroupant les états **D** pris en cas de défaillance de la partie opérative.

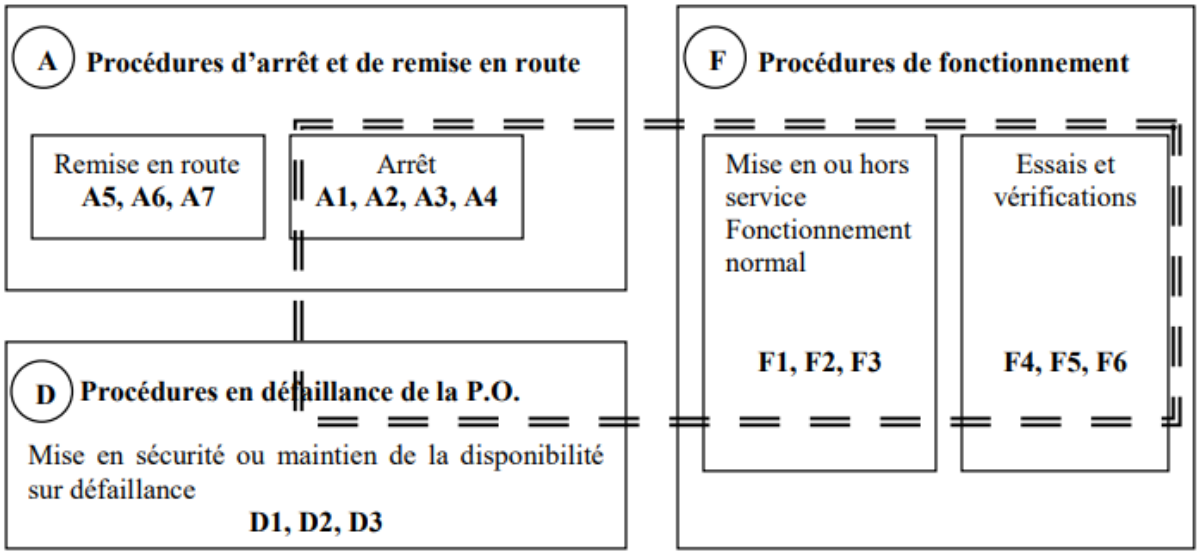


Figure 2-3 : Les familles et les sous-familles de procédures

Une distinction supplémentaire est faite parmi ces trois familles de procédures. On distingue la zone de production de la zone hors production par un double encadrement de la zone de production en pointillés. La zone de production de trouve à cheval sur les trois types de procédures.

Le guide graphique GEMMA porte les **rectangles-états** dans lesquels seront exprimés les différents états de Marches et d'Arrêts (M/A) pris par la machine.

Les états F

ETATS F: PROCEDURES DE FONCTIONNEMENT		
Repère	**Désignation**	**Description**
F1	<production normale>	Dans cet état la machine produit normalement : c'est l'état pour lequel elle a été conçue. On peut souvent faire correspondre à cet état un grafcet que l'on appelle "grafcet de base". Note : A cet état ne correspond pas nécessairement une marche automatique
F2	<marche de préparation>	Cet état est utilisé pour les machines nécessitant une préparation préalable à la production normale :

		Préchauffage de l'outillage, remplissage, mises en routes diverses ...
F3	<marche de clôture>	C'est l'état nécessaire pour certaines machines devant être vidées, nettoyées ... en fin de journée ou en fin de série.
F4	<marche de vérification dans le désordre>	Cet état permet de vérifier certaines fonctions ou certains mouvements sur la machine sans respecter l'ordre de déroulement du cycle.
F5	<marche de vérification dans l'ordre>	Dans cet état, le cycle de production peut être exploré au rythme de production voulu par la personne effectuant la vérification
F6	<marche de test>	Les machines de contrôle, de tri, de mesure... comportent des capteurs qui doivent être réglés ou étalonnés : cet état permet les différentes opérations.

Les états A

ETATS A: PROCEDURES D'ARRET		
Repère	**Désignation**	**Description**
A1	<Arrêt dans état initial>	C'est l'état repos de la machine. Il correspond en général à la situation initiale du grafcet.
A2	<Arrêt demandé en fin de cycle>	Lorsque l'arrêt est demandé, la machine continue de produire jusqu'à la fin de cycle; l'état A2 est donc un état transitoire vers l'état A1
A3	<Arrêt demandé dans état déterminé>	La machine continue de produire jusqu'à un arrêt en une position autre que la fin de cycle; c'est un état transitoire vers A4
A4	<Arrêt Obtenu>	La machine est alors arrêtée dans un état autre que la fin de cycle.
A5	<Préparation pour remise en route après défaillance>	C'est dans cet état que l'on procède à toutes les opérations (désengagements, nettoyages ...) nécessaires à une remise en route après défaillance
A6	<Mise PO dans état initial>	La machine étant en A6, on remet manuellement ou automatiquement la partie opérative en position initiale pour un redémarrage dans l'état initial.
A7	<Mise PO dans état déterminé>	La machine étant en A7, on remet la partie opérative en position pour un redémarrage dans une position autre que l'état initial.

Les états D

ETATS D: PROCEDURES DE DEFAILLANCE		
Repère	**Désignation**	**Description**
D1	<Arrêt d'urgence>	C'est l'état pris lors d'un arrêt d'urgence : on y prévoit non seulement les arrêts, mais aussi les cycles de dégagement, les procédures et précautions nécessaires pour éviter ou limiter les conséquences dues à la défaillance.
D2	<Diagnostic et/ou traitement de la défaillance>	C'est dans cet état que la machine peut être examinée après défaillance et qu'il peut être apporté un traitement permettant le redémarrage.
D3	<Production tout de même>	Il est parfois nécessaire de continuer la production même après une défaillance de la machine : on aura alors une production dégradée, forcée ou aidée par des opérateurs non prévus en production normale

3.3 Méthode d'utilisation du GEMMA

L'étude des modes de marches et d'arrêts est prévue dès la conception de la machine et intégrée dans sa réalisation. Après l'établissement du **GRAFCET** de production normale (**GPN**), on met en œuvre le guide graphique **GEMMA** pour la sélection des modes de marches et d'arrêts.

La démarche comporte deux phases :

- Le recensement des différents modes envisagés pour le système et la mise en évidence des enchaînements qui les relient.
- La détermination des conditions de passage d'un mode à l'autre.

3.4 Sélection des Modes de Marches et d'Arrêts :

La PO de la machine étant définie, ainsi que le Grafcet du cycle de production normale (**GPN**), il faut sélectionner et préciser les modes de M/A nécessaires.

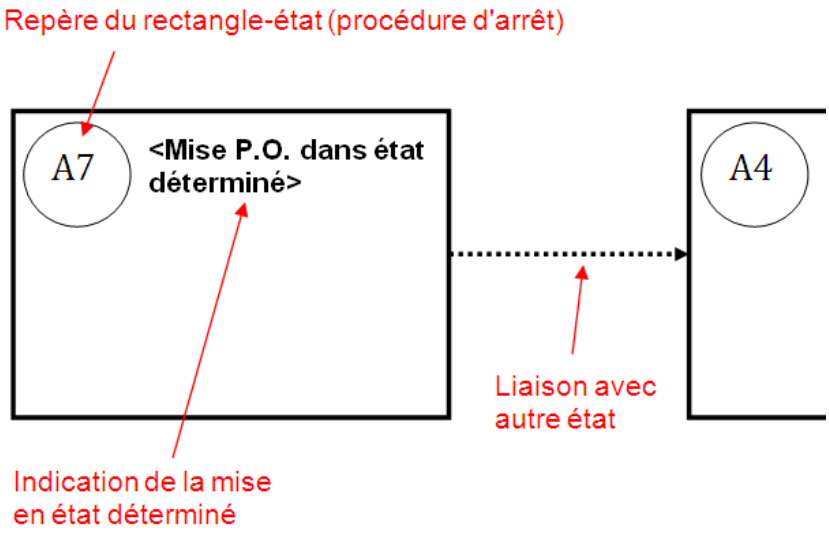


Figure 2.4 : Exemple d'un « rectangle-état » proposé.

Pour une machine donnée, il est important d'examiner le cas de chaque rectangle état :

- Si le mode proposé est retenu, il sera précisé dans le "rectangle - état ".
- Un « rectangle-état » retenu se complète de façon manuscrite : en précisant l'opération exécutée propre à la machine étudiée ; en surlignant la (les) liaison(s) orientée(s) ; en indiquant la (les) condition(s) d'évolution

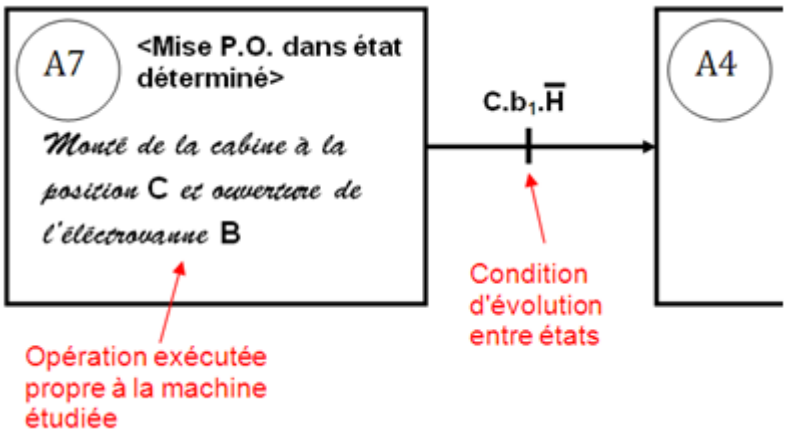


Figure 2.5 : Exemple d'un « rectangle-état » retenu.

- Si le mode proposé n'est pas nécessaire pour la machine, une croix sera portée dans le "rectangle - état" pour signifier qu'il n'est pas retenu.

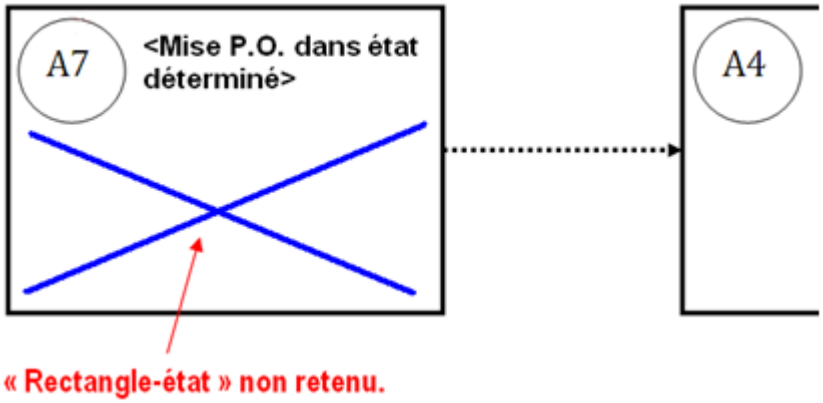


Figure 2.6 : Exemple d'un « rectangle-état » non retenu.

Deux états essentiels, définis dès le début de l'étude, se retrouvent sur toutes les machines :

- ✓ L'état **A1**, dit, ou de la machine.
- ✓ L'état **F1**, mode de pour lequel la machine a été conçue.

En partant de chacun des deux états essentiels, **A1** et **F1**, on recherche les évolutions vers les autres états. On se demande d'abord quelle évolution suivre lors du démarrage.
Le choix est :

- ✓ **A1 → F1** : démarrage sans marche de préparation ;
- ✓ **A1 → F2 → F1**: démarrage avec marche de préparation.

Ce qui permet de répondre à la question suivante : « **une marche de préparation est-elle nécessaire ?** » Ensuite, on se demande **quelle évolution suivre lors de l'arrêt normal de production**. Le choix est alors :

- ✓ **F1 → A2 → A1** : arrêt en fin de cycle sans marche de clôture ;
- ✓ **F1 → F3 → A1** : arrêt avec une marche de clôture ;
- ✓ **F1 → A3 → A4** : arrêt dans un état autre que la condition initiale.

Ce qui permet de répondre aux questions suivantes :

- ✓ Une marche de clôture est-elle nécessaire ?
- ✓ Un arrêt dans un état autre que la condition initial est-il nécessaire?

Puis, on se demande **quelle évolution suivre lors d'une défaillance de l'automatisme**. Un grand nombre de choix est disponible :

- ✓ **F1 → D3** : défaillance légère permettant une marche de production tout de même ;
- ✓ **D1 → D2 → A5 → A7 → A4 → D3** : arrêt d'urgence puis évolution pour production tout de même (défaillance légère impliquant l'arrêt d'un poste) ;
- ✓ **D1 → A5 → A6 → A1** : arrêt d'urgence puis évolution pour un arrêt en condition initiale (défaillance majeure) ;
- ✓ **D1 → A5 → A7 → A4** : arrêt d'urgence puis évolution pour un arrêt dans le même état que lors de l'apparition de l'arrêt d'urgence (défaillance mineure) ;

- ✓ **D1 → D2 → A5 → A6 → A1** : arrêt d'urgence avec diagnostic et traitement, puis évolution pour un arrêt en condition initiale (défaillance majeure) ;
- ✓ **D1 → D2 → A5 → A7 → A4** : arrêt d'urgence avec diagnostic et traitement, puis évolution pour un arrêt dans le même état que lors de l'arrêt d'urgence (défaillance mineure).

Ce qui permet de répondre aux questions suivantes :

- ✓ quel genre de défaillance peut affecter l'automatisme ?
- ✓ y-a-t-il des défaillances pour lesquelles on peut réussir de produire tout de même de façon sécuritaire ?
- ✓ qu'elle doit être le comportement de la machine en arrêt d'urgence ?
- ✓ la machine est-elle suffisamment compliquée pour exiger un diagnostic élaboré ?
- ✓ suite à une défaillance, la machine doit-elle être retournée en condition initiale ?
- ✓ suite à une défaillance, la machine peut-elle être remise en marche en repartant de l'état ou elle était lors de l'apparition de l'arrêt d'urgence ?

3.5 Conditions d'évolution entre modes de marches et d'arrêts

Les modes de marches et d'arrêt ayant été sélectionnés et explicités, il convient de préciser le passage d'un état à l'autre.

L'élaboration de ces conditions de passage rend possible **la conception du pupitre de commande** et entraîne éventuellement **l'adjonction de capteurs supplémentaires.** Le passage d'un état à un autre s'effectue de 2 façons :

- Soit avec une **condition d'évolution**.
- Soit sans **condition d'évolution**

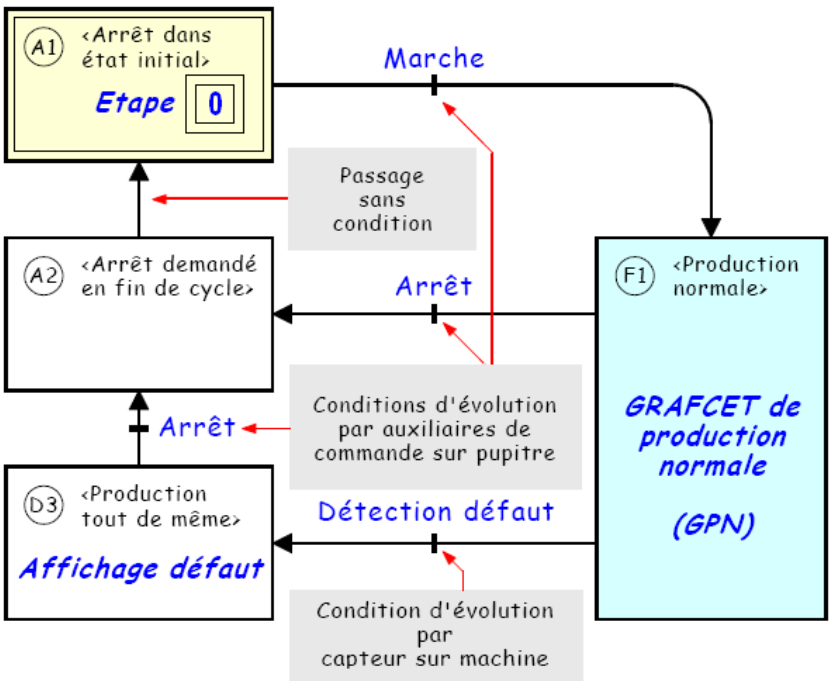


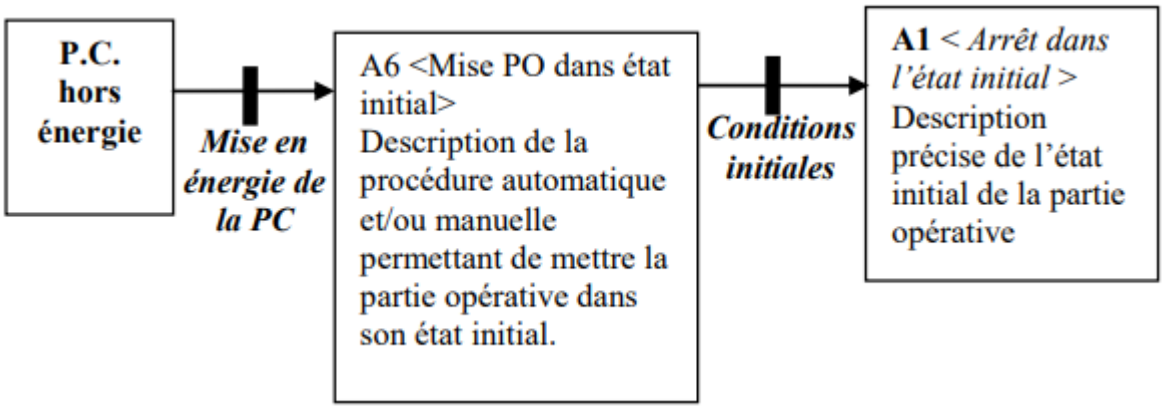
Figure 2.7 : Exemple de conditions d'évolution entre modes de marches et d'arrêts.

3.6 Exemples types de GEMMA :

Sur le GEMMA on caractérise plusieurs "boucles". Une boucle est une succession d'états caractérisant le fonctionnement du système. En effet il n'est possible de passer d'un état à un autre que si les conditions d'évolutions sont respectées, mais il est parfois impossible de passer d'un état à un autre sans utiliser un état intermédiaire. Cet état intermédiaire permettra d'atteindre l'état final sans risque pour le système.

Boucle PC hors énergie – A6 - A1 :

Cette « boucle » opérationnelle correspond au démarrage de la machine.



Les boucles de marches de production :

- ✓ **GEMMA- Marche de production à cycles répétés**

Après l'information de départ donnée par l'opérateur, les cycles se succèdent sans nouvelle intervention de celui-ci. L'arrêt doit être demandé par l'opérateur. Ceci correspond à la boucle **A1 – F1 - A2** (boucle de fonctionnement normal) : Arrêt – Fonctionnement normal – Demande d'arrêt – Arrêt – Fonctionnement normal etc.

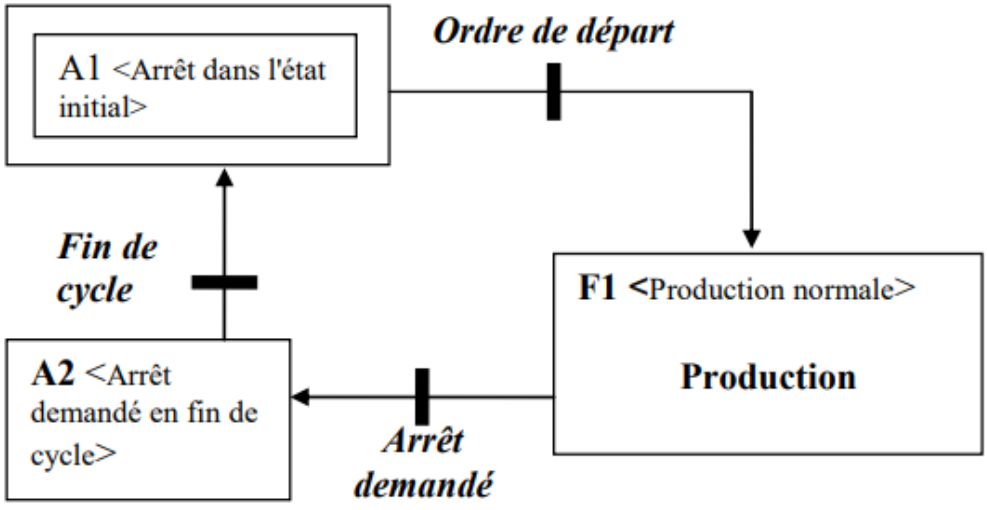


Figure 2.8 : GEMMA d'une machine avec marche de production à cycles répétés

✓ **GEMMA Marche de production cycle par cycle** :

L'information départ doit être réalisée à la fin de chaque cycle.

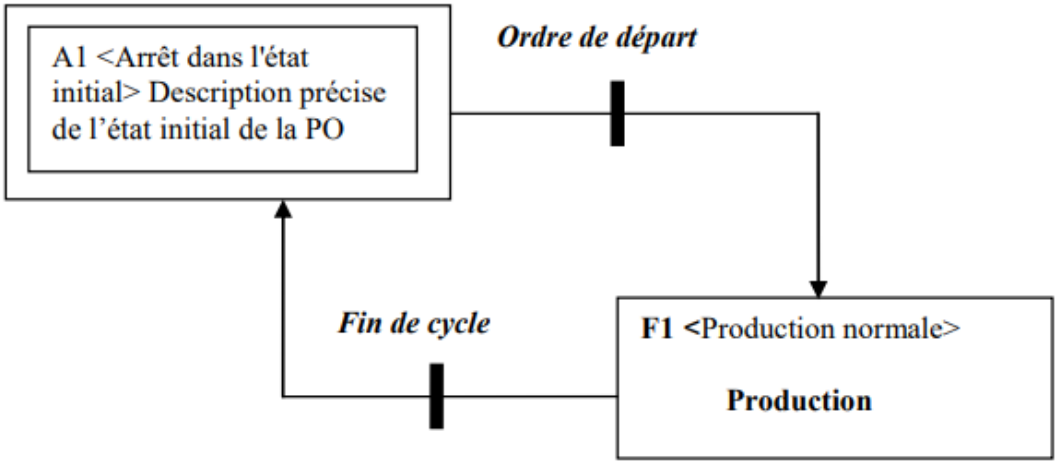


Figure 2.9 : GEMMA d'une machine avec marche de production cycle par cycle.

✓ **GEMMA d'une machine exigeant une marche de préparation et une matche de clôture :**

Si une machine nécessite une marche de clôture, par exemple pour vider un convoyeur à la fin de la journée, il faut prévoir un signal demandant l'exécution de la marche de clôture.

La machine ayant le GEMMA de la figure ci-dessous peut être arrêtée en fin de cycle pour une courte période (sans vider le convoyeur) avec le signal ACY.

Donc la condition initiale de ce GEMMA c'est que la machine soit en condition initiale, avec un convoyeur vide ou plein. Cela explique que pour la mise en route, il faut vérifier si le convoyeur est plein ou vide pour savoir si la marche de préparation est nécessaire.

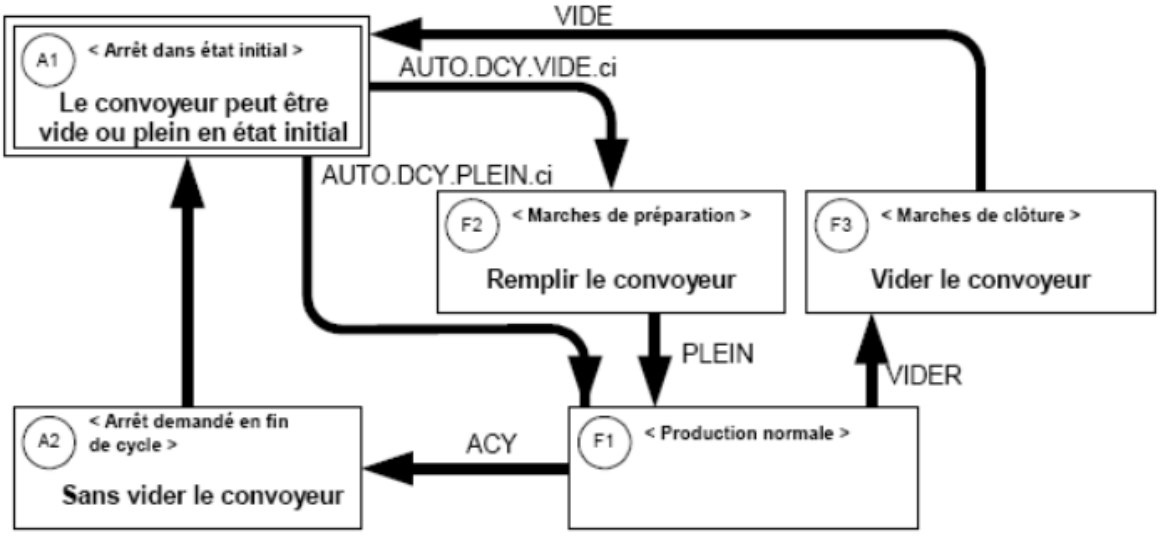


Figure 2.10 : GEMMA d'une machine avec marche de préparation et marche de clôture.

Les marches de vérification :

✓ **Dans l'ordre de cycle :**

Cette marche étape par étape a pour but de vérifier la conformité du déroulement du cycle en prenant en compte toutes les conditions réelles d'une marche de production, le déroulement s'effectuant sous le contrôle permanent de l'opérateur.

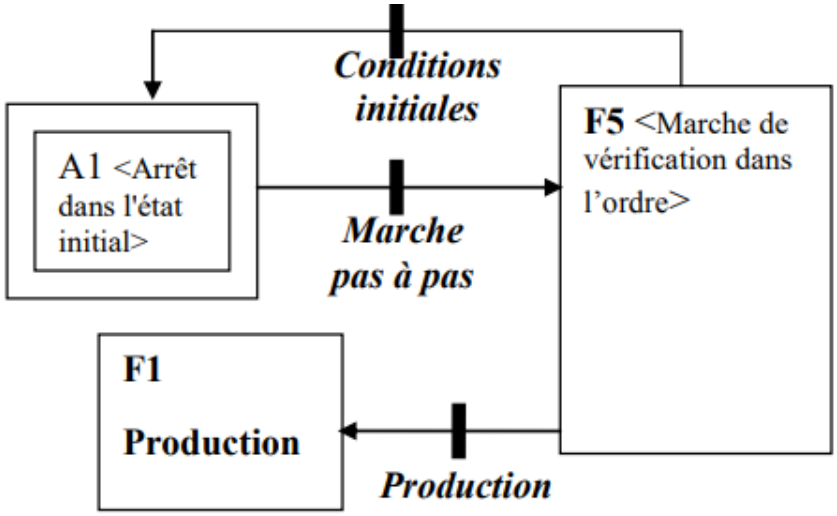


Figure 2.11 : GEMMA d'une machine avec marche de vérification dans l'ordre de cycle.

✓ **Dans le désordre :**

Dans ce mode, l'opérateur peut actionner les actionneurs dans l'ordre qu'il le désire. Par contre, il existe toujours un risque que l'opérateur ne remette pas la machine en condition initiale. Donc du **rectangle-état F4**, il faut repasser obligatoirement par le rectangle-état A6 pour que cette remise en condition initiale ait lieu.

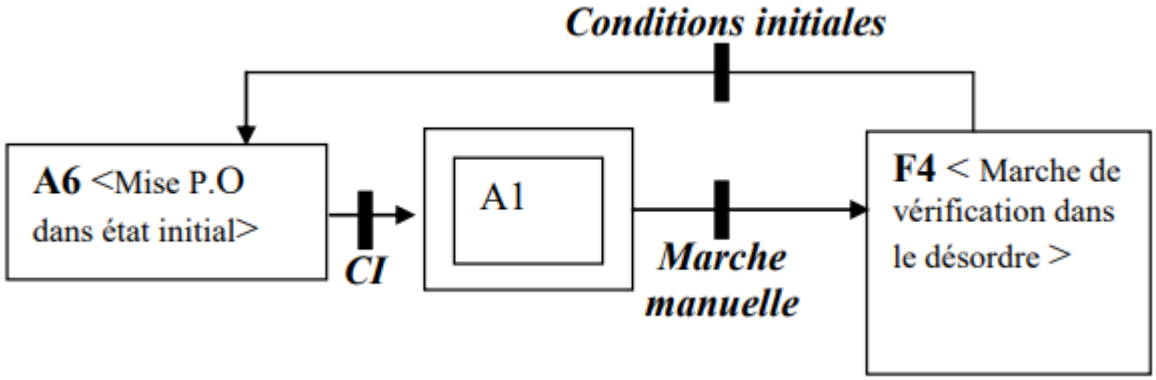


Figure 2.12 : GEMMA d'une machine avec marche de vérification dans le désordre.

<u>Les arrêts normaux :</u>

✓ **Arrêt normal en cours de cycle :**

L'étape en cours termine son action et la machine s'arrête avant l'enclenchement de l'étape suivante (arrêt en une position donné du cycle).

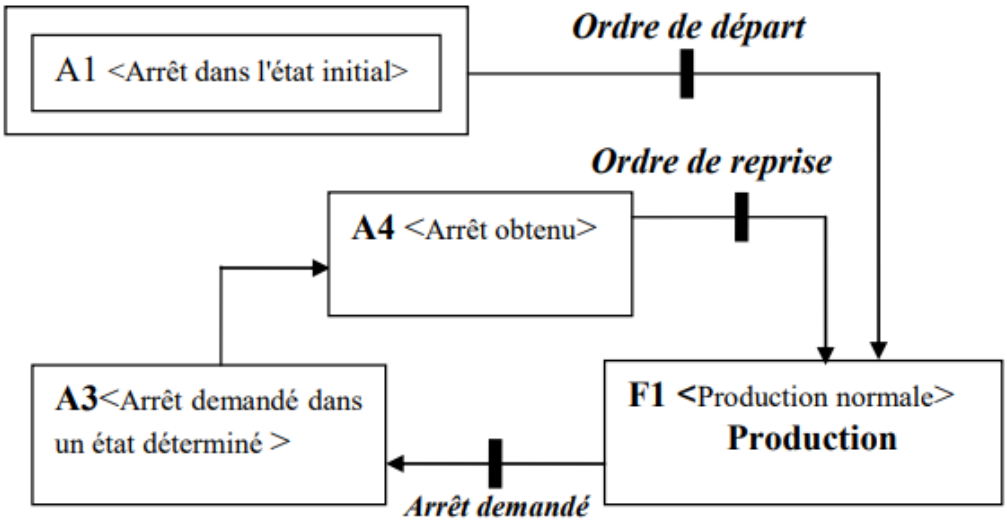


Figure 2.13 : GEMMA d'une machine avec arrêt normal en cours de cycle.

