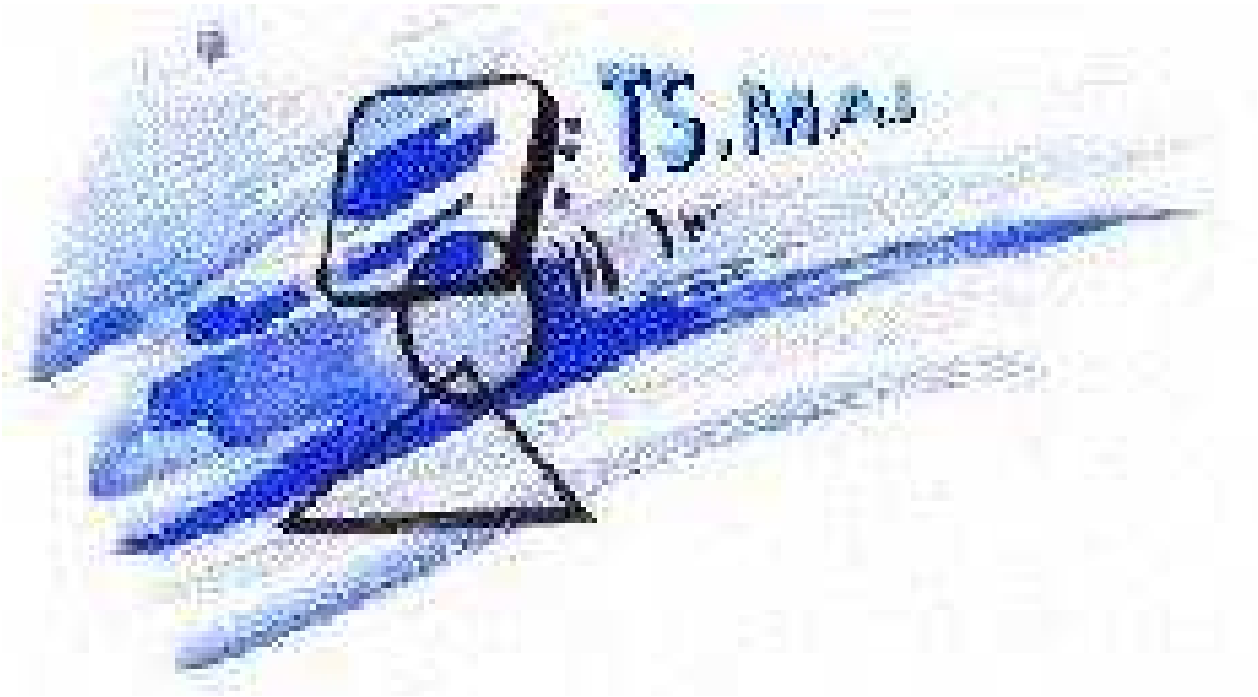
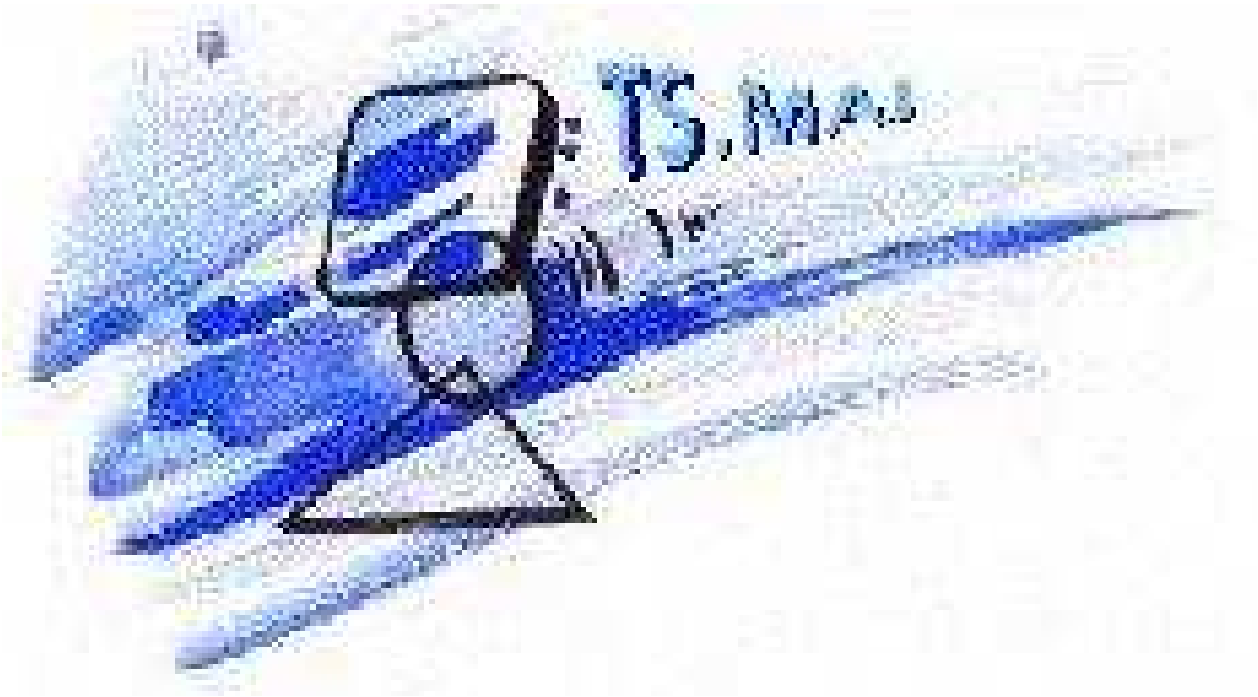