✓ **Arrêt normal en fin de cycle :**

Quel que soit le moment d'émission de la demande d'arrêt, la machine s'arrête en fin de cycle.

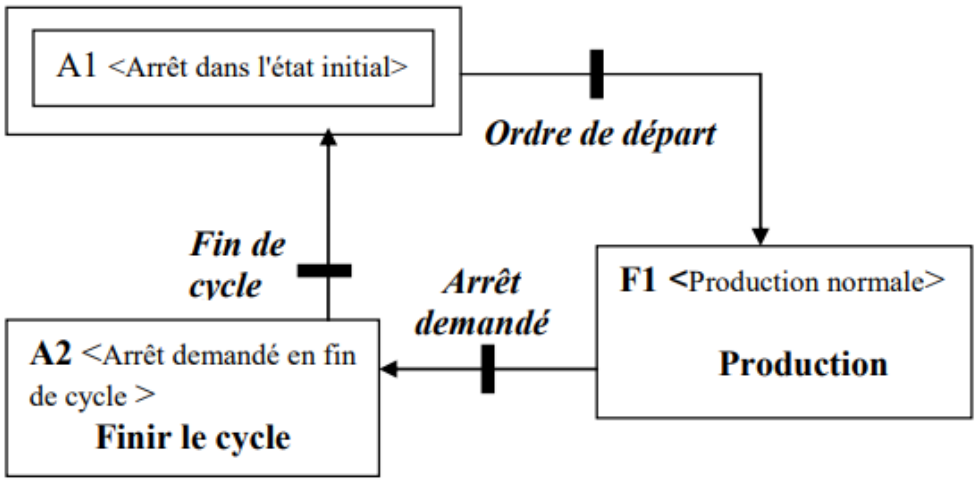


Figure 2.14 : GEMMA d'une machine avec arrêt normal en fin de cycle.

Les arrêts d'urgence :

Dans le cas où un arrêt d'urgence causé par une défaillance grave est envisagé, il faut pouvoir aller au **rectangle-état D1** lorsque cet arrêt d'urgence se produit. Et cela, quel que soit le rectangle-état ou la machine se situe.

Pour éviter d'encombrer le GEMMA, on met simplement en évidence l'évolution de F1 vers D1 et on ajoute un symbole de regroupement avec la mention «Depuis tout état». Après l'arrêt d'urgence, il faut préparer la machine à sa remise en route en la nettoyant ou en

dégageant les pièces coincées (A5), puis remettre la partie opérative en condition initiale de façon manuelle ou par une initialisation automatisée (A6). La machine sera alors prête à être redémarrée.

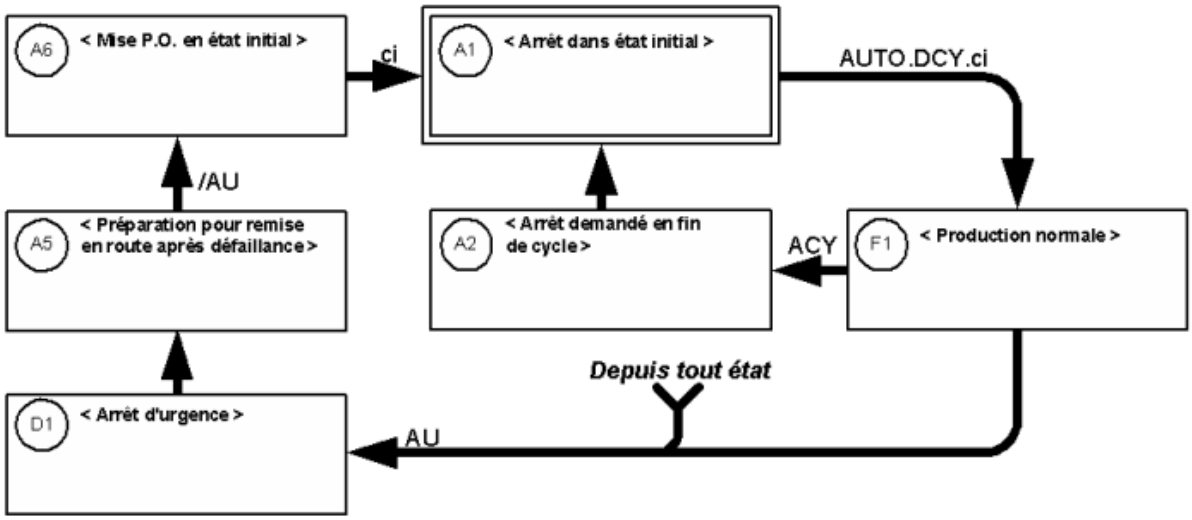


Figure 2.15 : GEMMA d'une machine avec arrêt d'urgence

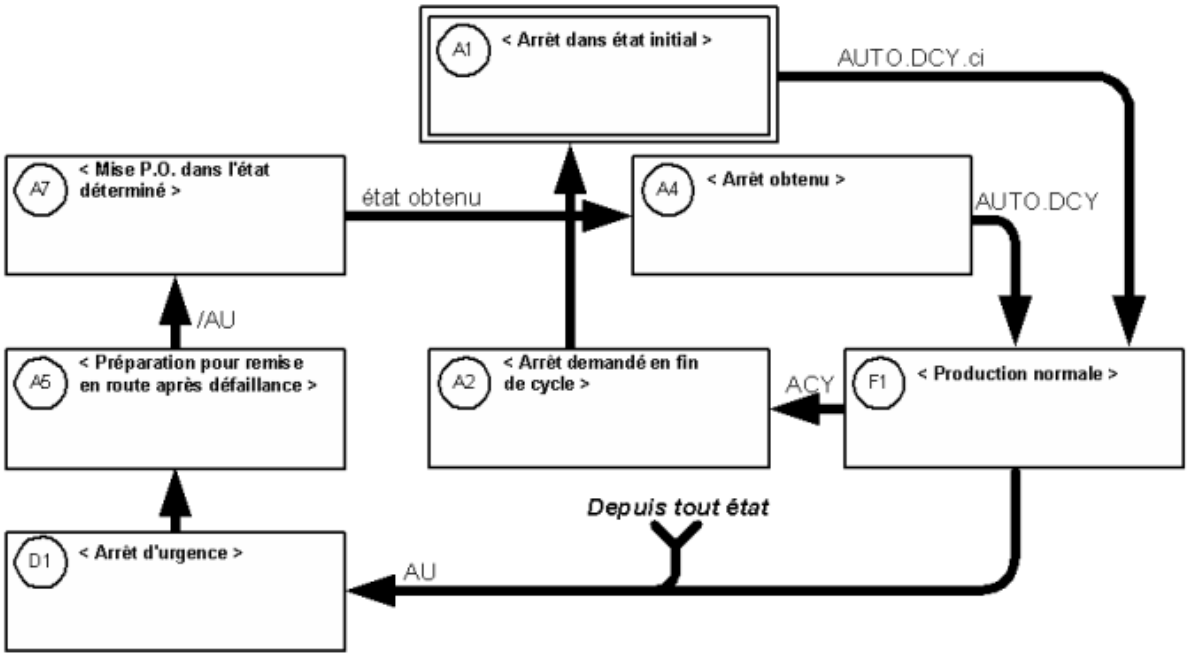


Figure 2.16 : GEMMA d'une machine avec arrêt d'urgence et remise en route à l'état ou la machine était lors de l'arrêt d'urgence.

Les redémarrages :

En fonction de la cinématique de la machine et des conséquences physiques des choix faits précédemment, le redémarrage peut être au choix :

✓ **Redémarrage à l'étape d'arrêt :**

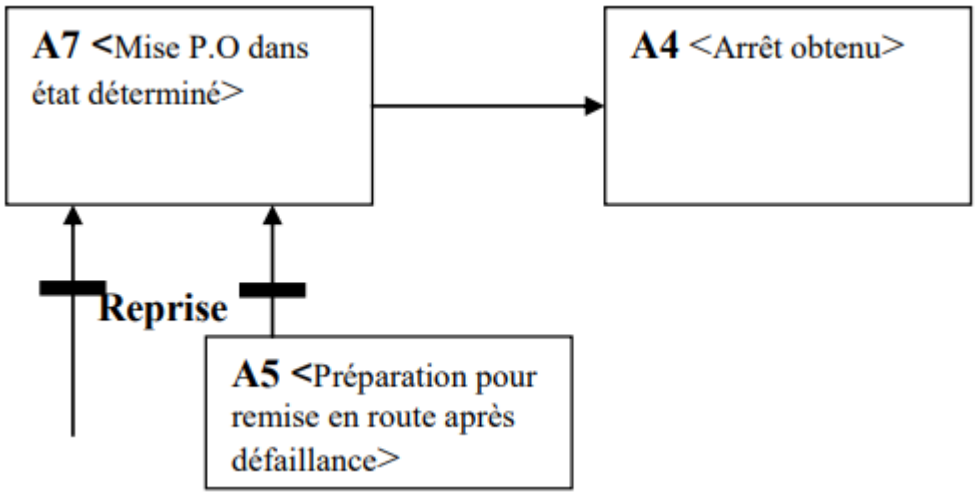


Figure 2.17 : GEMMA d'une machine avec redémarrage à l'étape d'arrêt

✓ **Redémarrage à l'étape initiale :**

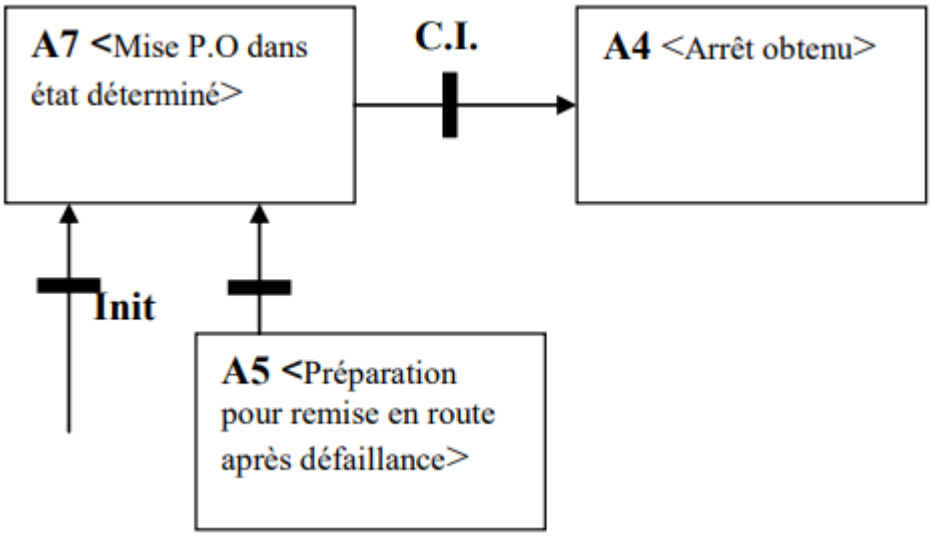


Figure 2.18 : GEMMA d'une machine avec redémarrage à l'étape initiale.

3.7 Exemple d'application : (machine à remplir et à boucher) :

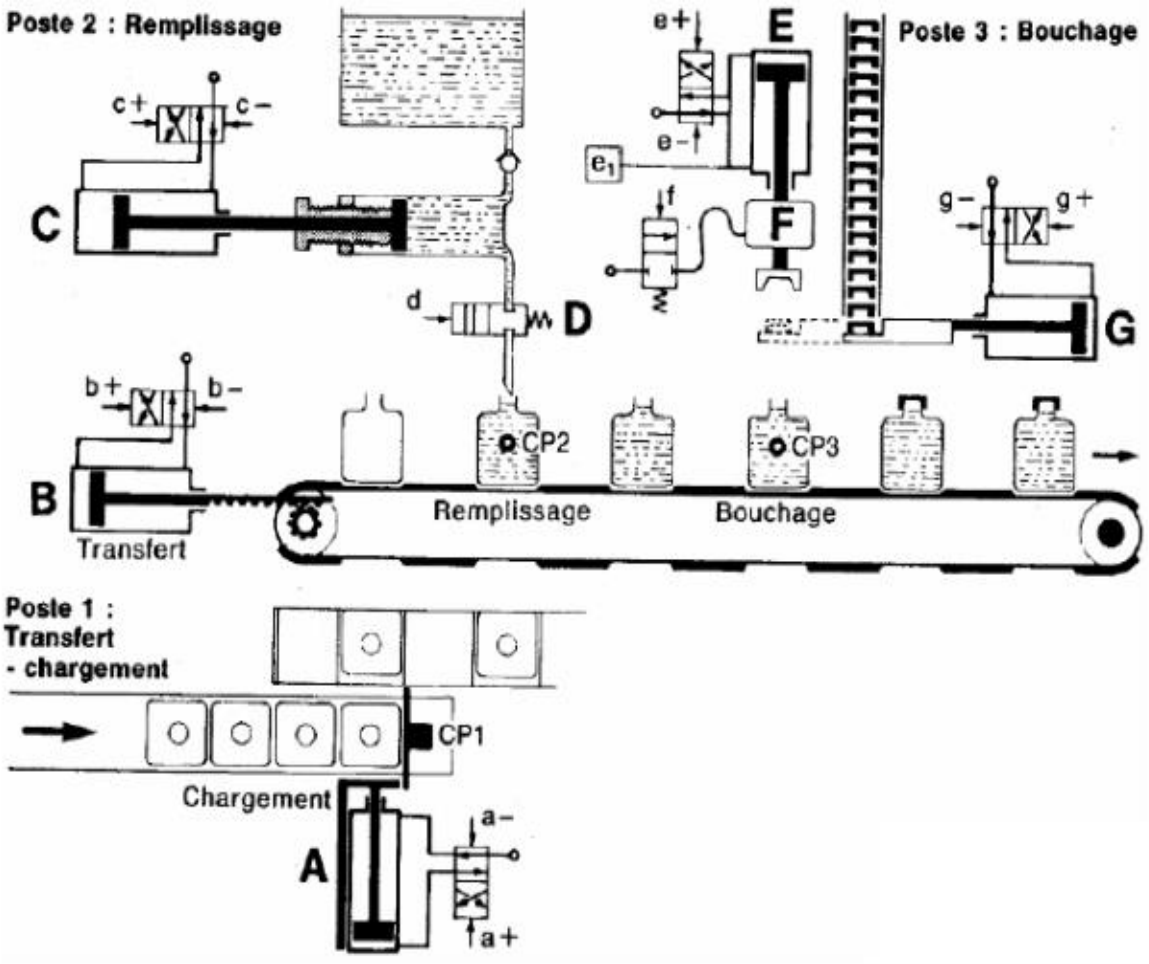


Figure 2.19 : Schéma de principe de la Machine à remplir et à boucher

La machine montrée ci-dessous est composée de trois postes.

- ✓ **Le poste 1** sert **au transfert et au chargement**. Le vérin de transfert B sert à décaler le convoyeur d'une position vers la droite. Le vérin A sert au chargement d'une nouvelle bouteille vide.

- ✓ **Le poste 2** sert au **remplissage des bouteilles**. Le vérin C actionne le piston du cylindre doseur et la vanne D sert à permettre le remplissage de la bouteille.

- ✓ **Le poste 3** est le **poste de bouchage**. Le vérin G présente un nouveau bouchon sous le dispositif de vissage composé du vérin E et du moteur F. Le vérin E et le moteur F servent au vissage du bouchon sur la bouteille. Les figures 1.41 et 1.42 représentent respectivement, le grafcet de niveau 1 et le grafcet de niveau 2 du système.

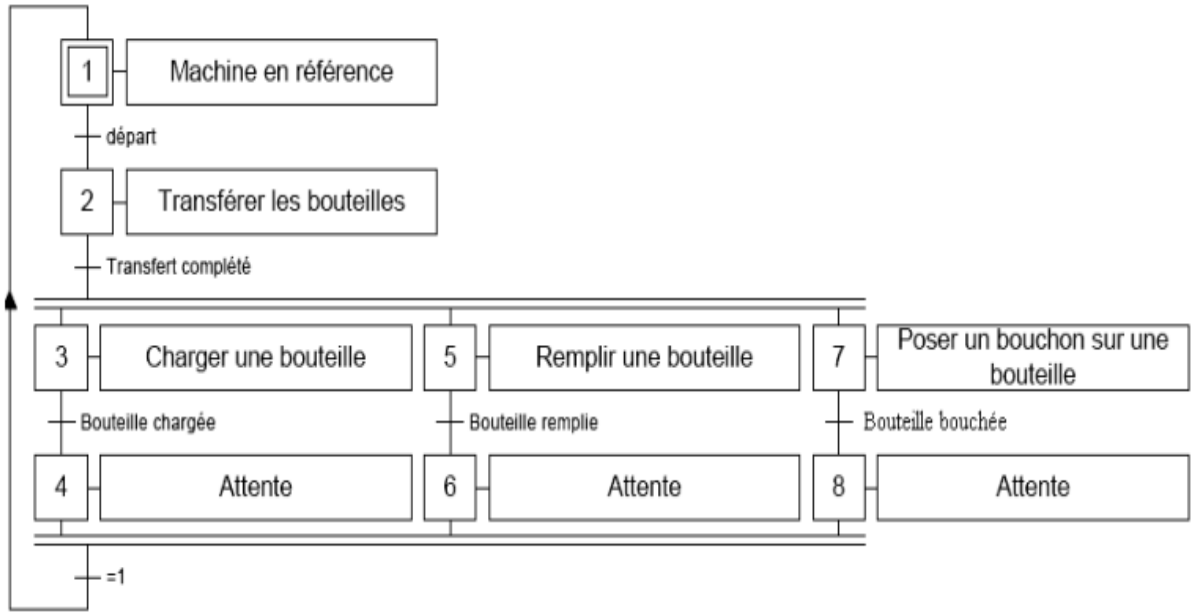


Figure 2.20 : (Grafcet niveau 1) de la Machine à remplir et à boucher

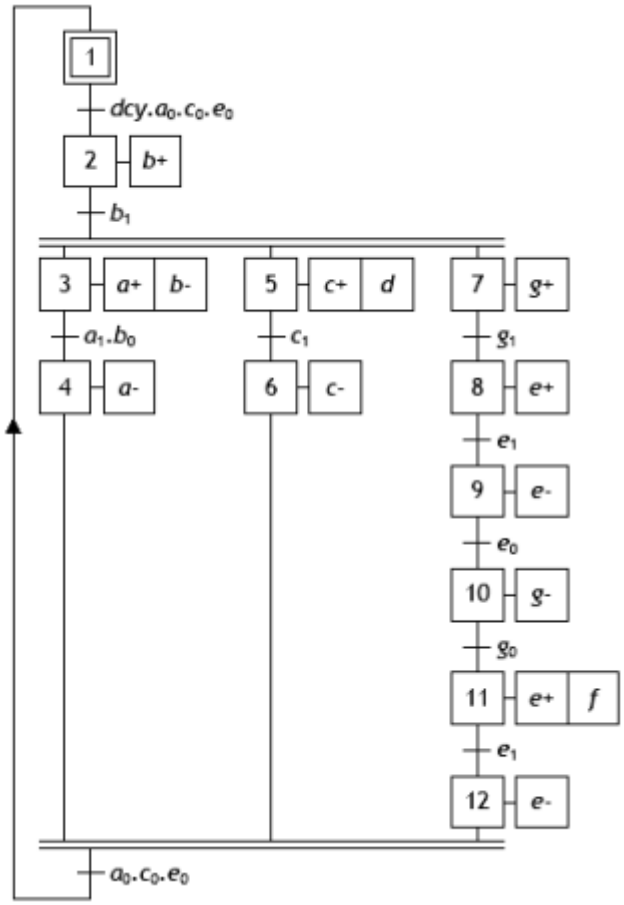


Figure 2.21 : (Grafcet niveau 2) de la Machine à remplir et à boucher

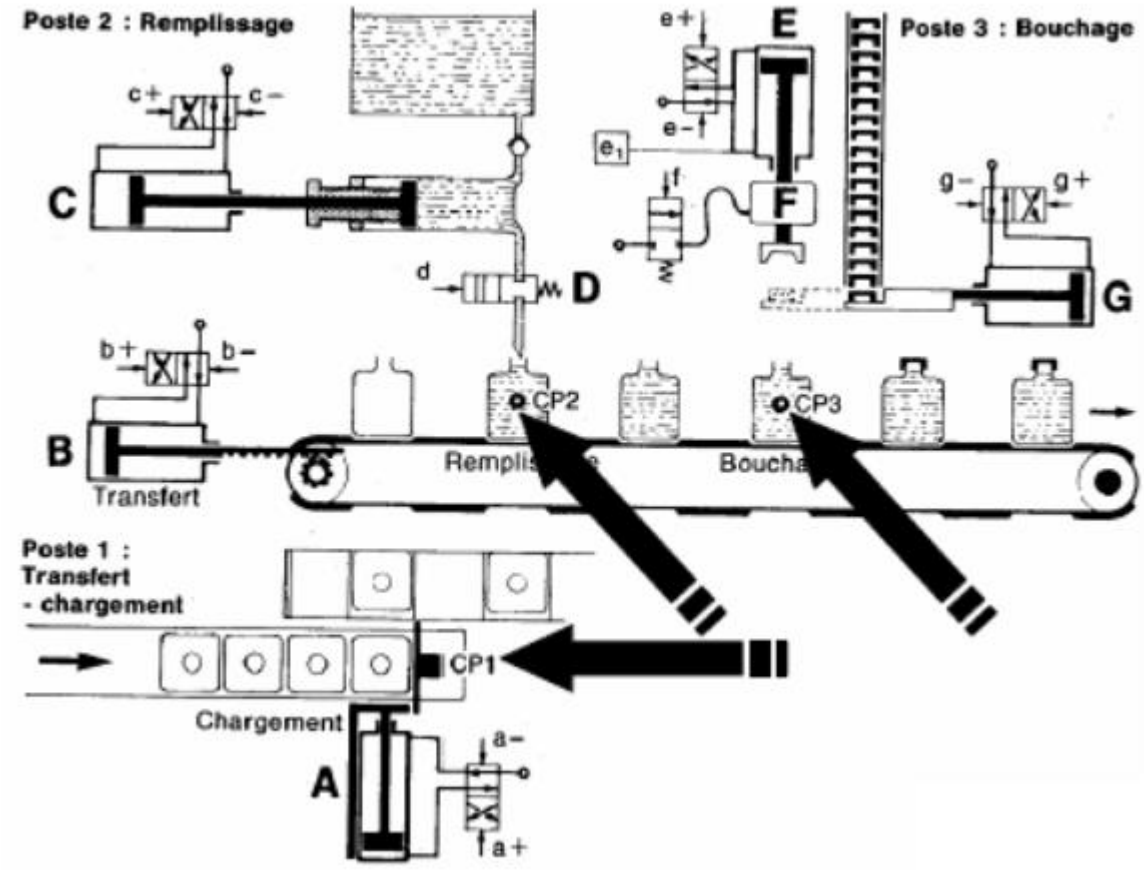
Le GRAFCET de niveau 2 assume :

- qu'il ne manque jamais de bouteilles ;
- Si une bouteille manque, on vide le liquide sur le tapis !!!
- qu'il n'y a aucun arrêt d'urgence ;
- qu'il n'y a aucune défaillance.

Question1 : Peut-on partir la machine sans bouteilles avec le GRAFCET actuel ?

Réponse 1: NON, donc marche de préparation nécessaire (**Remplir le rectangle F2**)

La marche de préparation fera en sorte de démarrer la machine de façon progressive.
➔ Il faut donc ajout des capteurs CP1, CP2 et CP3 (capteurs de présence de bouteilles aux postes 1, 2 et 3).



Question 2 : Peut-on arrêter la machine et faire en sorte qu'il n'y ait plus de bouteilles avec le GRAFCET actuel ?

Réponse 2: NON, donc marche de clôture nécessaire ➔ Donc il faut ajouter le rectangle état F3 et barrer le rectangle état A2 (car avant d'arrêter la machine il faut passer par F3).

La marche de clôture fera en sorte de vider la machine avant l'arrêt complet.
Impossible d'avoir un convoyeur sans bouteilles avec un arrêt en fin de cycle du GRAFCET.

Question 3 : Supposons que la machine exige 5 minutes pour se mettre en opération et une autre 5 minute pour s'arrêter. Que faire si l'opérateur doit prendre une pause de 5 minutes ? Peut-on se permettre de perdre 15 min ?

Réponse 3 : il faut prévoir un arrêt de courte durée, sans vider le convoyeur. Pour ce faire, il faut terminer le cycle en cours sans vider le tapis (ajout A3) et puis arrêter la machine de courte durée (Ajouter le rectangle état A4).

Question 4 : que doit faire l'opérateur suite à un arrêt d'urgence ?

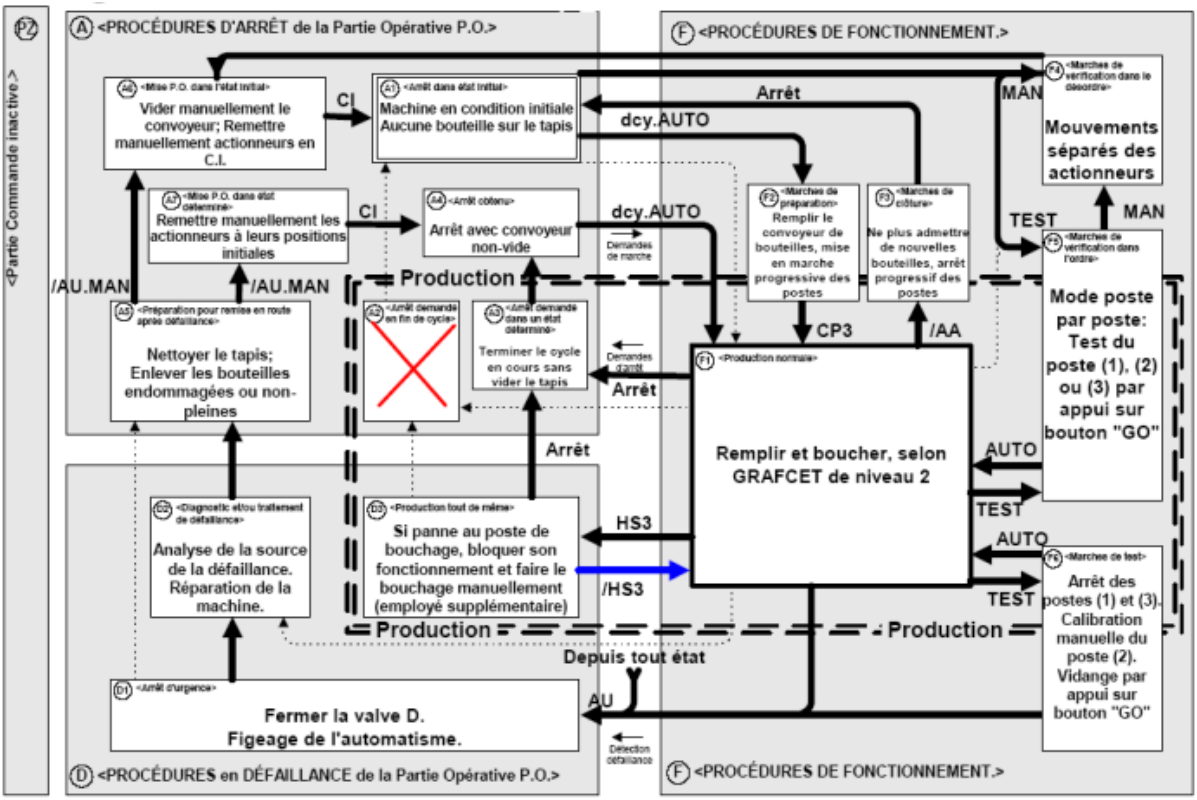
Réponse 4 :L'opérateur devrait, suite à un arrêt d'urgence, préparer la machine en nettoyant le tapis et en enlevant les bouteilles endommagées ou non pleines.

- Si la défaillance était grave, répartir la production du début en enlevant manuellement les bouteilles du convoyeur et en remettant les actionneurs en conditions initiales de façon manuelle.
- Si la défaillance était sans gravité, répartir la production du début du cycle, sans retirer les bouteilles présentes et en remettant les actionneurs en conditions initiales de façon manuelle.
- Si la défaillance se produit au poste de bouchage, bloquer le fonctionnement de ce poste et faire le bouchage manuellement.

Après l'analyse des différents modes de marches et d'arrêts, il faut définir les réceptivités de transition entre chaque mode.

- Démarrage en mode automatique,
- Si une bouteille se présente au poste 3, la machine est alors complètement remplie de bouteilles,
- Quand l'opérateurconstate qu'il ne reste plus de bouteilles sur le tapis, il arrête la machine,
- Pour un arrêt complet, interdire l'admission de nouvelles bouteilles sur le tapis,

Le GEMMA qui décrit les modes de fonctionnement du système est donne par la figure suivante :



3.8 Passage du GEMMA à une spécification GRAFCET :

Le passage du GEMMA vers le Grafcets conduit à une structure multi-grafcets hiérarchisées en utilisant les moyens de structuration définis dans la norme IEC 60848 (Macro-étape, Grafcet partiel, Structuration par forçage, Structuration par encapsulation) :

- ✓ **GRAFCET DE CONDUITE (GC) : Grafcet gérant tous les modes marches et d'arrêts normaux**. IL peut faire appel à des grafcets spécifiques (initialisation, préparation, clôture...). Le dialogue entre les grafcets est réalisé généralement par les variables internes d'étapes Xi.

- ✓ **GRAFCET DE PRODUCTION NORMALE (GPN) :** Grafcet **décrivant le fonctionnement normal de production du système**. Il peut faire appel à des grafcets de tâche ou de procédure. Le dialogue entre les grafcets est réalisé par les variables Xi.

- ✓ **GRAFCET DE SURETE (GS) :** Grafcet **gérant les procédures de défaillance**. Il est Hiérarchiquement supérieur à tous les autres grafcets. Cette hiérarchie est réalisée par des ordres de Forçage.

Exemple 1

Elaboration du Grafcet de conduite d'un système à partir du GEMMA :

Le GEMMA de fonctionnement d'un système est donné par la figure suivante :

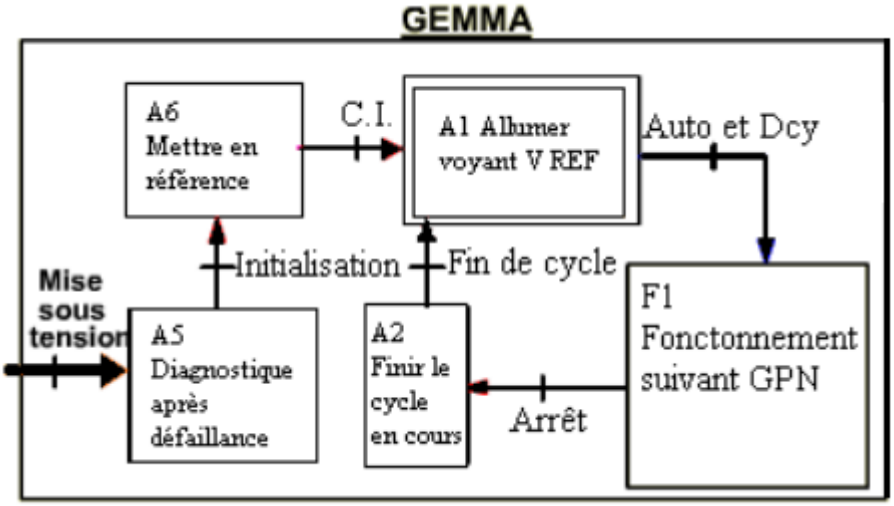


Figure 2.22 : Exemple de GEMMA d'un système.

Le GRAFCET de conduite correspondant est donné par la figure suivante :

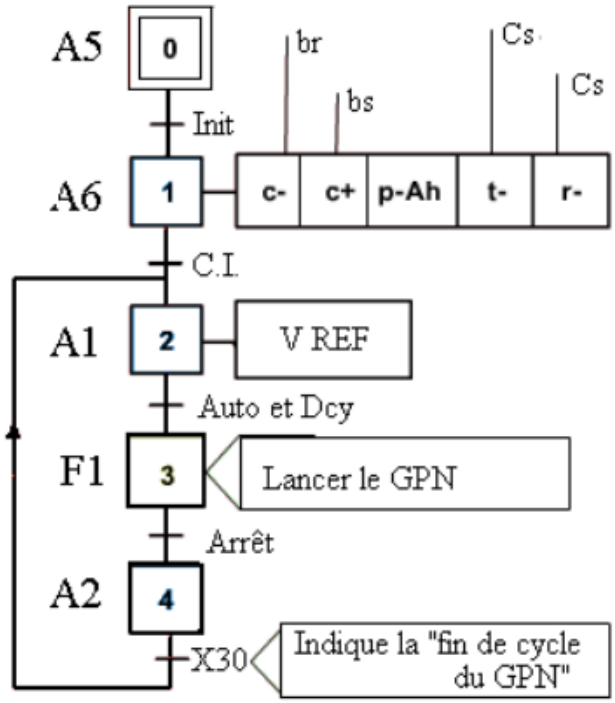
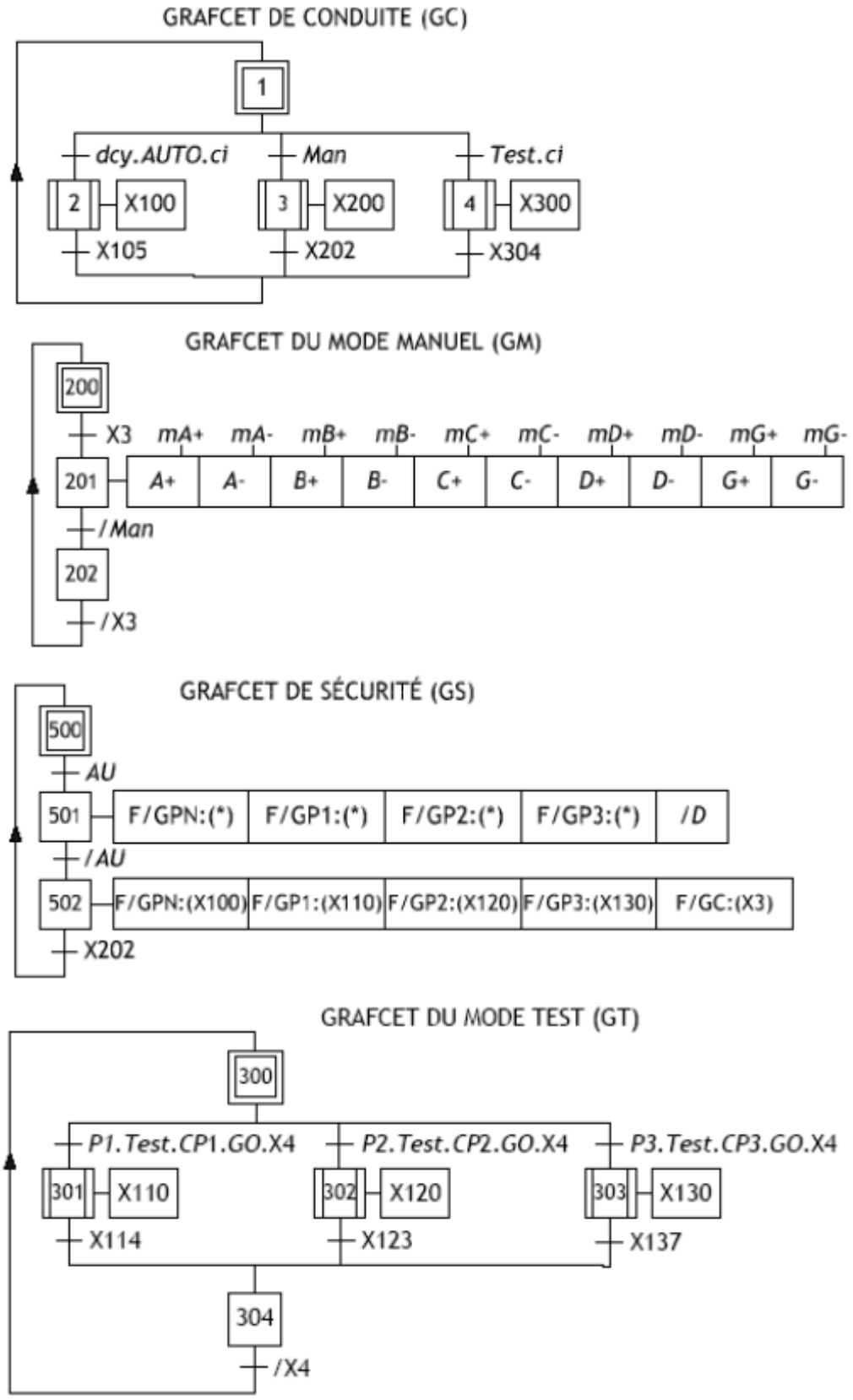
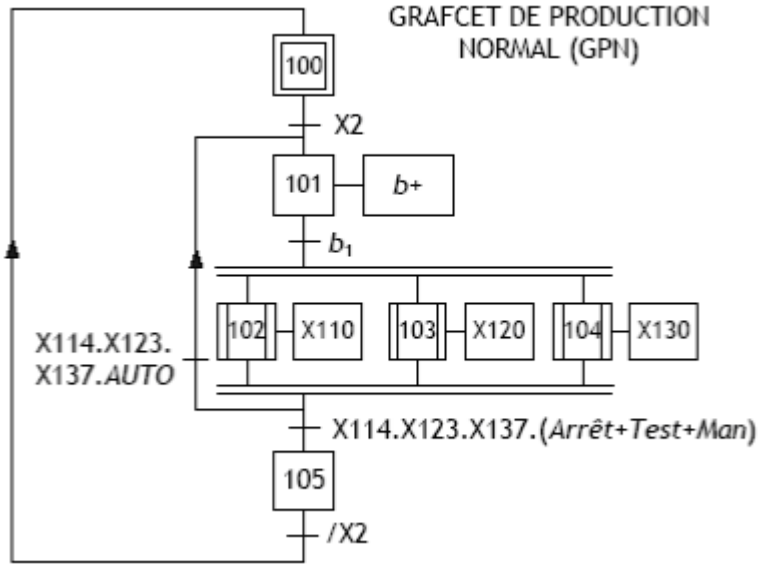


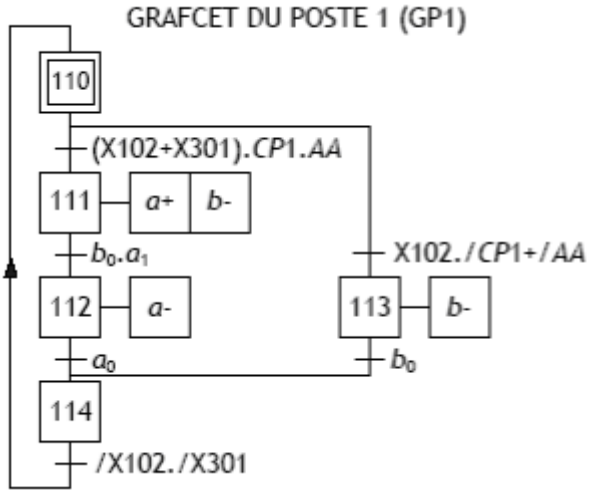
Figure 2.23 : GRAFCET de conduite déduit du GEMMA de la figure précédente.

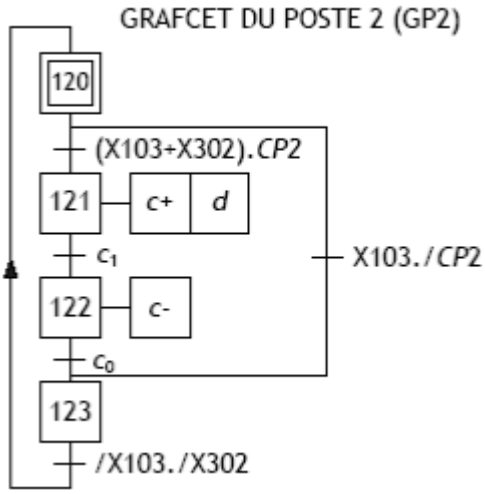
Exemple 2

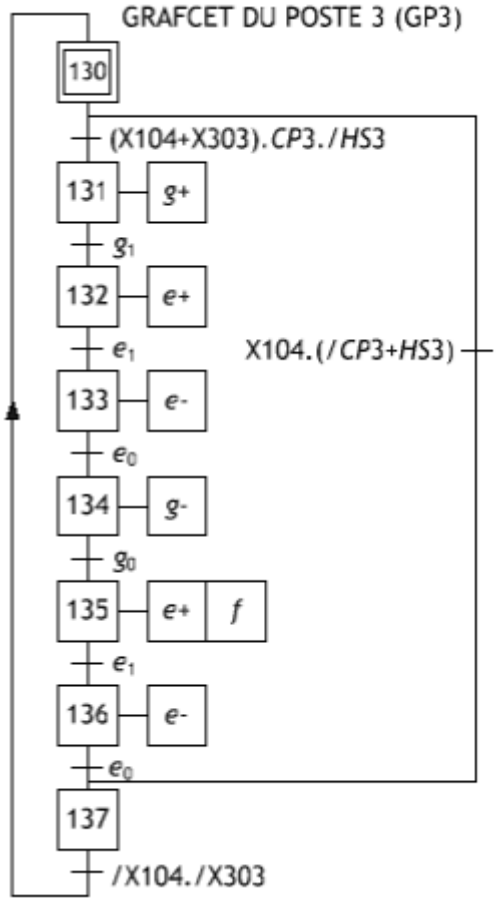
Elaboration des Grafcets de conduite du système Machine à remplir et à boucher




GRAFCET DE PRODUCTION NORMAL (GPN)
100
X2
101
b+
b1
102
X110
103
X120
104
X130
X114.X123.
X137.AUTO
X114.X123.X137.(Arrêt+Test+Man)
105
/X2


GRAFCET DU POSTE 1 (GP1)
110
(X102+X301).CP1.AA
111
a+
b-
b0.a1
X102./CP1+/AA
112
a-
113
b-
a0
b0
114
/X102./X301


GRAFCET DU POSTE 2 (GP2)
120
(X103+X302).CP2
121
c+
d
c1
X103./CP2
122
c-
c0
123
/X103./X302



3.9 Implémentation du programme dans l'API

La Partie Opérative du système, les grafcets de Production Normale, le Dialogue, le GEMMA (Modes de Marches et d'Arrêts), les GRAFCET de Sécurité et de Conduite étant définis, il reste à définir la Partie Commande.

Si le choix se porte sur un automate programmable, celui-ci étant relié aux pré actionneurs (affectation Entrées/ Sorties) et ayant son propre langage de programmation, il faut traduire les GRAFCET précédents en un programme.

Tracer les GRAFCET adaptés à l'automate programmable.	Remplacer les réceptivités et les actions par les affectations des variables d'Entrées/Sorties Modifier les structures GRAFCET si nécessaire en fonction des possibilités du langage de programmation. Préparer la programmation pour les temporisations, les compteurs, les mémorisations d'action etc.. en respectant la syntaxe du langage de programmation.

Ecrire les équations de sorties	Recherche des conditions d'exécution des actions dans l'ensemble des grafcets et des équations logiques
Noter l'état initial des variables	Etapes actives au démarrage, mots de données pour tempo ou compteur)
Ecrire le programme.	Il existe 2 possibilités d'édition de Programme: Ecrire le programme directement dans le langage programmable sur feuille de programmation. Ecriture de l'ossature GRAFCET et des réceptivités, puis des équations de sorties. Utiliser un logiciel d'assistance à la Programmation (en général GRAPHIQUE) exemple AUTOMGEN

REMARQUE: Le logiciel AUTOMGEN permet l'édition graphique proche des grafcets, puis l'affectation des entrées/sorties, la génération du programme pour l'automate concerné, la simulation du programme, le transfert et la supervision de son exécution.

a) Terminaux de programmation et de réglage

L'API doit permettre un dialogue avec :

- ✓ Le personnel d'étude et de réalisation pour réaliser la première mise en œuvre (Edition programme, Transfert, Sauvegarde...)
- ✓ Le personnel de mise au point et de maintenance de réaliser des opérations sur le système (Forçage, Visualisation de l'état, Modification de paramètres temporisation, compteurs.)

Ce dialogue peut être réalisé par :

- ✓ **Une Console** : Elle sera utilisée sur site. Elle comporte un clavier, un écran de visualisation et le langage de programmation.
- ✓ **Un Micro-ordinateur avec un logiciel d'assistance à la programmation** : Il sera utilisé hors site. Il comprend plusieurs modules pour permettre l'édition, l'archivage, la mise au point des applications.

b) Transfert du programme dans l'automate programmable

Le transfert du programme peut être fait soit :

- ✓ manuellement en entrant le programme et l'état initial à l'aide d'une console de programmation

- ✓ Automatiquement en transférant le programme à l'aide du logiciel d'assistance, et en réalisant la liaison série entre l'ordinateur et l'automate.

c) Vérification du fonctionnement

Lors de sa première mise en œuvre il faut réaliser la mise au point du système.

Prendre connaissance du système (dossier technique, des grafcets et du GEMMA, affectation des entrées / sorties, les schémas de commande et de puissance des entrées et des sorties).

Lancer l'exécution du programme (RUN ou MARCHE), visualiser l'état des GRAFCET, des variables...

Il existe deux façons de vérifier le fonctionnement :

- ✓ En simulation (sans Partie Opérative).
- ✓ En condition réelle (avec Partie Opérative).

Simulation sans P.O.	**Condition réelle**
Le fonctionnement sera vérifié en simulant le comportement de la Partie Opérative, c'est à dire l'état des capteurs, en validant uniquement des entrées. ✓ Valider les entrées correspondant à l'état initial (position) de la Partie Opérative. ✓ Valider les entrées correspondant aux conditions de marche du cycle. ✓ Vérifier l'évolution des grafcets (étapes actives). ✓ Vérifier les ordres émis (Leds de sorties). ✓ Modifier l'état des entrées en fonction des ordres émis (état transitoire de la P.O.). ✓ Modifier l'état des entrées en fonction des ordres émis (état final de la P.O.). Toutes les évolutions du GEMMA et des grafcets doivent être vérifiées.	Le fonctionnement sera vérifié en suivant le comportement de la P.O. ✓ Positionner la P.O. dans sa position initiale. ✓ Valider les conditions de marche du cycle. ✓ Vérifier l'évolution des grafcets et le comportement de la P.O. Toutes les évolutions du GEMMA et des grafcets doivent être vérifiées.

Chapitre III

PROGRAMMATION D'UN API (MISE EN ŒUVRE DE L'API S7-300)

1 Introduction

Les automates SIMATIC S7-300 sont des API modulaires destinés à des applications industrielles plus ou moins complexes. Ils peuvent être couplés avec des coupleurs réseaux, des cartes d'entrées/sorties logiques et analogiques, des cartes spéciales pour le positionnement, la régulation, le comptage etc…

La série S7-300 possède une vingtaine de CPU standards avec des spécificités différentes allant de versions compactes à des versions modulaires.

Les automates S7-300 se programment avec le logiciel STEP 7. Dans ce chapitre on s'intéresse aux API S7-300 modulaires.

2 Architecture matérielle du S7-300

L'API S7-300 modulaire, comporte des modules d'alimentation (**PS**), une unité centrale (**UC**), des modules d'entrées/sorties (**SM**), des modules de fonctions spéciales (**FM**), des modules de communication pour les liaisons réseau (**CP**) qui sont placés tous sur un ou plusieurs racks.

La figure 3- 1 montre un exemple de la disposition des modules sur un châssis en configuration avec 8 modules de signaux.

Emplacement 1: Module d'alimentation (PS)
Emplacement 2 : CPU,
Emplacement 3 : coupleur d'extension (IM)
Emplacements 4→ 11 : modules de signaux (SM)

1	2	3	4	5	6	7	8	9	10	11
PS	CPU	IM	SM 1	SM 2	SM 3	SM 4	SM 5	SM 6	SM 7	SM 8

Figure 3- 1 : Exemple de disposition des modules sur un châssis d'un API S7-300

La disposition des modules sur plusieurs châssis doit respecter les critères suivants:

- ✓ Le coupleur d'extension occupe toujours l'emplacement 3. Il se situe toujours à gauche du premier module de signaux,
- ✓ Chaque unité (profilé-support) peut recevoir au maximum 8 modules (SM, FM, CP),

- ✓ Le nombre des modules enfichés (SM, FM, CP) est limité par la consommation de courant autorisée sur le bus de fond de panier S7-300. La consommation totale ne doit pas dépasser 1,2 A par unité (avec la CPU 312 IFM : 0,8 A).

Les modules traditionnels de la famille SM :

DI : Entrées logiques
DO : Sorties logiques
AI : Entrées analogiques
AO : Sorties analogiques

Les modules spéciaux :

CP : Modules de communications (Point à point, ASI, PROFIBUS, ETHERNET),
FM : Modules spéciaux (comptage, positionnement, régulation PID),

La figure 3- 2 présente la disposition des modules dans un montage S7-300 sur 4 châssis.

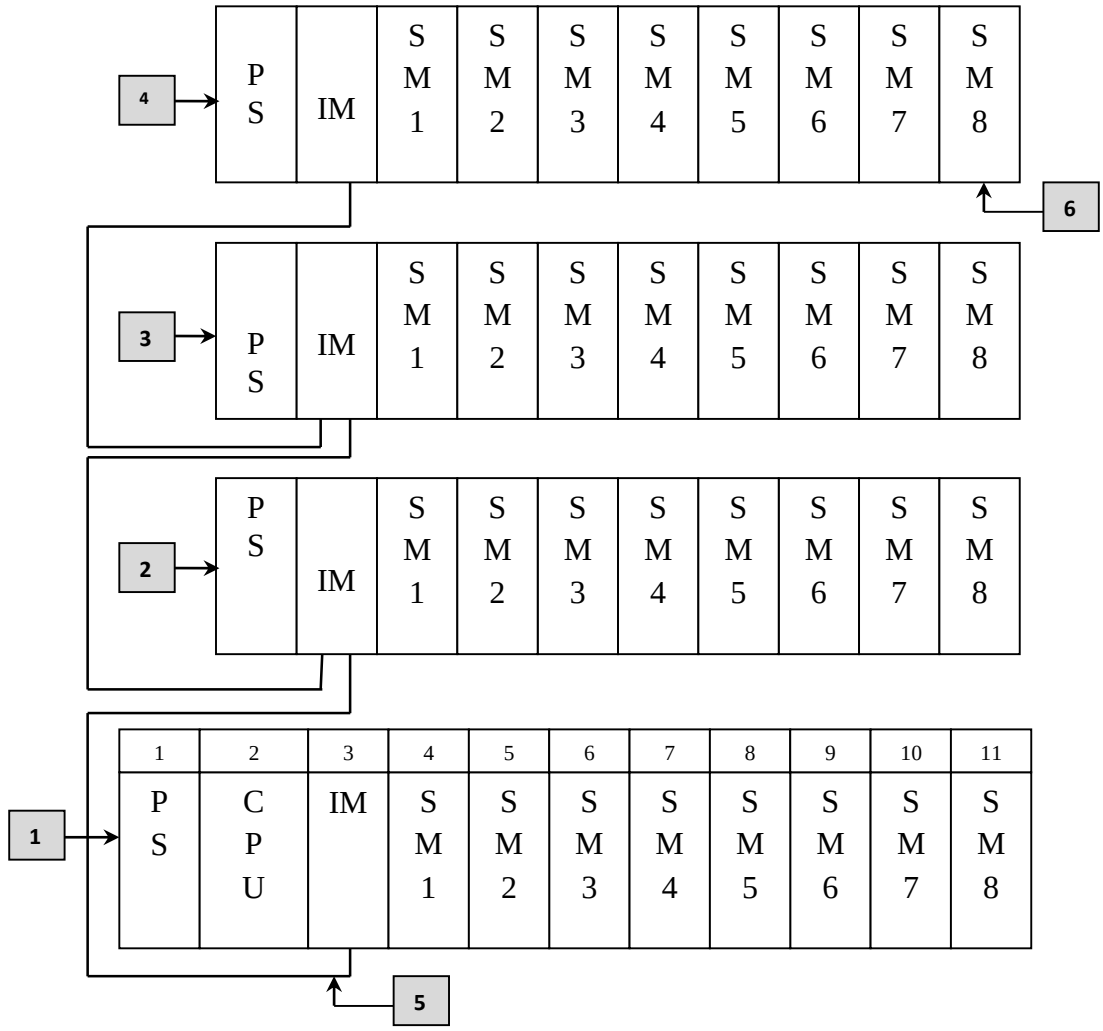


Figure 3- 2 : Disposition des modules dans un montage sur 4 châssis

(1) le châssis 0 (châssis de base)
(2) le châssis 1 (châssis d'extension)
(3) le châssis 2 (châssis d'extension)
(4) le châssis 3 (châssis d'extension)
(5) le câble de liaison 368
(6) les restrictions pour la CPU 314 IFM. Ne pas enficher le module de signaux 8 dans le châssis 4.

Les fonctions des différents modules spéciales enfichables sur les châssis de l'API S7-300 sont données par le tableau suivant :

Module	Fonction
CP340	Communication point à point
CP341	
CP 342-2 CP343-2	Communication ASI
CP 342-5	Communication PROFIBUS
CP 342-5FO CP 345-5	
CP 342-5	
CP 343-1	Communication ETHERNET
FM 350-1	Module de comptage
FM 350-2	
FM 351	Module de positionnement
FM 353	
FM 354	
FM 355 C	Module de régulation
FM 355 S	
FM 352-2S	
FM 352-2C	
IM 360	Coupleurs
IM 361	
IM 365	

Tableau 3-1 : Fonction des modules enfichables sur les châssis de l'API S7-300

3 Adressage des voies des modules

Les possibilités d'adressage des différentes voies des modules sont :

3-1- Adressage axé sur les emplacements

L'adressage axé sur les emplacements des modules est l'adressage par défaut. En effet, *STEP 7* attribue à chaque numéro d'emplacement une adresse initiale définie pour les modules selon leurs types (numérique ou analogique).

La figure 3- 3 présente le montage d'un API S7-300 sur 4 châssis et les emplacements possibles des modules avec leurs adresses initiales correspondantes.

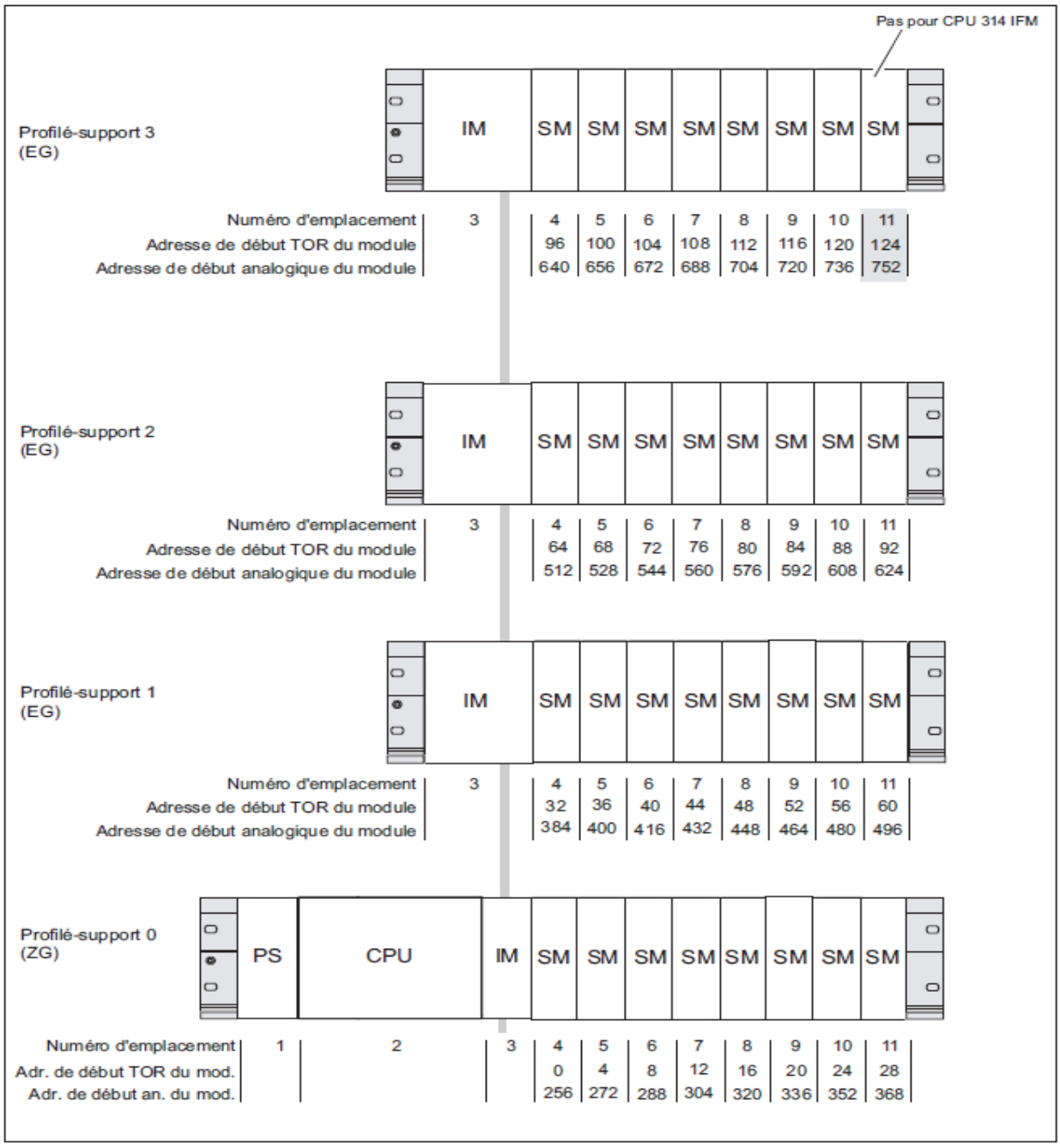


Figure 3- 3 : Emplacements du S7-300 et adresses initiales des modules correspondantes

3-2- Adressage libre

L'adressage libre signifie qu'on peut affecter à chaque module (SM/FM/CP) une adresse choisie par l'utilisateur à l'intérieur de la plage d'adresses gérée par la CPU. Cet adressage sera réalisé dans STEP 7. On définit ainsi l'adresse initiale des modules sur laquelle sont basées toutes les autres adresses du module.

L'adressage libre n'est possible que pour des API S7-300 comportant les CPU 315, 315-2 DP, 316-2 DP et 318-2 DP.

3-3-Adressage des modules de signaux (SM)

Une entrée ou une sortie est désignée dans le programme à l'aide d'une adresse qui indique clairement quel est son adresse sur l'automate. Pour les modules d'**entrées/sorties logiques**, cette adresse est composée :

- d'une lettre qui indique la nature de la variable
  - **I** (ou **E**) pour une entrée
  - **Q** (ou **A**) pour une sortie
- d'un chiffre, appelé **adresse d'octet**, qui indique l'emplacement du module sur le châssis.
- d'un point (**.**)
- d'un chiffre, appelé **adresse du bit** qui indique l'emplacement de la variable sur le module (de 0 à 7).