LA PROGRAMMATION

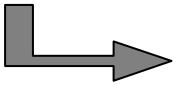


LA NORME CEI 1131

La Norme CEI 1131

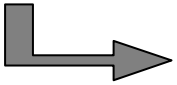
Définitions et informations générales

CEI 1131_1 **Définitions et informations générales**



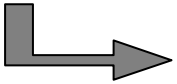
définir et identifier

CEI 1131_2 **Spécifications et essais matériels**



prescrire un minima

CEI 1131_3 **Langages de programmation**



Définir les langages

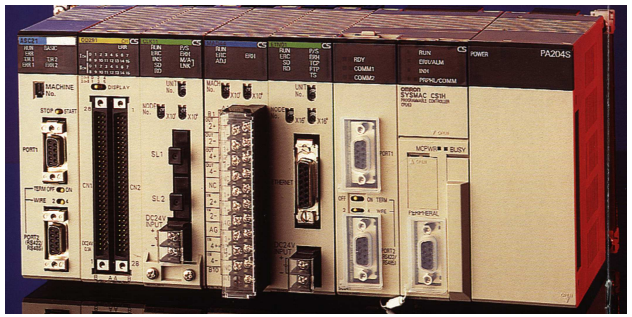
CEI 1131_4 **documentations**

CEI 1131_5 **communications**

La Norme CEI 1131

Définitions et informations générales

L’automate programmable

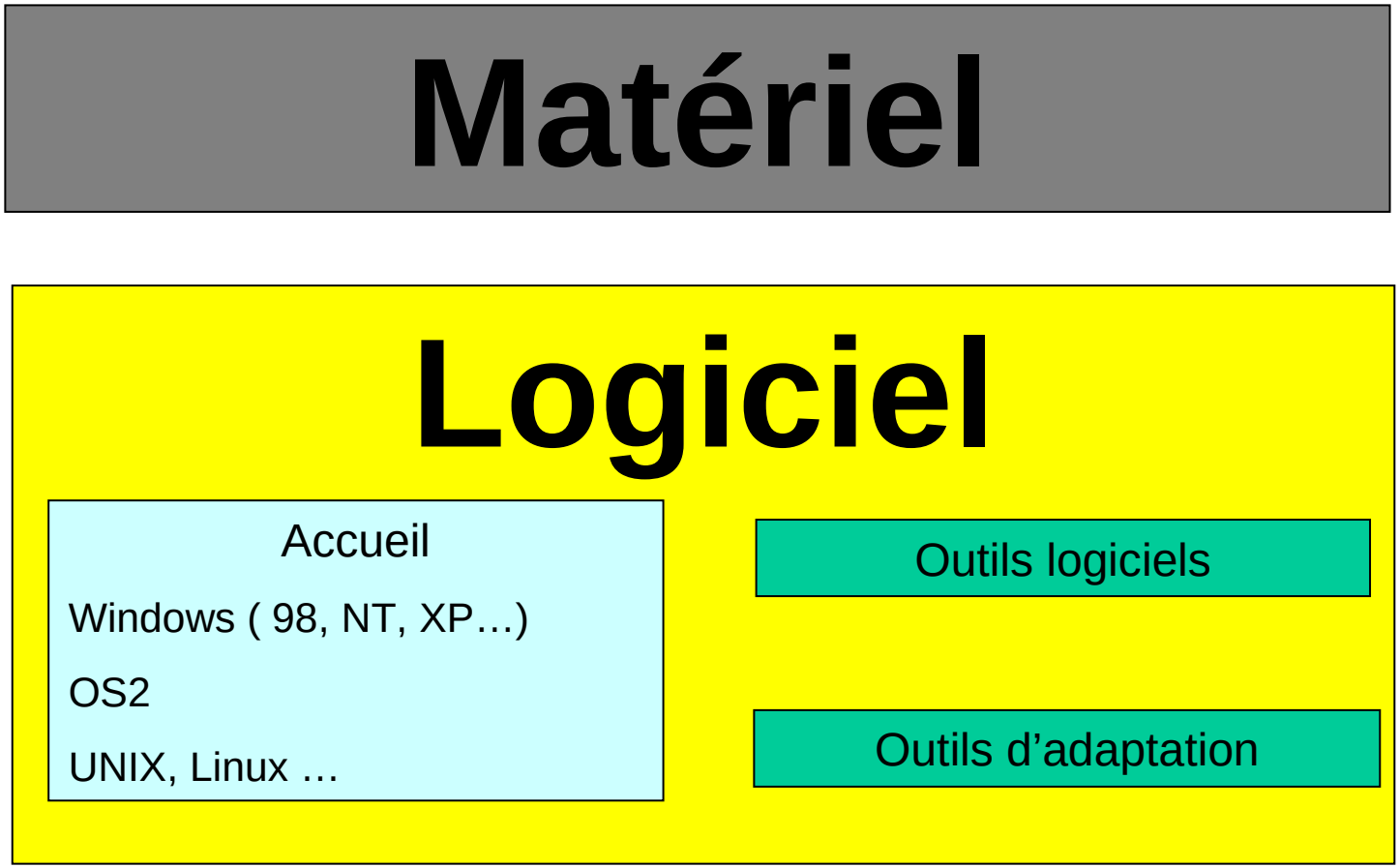


Système électronique fonctionnant de manière numérique, destiné à être utilisé dans un environnement industriel, qui utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur aux fins de mise en oeuvre de fonctions spécifiques, telles que des fonctions de logique, de mise en séquence, de temporisation, de comptage