La figure 3- 4, donne un exemple d'adressage des modules TOR.

1	2	3	4	5	6	7	8	9	10	11
PS	CPU	IM	X0.0 X3.7	X4.0 X7.7	X8.0 X11.7	X12.0 X15.7	X16.0 X19.7	X20.0 X23.7	X24.0 E27.7	X28.0 X31.7

Entrée : X = E ou I

		4	5	6	7	8	9	10	11
PS	IM	X32.0 X35.7	X36.0 X39.7	X40.0 X43.7	X44.0 X47.7	X48.0 X51.7	X52.0 X55.7	X56.0 X59.7	X60.0 X63.7

Figure 3- 4 : Exemple d'dressage des modules d'entrées/sorties TOR

Pour les modules d'**entrées/sorties analogiques**, l'adresse d'une voie d'entrée ou de sortie analogique est toujours une adresse de mots. Cette adresse est composée :

- d'une combinaison de lettres qui indique la nature de la variable
  - **PIW** (ou **PEW**) pour les voies des modules d'entrées
  - **PQW** (ou **PAW**) pour les voies des modules de sorties
- d'une combinaison de chiffres qui indique l'adresse de la voie sur le châssis qui est basée sur l'adresse initiale des modules. L'adresse de chaque voie du module considéré occupe deux octets. L'adresse initiale de chaque module analogique suivant augmente de 16 par emplacement.

Si le premier module analogique se trouve sur l'emplacement **25** du troisième châssis par exemple, son adresse initiale par défaut est **PEW592** ou **PAW592**, l'adresse initiale de l'emplacement suivant sera **PEW608** ou **PAW608.** Les adresses des 8 voies d'entrées d'un module analogique placé à l'emplacement 27 du troisième châssis par exemple, seront de **PEW624** à **PEW 638**. Les adresses des 8 voies de sorties d'un module analogique placé à l'emplacement 35 du quatrième châssis par exemple, seront de **PAW752** à **PEW 766**.

La figure 3-5 donne un exemple d'adressage des modules d'entrées/sorties analogiques.

1	2	3	4	5	6	7	8	9	10	11
PS	CP	IM	PXW 256	PXW 272	PXW 288	PXW 304	PXW 320	PXW 336	PXW 352	PXW 368

Entrée : X = E ou I

4	5	6	7	8	9	10	11
PS IM	PXW 384	PXW 400	PXW 416	PXW 432	PXW 448	PXW 464	PXW 480	PXW 496

Figure 2- 5 : Exemple d'dressage des modules d'entrées/sorties analogiques

Remarque :

Dans la configuration de l'automate, réalisée à l'aide de l'outil **Hardware Config**, il faut inclure les différents modules et leurs paramètres (type de transmetteur, plage de variation du signal,…).

4 Les variables internes de l'automate S7-300

4-1- Types de variables

L'automate S7-300 dispose de différentes zones de mémoires qui sont définies comme suit :

Zone E (ou **I):** Mémoire image des entrées sur bus locale ou bus de terrain tel que PROFIBUS,
Zone A (ou **Q) :** Mémoire image des sorties sur bus locale ou bus de terrain tel que PROFIBUS,
Zone M : Mémoire utilisateur,
Zone L : Mémoire locale, associée à un module de programme,
Zone P : Accès à la périphérie,
Zone T : Mémoire des temporisations. Deux variables sont associées à chaque temporisation :

- La valeur en cours qui contient la durée comptabilisée par la temporisation.
- Le bit de temporisation qui est mis à 1 ou à 0 selon le résultat de comparaison entre la valeur en cours et la valeur prédéfinie.

Zone Z (ou **C):** Mémoire des compteurs,
Zone DB : Mémoire utilisateur ou système structuré dans des blocs de données.

Les symboles des différentes variables sont décrits dans le tableau suivant:

Définition	Symbole CEI	Symbole SIMATIC
Bits d'entrée	I	E
Octet d'entrée	IB	EB
Mot d'entrée	IW	EW
Double mot d'entrée	ID	ED

Bits de sortie Octet de sortie Mot de sortie Double mot de sortie	Q QB QW QD	A AB AW AD
Mémoires utilisateurs Octet mémoire Mot mémoire Double mot mémoire	M MB MW MD	M MB MW MD
Bit dans la mémoire locale Octet dans la mémoire locale Mot dans la mémoire locale Double mot dans la mémoire locale	L LB LW LD	L LB LW LD
Octet de périphérie d'entrée Octet de périphérie de sortie Mot de périphérie d'entrée Mot de périphérie de sortie Double mot de périphérie d'entrée Double mot de périphérie de sortie		PEB PAB PEW PAW PED PAD
Temporisation Compteur	T C	T Z
Bit dans un bloc de donnée Octet dans un bloc de donnée Mot dans un bloc de donnée Double mot dans un bloc de donnée		DBX DBB DBW DBD

Tableau 3-2 : Symboles des variables de l'API S7-300

4-2- Adressage des variables

Les adresses des différentes variables de l'API S7-300 sont rangées dans des octets (8 bits), des mots (16 bits) ou des doubles mots (32 bits) selon le type de la variable traitée.

Pour les entrées E (I) (lecture dans la Mémoire MIE)

- E y.x désigne une entrée, y est le numéro de voies (0 à 127), x sa position (0 à 7).
- EB y désigne un octet d'entrées.
- EW y désigne un mot d'entrées (16 bits).
- ED y désigne un double mot d'entrées (32 bits).

Les mêmes termes, en remplaçant E par P accèdent directement à la variable de la périphérie d'entrée.

*Pour les sorties **A (Q)** (sortie dans la Mémoire MIS)*

- A y.x désigne une sortie y est le numéro de voies (0 à 127), x sa position (0 à 7).
- AB y désigne un octet de sorties.
- AW y désigne un mot de sorties (16 bits).
- AD y désigne un double mot de sorties (32 bits).

Les mêmes termes, en remplaçant A par P accèdent directement à la variable de la périphérie de sortie.

*Pour les mémentos **M** (lecture dans la mémoire interne)*

- M y.x désigne un bit de mémoire y est le numéro d'octets (0 à 127), x sa position (0 à 7).
- MB y désigne un octet de mémoire.
- MW y désigne un mot de mémoire (16 bits).
- MD y désigne un double mot de mémoire (32 bits).

Le tableau 3-3 donne le mode d'adressage des différentes variables :

Bits	E**0.0** A**4.0** M**0.0** L**0.0** DB1.DB**X0.0**	Entrée (Eingang) digitales de la MIE Sortie (Ausgang) digitales de la MIS Mémoires Internes Globales (mémentos) Variables locales à chaque bloc Bit d'un bloc de données (DB1)
Byte : 8 bits	E**B0** A**B4** M**B0** L**B0** DB1.DB**B0**	8 bits d'entrées de E0.0 à E0.7 8 bits de sortie de A4.0 à A4.7 8 bits mémento de M0.0 à M0 .7 8 bits local de L0.0 à L0.7 8 bits de DB1 de DB1.DBX.0 à DB1.DBX.7
Word : 16 bits	E**W0** A**W4** PE**W256** PA**W304** M**W0** L**W0** DB1.DB**W0**	Désigne les octets EB0 et EB1 Désigne les octets AB4 et AB5 Entrée périphérique d'adresse 256 Sortie périphérique d'adresse 304 Désigne les octets MB0 et MB1 Désigne les octets LB0 et LB1 Désigne les octets DB1.DBB0 et DB1.DBB1
Dword : 32 bits	E**D0** A**D4** PE**D256** PA**D304** M**D0** L**D0** DB1.DB**D0**	Désigne les mots EW0 et EW2 Désigne les mots AW4 et AW6 Désigne les mots PEW256 et PEW258 Désigne les mots PAW304 et PAW306 Désigne les mots MW0 et MW2 Désigne les mots LW0 et LW2 Désigne les mots DB1.DBW0 et DB1.DBW2

Tableau 3-3 : Mode d'adressage des variables de l'API S7-300

Pour accéder à un bit dans une des zones mémoires, il faut préciser son adresse composée d'un identificateur de la zone mémoire, de l'adresse de l'octet et du rang du bit.

Exemple N°1:

E1.3 : 4 ème bit de l'octet 1 de la zone mémoire MIE

Pour accéder à un octet (8bits), dans une des zones mémoires, il faut préciser son adresse composée d'un identificateur de la zone mémoire, de l'adresse de l'octet.

Exemple N°2:

MB100 : Octet mémento d'adresse 100

EB20 : Octet de la zone mémoire MIE d'adresse 20

Pour accéder à un mot (16bits), dans une des zones mémoires, il faut préciser l'identificateur de la zone mémoire, l'adresse de l'octet du poids le plus fort du mot.

Exemple N°3:

MW1 : accès à un mot de la zone mémoire des mémentos formé par les octets d'dresses **MB1** et **MB2**.

MB1								MB2							
M1.7	M1.6	M1.5	M1.4	M1.3	M1.2	M1.1	M1.0	M2.7	M2.6	M2.5	M2.4	M2.3	M2.2	M2.1	M2.0

Pour accéder à un double mot (32bits), dans une des zones mémoires, il faut préciser l'identificateur de la zone mémoire, l'adresse de l'octet du poids le plus fort du double mot.

Exemple N°4:

MD10 : accès à un double mot de la zone mémoire des mémentos formé par les mots d'dresses **MW10** et **MW12**.

MW10		MW12	
MB10	MB11	MB12	MB13

Ce type de numérotation s'applique à tous les types de variables (A.., E..,P..., DB..,L..)
La cartographie d'adressage des différentes variables est donnée par la figure 2- 6.

BIT X.0	BIT X.1	BIT X.2	BIT X.3	BIT X.4	BIT X.5	BIT X.6	BIT X.7	OCTET X	MOT X		DOUBLE MOT X			
M0.0	M0.1	M0.2	M0.3	M0.4	M0.5	M0.6	M0.7	MB0	MW0					
M1.0	M1.1	M1.2	M1.3	M1.4	M1.5	M1.6	M1.7	MB1		MW1	MD0			
M2.0	M2.1	M2.2	M2.3	M2.4	M2.5	M2.6	M2.7	MB2	MW2			MD1		
M3.0	M3.1	M3.2	M3.3	M3.4	M3.5	M3.6	M3.7	MB3		MW3			MD2	
M4.0	M4.1	M4.2	M4.3	M4.4	M4.5	M4.6	M4.7	MB4	MW4		MD4			MD3
M5.0	M5.1	M5.2	M5.3	M5.4	M5.5	M5.6	M5.7	MB5		MW5		MD5		
M6.0	M6.1	M6.2	M6.3	M6.4	M6.5	M6.6	M6.7	MB6	MW6				MD6	
M7.0	M7.1	M7.2	M7.3	M7.4	M7.5	M7.6	M7.7	MB7		MW7				MD7
M8.0	M8.1	M8.2	M8.3	M8.4	M8.5	M8.6	M8.7	MB8	MW8		MD8			
M9.0	M9.1	M9.2	M9.3	M9.4	M9.5	M9.6	M9.7	MB9		MW9		MD9		
M10.0	M10.1	M10.2	M10.3	M10.4	M10.5	M10.6	M10.7	MB10	MW10				MD10	
M11.0	M11.1	M11.2	M11.3	M11.4	M11.5	M11.6	M11.7	MB11		MW11				MD11
M12.0	M12.1	M12.2	M12.3	M12.4	M12.5	M12.6	M12.7	MB12	MW12		MD12			
M13.0	M13.1	M13.2	M13.3	M13.4	M13.5	M13.6	M13.7	MB13		MW13		MD13		
M14.0	M14.1	M14.2	M14.3	M14.4	M14.5	M14.6	M14.7	MB14	MW14				MD14	
								MB15		MW15				MD15
								MB16	MW16		MD16			
								MB17		MW17		MD17		
								MB18	MW18				MD18	
								MB19		MW19				MD19
								MB20	MW20					
								MB21		MW20				
M22.0	M22.1	M22.2	M22.3	M22.4	M22.5	M22.6	M22.7	MB22						

Figure 2- 6 : Cartographie d'adressage des variables mémoire de l'API S7-300

4-3- Le mémento de cadence

Le mémento de cadence est un octet dont la valeur binaire de chaque bit est modifiée périodiquement par le système d'exploitation de la CPU. Le mémento de cadence est utilisé

pour déclencher des opérations périodiques. Une période/fréquence est attribuée à chaque bit de l'octet du mémento de cadence.

Le tableau 3-4 donne l'attribution des fréquences à chaque bit du mémento de cadence :

Bit	7	6	5	4	3	2	1	0
Période (s)	2	1,6	1	0,8	0,5	0,4	0,2	0,1
Fréquence (Hz)	0,5	0,625	1	1,25	2	2,5	5	10

Tableau 3-4 : Attribution des fréquences à chaque bit du mémento de cadence

L'adresse du mémento de cadence se configure dans la propriété de la CPU lors de la configuration matérielle de l'application. La figure 2- 7 donne un exemple de configuration d'un mémento de cadence lors de la configuration de la CPU. Il s'agit de l'octet mémento **M255**.

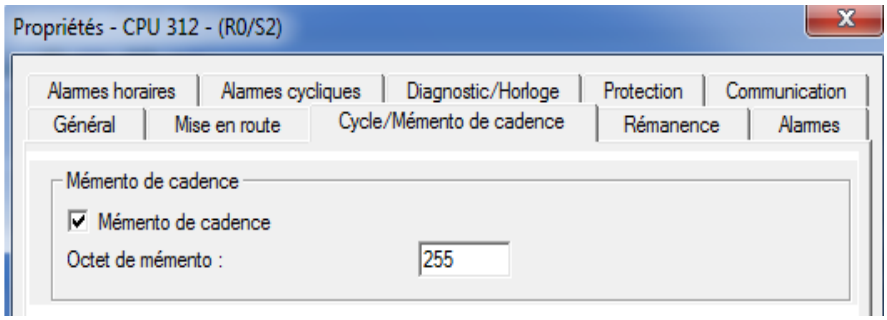


Figure 2- 7 : Configuration du mémento de cadence lors de la configuration de la CPU

4-4- Les zones de rémanence de la CPU

Les zones de rémanence sont des zones où les données sont inscrites dans des mémoires non volatiles. Les mémentos, les temporisations et les compteurs qui sont inscrits dans ces zones conservent leurs valeurs lors d'une coupure d'alimentation, même sans pile de sauvegarde.

Ces zones sont à définir lors de la configuration matérielle de la CPU. La figure 2- 8 donne un exemple de zones de rémanence pour une CPU 314.

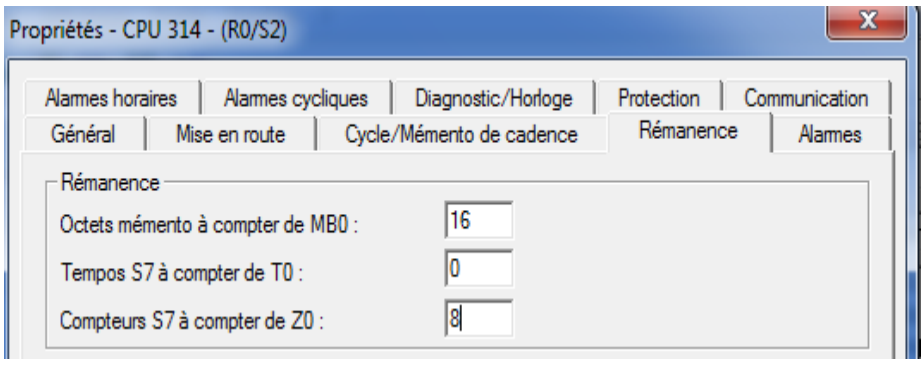


Figure 3- 8: Zones de rémanence de la CPU 314

5 Organisation d'un programme dans l'automate S7-300

5-1- Organisation d'un programme

Les API S7-300 se programment avec le logiciel STEP7. Deux programmes différents s'exécutent dans une CPU : le système d'exploitation et le programme utilisateur.

Système d'exploitation

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

- ✓ Le déroulement du démarrage et du redémarrage,
- ✓ L'actualisation de la mémoire image des entrées et l'émission de la mémoire image des sorties,
- ✓ L'appel du programme utilisateur,
- ✓ L'enregistrement des alarmes et l'appel des OB d'alarme,
- ✓ La détection et le traitement d'erreurs,
- ✓ La gestion des zones de mémoire,
- ✓ La gestion des interruptions,
- ✓ La communication avec des consoles de programmation et d'autres partenaires de communication.

Programme utilisateur

Le programme utilisateur contient toutes les fonctions nécessaires au traitement d'une tâche d'automatisation spécifique. Il doit entre autres:

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

La figure 2- 9 donne le cycle de déroulement d'un programme dans la CPU.

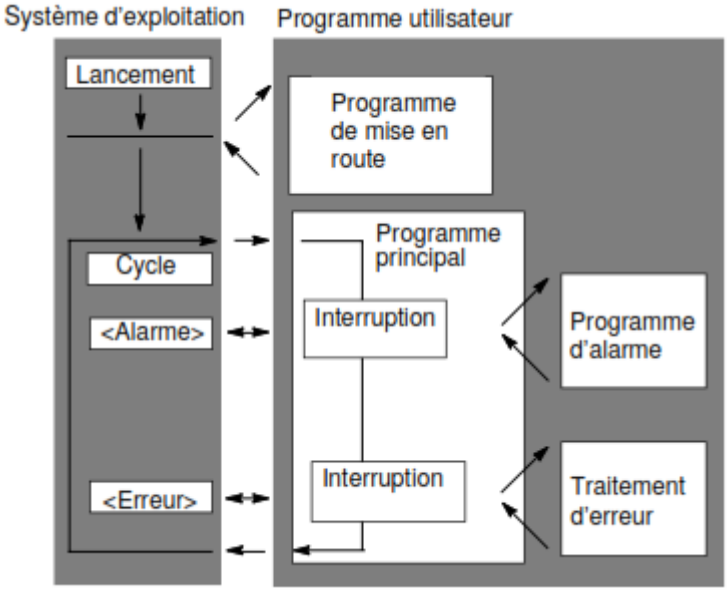


Figure 3- 9: Cycle de déroulement d'un programme dans la CPU

Le programme utilisateur peut être écrit complètement dans l'OB1 (programmation linéaire). Cela n'est toutefois recommandé que pour des programmes simples s'exécutant sur des CPU S7-300 avec une mémoire peu importante.

Les automatismes complexes seront mieux traités en les subdivisant en parties plus petites qui correspondent aux fonctions technologiques du processus d'automatisation ou qui peuvent être utilisées plusieurs fois. Dans le programme utilisateur, ces tâches partielles sont représentées par des parties de programme correspondantes : les blocs (programmation structurée).

La figure 3- 10 montre les approches linéaire et structurée du programme utilisateur.

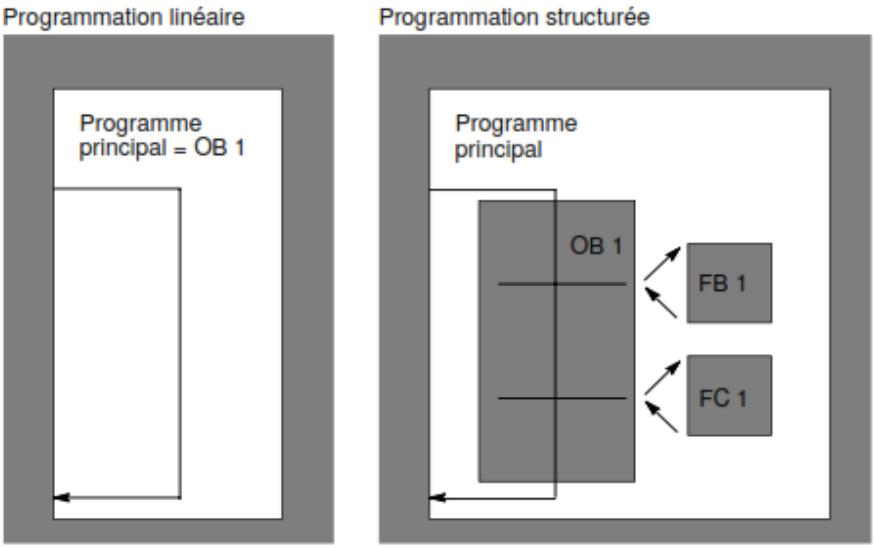


Figure 3- 10: Programmation linéaire et programmation structurée

Pour faire fonctionner le programme utilisateur, les blocs qui le composent doivent être appelés selon une hiérarchie d'appel comme le montre la figure 3- 11.

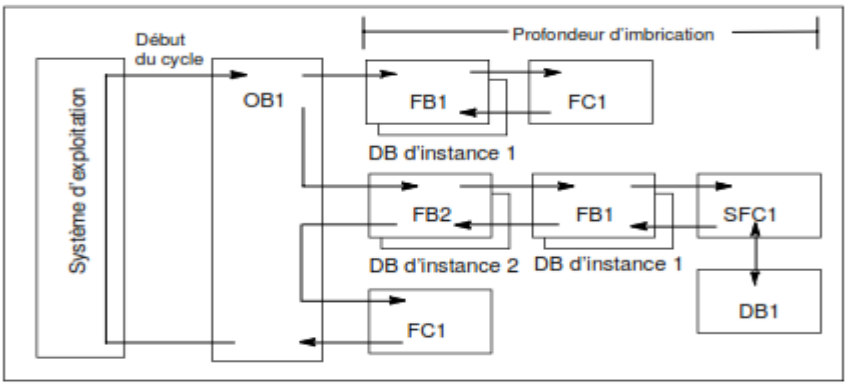


Figure 3- 11: Exemple d'hiérarchie d'appel des blocs fonctionnels dans un cycle de traitement.

Le cycle de déroulement d'un programme dans la CPU d'un automate S7-300 est donné par la figure 3-12.

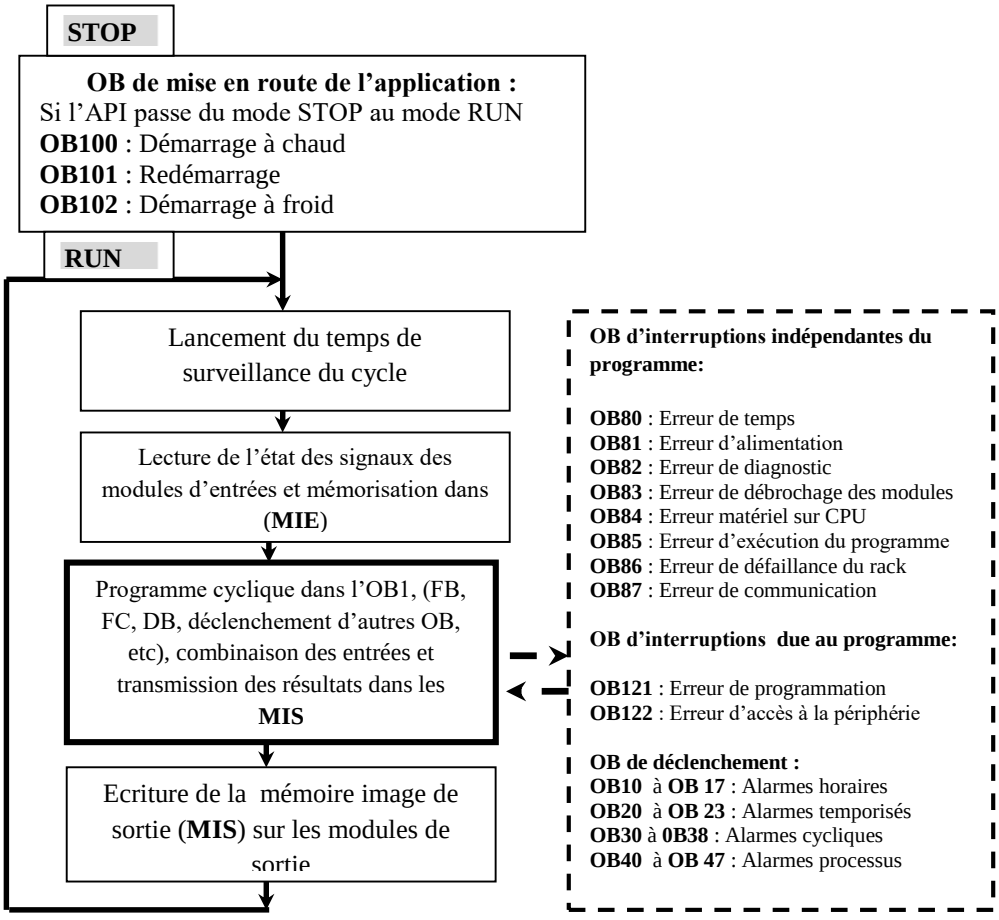


Figure 3- 12: Cycle de déroulement d'un programme dans la CPU

Les alarmes ne peuvent être traitées que si un bloc d'organisation correspondant est contenu dans le programme utilisateur. Dans l'annexe 1, on présente les types d'alarmes et leurs applications pour les API S7-300.

5-2- Types de données utilisées

Lors de la programmation de l'API S7-300, les types de données utilisées par la CPU sont donnés par le tableau 3-5:

Types de données	Taille	Description	plage
BOOL	1 Bit	Texte Booléen	**TRUE / FALSE**
BYTE	8 Bits	Nombre Hexadécimal Nombre Binaire	B#16#00 à B#16#FF B#2#0000_0000 à B#2#1111_1111
CHAR	8 Bits	Caractère ASCII	"A" "B" ….etc...
WORD	16 Bits	Nombre Hexadécimal Nombre DCB Nombre Décimal non Signé	W#16#0000 à W#16#FFFF C#0 à C#999 B#(0,0) à B#(255,255)
INT	16 Bits	Nombre entier signé	-32768 à 32767
DWORD	32Bits	Nombre Hexadécimal Nombre Binaire Nombre Décimal Non Signé	DW#16#00000000 à DW#16#FFFFFFFF 2#0 à 2#1111_…_1111 B#(0,0,0,0) à B#(255,255,255,255)
DINT	32 Bits	Nombre entier signé	-2^{31} à $(2^{31}-1)$ = L#-2147483648 à L#2147483647
REAL	32 Bits	Nombre réel à virgule flottante	Limite Supérieure: +/- 3.402823 $.10^{38}$ Limite Inférieure: +/- 1.175495.10^{-38}
TIME	16 Bits	Durée IEC en pas de ms	T#24D_20H_31M_23S_648M à T#24D_20H_31M_23S_647MS
S5TIME	16 Bits	Durée Tempos S7 (*10ms)	S5T#10 ms à S5T#2H46M30S
DATE	16 Bits	Date IEC en pas de 1 jour	D#1990_1_1 à D#2168_12_31

Tableau 3-5 : Types de données utilisées par la CPU de l'API S7-300

5-3- Blocs S7 dans le programme utilisateur

L'organisation du programme utilisateur avec STEP 7 est conçue à partir de blocs, qui sont :

- Les blocs utilisateurs (**OB**, **FC**, **FB, DB**) où on écrit le programme:
    - Les **blocs d'organisation (OB)** constituent l'interface entre le système d'exploitation de la CPU et le programme utilisateur. Ils permettent de déclencher l'exécution de certaines parties du programme. Le plus important est l'**OB1** (programme principal) qui est exécuté d'une manière cyclique.
    - Les **blocs fonctions (FC)** qui sont des blocs à programmer par l'utilisateur permettant la décomposition de l'application en sous parties.
    - Les **blocs fonctionnels (FB)** qui sont des blocs à programmer par l'utilisateur permettant la décomposition de l'application en fonctions répétitives. On ne l'écrit qu'une fois, et on peut l'utiliser autant de fois que nécessaire.
    - Les **blocs de données (DB)** qui sont des blocs à programmer par l'utilisateur permettant de créer des zones de mémorisations de données. Ils constituent la mémoire des blocs FB paramétrés (DB D'INSTANCE).
- Les blocs systèmes qui sont des ressources prédéfinies exploitables par l'utilisateur (**SFC**, **SFB, FC** et **FB**) qui sont appelés aussi blocs SIEMENS.

- Ils sont déjà programmés par Siemens, donc ils sont protégés.
- Les SFC et SFB sont des programmes résidents dans la mémoire de l'automate.
- Les FC et FB prédéfinies sont fournis lors de l'acquisition d'un module ou d'un logiciel spécifique. **Exemple** : Le FB PID_FM permet la lecture des données du module de régulation FM 355 dans la CPU de l'API.

- Les **programmes d'interruptions** qui sont des blocs d'organisations autres qu'OB1. Ils sont exécutés lorsqu'un événement d'interruption se produit.

  - **OB100** est un bloc qui est exécuté lorsque l'automate passe en mode RUN (démarrage à chaud).
  - **OB35** est un bloc qui peut être exécuté toutes les 100ms.
  - **OB80** est un bloc qui est exécuté en cas de débordement du temps cyclique (chien de garde).
  - **OB82** est un bloc qui est exécuté en cas de défaillance d'un module hardware.
  - **OB122** est un bloc qui est exécuté en cas d'erreur de programmation.

Le fonctionnement du programme principal est donné par la figure 3-13.

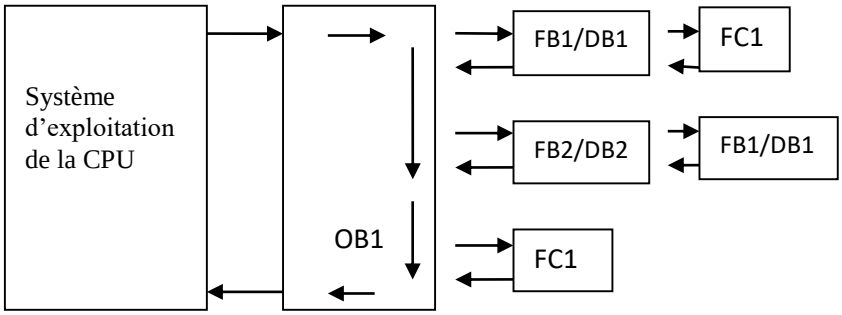


Figure 3- 13: Déroulement du programme principal dans la CPU

6 Programmation de l'API S7-300

La programmation structurée permet la rédaction claire et transparente des programmes. Elle permet la construction d'un programme complet à l'aide de modules qui peuvent être échangés et/ou modifiés à volonté.