et de calcul arithmétique, pour commander au moyen d'entrées et de sorties tout ou rien ou analogiques divers types de machines ou de processus. L' AP et ses périphériques associés sont conçus pour pouvoir facilement s'intégrer à un système d'automatisme industriel et être facilement utilisés dans toutes leurs fonctions prévues.

La Norme CEI 1131

Définitions et informations générales

L’atelier logiciel de génie automatique

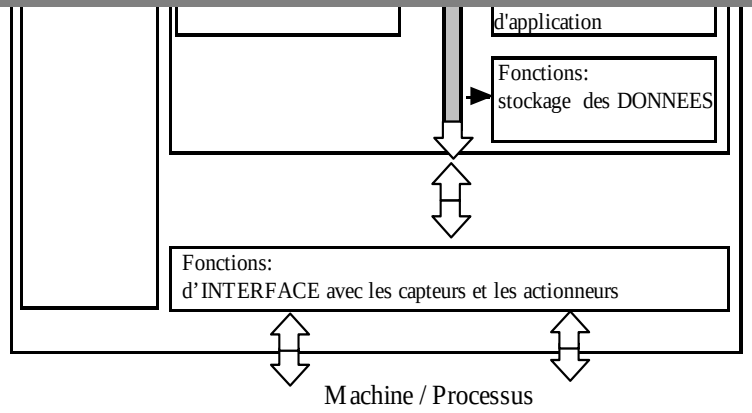


La Norme CEI 1131

Définitions et informations générales

Configuration d'automate programmable

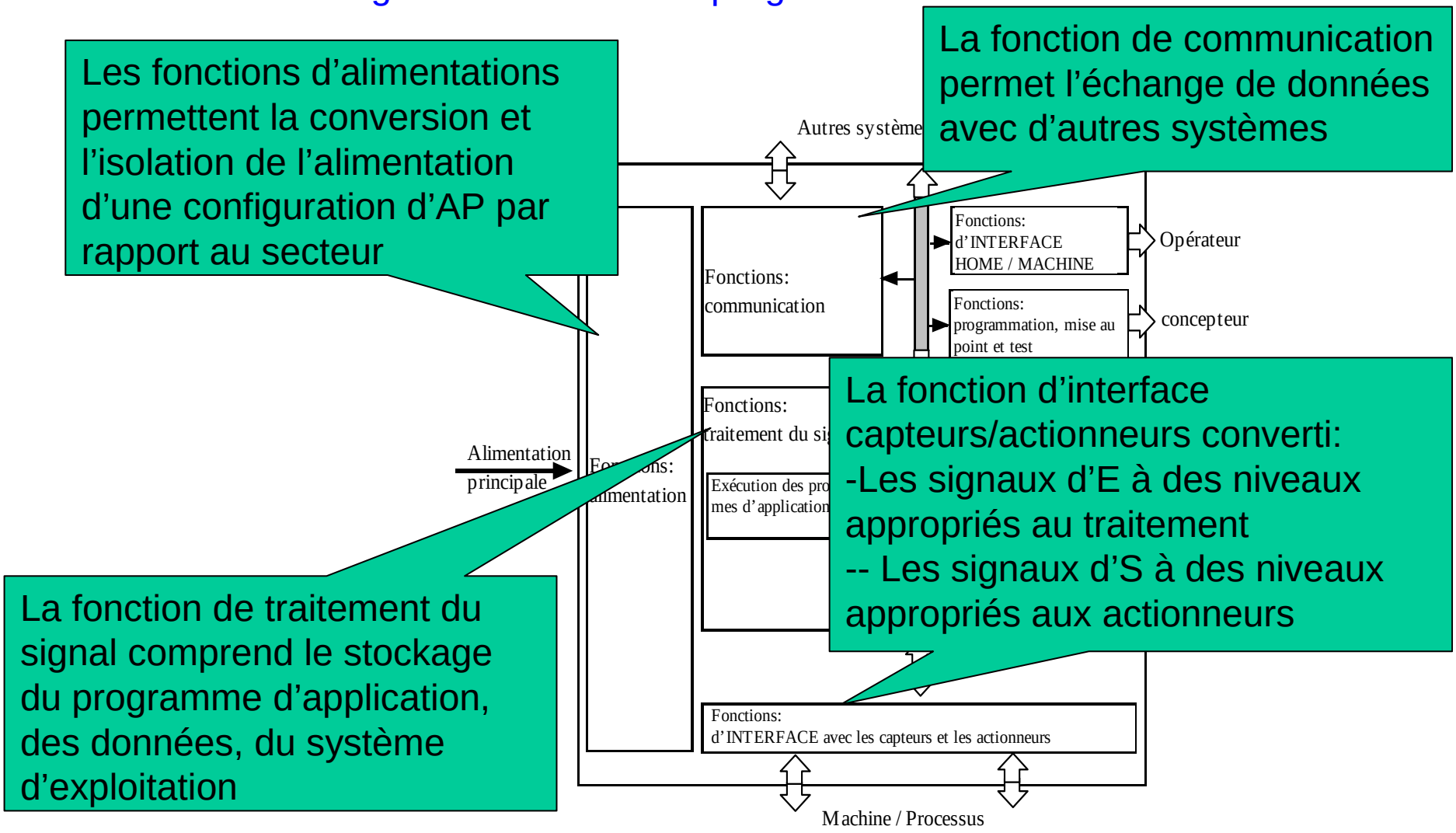
Configuration réalisée par l'utilisateur et comportant un automate programmable et des périphériques associés, nécessaires au système automatisé prévu. Elle consiste en des unités interconnectées au moyen de câbles ou de raccords enfichables pour l'installation permanente, et au moyen de câbles ou d'autres dispositifs pour des périphériques portables et transportables.



La Norme CEI 1131

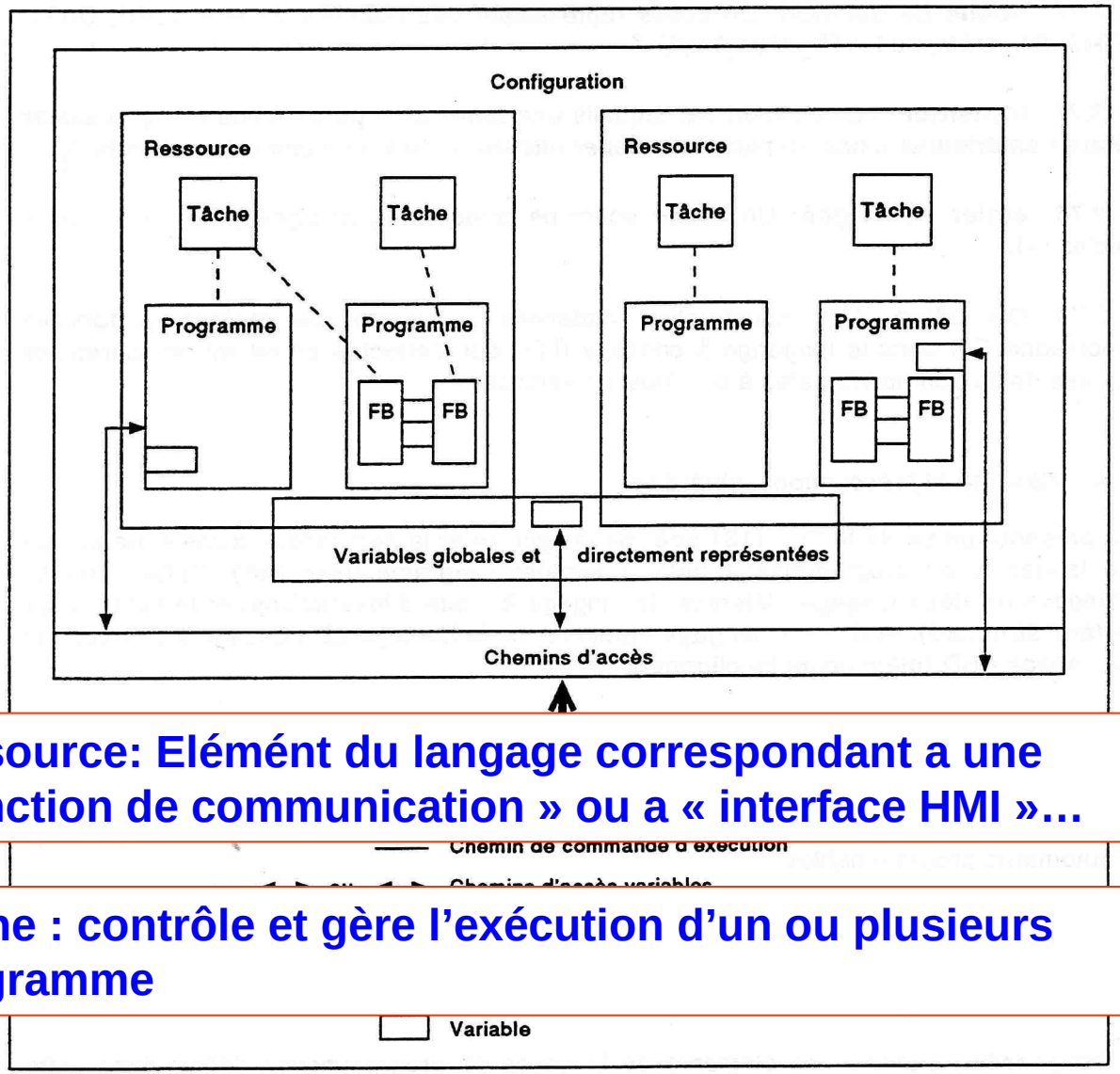
Définitions et informations générales

Configuration d'automate programmable



La Norme CEI 1131

Définitions et informations générales



Ressource: Elémént du langage correspondant a une « fonction de communication » ou a « interface HMI »…

Tâche : contrôle et gère l'exécution d'un ou plusieurs programme

La Norme CEI 1131

Définitions et informations générales

Reprise à froid

Reprise immédiate

Reprise à chaud

Reprise de la configuration d'AP et de son programme d'application après que toutes les données dynamiques {variables telles qu'image *E/S,* mémoires internes, temporisateurs, compteurs, etc. et contextes du programme) aient été ramenées à un état prédéterminé. Une reprise à froid peut être automatique (par exemple après une coupure de courant, une perte d'information dans la ou les parties dynamiques de la ou des mémoires, etc.) ou manuelle (par exemple bouton de réinitialisation, etc.).

La Norme CEI 1131

Définitions et informations générales

Reprise à froid

Reprise immédiate

Reprise à chaud

Reprise après une coupure d'alimentation. avec un ensemble de données dynamiques prédéterminé et programmé par l'utilisateur et un contexte programme d'application prédéterminé par le système. Une reprise à chaud se caractérise par une signalisation d'état ou tout autre moyen apparenté mis à la disposition du programme d'application. indiquant que l'interruption d'alimentation de la configuration d'AP à été détectée en mode run

La Norme CEI 1131

Définitions et informations générales

Reprise à froid

Reprise immédiate

Reprise à chaud

Reprise après une coupure d'alimentation intervenant pendant le laps de temps maximal fonction du processus alloué à la configuration d'AP pour rétablir son fonctionnement comme s'il n'y avait pas eu de coupure d'alimentation. Toutes les informations d'E/S et toutes les autres données dynamiques ainsi que le contexte du programme d'application sont restaurée ou demeurent inchangés.

La Norme CEI 1131

La programmation

Langages LITTERAUX