Pour permettre une programmation structurée confortable, il faut prévoir plusieurs types de modules : les modules d'organisation (OB), de programmes (FB), fonctionnels (FC), de données (DB). Dans le dossier Blocs, on retrouve l'OB1 ainsi que tous les Blocs appelés par celui-ci (FB, FC, DB, …)

6-1- l'OB1

L'OB1 gère le programme en fonctionnement normal, on peut utiliser l'OB1 pour programmer directement à l'intérieur de celui-ci des ordres ou alors appeler des Blocs (FCs ou FBs) qui permettent de structurer le programme. Ainsi on peut créer un bloc pour un moteur avec tous

ses défauts, ses status (HMI), ses modes opératoires (Auto/Manu), ces commandes (Marche/Arrêt), ceci est valable pour des vannes où autres.

6-2- Les DB

Les DBs (Data Base) permettent de stocker des valeurs dans une base de données. Un DB peut être de deux types différents:

- DB d'instance (générer automatiquement par le système et qui sont associées aux FB et SFB),
- DB global (permet de structurer les données. Ils sont interrogeable dans le programme par n'importe quel bloc).

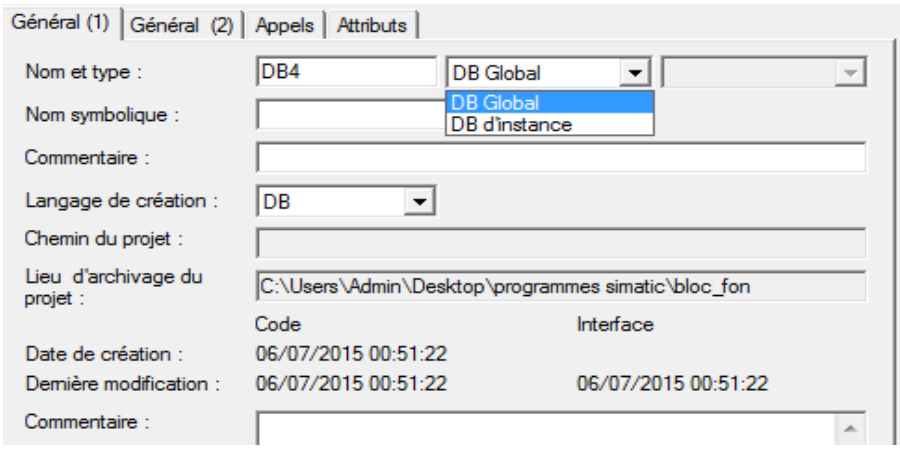


Figure 3- 14: Ecran de création d'une DB

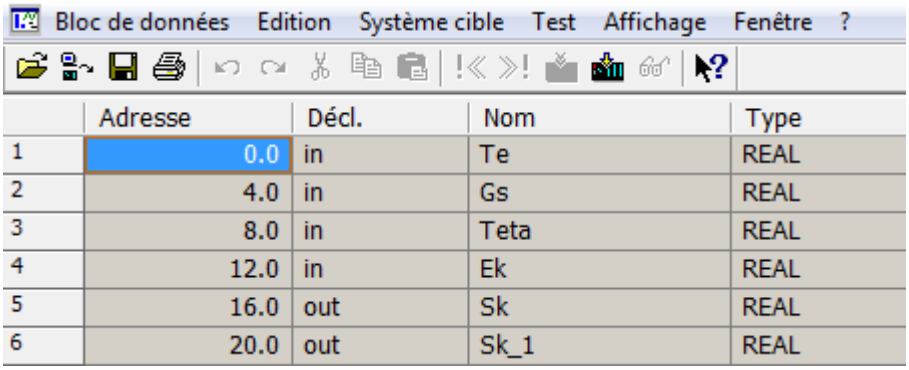
Les blocs de données (DBs) ont les caractéristiques suivantes :

- Les valeurs des DBs sont toutes sauvegardées hors tension,
- Le nombre et la taille des DBs ne sont pas fonction de la CPU mais de la taille mémoire utilisateur. En effet les DBs prennent de la place mémoire, de la même manière que les FCs, FBs, OBs.
- On peut stoker dans une DB tous les types de variables: BOOL, BYTE, WORD, INT, DWORD, DINT, REAL.

La syntaxe pour appeler une variable dans une DB est la suivante:

DB100.DBX10.0 → interroge le bit 0 de l'octet 10 de la DB100.
DB100.DBB10 → interroge l'octet 10 de la DB100.
DB100.DBW10 → interroge le mot 10 de la DB100.
DB100.DBD10 → interroge le double mot 10 de la DB100.
***Remarque** :*

La figure 2-15 montre la déclaration des variables de type REAL dont les adresses sont des multiples de "**4.0**" (0.0; 4.0; 8.0; 12.0; …) car une variable de type REAL est composée de **4 octets.**



	Adresse	Décl.	Nom	Type
1	0.0	in	Te	REAL
2	4.0	in	Gs	REAL
3	8.0	in	Teta	REAL
4	12.0	in	Ek	REAL
5	16.0	out	Sk	REAL
6	20.0	out	Sk_1	REAL

Figure 3- 15: Ecran de déclaration des variables d'une DB

On obtient le type de fenêtre ci-dessus à l'ouverture d'une DB. Dans la barre d'outils on peut choisir dans "**Affichage**" soit d'être en "**Vue des données**" ou en "**Vue des déclarations**". La première option permet de voir et de modifier les données **en état actuel** et la seconde permet de modifier les valeurs à **l'état initial.**

6-3- Les FCs

Les FCs sont des fonctions permettant la programmation de sous-programmes. Le plus souvent on les appellera par l'intermédiaire de l'OB1.

Le bloc utilise des variables temporaires pour ces calculs et aussi pour l'affectation de ces E/S. En appelant ce bloc FC dans un autre endroit du programme (FC, OB, …), alors celui-ci exécutera son contenu à l'aide de ses paramètres d'entrées et modifiera des valeurs de sorties. On affectera aux E/S du FC les variables faisant partie de la table de mnémoniques.

Dans la partie de déclaration du bloc on trouve :

IN: Paramètres entrant dans le bloc.
OUT: Paramètres sortant du bloc.
IN/OUT: Paramètres entrant et sortant du bloc.
TEMP: Mémoire Interne (L) remise à zéro à chaque cycle.

Un exemple de déclaration des entrées/sorties d'une fonction est donné par la figure 3-16.

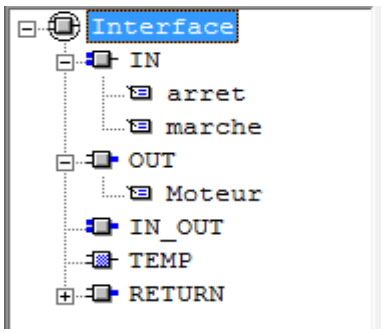


Figure 3- 16: Exemple de déclaration des variables d'une fonction

Un exemple du contenu d'une fonction est donné par la figure 3-17.

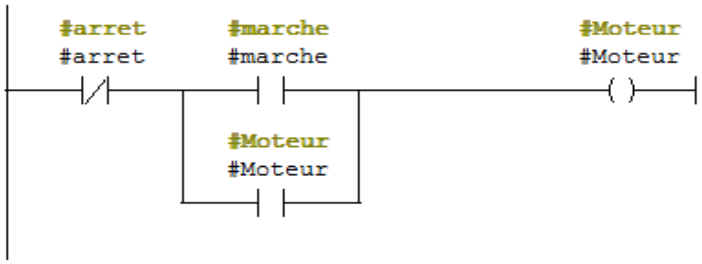


Figure 3- 17: Exemple du contenu d'une fonction

L'appel à une fonction à partir du bloc d'organisation OB1 est donné par la figure 3-18.

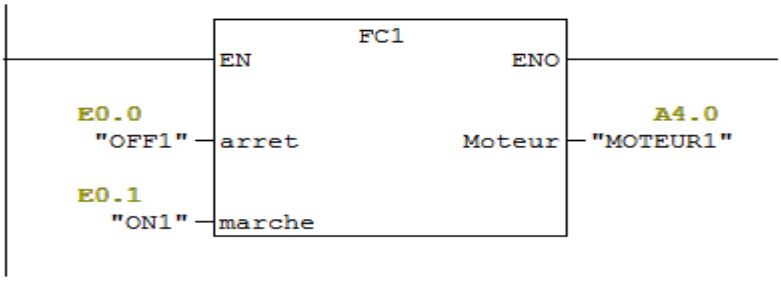


Figure 3- 18: Exemple d'appel à une fonction à partir d'OB1

L'appel à une fonction peut être conditionné par l'entrée EN. La sortie ENO permet de contrôler l'exécution sans erreur de la fonction.

6-4- Les FBs

Les FBs obéissent aux mêmes principes que les FCs, à la différence qu'un FB doit être impérativement associé à une DB d'instance. A chaque fois que l'on utilisera un FB, un DB différent devra lui être associé.

Dans la partie déclarative du bloc on trouve :

IN: Paramètres entrant dans le bloc.
OUT: Paramètres sortant du bloc.
IN/OUT: Paramètres entrant et sortant du bloc.
STAT: Mémoire sauvegardée et accessible au niveau du DB d'instance
TEMP: Mémoire Interne remise à zéro à chaque cycle.

Un exemple de déclaration des entrées/sorties d'un bloc fonctionnel est donné par la figure 3-19.

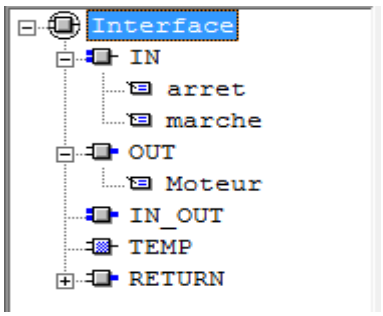


Figure 3- 19: Exemple de déclaration des variables d'un bloc fonctionnel

La figure 3-20 donne un exemple de contenu d'un bloc fonctionnel

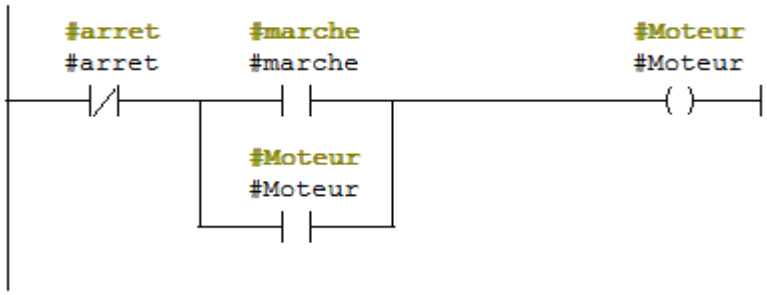


Figure 2- 20: Exemple de contenu d'un bloc fonctionnel

L'appel à un bloc fonctionnel à partir du bloc d'organisation OB1est donné par la figure 3-21.

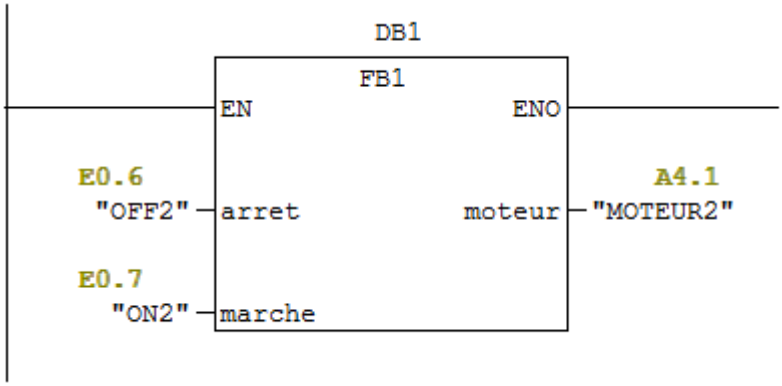


Figure 3- 21: Exemple d'appel à un bloc fonctionnel à partir d' OB1

Chaque appel d'un même FB est associé à un DB d'instance différent. Ce DB permet en « Vue des données » de visualiser les variables internes du FB.

Premier appel du FB1 ***Deuxième appel du FB1***

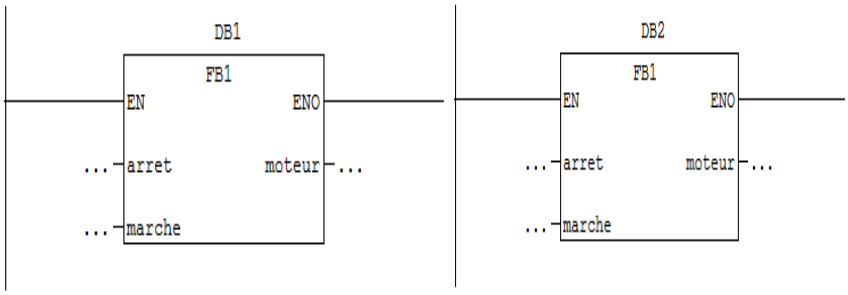


Figure 3- 22: Appel multiple à un bloc fonctionnel à partir de OB1

Le bloc fonctionnel garde l'état précédent de ses variables en cas d'arrêt du programme contrairement à une fonction.

La différence entre un FB et un FC est que le FB permet de mémoriser l'état de ses variables dans un DB, alors que le FC ne le permet pas.

6-5- Les SFB et les SFC

Ce sont des blocs fonctionnels préprogrammés par Siemens, on ne peut que les utiliser. Il n'est pas nécessaire de les transférer puisqu'ils sont déjà dans la CPU.

Exemple de SFB:

SFB 0 : Compteur IEC.
SFB 1 : Décompteur IEC.
SFB 2 : Compteur/Décompteur IEC.
SFB 4 : Temporisation TON IEC.
SFB 5 : Temporisation TOFF IEC.
SFB 8 à SFB 16 : Fonctions de Communication S7.
SFB 19 : Mise en Marche d'un appareil distant.
SFB 20 : Mise en Arrêt d'un appareil distant.
SFB 21 à SFB 23 : État d'un appareil distant

Exemple de SFC:

SFC 0 et SFC 1 : Mise à l'heure et Lecture de l'horloge Automate.
SFC 13 : Diagnostique Esclave DP Intelligent.
SFC 14 et SFC 15 : Lecture et Écriture de données cohérentes Esclave
SFC 20 : Copie de bloc (BLKMOV).
SFC 21 : Copie et remplissage de zones (FILL).
SFC 22 à SFC 25 : Création, Effacement, Test et Compression de DB.
SFC 26 : Mise à jour des MIE.
SFC 27 : Mise à jour des MIS.
SFC 28 : Désactivation des alarmes horaires...
SFC 43 : Réarmement du Chien de Garde (RE_TRIG).
SFC 46 : Mise en STOP de la CPU.
SFC 47 : Attente au niveau de la scrutation.
SFC 58 et SFC 59 : Écriture et Lecture d'esclave ASI2 (E/S B et Analogique
SFC 60 à SFC 69 : Communication MPI (SFC65=> et SFC66 <=)

7 Instructions de base pour la programmation en STEP 7

7-1- Opérations combinatoires sur bits

a) Description

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

Le résultat d'une opération combinatoire sur bits est affecté au bit de résultat logique (RLG).

Il existe des opérations combinatoires sur bits pour effectuer les fonctions suivantes :

- ---| |--- Contact à fermeture
- ---| / |--- Contact à ouverture
- ---( ) Bobine de sortie
- ---(#)--- Connecteur
- ---|NOT|--- Inverseur du bit RLG
- ---(SAVE) Sauvegarder le bit RLG dans RB (RB : résultat binaire du mot d'état)

Les opérations suivantes réagissent à un bit RLG égal à 1 :

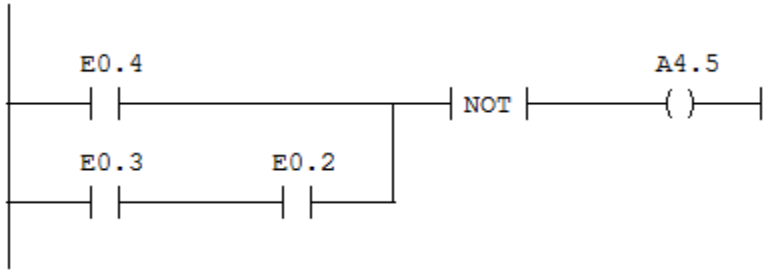
- ---(S) Mettre à 1
- ---(R) Mettre à 0
- SR Bascule mise à 1, mise à 0
- RS Bascule mise à 0, mise à 1

D'autres opérations exécutent les fonctions suivantes en cas de front montant ou descendant :

- ---(N)--- Détecter front descendant
- ---(P)--- Détecter front montant
- NEG Détecter front descendant de signal
- POS Détecter front montant de signal

b) Exemples

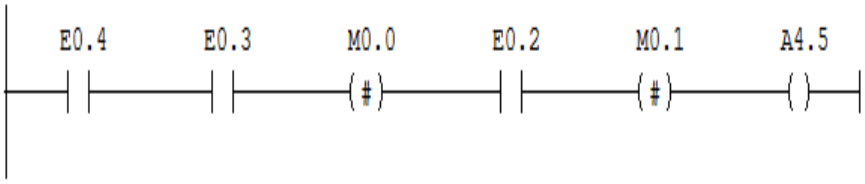
Fonction inverser RLG



RLG = E0.4 + E0.2 . E0.3
La sortie A4.5 = 0 si RLG = 1 et A4.5 = 1 si RLG = 0

Figure 3- 23: Exemple d'utilisation de la fonction inverser RLG

Fonction connecteur



Le RLG du E0.4.E0.3 est placé dans M0.0
Le RLG du E0.4.E0.3.E0.2 est placé dans M0.1
La sortie A4.5 = 1 si (M0.0 = 1et M0.1 = 1)

Figure 3- 24: Exemple d'utilisation de la fonction connecteur

Fonction détecter front montant

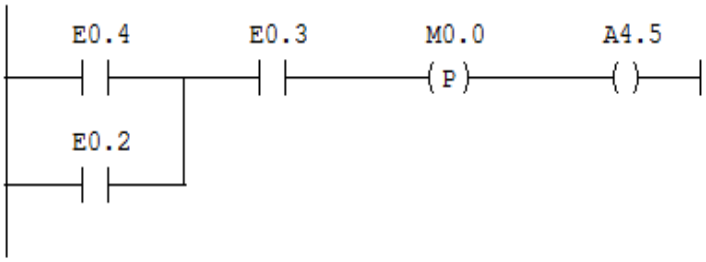


Figure 3- 25: Exemple d'utilisation de la fonction détecter front montant

Le mémento M0.0 mémorise l'état du signal RLG de toute combinaison binaire. En cas de passage de 0 à 1 du bit RLG, A4.5 = 1.

7-2- Opérations de comparaison

a) Description

Ce type d'instructions est utilisé pour comparer deux entrées IN1 et IN2 selon les types de comparaison suivants :

= = IN1 égal à IN2
<> IN1 différent de IN2
> IN1 supérieur à IN2
< IN1 inférieur à IN2
>= IN1 supérieur ou égal à IN2
<= IN1 inférieur ou égal à IN2

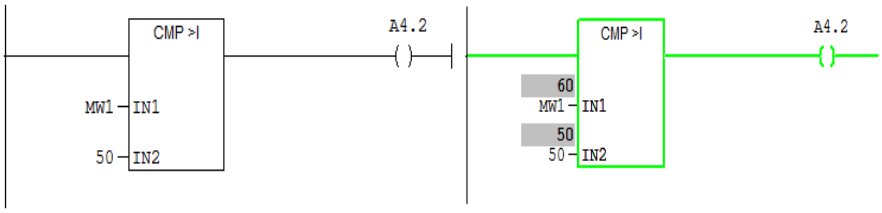
Si la comparaison est vraie, le bit de résultat logique (RLG) est à 1.

Dans STEP 7, on dispose des opérations de comparaison suivantes :

- CMP ? I pour comparer deux entiers de 16 bits (16 Bit)
- CMP ? D pour comparer deux entiers de 32 bits (32 Bit)
- CMP ? R pour comparer deux nombres réels

b) Exemples

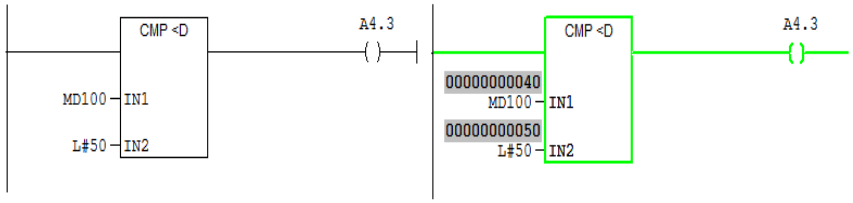
Comparaison de deux entiers simples (16 bits)



Comparaison entre l'entier MW1 et 50.
Si (MW1 = 60) > 50 alors RLG = 1 et A4.2 = 1

Figure 3- 26: Exemple de programmation d'un comparateur de deux entiers simples

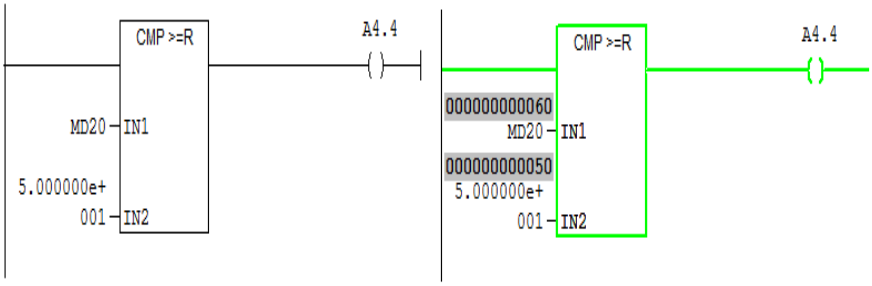
Comparaison de deux entiers doubles (32 bits)



Comparaison entre l'entier double MD100 et 50.
Si (MD100 = 40) < 50 alors RLG =1 et A4.3 = 1.

Figure 3- 26: Exemple de programmation d'un comparateur de deux entiers doubles

Comparaison de deux réels



Comparaison entre le réel MD20 et 50.0
Si (MD20 = 60) >= 50.0 alors RLG = 1 et A4.4 = 1

Figure 3- 27: Exemple de programmation d'un comparateur de deux réels

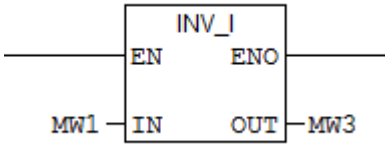
7-3- Opérations de conversion

a) Description

Les opérations de conversion sont utilisées pour lire le contenu du paramètre d'entrée IN, et le convertir selon le format souhaité. Le résultat de conversion est rangé dans le paramètre de sortie OUT.

Dans STEP 7, on dispose des opérations de conversion suivantes :

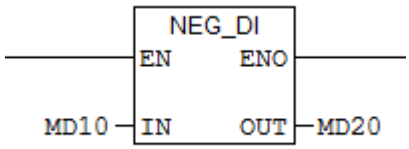
- BCD_I Convertir nombre DCB en entier de 16 bits
- I_BCD Convertir entier de 16 bits en nombre DCB
- BCD_DI Convertir nombre DCB en entier de 32 bits
- I_DI Convertir entier de 16 bits en entier de 32 bits
- DI_BCD Convertir entier de 32 bits en nombre DCB
- DI_R Convertir entier de 32 bits en réel
- **INV_I** Complément à 1 d'entier de 16 bits
  (**OUT = - (IN+1)**)



Cette opération lit le contenu du paramètre d'entrée IN et exécute l'opération de combinaison OU exclusif avec le masque hexadécimal W#16#FFFF afin d'inverser la valeur de chaque bit.

MW1 = 01000001 10000001 est converti en
MW3 = 10111110 01111110.

- INV_DI Complément à 1 d'entier de 32 bits
(**OUT = - (IN+1))**
- NEG_I Complément à 2 d'entier de 16 bits (**OUT = - IN)**
- **NEG_DI** Complément à 2 d'entier de 32 bits (**OUT = - IN)**

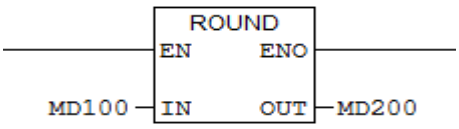


Cette opération lit le contenu du paramètre d'entrée IN et en change le signe (par exemple, valeur positive en valeur négative).

MD10 = + 1000 donne **MD20 = - 1000**.

- NEG_R Inverser le signe d'un nombre réel (**OUT = - IN)**
- **ROUND Arrondir**

Cette opération lit le contenu du paramètre d'entrée IN comme nombre à virgule flottante et le convertit en nombre entier de 32 bits. Le résultat, qui est le nombre entier le plus proche, est rangé dans le paramètre de sortie OUT. Si le nombre en virgule flottante se situe exactement entre deux nombres entiers, le nombre pair est pris comme résultat.



Le contenu du double mot de mémento MD100 est lu comme nombre à virgule flottante et converti en nombre entier de 32 bits. Le résultat de cette fonction "Arrondir par excès ou par défaut" est rangé dans le double mot de mémento MD200.

- TRUNC Tronquer à la partie entière
- CEIL Convertir réel en entier supérieur le plus proche
- FLOOR Convertir réel en entier inférieur le plus proche

b) Exemples

Conversion d'un nombre entier en un nombre réel

Pour convertir un nombre entier en un nombre réel, on doit passer par une conversion d'un entier simple en un entier double, puis du double entier en un nombre réel. (Cette conversion est nécessaire lors de la manipulation des entrées analogiques).

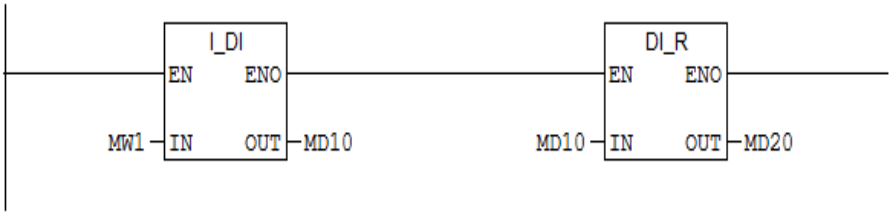


Figure 3- 28: Exemple de programmation d'une conversion nombre entier→ nombre réel

7-4- Opérations sur les nombres à virgule flottante (nombres réels)

a) Description

Les nombres à virgule flottante IEEE -754 de 32 bits ont le type de données REAL.

Les opérations arithmétiques sur les nombres réels à virgule flottante permettent d'exécuter les fonctions arithmétiques suivantes:

- **ADD_R :** Addition de deux nombres réels,
- **SUB_R** : Soustraction de deux nombres réels,
- **MUL_R** : Multiplication de deux nombres réels,
- **DIV_R** : Division de deux nombres réels,
- **ABS** : Valeur absolue d'un nombre réel,
- **SQR** : Carré d'un nombre réel,
- **SQRT** : Racine carrée d'un nombre réel,
- **LN :** Logarithme naturel d'un nombre réel,
- **EXP :** Valeur exponentielle sur la base e (= 2,71828) d'un nombre réel,
- Fonctions trigonométriques d'angles représentés :

  - Sinus (**SIN**) et Arc sinus (**ASIN**),
  - Cosinus (**COS**) et Arc cosinus (**ACOS**),
  - Tangente (**TAN**) et Arc tangente (**ATAN**),

b) Exemples

Mise à l'échelle d'une entrée entière en une valeur réelle

Format unipolaire :

La fonction prend une valeur entière (IN) de l'entrée analogique (MW10) en format (unipolaire : entre 0 et 27648) et la convertit en une valeur réelle (OUT) placée dans MD 20 exprimée en unités physiques, comprises entre une limite inférieure (0 %) et une limite supérieure (100 %). L'équation de conversion est donnée par la relation suivante :

$$OUT = \frac{100}{27648}.IN$$

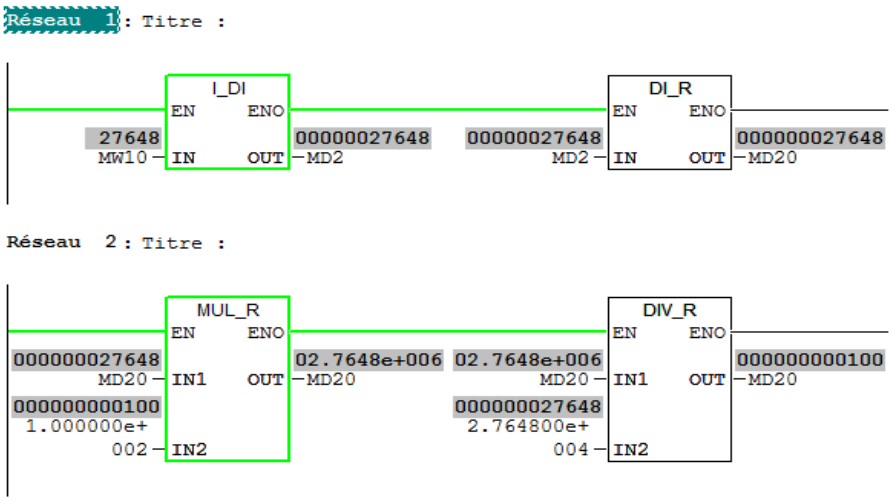


Figure 3- 29: Exemple de programmation d'une mise à l'échelle d'une entrée de type entier en format unipolaire (Résultat de simulation pour MW10 = 27648)

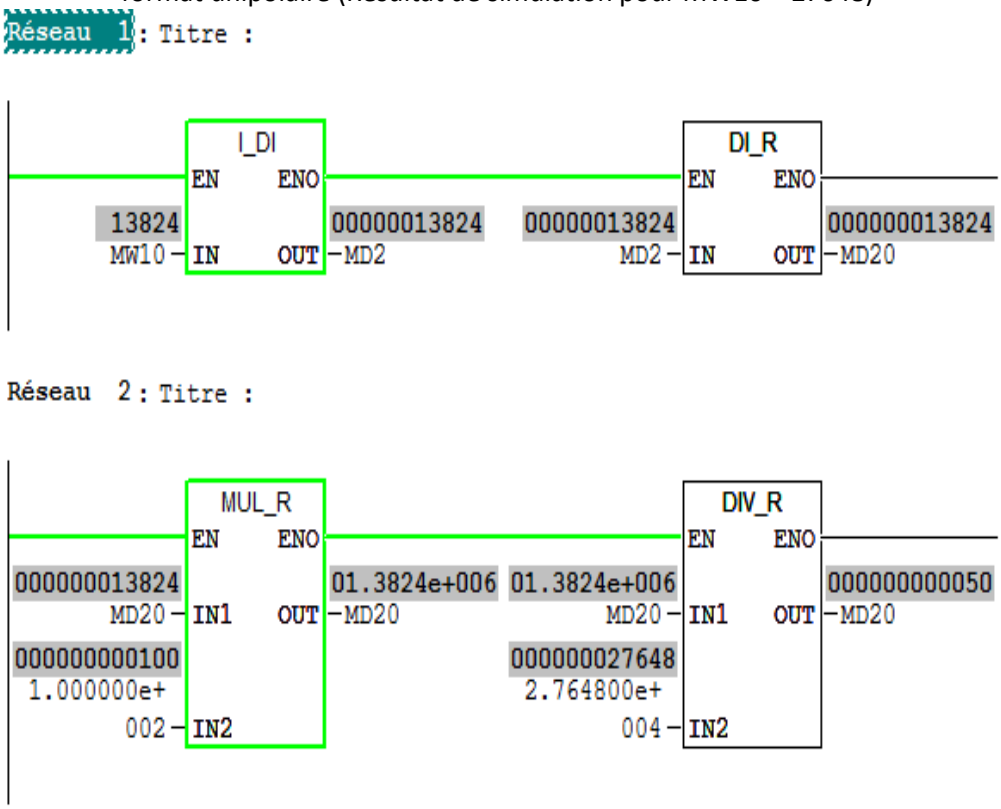


Figure 3- 30: Exemple de programmation d'une mise à l'échelle d'une entrée de type entier en format unipolaire (Résultat de simulation pour MW10 = 13824)

Format bipolaire :

La fonction prend une valeur entière (IN) de l'entrée analogique (MW10) en format (bipolaire : entre -27648 et 27648) et la convertit en une valeur réelle (OUT) placée dans MD 20 exprimée en unités physiques, comprises entre une limite inférieure (0 %) et une limite supérieure (100 %).