Langage IL : liste d'instruction

Langage ST : littéral structuré

Langages GRAPHIQUES

Langage LD : langage à contact (LADDER)

Langage FBD : diagramme fonctionnel (logigramme)

Langage SFC

Diagramme fonctionnel en séquence (GRAFCET)

La Norme CEI 1131

La programmation

Eléments communs

Un identificateur est un cordon de lettres, de chiffres et de caractères de soulignement qui doit commencer par une lettre.

L'identificateur ne doit pas comporter de caractère espace (SP)

Exemple:

bp_dcy

cp_avant

Les commentaires de l'utilisateur doivent être respectivement délimités, à leur début et à leur fin, par la combinaison de caractères spéciaux « (* » et « *) ».

Exemple

bp_dcy : (* bouton poussoir NO de départ cycle *)

La Norme CEI 1131

La programmation

les Données

Les données sont définies par des mots clé relatifs à chaque type et par le nombre de bits par élément d'information. Les différents types reconnus par la norme sont:

- types de données élémentaires

N°	Mot clé	Type de donnée	Bits
1	BOOL	Booléen	1

ANY
ANY_NUM
ANY_REAL

Les types de données dérivés, c'est à dire spécifiés par l'utilisateur peuvent être déclarés à l'aide de la construction littérale :

TYPE …… END_TYPE.

Exemple

```
TYPE
        ANALOG_DATA : INT(-4095..4096)
END_TYPE
```

4 Le type générique de tous les autres types dérivés définis au tableau 12 doit être ANY.

La Norme CEI 1131

La programmation

les Variables

Les variables permettent d'identifier des objets de données dont le contenu peut varier, comme par exemple des donnée associées aux E/S ou aux données internes de l'API. Une variable sera déclarée comme appartenant à l'un des types de données

N°	Préfixe	Signification
1	I	Emplacement d'entrée
2	Q	Emplacement de sortie
3	M	Emplacement de mémoire
4	X	Taille d'un seul bit
5	Aucun	Taille d'un seul bit
6	B	Taille d'un octet (8 bits)
7	W	Taille d'un mot (16 bits)
8	D	Taille d'un double mot (32 bits)
9	L	Taille d'un mot long (Quad) (64 bits)

NOTES

1 Sauf déclaration contraire, le type de donnée d'une variable directement adressable, de la taille d'un "seul bit" doit être BOOL.