L'équation de conversion est donnée par la relation suivante :

$$OUT = \frac{100}{55296}.(IN + 27648)$$

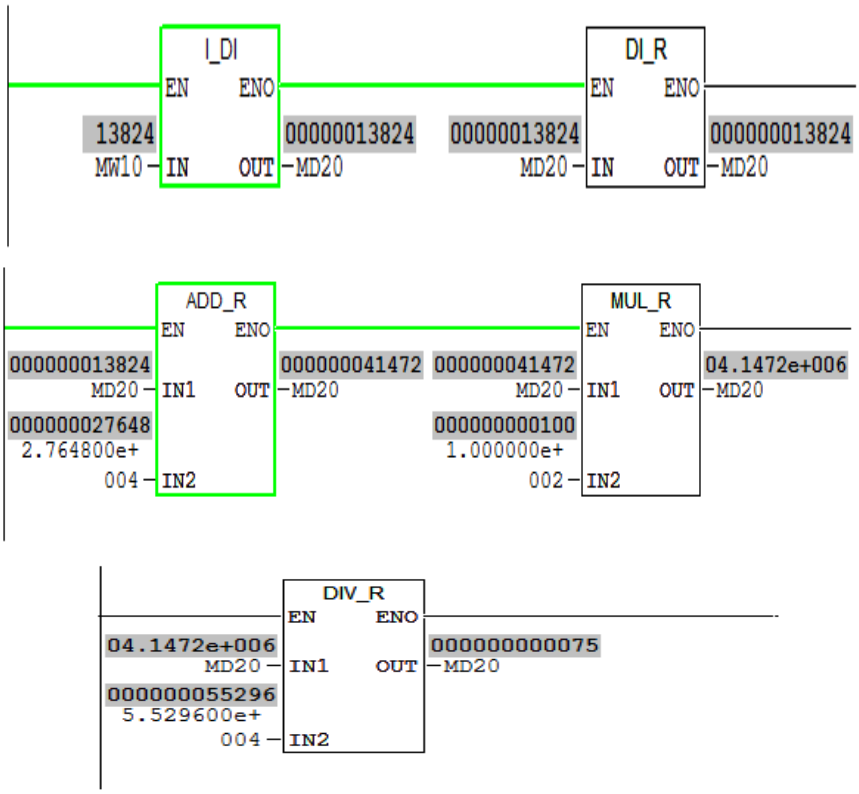
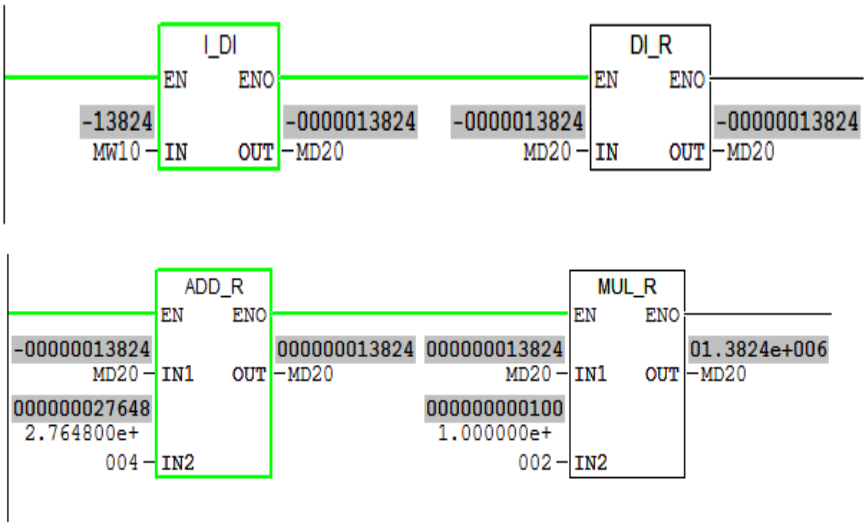


Figure 3- 31: Exemple de programmation d'une mise à l'échelle d'une entrée de type entier en format bipolaire (Résultat de simulation pour MW10 = 13824)



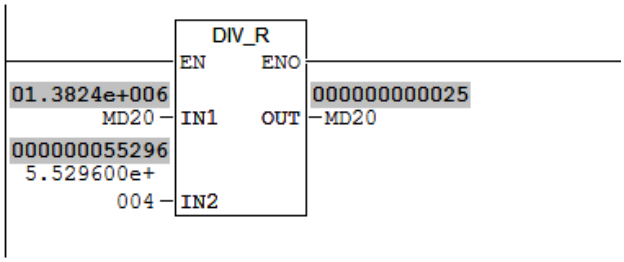
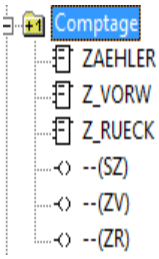


Figure 3- 32: Exemple de programmation d'une mise à l'échelle d'une entrée de type entier en format bipolaire (Résultat de simulation pour MW10 = -13824)

7-5- Opérations de comptage

a) Description

La programmation des API S7-300 permet d'utiliser jusqu'à 256 compteurs (un compteur est numéroté de **Z 0** à **Z 255** ou de **C0** à **C255)**. La plage de la valeur de comptage de chaque compteur est comprise entre 0 et 999 au format suivant : C# 127 (C# correspond au format décimal) et elle peut être modifiée en utilisant les opérations suivantes:



ZÄHLER Paramétrage et compteur d'incrémentation /décrémentation
Z_VORW Paramétrage et compteur d'incrémentation
Z_RUECK Paramétrage et compteur de décrémentation
---(SZ) Initialiser compteur
---(ZV) Incrémenter
---(ZR) Décrémenter

Pour le paramétrage des compteurs dans une application, on utilise les blocs suivants :

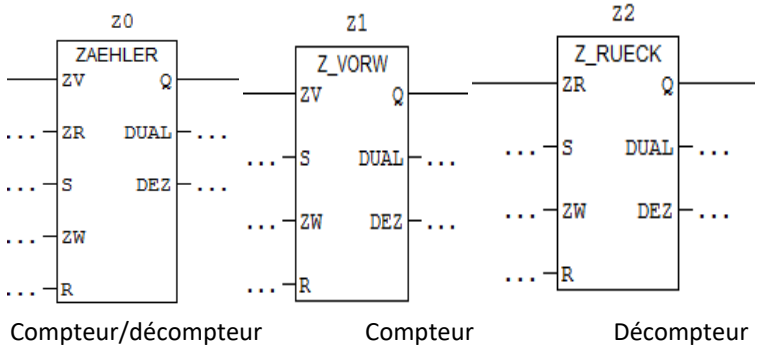


Figure 3- 33: Représentation allemande des compteurs

Z : Numéro d'identification du compteur.
ZV : (BOOL) Entrée d'incrémentation
ZR : (BOOL) Entrée de décrémentation
S : (BOOL) Entrée d'initialisation du compteur
ZW : (WORD) Valeur de comptage entrée sous forme C#<valeur> dans la plage comprise entre 0 et 999 (Valeur d'initialisation du compteur)
R : (BOOL) Entrée de remise à zéro
DUAL : (WORD) Valeur de comptage en cours (format hexadécimal)
DEZ : (WORD) Valeur de comptage en cours (format DCB)
Q : (BOOL) Etat du compteur

Un front montant à l'entrée S de cette opération initialise le compteur à la valeur figurant dans l'entrée ZW. Un état logique 1 à l'entrée R remet le compteur, et donc la valeur de comptage, à zéro.

Le compteur est incrémenté d'une unité si l'état de signal à l'entrée ZV passe de 0 à 1 « front montant » et que la valeur du compteur est inférieure à 999.

Le compteur est décrémenté d'une unité si l'état de signal à l'entrée ZR passe de 0 à 1 « front montant » et que la valeur du compteur est supérieure à 0.

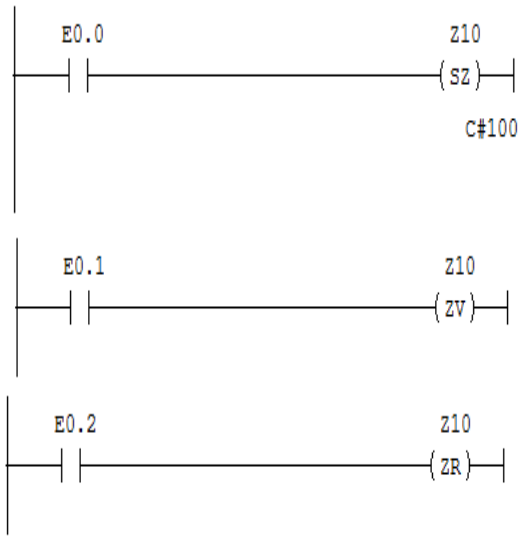
En cas de front montant aux deux entrées de comptage, les deux fonctions sont exécutées et la valeur de comptage reste inchangée.

Si le compteur est mis à 1 et si le RLG = 1 aux entrées ZV/ZR, le compteur compte une fois dans le cycle suivant, même si aucun changement de front n'a eu lieu.

L'état du signal à la sortie Q est à 1 lorsque la valeur de comptage est supérieure à 0 ; il est à 0 lorsque la valeur de comptage est égale à 0.

b) Exemple

Cette opération incrémente d'un la valeur du compteur précisé si le RLG présente un front montant et si la valeur du compteur est inférieure à 999. En l'absence de front montant au RLG ou si le compteur est déjà égal à 999, la valeur du compteur reste inchangée.



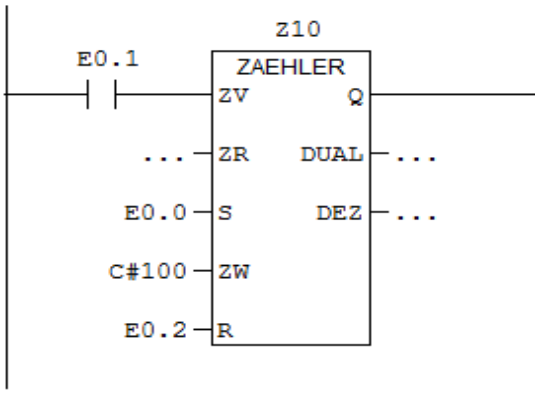


Figure 3- 34: Exemple de programmation d'un compteur

Si l'état de signal en E 0.0 passe de 0 à 1 (front montant du RLG), le compteur Z10 est initialisé avec la valeur 100.

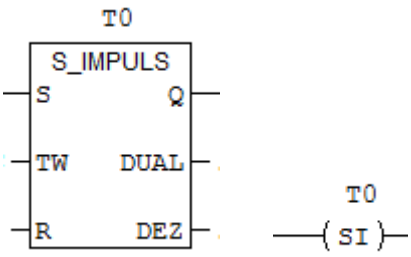
Si l'état de signal en E 0.1 passe de 0 à 1 (front montant du RLG), la valeur de comptage du compteur Z10 est incrémentée d'un, à moins qu'elle ne soit déjà à 999. En l'absence de front montant au RLG, la valeur de Z10 reste inchangée.

Si l'état de signal à l'entrée E 0.2 est égal à 1, le compteur est mis à zéro.

7-6- Opérations de temporisation

a) Description

Les temporisateurs sont numérotés de **T0** à **T255**. Les temporisations peuvent se représenter sous forme de bobine ou de bloc comme le montre la figure 3-35.



S: Activation de la Temporisation.
TW: Valeur de Consigne
Format : **TW = S5T#wH xxM yyS zzzmS**
R: Remise à zéro de la temporisation.
Q: État de la Tempo (fonction du type).
DUAL: Valeur actuelle de la temporisation.
Cette valeur est comprise entre 0 et 999.
DEZ: Valeur actuelle en DCB

Figure 3- 35: représentation d'une temporisation

Dans STEP 7, on dispose des opérations de temporisations suivantes :

- **S_IMPULS** :

Paramétrer et démarrer temporisation sous forme d'impulsion,

- **S_VIMP** :

Paramétrer et démarrer temporisation sous forme d'impulsion prolongée,

- **S_EVERZ** :

Paramétrer et démarrer temporisation sous forme de retard à la montée,

- **S_SEVERZ** :

Paramétrer et démarrer temporisation sous forme de retard à la montée mémorisé,

- **S_AVERZ** :

Paramétrer et démarrer temporisation sous forme de retard à la retombée,

- ---(SI) Démarrer temporisation sous forme d'impulsion
- ---(SV) Démarrer temporisation sous forme d'impulsion prolongée
- ---(SE) Démarrer temporisation sous forme de retard à la montée
- ---(SS) Démarrer temporisation sous forme de retard à la montée mémorisé
- ---(SA) Démarrer temporisation sous forme de retard à la retombée

On peut charger une valeur de temps prédéfinie en utilisant l'un des deux formats suivants :

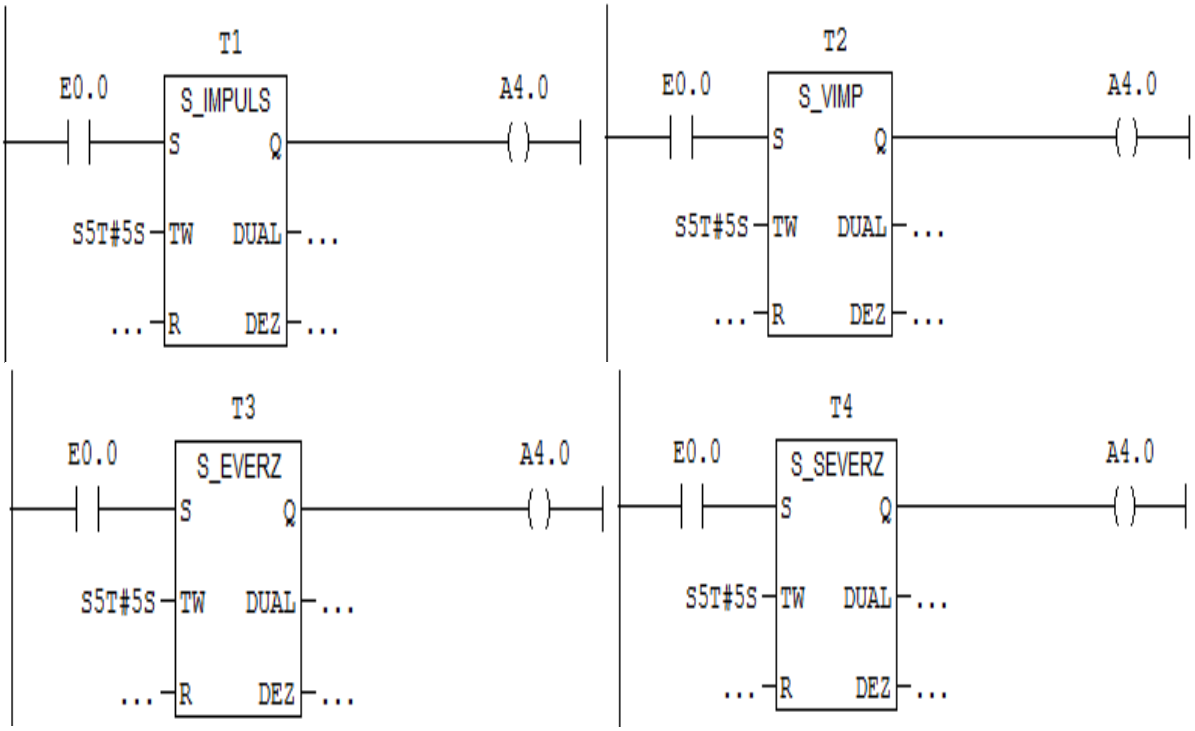
- **w#16#wxyz** : (**w** = base de temps (c'est-à-dire l'intervalle de temps ou la résolution) et **xyz** = valeur de temps en format décimal codé binaire (DCB)).

- **S5T#aH_bM_cS_dMS (**H (heures), M (minutes), S (secondes) et MS (millisecondes); a, b, c, d sont des valeurs définies par l'utilisateur, la base de temps est choisie automatiquement et la valeur est arrondie au nombre inférieur le plus proche avec cette base de temps).

La valeur maximale du temps est égale à 9990 secondes ou 2H_46M_30S.

S5TIME#4S = 4 secondes
S5T#2h_15m = 2 heures et 15 minutes
S5T#1H_12M_18S = 1 heure, 12 minutes et 18 secondes

La vue d'ensemble des cinq types de temporisations est donnée par la figure 3-36 :



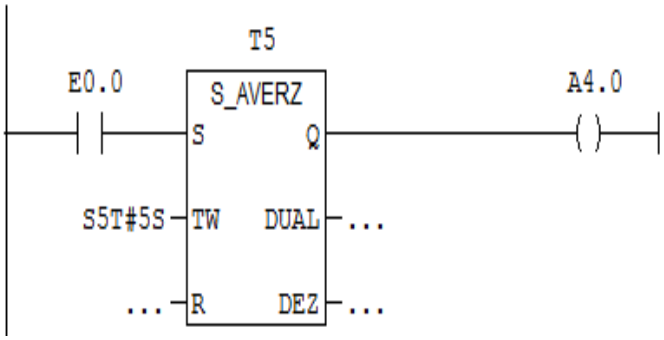


Figure 3- 36: Exemple de programmation des temporisateurs

Les chronogrammes de fonctionnement des cinq temporisateurs sont donnés par la figure 3-37.

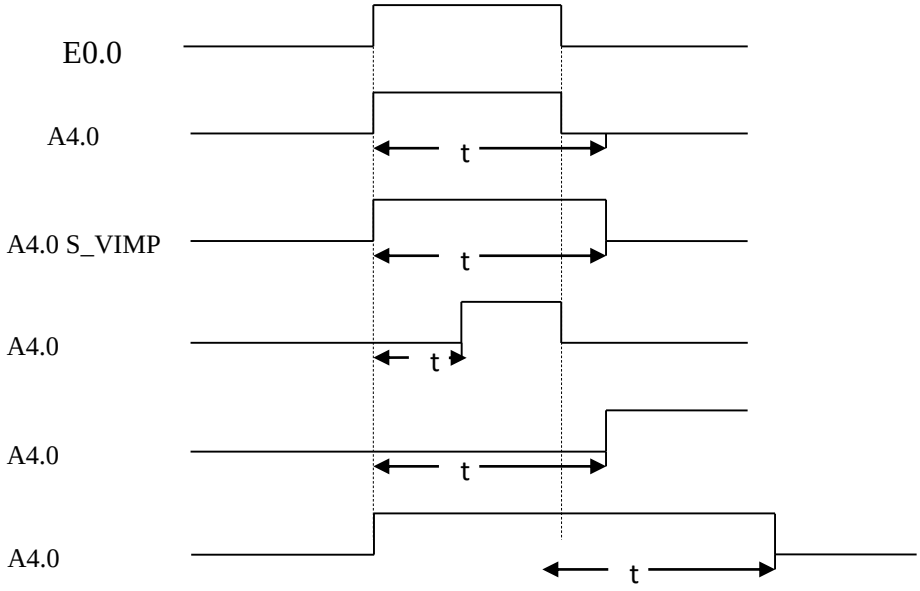


Figure 3- 37: Chronogrammes de fonctionnement des cinq temporisateurs

b) Exemples

Exemple N°1 :

Déclenchement d'une temporisation T5 (S_IMPULS) de durée 2 secondes par action sur l'entrée E0.0. L'état de signal à l'entrée E0.1 remet à zéro le temporisateur. L'état du signal à la sortie A4.0 est 1 tant que la temporisation s'exécute. La configuration de cette temporisation est donnée par la figure 3-38.

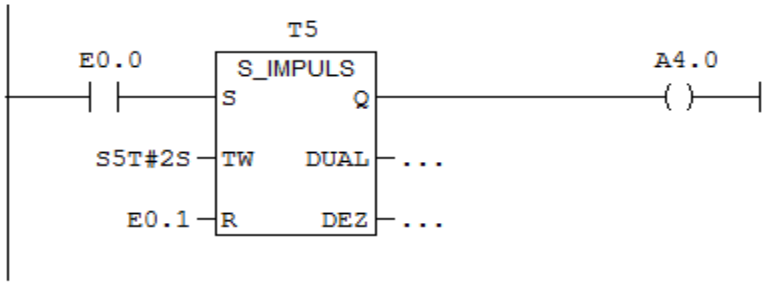


Figure 3- 38: Exemple de programmation d'un temporisateur S_IMPULS

Exemple N°2 :

Déclenchement d'une temporisation T5 (SV) de durée 2 secondes par action sur l'entrée E0.0. L'état de signal à l'entrée E0.1 remet à zéro le temporisateur. L'état du signal à la sortie A4.0 est 1 tant que la temporisation s'exécute. La configuration de cette temporisation est donnée par la figure 3-39.

Figure 3- 39: Exemple de programmation d'un temporisateur SV

8 Exemples de programmation

8-1- Démarrage Etoile-Triangle

On veut réaliser la commande (étoile / triangle) de démarrage d'un moteur asynchrone triphasé. Le moteur démarre en couplage Y pendant 5 s, puis en couplage triangle.

L'entrée **E0.0** (**BP_ DEMARRE**) démarre le moteur, l'arrêt s'obtient avec l'entrée **E0.1** (**BP_ARRÊT**). En sortie, **KM_LIGNE** (sortie **A4.0**), **KM_ETOILE** (sortie **A4.1**) et **KM_TRIANGLE** (sortie **A4.2**).

Ecrire le programme ladder à implanter dans l'automate S7-300 pour commander le moteur.

Pour réaliser ce type de commande avec un API S7-300, on va créer une fonction **FC1** permettant la commande de démarrage du moteur. Les éléments d'entrée/sortie de la fonction **FC1** sont définis comme l'indique la figure 3-40.

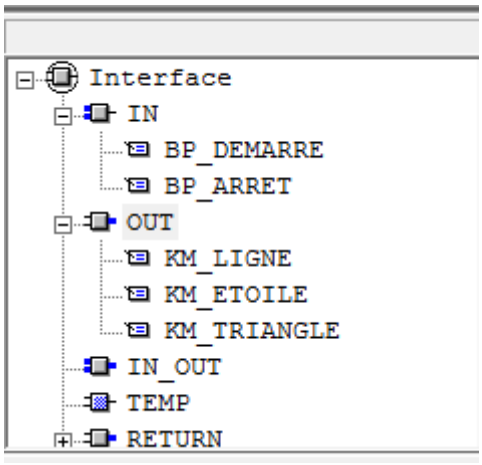


Figure 3- 40: Configuration des variables du bloc fonctionnel FC1

Le programme ladder de la fonction est organisé de la façon suivante :

Le réseau de commande du contacteur de ligne est donné par la figure 3-41.

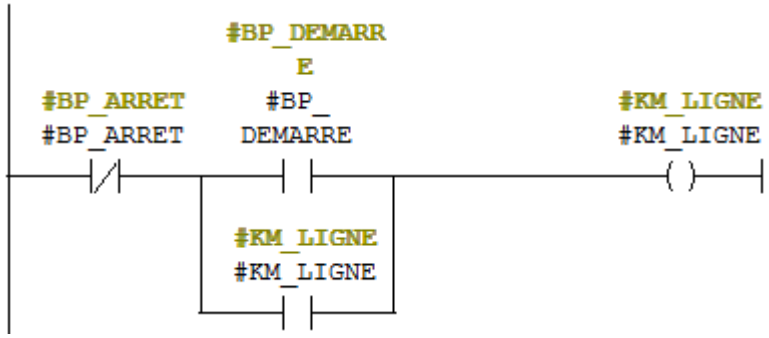


Figure 3- 41: Réseau de commande du contacteur de ligne (KM_Ligne)

Le réseau de commande du contacteur de démarrage en couplage étoile nécessite un temporisateur sous forme d'impulsion prolongée :

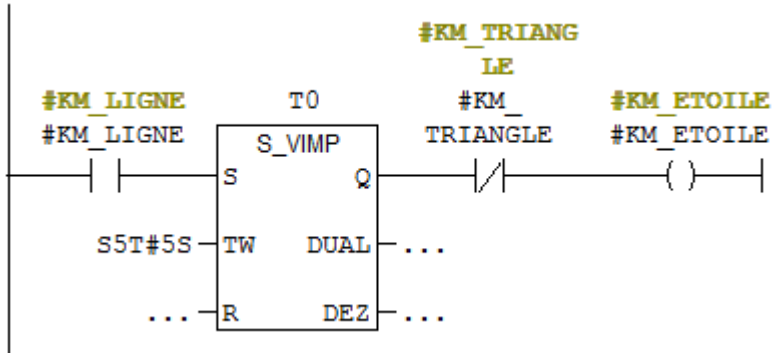


Figure 3- 42: Réseau de commande du contacteur (KM_Etoile)

Le réseau de commande du contacteur de démarrage en couplage triangle est donné par la figure 3-43.

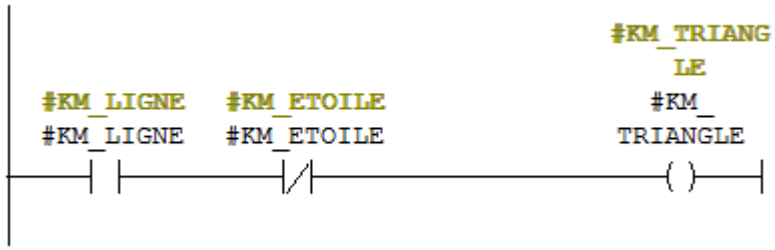


Figure 3- 43: Réseau de commande du contacteur (KM_Triangle)

L'appel de la fonction FC1 se fait selon la figure 3-44.

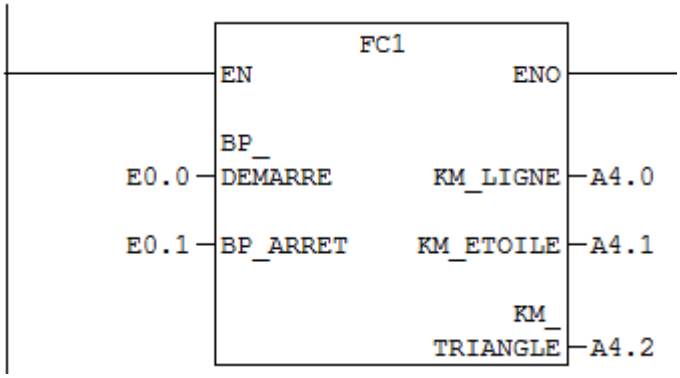


Figure 3- 44: Appel de la fonction FC1

8-2- Démarrage Etoile-Triangle avec restriction du nombre de démarrage (nombre limité par minute)

On veut limiter le nombre de démarrage du moteur de telle sorte qu'il peut effectuer au maximum 3 démarrages par minute. Lorsque le nombre de démarrage par minute dépasse 3, un voyant (**NB_D >3**) connecté sur la sortie **A4.7** s'allume et le système s'arrête (tous les contacteurs se désamorcent). Pour reprendre le fonctionnement normal du système, il faut appuyer sur un interrupteur de **DESARMEMENT** (**E0.7**) et attendre l'écoulement des 60 secondes.

Ecrire le programme ladder à implanter dans l'automate S7-300 pour commander le moteur.

Pour limiter le nombre de démarrage du moteur de telle sorte qu'il peut effectuer au maximum 3 démarrages par minute, on utilise un bloc de comptage **Z0** et un temporisateur **T1** de durée 1 minute.

Le démarrage du moteur est conditionné par le fonctionnement du voyant A4.7.

Pour réaliser cette commande avec un API S7-300, utilise la fonction **FC1** déjà créé dans l'exemple précédent. L'appel de la fonction **FC1** se fait selon la figure 3-45.