2 Les organismes nationaux de normalisation peuvent publier des tables de traduction de ces préfixes.

La Norme CEI 1131

La programmation

Déclaration de variables

Mot clé	Utilisation de variables
VAR	Interne à l'unité d'organisation
VAR_INPUT	Fournie de l'extérieur, non modifiable dans l'unité d'organisation
VAR_OUTPUT	Fournie par l'unité d'organisation aux entités externes
VAR_IN_OUT	Fournie par des entités externes Peut être modifiée dans l'unité d'organisation NOTE - Des exemples de l'utilisation de ces variables sont donnés dans les figures 11b et 12.
VAR_EXTERNAL	Fournie par la configuration, par l'intermédiaire de VAR_GLOBAL (2.7.1) Peut être modifiée dans l'unité d'organisation
VAR_GLOBAL	Déclaration de variable globale (2.7.1)
VAR_ACCESS	Déclaration de chemin d'accès (2.7.1)
RETAIN	Variables non volatiles (voir texte précédent)
CONSTANT	Constante (variable qui ne peut être modifiée)
AT	Enoncé d'emplacement (voir 2.4.3.1)
NOTE - L'utilisation de ces mots clés est une caractéristique de l'unité d'organisation de programme ou un élément de configuration dans lequel ils sont utilisés; voir 2.5 et 2.7.	

La Norme CEI 1131

La programmation

Le langage IL

Tableau 51 – Exemples de champs d'instruction

Etiquette	Opérateur	Opérande	Commentaire
START:	LD	%IX1	(* BOUTON POUSSOIR *)
	ANDN	%MX5	(* NON INHIBÉE *)
	ST	%QX2	(* MARCHE VENTILATEUR *)

La Norme CEI 1131

La programmation

Le langage ST

ENONCES

Affectation : A := B

Sélection : IF … THEN … ELSE …. END_IF

CASE … OF ; END_CASE

Itération : FOR … to .. ; END_FOR

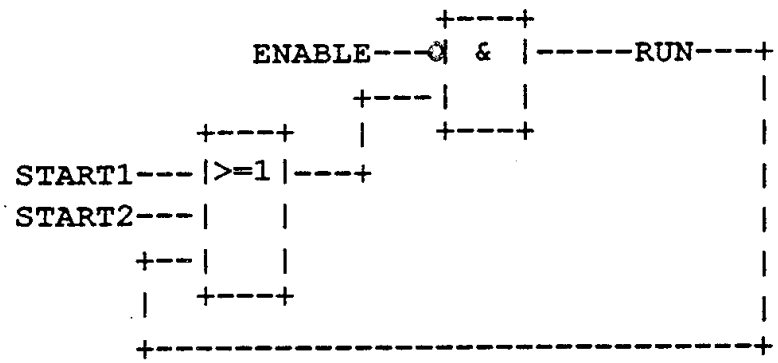
WHILE …. ;END_WHILE

REPEAT … UNTIL ; END_REPEAT

La Norme CEI 1131

La programmation

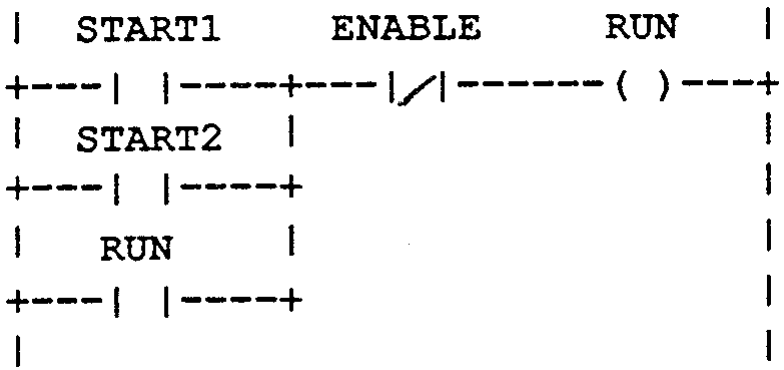
Le langage FBD



La Norme CEI 1131

La programmation

Le langage LD



La Norme CEI 1131

La programmation

La FONCTION

1_ Définition

2 _ Représentation

3 _ Déclaration

La fonction est définie comme un module logiciel pouvant avoir plusieurs variables d'entrées, mais une seule variable de sortie.

Le lancement d'une fonction dotée des mêmes valeurs d'entrées doit toujours donner la même valeur de sortie.

Une fonction n'a pas de mémoire

La Norme CEI 1131

La programmation

Le BLOC FONCTIONNEL

1_ Définition

2 _ Représentation

3 _ Déclaration

Le bloc fonctionnel est défini comme un module logiciel pouvant avoir plusieurs sorties.