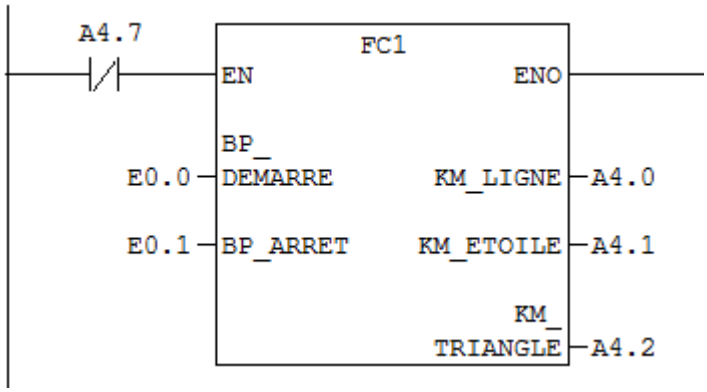


Figure 3- 45: Appel de la fonction FC1 conditionné par A4.7

Le nombre de démarrage est stocké dans la variable **MW10** en utilisant un compteur Z0 comme le montre la figure 3-46.

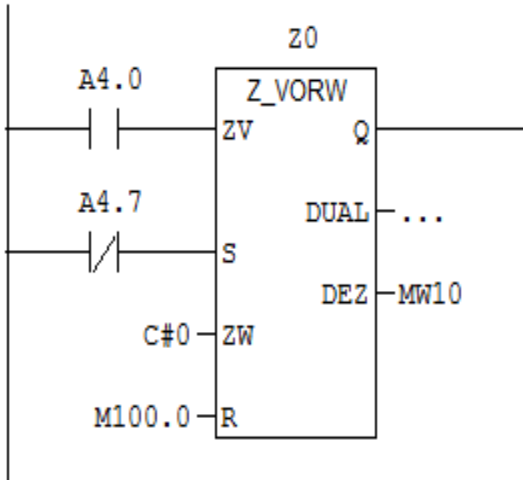


Figure 3- 46: Programme de comptage du nombre de démarrage

Le déclenchement de la temporisation de 1 minute est commandé par le premier démarrage (**MW10 > 0**). Le temporisateur T1 démarre après le premier démarrage du moteur. Le temps en cours est stocké dans la variable **MW20**.

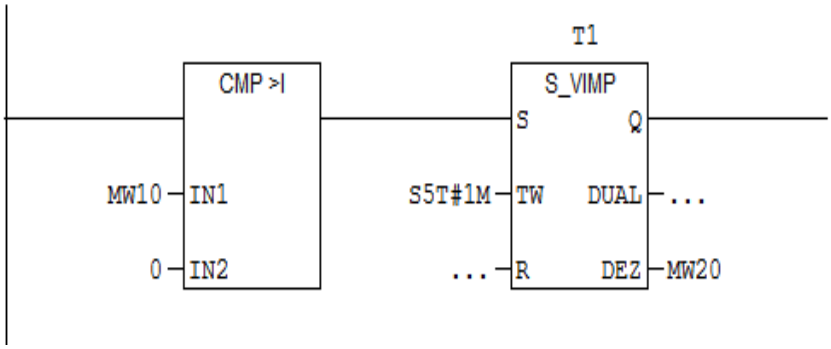


Figure 3- 47: Programme de déclenchement de la temporisation

Le comptage de la temporisation de durée 1 minute (**MW20 = 4097**) est surveillé par le mémento **M100.0** (**M100.0 = 1** lorsque **MW20 = 4097**)

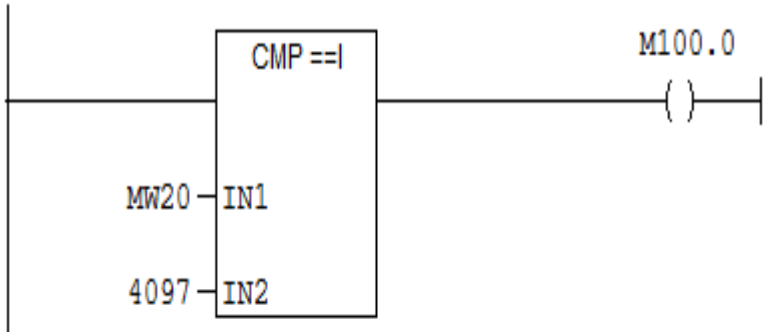


Figure 3- 48: Programme de comptage d'une temporisation de 1 min

Le comptage du nombre de démarrage (**MW10 > 3**) est surveillé par le mémento M100.1

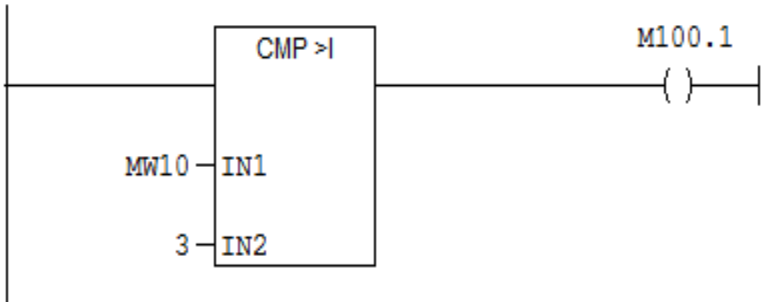
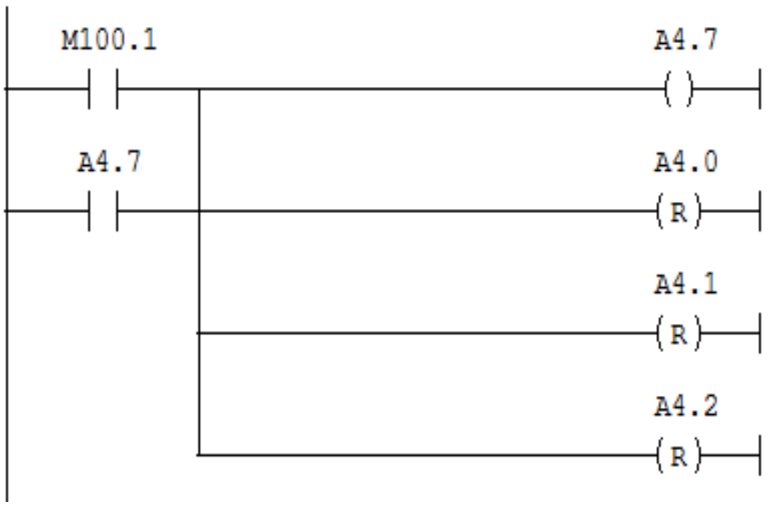


Figure 3- 49: Programme de comparaison du nombre de démarrage à 3

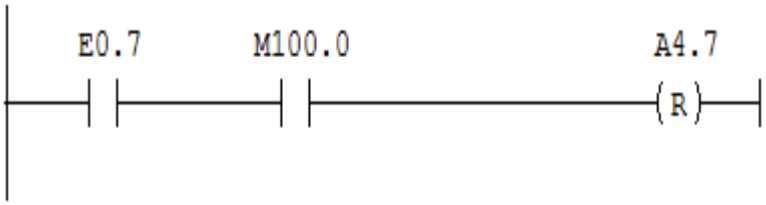
L'activation du voyant NB_D>3 est gérée par le programme donné par la figure 3-50.



(**A4.7** est activé lorsque **M100.1= 1**)
Lorsque **A4.7= 1**, **A4.0** = **A4.1** = **A4.2** = **0**

Figure 3- 50: Programme d'activation du voyant NB_D

Le désarmement du blocage du système est géré par le programme donné par la figure 3-51.



A4.7 = 0 lorsque **E0.7 = 1** et **M100.0 = 1**.

Figure 3- 51: Programme de désarmement du blocage du système

Chapitre IV

MAINTENANCE DES SYSTEMES INDUSTRIELS A BASE D'API

1 Recherche des dysfonctionnements

8.1 Causes de dysfonctionnements

Un dysfonctionnement peut avoir pour origine :

- ✓ un composant mécanique défaillant (pré actionneur, actionneur, détecteur,...).
- ✓ un câblage incorrect ou défaillant (entrées, sorties).
- ✓ un composant électrique ou électronique défectueux (interface d'entrée ou de sortie).
- ✓ une erreur de programmation (affectation d'entrées-sorties, ou d'écriture).
- ✓ un système non initialisé (étape, conditions initiales...).

8.2 Méthode de recherche des causes de dysfonctionnement

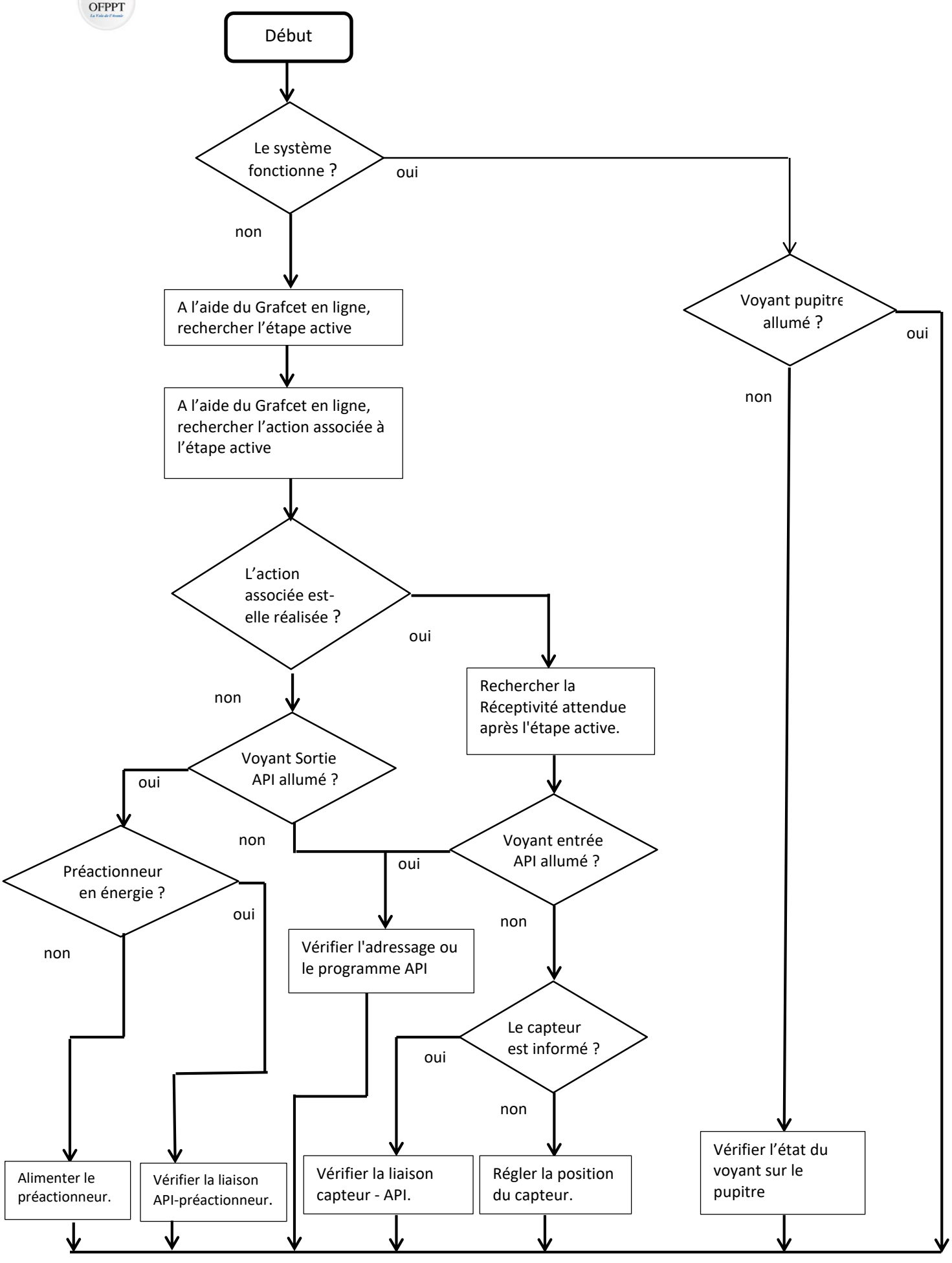
- ✓ Méthode générale de dépannage d'un automate simple commandé par un automate.
  - Tout système est divisé en quatre blocs :
    - Bloc 1 : alimentation ;
    - Bloc 2 : commande ;
    - Bloc 3 : puissance ;
    - Bloc 4 : sortie.
- ✓ Faire une vérification visuelle des composants :
  - Si le composant défectueux est facilement repérable, le remplacer. Faire une vérificationobligatoire des causes du défaut avant de passer à l'étape suivant.
  - Si le composant défectueux est difficilement repérable, aller directement à l'étape suivante.
  - S'il n'y a pas de danger, débranchement de la sortie du système et branchement sur unecharge factice («dummy load ») avant de procéder à un essai de mise en marche.
- ✓ Vérification de l'état des composants durant essai (vue, odorant, ouïe, toucher).
- ✓ Si rien ne fonctionne, vérification des blocs selon l'endroit où le défaut est le plus susceptible de s'être produit, 1, 4, 3, 2.
- ✓ En cas de fonctionnement partiel, vérification des blocs selon l'endroit où le défaut est leplus susceptible de s'être produit, 4, 3, 1, 2.
- ✓ Une fois la réparation effectuée, essai de fonctionnement du système pendant un tempssuffisant pour permettre de conclure que le système est fonctionnel.
- ✓ Application des normes du fabricant (attendre le temps nécessaire pour que tous les appareils entrent en action). Poursuite de l'attente jusqu'à ce que les appareils atteignent leur température de fonctionnement de manière à s'assurer que le système ne tombe pas en panne à cause d'une dérive thermique des composants.

- ✓ S'il faut débrancher un composant, s'assurer que toutes les alimentations sont hors fonction.

- ✓ Les appareils de mesure suggérés pour le dépannage du système automatisé sont : ordinateur, voltmètre, pince ampèremétrique et oscilloscope (éviter le plus possible l'utilisation de l'ampèremètre série pour minimiser les risques de défaut par mauvais branchement). L'ohmmètre devrait être utilisé avec beaucoup de précautions.

- ✓ Utiliser la caractéristique de l'automate en mode :
  - Manuel ;
  - Automatique ;
  - Étape par étape.
- ✓ Décoder les messages d'erreurs

8.3 Organigramme de maintenance

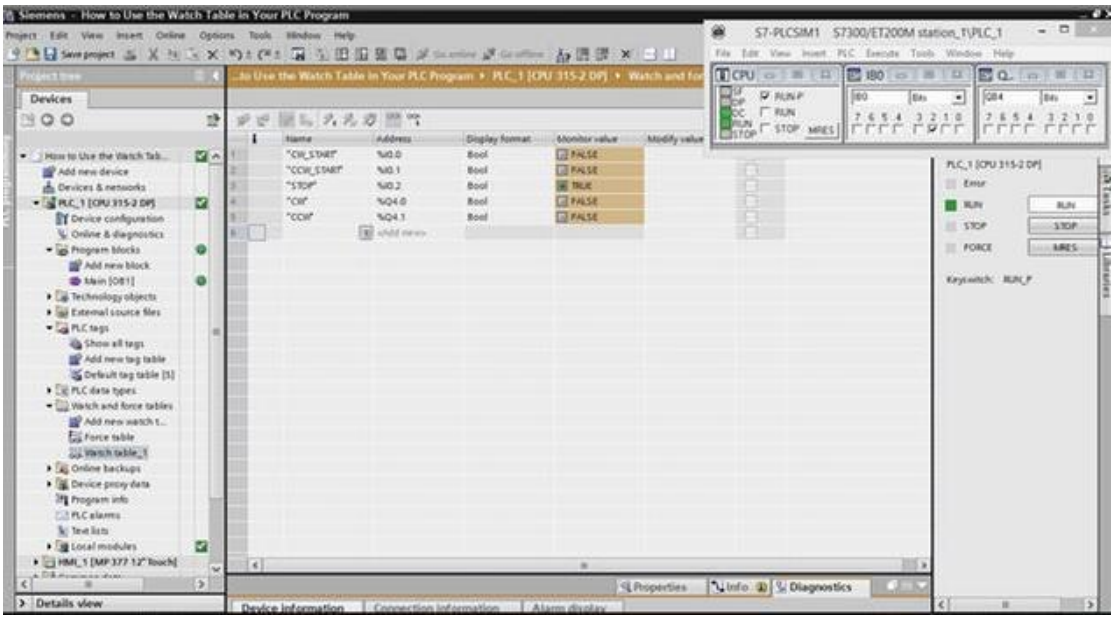

Début
Le système fonctionne ?
oui
non
A l'aide du Grafcet en ligne, rechercher l'étape active
A l'aide du Grafcet en ligne, rechercher l'action associée à l'étape active
L'action associée est-elle réalisée ?
oui
non
Rechercher la Réceptivité attendue après l'étape active.
Voyant Sortie API allumé ?
oui
non
Voyant entrée API allumé ?
oui
non
Préactionneur en énergie ?
oui
non
Vérifier l'adressage ou le programme API
Le capteur est informé ?
oui
non
Voyant pupitre allumé ?
oui
non
Alimenter le préactionneur.
Vérifier la liaison API-préactionneur.
Vérifier la liaison capteur - API.
Régler la position du capteur.
Vérifier l'état du voyant sur le pupitre

2 La vérification du matériel :

Sur une machine automatisée, chaque capteur, interrupteur et bouton poussoir est connecté à une entrée spécifique de l'automate et chaque actionneur (vérin, moteur, voyant) à une sortie spécifique de celui-ci.

Il faudra donc veiller à bien vérifier que chaque capteur/actionneur soit connecté à la bonne entrée/sortie

2.1 Vérification des entrées/sorties par table de forçage



Le forçage des entrées / sorties consiste à mettre à 1 le bit image de ces dernières grâce au logiciel de programmation. On procède au forçage lorsque :

- ✓ On ne dispose pas matériellement des entrées ou sorties. (absence d'un bouton poussoir pour la mise en marche d'un engin, absence d'un contacteur…)
- ✓ On veut déceler les défauts provenant E/S.
- ✓ On veut voir l'évolution de la programmation du processus automatisme avant de passer au câblage.

Le forçage est prioritaire, il est conseillé de l'utiliser avec précaution car il présente certains dangers.

Exemple 1 :

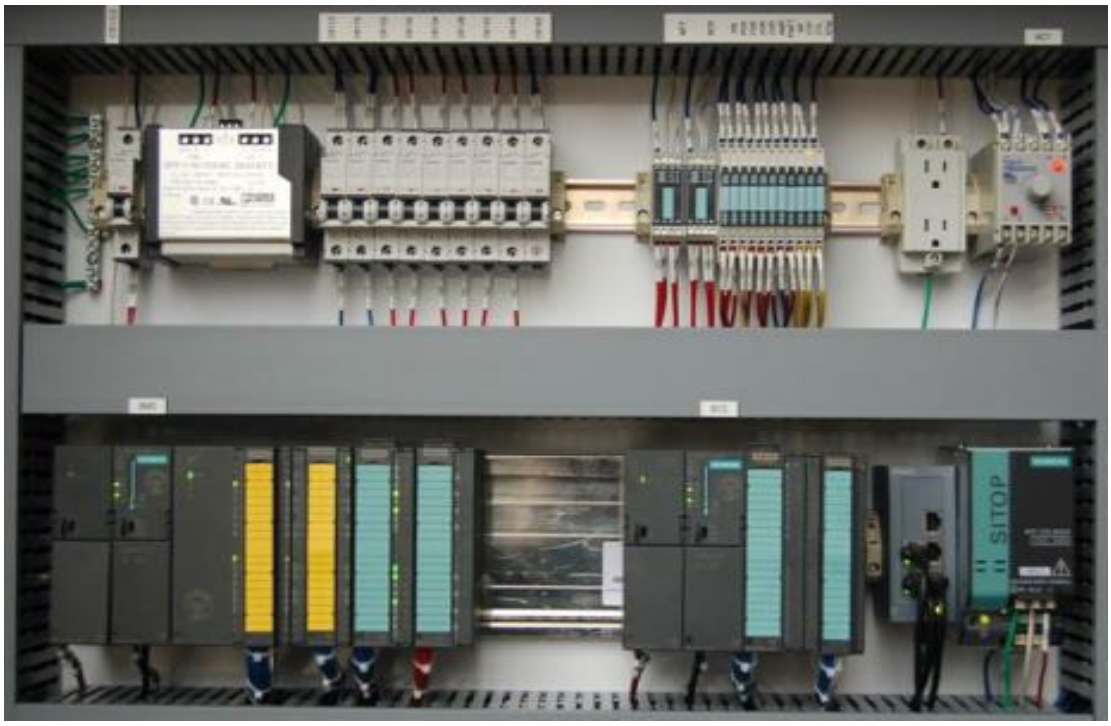
Lorsqu'on force à 1 un sortie automate qui commande l'ouverture d'une vanne évacuant de l'air chaude, celle-ci peut brûler les travailleurs, endommager le matériel autour.

Exemple 2 :

Le forçage d'une entrée commandant la fermeture, automatisée de l'issue de secours,ou la sortie d'un vérin.

Ajoutons que le verrouillage par logiciel s'avère insuffisant comme moyens desécurité. Il est indispensable de procéder à un verrouillage matériel ainsi qu'à la procédure decadenassage parfois car il est plus sécuritaire.

a) Vérification des LED(s) d'entrées TOR de l'API



Les automates sont équipés d'un bloc de visualisation centralisant toutes les informations nécessaires au contrôle, au diagnostic et à la maintenance de l'automate et de ses modules, et des fonctions simples de dialogue opérateur.

- La visualisation de l'état des voies d'entrées / sorties locales ou distantes (entrées / sorties des automates Nano).
- La visualisation des équipements sur le bus AS-i et le diagnostic de ce dernier,
- Le diagnostic des voies ou des modules en défaut.
- La visualisation de données internes :
    - Bits,
    - Chaînes de bits,
    - Chaînes de mots,
    - Variables du programme (étapes actives, informations d'application…)
- Une visualisation numérique multiple sur 4 digits.

Le bloc de visualisation centralisée comprend :

1. Trois ensembles de 32 voyants (DEL) représentant les emplacements des modules implantés dans le bac de base ou le mini bac d'extension.
2. Une ligne d'information formée de voyants (DEL) signalant les modes de fonctionnement de la visualisation.
3. Un bouton-poussoir de commande donnant accès aux différents modes de fonctionnement de la visualisation.
4. Cinq voyants (DEL) :
   - RUN, marche / arrêt de l'automate,
   - TER, trafic sur la prise terminal,
   - I/O, défaut processeur ou application,
   - BAT, défaut ou absence de pile

b) Vérification par multimètre

Vérification du câblage d'une entrée à masse commune

- ✓ Vérifier l'alimentation des entrées à l'aide d'un voltmètre.
- ✓ Pour vérifier le capteur et son câblage, tester aux différents points indiqués à l'aide d'un ohmmètre, contact du capteur ouvert, contact du capteur fermé.
- ✓ Pour vérifier l'interface d'entrée court-circuiter le capteur par un shunt, le voyant d'entrée doit s'allumer.

Vérification du câblage d'une sortie à relais

- ✓ Vérifier que U alimentation existe à l'aide du voltmètre.

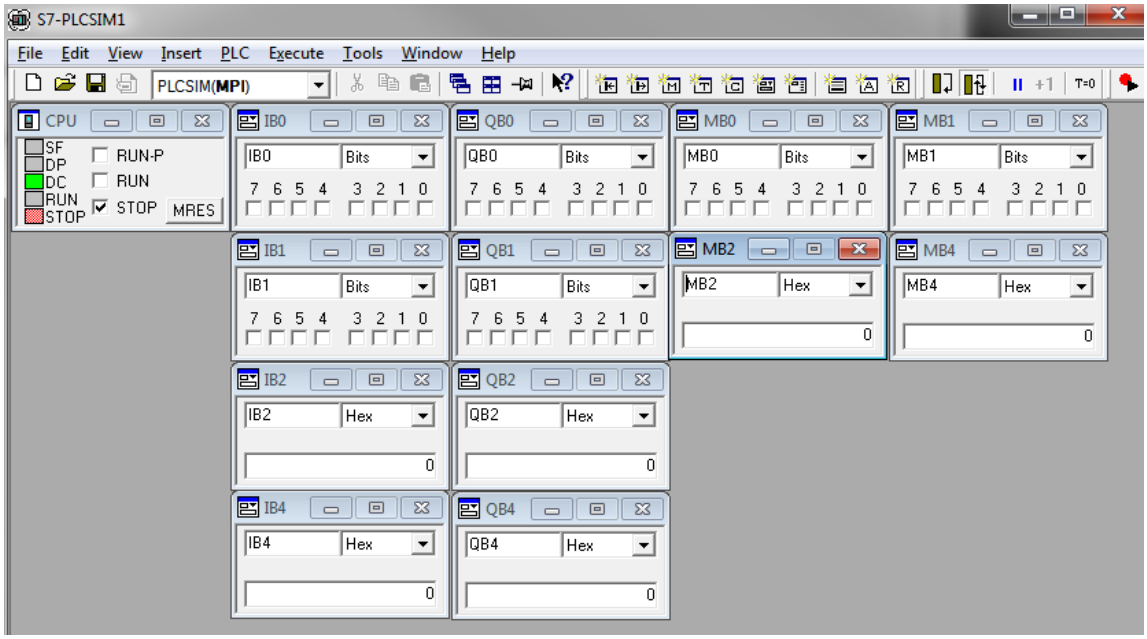
- ✓ Forcer à l'aide du shunt la sortie automate. Si le pré actionneur fonctionne, c'est le module de sortie qui est défectueux. Sinon vérifier le pré actionneur et son câblage.
- ✓ Pour vérifier le câblage tester aux différents points de connexion à l'aide d'un Ohmmètre en laissant le shunt.

2.2 La vérification du programme :

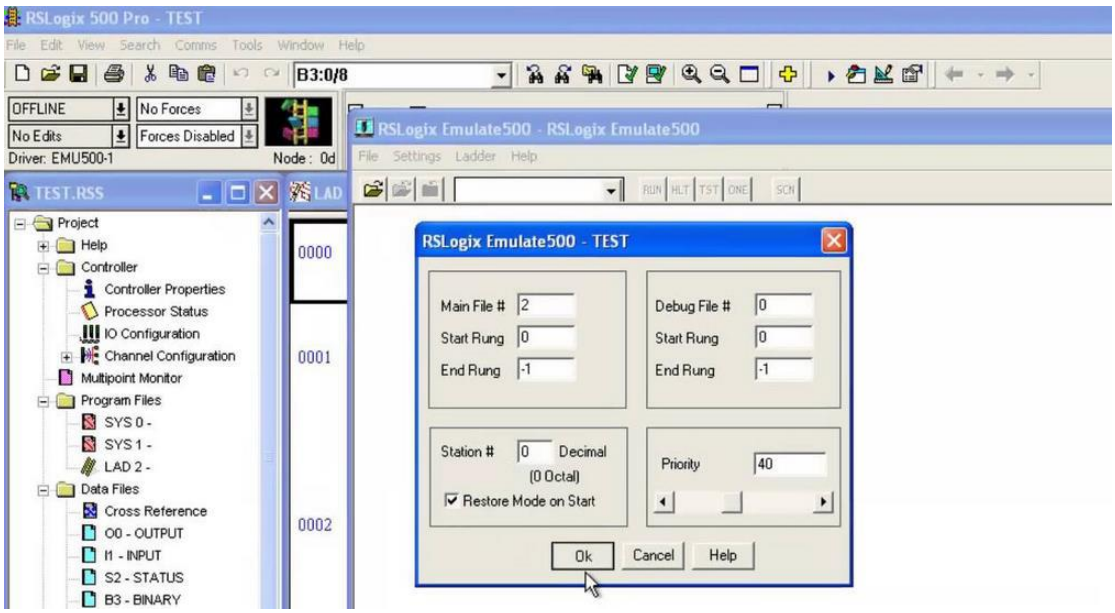
Pendant la phase de vérification du programme, l'automaticien pourra utiliser des outils de simulation afin de détecter tous les défauts émanants du programme.

Si vous utilisez un automate Siemens, vous pourrez utiliser l'application PLCSIM comme automate virtuel. Pour un automate Allen Bradley, vous pourrez utiliser RSEmulate. Ces outils permettent de simuler le fonctionnement d'un automate et de faciliter le test du programme ainsi conçu.

- ✓ ***PLCSIM***



✓ ***RSEmulate***



L'automaticien devra aussi **tester l'interface homme-machine utilisée, vérifier sa bonne configuration et s'assurer de la communication entre les différents dispositifs du système.** Ceux-ci pouvant communiquer via des bus de terrain traditionnels comme le Profibus, le modbus, l'ASI ou encore via les réseaux d'Ethernet industriel.

Après cela, les parties individuelles du programme et les fonctions spéciales du système seront testées: fonctionnement en mode manuel et automatique, configuration et vérification des bases de données d'archivage etc.

On doit confirmer qu'un automatisme commandé par automate est fonctionnel, après avoir lancé l'exécution et vérifier que :

- ✓ L'ensemble fonctionne pendant une durée assez suffisante permettant d'atteindre les températures de fonctionnement afin de s'assurer que le système est fonctionnel et qu'il ne va pas tomber en panne à cause d'une dérive thermique de composants.

- ✓ Si on effectue un essai après avoir changé un ou plusieurs composants défectueux, on vérifie que tout entre en action après le changement effectué.

3 Méthode d'organisation des postes de travail (Méthode 5 S)

La **méthode 5S**, applicable à tout espace de travail (y compris les bureaux), constitue une forme de standardisation des postes de travail, par gestion visuelle.

Elle fait en sorte qu'un employé puisse avoir quotidiennement, à portée de la main, l'information, les outils et le matériel requis pour effectuer sa tâche à son poste de travail, et ainsi travailler efficacement et de façon sécuritaire.

L'objectif est d'instaurer des espaces de travail efficaces qui s'expliquent et se réglementent par eux-mêmes en tout temps, à l'aide de procédures et d'instructions visuelles.