Le bloc fonctionnel possède une mémoire interne, les variables internes sont transparentes pour l'utilisateur. Il est possible de créer plusieurs instances d'un bloc fonctionnel, chaque instance étant identifiée par un nom

La Norme CEI 1131

La programmation

Le BLOC FONCTIONNEL

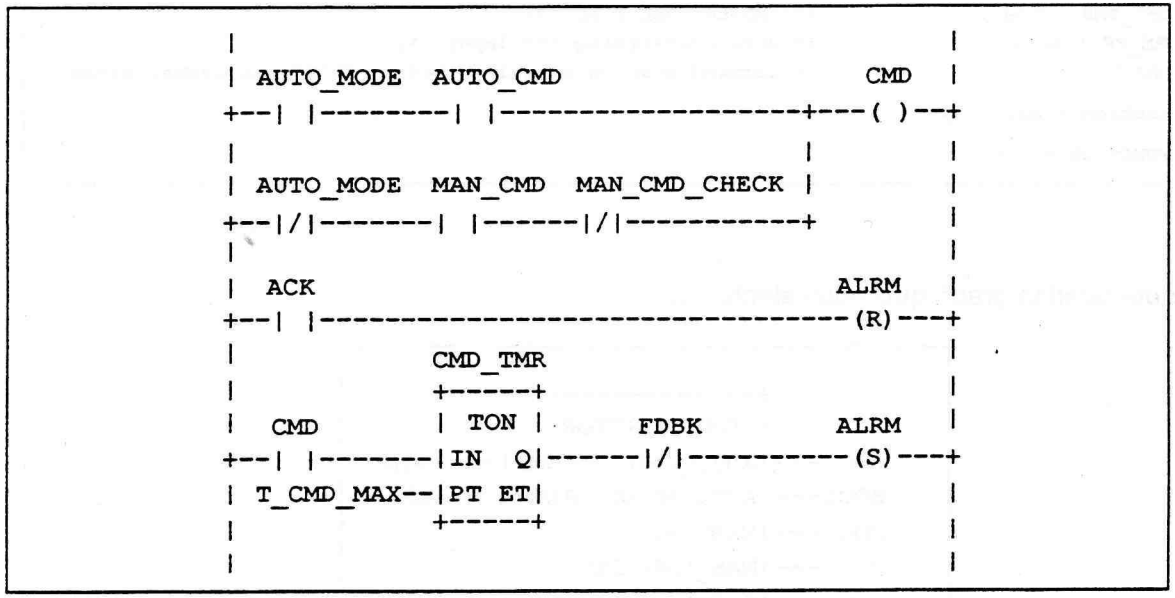
```
FUNCTION_BLOCK CMD_MONITOR
  VAR_INPUT AUTO_CMD : BOOL ;     (* Automated command *)
  AUTO_MODE : BOOL ;              (* AUTO_CMD enable *)
  MAN_CMD : BOOL ;                (* Manual command *)
  MAN_CMD_CHK : BOOL ;            (* Negated MAN_CMD to debounce *)
  T_CMD_MAX : TIME ;              (* Max time from CMD to FDBK *)
  FDBK : BOOL ;                   (* Confirmation of CMD completion by operative unit *)
  ACK : BOOL ;                    (* Acknowledge/cancel ALRM *)
END_VAR
VAR_OUTPUT CMD : BOOL ;           (* Command to operative unit *)
  ALRM : BOOL ;                   (* T_CMD_MAX expired without FDBK *)
END_VAR
VAR CMD_TMR : TON ;               (* CMD-to-FDBK timer *)
  ALRM_FF : SR ;                  (* Note over-riding "S" input: *)
END_VAR                           (* Command must be cancelled before "ACK" can cancel alarm *)
(* Function Block Body *)
END_FUNCTION_BLOCK
```

```
CMD := AUTO_CMD & AUTO_MODE
       OR MAN_CMD & NOT MAN_CMD_CHK & NOT AUTO_MODE ;
CMD_TMR (IN := CMD, PT := T_CMD_MAX) ;
ALRM_FF (S1 := CMD_TMR.Q & NOT FDBK, R := ACK) ;
ALRM := ALRM_FF.Q1 ;
```

La Norme CEI 1131

La programmation

Le BLOC FONCTIONNEL



La Norme CEI 1131

La programmation

Le PROGRAMME

1_ Définition

2 _ Représentation

3 _ Déclaration

Un programme est défini comme:

Un ensemble logique de tous les éléments et constructions des langages de programmations nécessaires pour le traitement des données requis pour contrôler une machine ou un processus au moyen d'une configuration d'automate programmable.

La Norme CEI 1131

La programmation

Phase de conception

ANALYSE

Etude et description d'un point de vue commande, outil : GRAFCET

SPECIFICATIONS

Déclaration des variables et spécifications logicielles dans un langage de la norme