3.1 Les cinq étapes de la méthode 5S

Les 5S correspondent à la première lettre de chacune des cinq étapes de la méthode :

1. **Sélectionner**

Trier tous les objets présents au poste de travail et ne conserver que ceux qui sont utiles au quotidien.

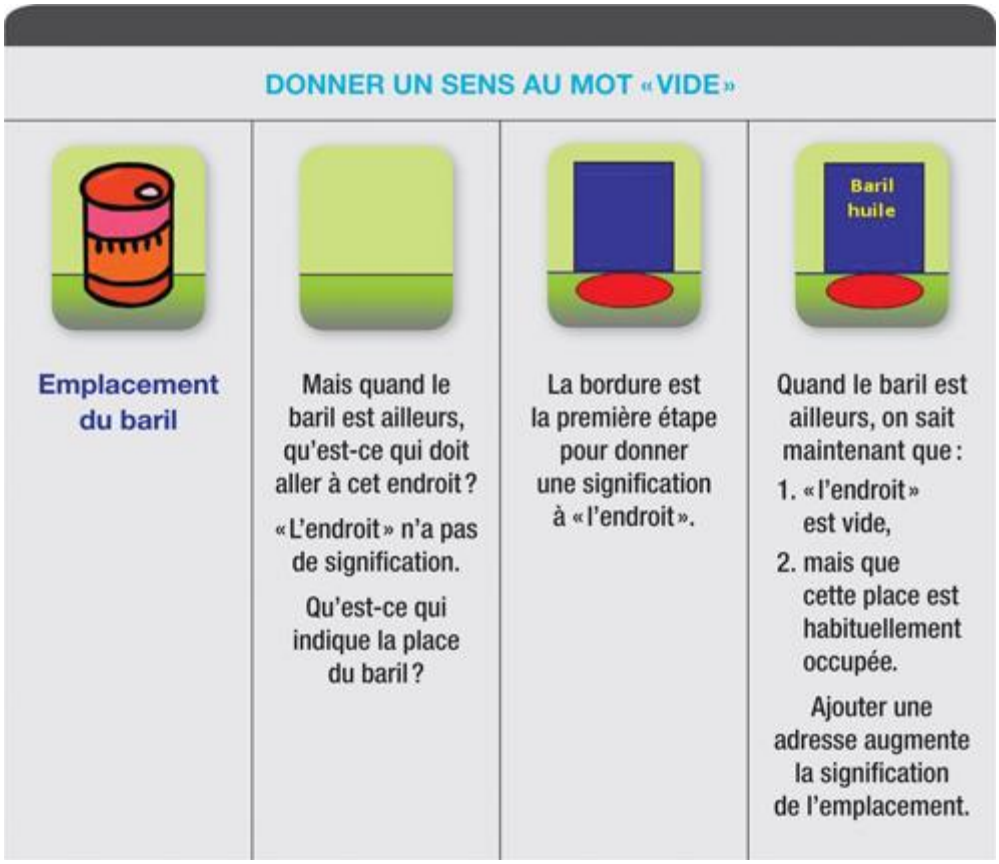
Ce qui n'est utilisé qu'à l'occasion ou ce qui n'ajoute aucune valeur devrait être rangé ailleurs, en dehors de la zone de travail, et ce qui n'a pas servi durant la dernière année et qui ne sera pas utilisé à court terme devrait être éliminé.

Utilisation	Décision à prendre
Jamais	À éliminer de l'aire de travail
Rarement	À mettre à l'extérieur de l'aire de travail
À l'occasion	À mettre dans l'aire de travail
Souvent	À garder près des opérations
Toujours	À garder à portée de la main

2. **Situer**

Déterminer un emplacement, c'est-à-dire réserver une place, pour chaque chose (outil, document, etc.) au poste de travail, de façon à pouvoir trouver la chose nécessaire au moment opportun, en facilitant son accès et son utilisation.

Une place pour chaque chose, chaque chose à sa place. En désignant des emplacements, un espace vide ne constitue plus qu'un simple trou où l'on peut y ranger n'importe quel objet qui traîne sur le plancher



3. Scintiller

Nettoyer, peinturer et s'attaquer aux sources de saleté. Plutôt que de nettoyer de manière répétitive, y aurait-il une méthode de travail plus efficace? Pourrait-on modifier les outils de manière à garder le poste de travail propre?

Il est important de déterminer les causes principales des problèmes de malpropreté afin de prendre les mesures nécessaires pour éliminer ces problèmes à la source.
Une approche efficace pour trouver les vraies causes des problèmes est la technique des « cinq pourquoi ».

Problème	**Solution**
Il y a une tache d'huile sur le plancher.	Enlever l'huile.
Pourquoi? Parce que la machine laisse échapper de l'huile.	Réparer la machine.
Pourquoi? Parce que les joints d'étanchéité sont usés.	Remplacer les joints.

Problème	Solution
Pourquoi? Parce que les joints achetés sont de moins bonne qualité.	Changer les spécifications.
Pourquoi? Parce que les prix obtenus sont concurrentiels.	Modifier la politique d'achat.
Pourquoi? Parce que le responsable des achats est évalué selon les coûts épargnés à court terme.	Changer la politique d'évaluation.

4. Standardiser

Désigner les emplacements et les objets en instaurant des méthodes de travail standard respectées par tous ceux qui travaillent à un même poste de travail et des contrôles visuels afin de pouvoir mesurer la performance des processus.

Par exemple, tous les outils d'un poste de travail pourraient être d'une même couleur (une tache de peinture ou un ruban adhésif de couleur suffisent) : bleu à un poste, jaune ou rouge à un autre.

Notez bien qu'avec les 5S, les outils sont attitrés aux postes de travail et non aux employés.

5. Suivre

Mettre de la rigueur dans le système de manière à ce que les 5S s'intègrent à la culture de l'entreprise.

Une bonne façon de procéder pour implanter les 5S consiste à commencer par un seul poste de travail, idéalement avec un employé volontaire et motivé. Ce poste pourra ensuite servir de modèle pour tous les autres.

3.2 Implanter les 5S

Voici maintenant, en gros, les étapes permettant d'implanter facilement et rapidement les 5S :

1. Faire un grand ménage des postes de travail et éliminer tout ce qui n'a aucune utilité (matières premières et produits défectueux ou invendables, matériel d'emballage inutilisable, etc.).

2. Isoler dans un coin ou désigner clairement à l'aide d'autocollants rouges, par exemple, les objets qui ne sont que peu ou pas utilisés (par exemple ceux n'ayant pas servi au cours de la dernière année).

   Il s'agit ensuite d'attendre quelques jours pour laisser le temps à l'équipe de retirer de ce coin les objets susceptibles de servir à court terme (d'ici un à deux mois) et d'enlever, le cas échéant, les autocollants rouges qui y sont apposés.

   Après cette période de réflexion, tout ce qui est encore dans le coin ou qui porte encore un autocollant rouge devrait être éliminé.

3. Séparer ce qui est utilisé quotidiennement de ce qui est utilisé occasionnellement.

   Les objets utilisés quotidiennement devraient être à portée de la main à chaque poste de travail (quitte à acheter quelques outils additionnels pour que les postes soient complets), alors que ce qui n'est utilisé qu'occasionnellement devrait être placé un peu plus loin, à un endroit connu de tous, de manière à ne pas nuire aux opérations.

   À titre d'exemple, il est possible de garder à un poste de travail servant aux expéditions un stock minimum de matériel d'emballage (cartons, ruban adhésif, etc.), le reste du matériel d'emballage pouvant être placé plus loin.

   Une fois les trois premières étapes terminées, vous aurez réussi à mettre en œuvre le premier « S » (**sélectionner**).

4. Pour mettre en pratique le deuxième « S » (**situer**), il suffit de prendre le temps de désigner une place pour chaque chose à l'intérieur de chaque poste de travail, alors que pour le troisième « S » (**scintiller**), il s'agit simplement de procéder à un nettoyage et de réfléchir à des moyens qui permettraient de réduire les sources de saleté.

5. En ce qui concerne le quatrième « S » (**standardiser**), il consiste habituellement en la rédaction de procédures et d'instructions de travail pour les tâches plus complexes ou susceptibles de causer des erreurs.

   Dans certains cas, de petits aide-mémoire visuels peuvent suffire (par exemple un tableau indiquant à quels endroits entreposer les différents produits), de pair avec une identification visuelle des objets aux postes de travail (par exemple les outils d'un poste de travail peuvent être identifiés par une couleur précise, et les contours des objets peuvent être dessinés de manière à ce que tous sachent facilement à quel endroit va chaque objet).

6. Quant au dernier « S », soit le cinquième (**suivre**), c'est généralement le plus difficile à mettre en pratique puisqu'il s'agit, une fois que les quatre premiers « S » ont été appliqués, de maintenir les lieux de travail propres et en ordre.

Pour ce faire, il est essentiel que les employés réservent un peu de temps à la fin de chaque journée pour ranger leurs outils au bon endroit et, possiblement, un peu plus de temps à la fin de la semaine pour faire le ménage de leur poste de travail.

En plus de réserver du temps au rangement et au ménage, il est tout aussi essentiel de mener des audits de façon régulière afin de s'assurer que le personnel respecte la mise en application des 5S tout en démontrant l'importance que cela revêt pour l'entreprise.

Chapitre V

TRAVAUX DIRIGES / AUTOEVALUATION

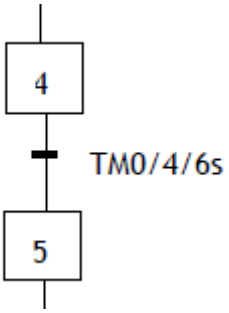
1- Travaux dirigés/Evaluation

1- Quel est le constituant d'un système automatisé qui agit sur la matière d'œuvre?

2- **Sur une ligne de production automatisée, une bouteille non remplie peut être** détectée par :

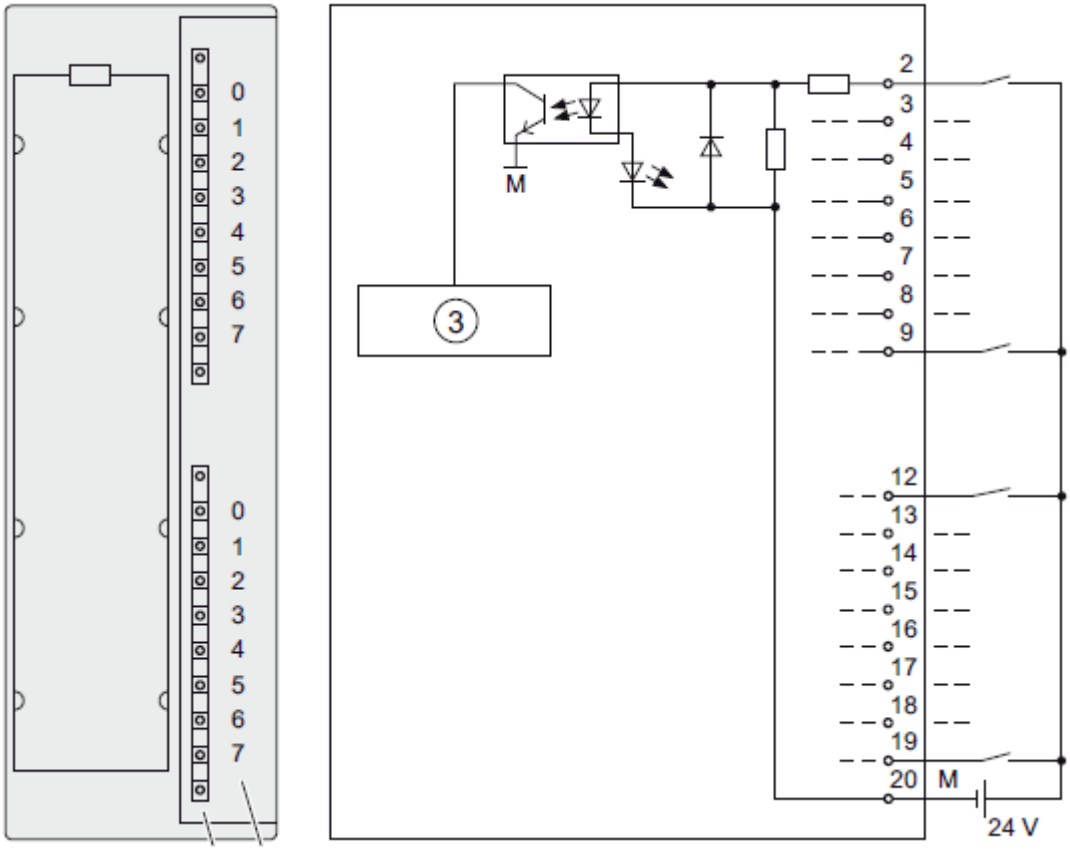
- ✓ Un capteur électromagnétique
- ✓ Un capteur de position
- ✓ Un capteur photo-électrique

3- On donne la séquence Grafcet suivant:



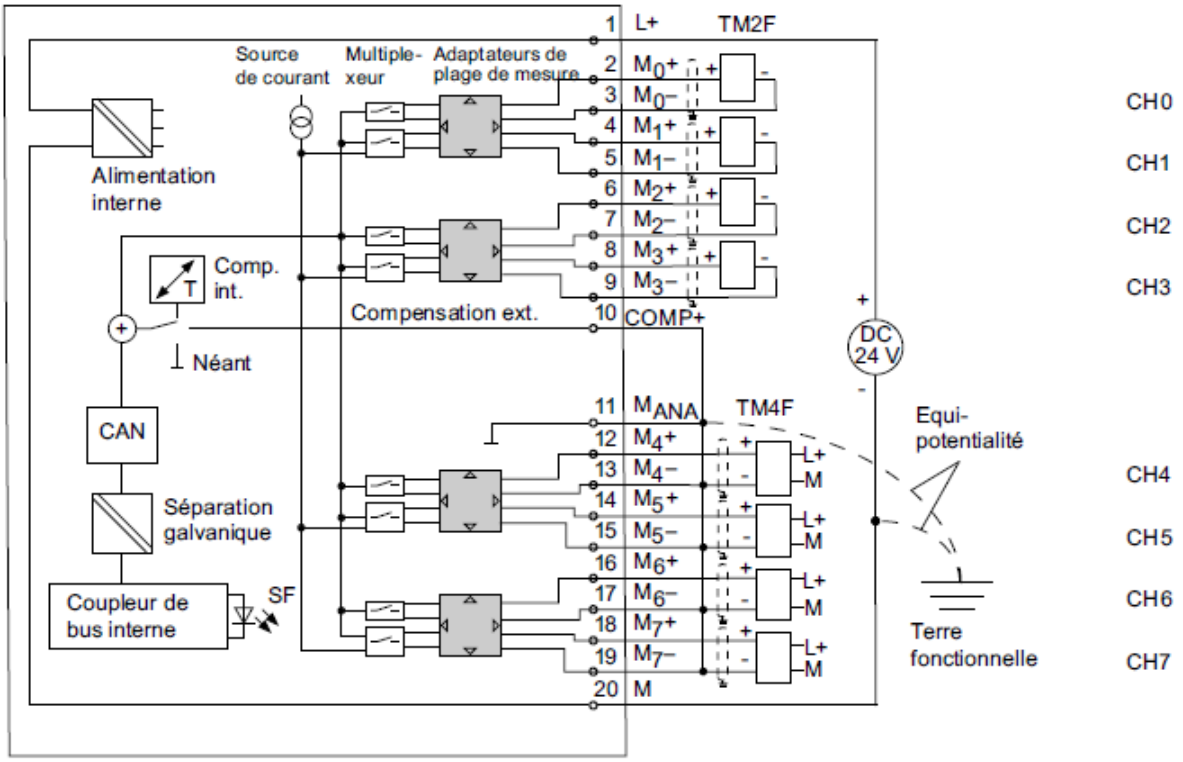
Interpréter cette séquence en langage LADDER et en IL:

4- On donne le schéma de principe d'un module d'entrée logique d'un API S7-300:



a) Combien de voies dispose ce module ?
b) S'agit –il d'une connexion en logique positive ou négative ?
c) Pour la lecture de ce module, l'API utilise le mot binaire %IW0. Sachant que le **%IW0 = 0x100A**, déterminer les états des voies de ce module.

5- On donne le schéma de principe d'un module analogique d'un API S7-300:



a) S'agit –il d'un module d'entrée ou de sortie ?
b) Combien de voies dispose ce module ?
c) Sachant que la plage des adresses de ce module commence à partir de **IW288**. Quelle est l'adresse de la voie CH2.

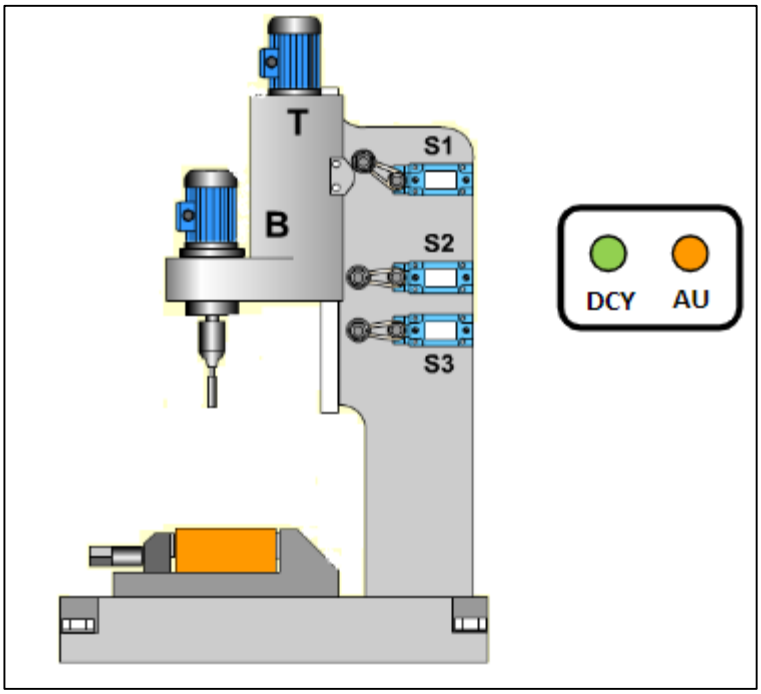
6- En utilisant la fonction de mise à l'échelle d'une entrée analogique présentée dans la figure ci-dessous :

Compléter sur la figure précédente, le programme en langage FBD permettant la mise à l'échelle de deux grandeurs analogiques (la température et la pression) en valeurs exprimées en pourcent (%), sachant que la température et la pression varient de 0% à 100%.

Pour la programmation on considère les variables suivantes :
%**MD30** : **Température _%** et %**MD40** : **Pression_%**
%IW256 : TT1001 et %IW272 :PT1001

7- **Etude d'une station d'usinage**:

Une station d'usinage automatisée commandée par un API est donnée par la figure suivante :



La partie opérative est composée par deux moteurs de type asynchrone triphasé.

- ✓ Le moteur T sert pour la monté et la descente du bloc de perçage,
- ✓ Le moteur B sert pour la rotation de la tête d'usinage (forêt).

Le système est contrôlé par trois capteurs S1, S2 et S3 et commandé par un bouton poussoir DCY pour le démarrage de la machine et un bouton poussoir AU pour l'arrêt d'urgence.

La partie commande est réalisé par un automate programmable.

Le fonctionnement du système est comme suit :

Au repos la tête d'usinage est en haut (capteur S1);
Après appui sur le bouton DCY (départ cycle) le système effectue le cycle suivant:

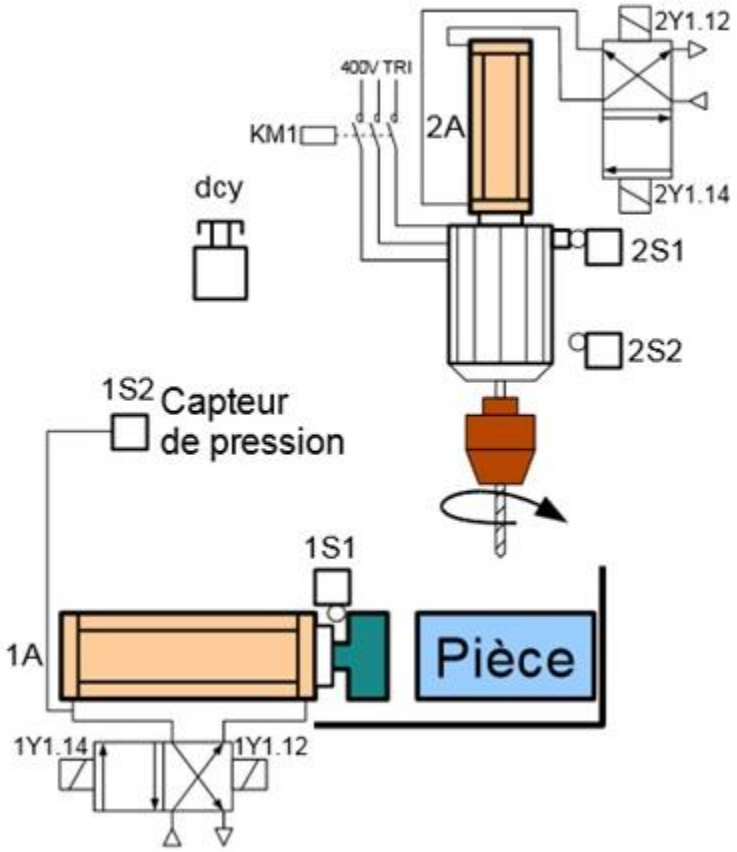
- ✓ De S1 à S2, La tête d'usinage descend en Grande Vitesse (GV).
- ✓ De S2 à S3, La tête d'usinage descend en Petite Vitesse (PV) avec rotation du bloc de perçage.

- ✓ De S3 à S1, La tête d'usinage monte en Grande Vitesse (GV) avec rotation du bloc de perçage.

a) Nommer les capteurs utilisés dans le système,
b) Nommer les actionneurs utilisés dans le système,
c) Etablir le Grafcet de point vue commande qui décrit le fonctionnement du système.
d) Traduire le GRAFCET obtenu par un programme en langage LADER.

8- **Etude d'une station de perçage**:

Une station de perçage automatique est donnée par la figure suivante :



Le fonctionnement du système est comme suit :
Le système est en attente, l'appui sur le bouton de mise en marche **dcy** provoque dans l'ordre :

- ✓ Serrage de la pièce (avance du vérin 1A),
- ✓ Perçage de la pièce (Rotation du moteur pendant la descente et la montée du vérin 2A) et Desserrage de la pièce (Recul du vérin 1A)
- ✓ Le système revient de nouveau à son état d'attente.

a) Nommer les capteurs utilisés dans le système,
b) Nommer les actionneurs utilisés dans le système,

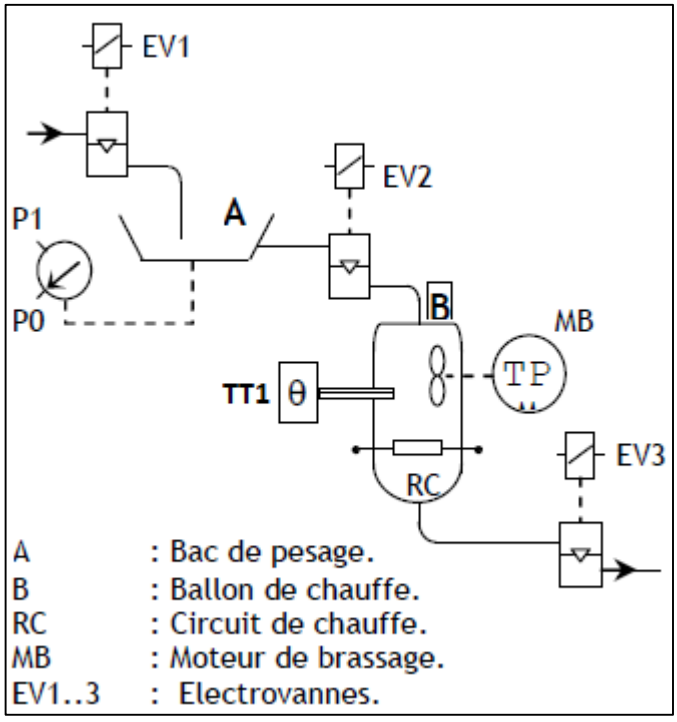
c) Etablir le Grafcet de point vue commande qui décrit le fonctionnement du système.

9- **Etude d'un Malaxeur agroalimentaire**:

Le malaxeur étudié est un système utilisé dans des usines de produits agro-alimentaires. Il décrit le processus de traitement d'un produit liquide assurant le dosage d'une certaine quantité du liquide pour la porter à une température donnée $\mathbf{T_0}$ (°C). Le système est réalisé autour de :

- Un bac de dosage A permettant de peser la quantité du liquide à chauffer ;
- Un ballon de chauffe permettant le chauffage et le brassage (mélange) du liquide pesé.

Le mode de marche du système est cycle par cycle.



Le début de chaque cycle est commandé par l'appui sur le bouton poussoir **Dcy**. Les étapes suivantes sont alors exécutées :

- ✓ **L'ouverture de EV1** autorise le remplissage du bac doseur A jusqu'à une valeur pré affichée P1 du système de pesage.
- ✓ Lorsque P1 est atteinte, on arrête le remplissage et on ouvre EV2 pour autoriser le déversement du liquide du bac vers le ballon de chauffe B.
- ✓ A la fin du déversement (information P0), le circuit de chauffage RC et le moteur de brassage MB sont alimentés.
- ✓ La température de chauffage est contrôlée par le capteur TT1. Lorsque la température T_0 est atteinte, le chauffage et le brassage sont arrêtés et on ouvre EV3 pour autoriser la circulation du liquide chauffé vers la suite du processus.

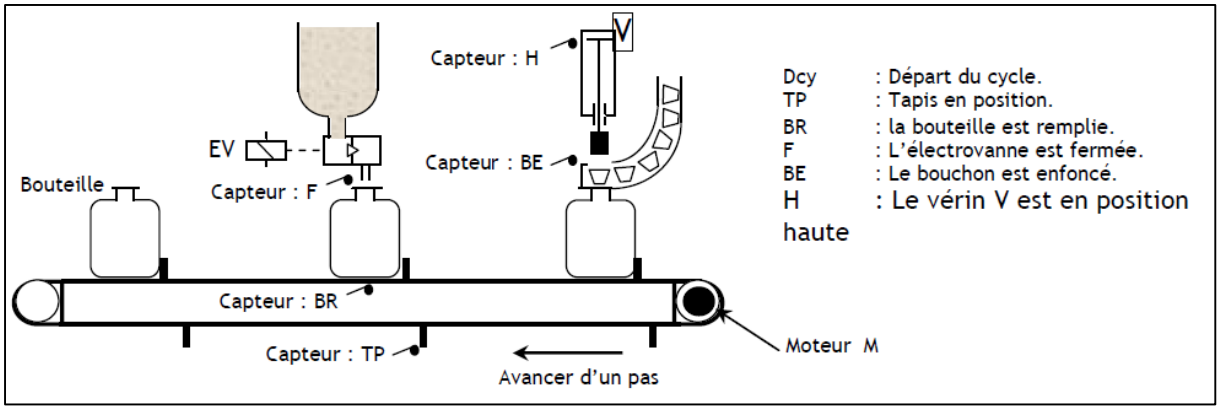
Au bout de 20 secondes, EV3 est désactivée et un nouveau cycle peut commencer.

a) Nommer les capteurs utilisés dans le système,
b) Nommer les actionneurs utilisés dans le système,
c) Etablir le Grafcet de point vue commande qui décrit le fonctionnement du système.

10- **Etude d'une chaîne d'embouteillage**:

Il s'agit d'un système utilisé dans les usines de production des boissons liquides. Il décrit une partie du processus assurant les fonctions de **remplissage** et de **bouchage** des bouteilles.
Le système est réalisé autour de :

- Un **tapis roulant** entrainé par un **moteur M** permettant le déplacement des bouteilles.
- Un **poste de remplissage P1** commandé par l'électrovanne EV.
- Un **poste de bouchage P2** commandé par un vérin presseur V à double effet.



Le déclenchement de la chaîne d'embouteillage se fait par action sur l'interrupteur **Dcy**.

Le moteur (**Avance Tapis** : **M)** tourne d'un pas jusqu'à l'action du capteur **(Tapis en position : TP**). Une bouteille est alors présente à chacun des postes P1 et P2.

Les opérations de remplissage et de bouchage s'effectueront **simultanément** sur les deux bouteilles :

- ✓ Le remplissage se fera en deux étapes :
  - **Ouverture** de l'électrovanne **EV (EV+)** ;
  - **Fermeture** de l'électrovanne **EV (EV-)** après le remplissage de la bouteille. Le capteur (**Bouteille remplie : BR**) permettra de contrôler le niveau de remplissage des bouteilles.
- ✓ Le bouchage se fera en deux étapes :
  - **Descente** du vérin presseur **V (V+)**;

- **Remonte** du vérin **V** (**V-**) après l'enfoncement du bouchon (**Bouchon Enfoncé ; BE**).

Il est à noter que le cycle ne recommencera que si les deux opérations de remplissage et de bouchage sont achevées.

a) Nommer les capteurs utilisés dans le système,
b) Nommer les actionneurs utilisés dans le système,
c) Etablir le Grafcet de point vue commande qui décrit le fonctionnement du système.

2- Bibliographie

[1] Hechmi KHATERCHI, « **Les automates S7-300, Manuel de programmation** », Centre de Publication Universitaire – Tunis 2017

[2] Manuel de programmation : **Logiciel système pour SIMATIC S7-300/400 Conception de programmes**.

[3] Guide de l'utilisateur : **Logiciel de base pour SIMATIC S7 et M7, STEP 7**.

[4] Manuel de référence : **Systèmes d'automatisation S7-300, M7-300**, Caractéristiques des modules.

[5] https://bpmei-prades.com/cours/

[6] http://robert.cireddu.free.fr/

[7] https://slideplayer.fr/slide/11620408/

[8] http://robert.cireddu.free.fr/TP/AII/TP%20AII.html

[9] http://terni.free.fr/cours.html#AUTOM