Phase de réalisation

CONFIGURATION

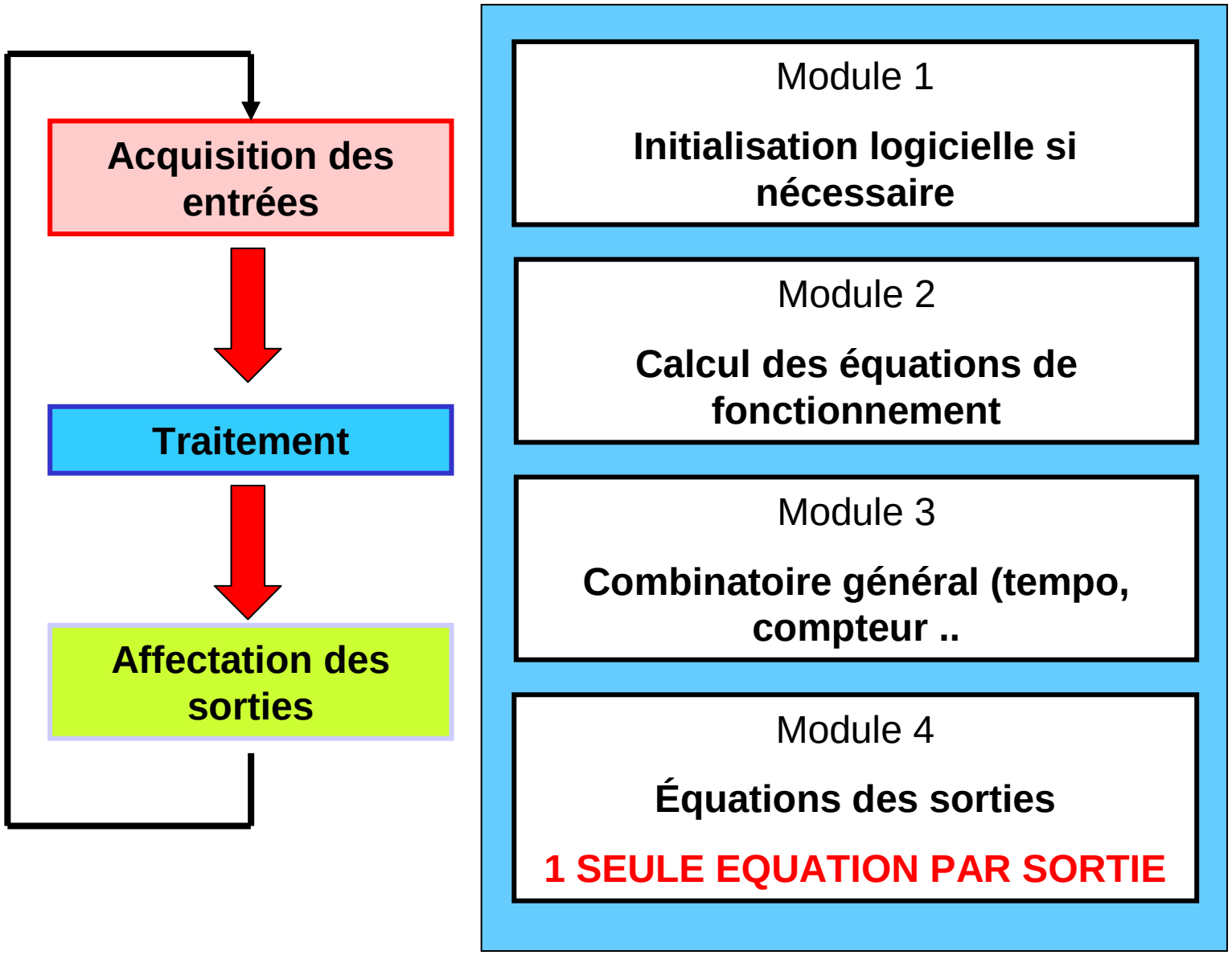
Etude et configuration matérielle de toutes les variables déclarées

CODAGE A.P.I

Transcodage des spécifications logicielles dans le code constructeur

La Norme CEI 1131

La programmation



La Norme CEI 1131

La programmation

F(A>B)=a3./b3+(a3Θb3)a2/b2+(a3Θb3)(a2Θb2)a1./b1+(a3Θb3)(a2Θb2)(a1Θb1)a0./b0

F(A<B)= /a3.b3+(a3Θb3)/a2.b2+(a3Θb3)(a2Θb2)/a1.b1+(a3Θb3)(a2Θb2)(a1Θb1)./a0.b0

F(A=B)= (a3Θb3)(a2Θb2)(a1Θb1)(a0Θb0)

La Norme CEI 1131

La programmation

COMPARATEUR BITS

FONCTION_COMPA
(* déclaration des variables*)

```
VAR_INPUT
a : ARRAY[0..3] OF BOOL        ;        (*valeur des poids binaire de A*)
b : ARRAY[0..3] OF BOOL        ;        (*valeur des poids binaire de B*)
END_VAR

VAR_OUPUT
Sup : BOOL ; (*valeur A>B *)
Inf :  BOOL ; (* valeur A<B *)
Egal : BOOL ; (* valeur A=B *)
END_VAR

VAR
Vi : ARRAY [0..15] OF BOOL ;     (* variables internes utilisées *)
END_VAR
```

La Norme CEI 1131

La programmation

COMPARATEUR BITS

FONCTION_COMPA
(* corps de fonction *)

```
Vi[0] := a[0] XOR b[0]
Vi[1] := a[1] XOR b[1]
Vi[2] := a[2] XOR b[2]
Vi[3] := a[3] XOR b[3]
Vi[4] := (NOT Vi[3]) AND (NOT Vi[2])
Vi[5] := (NOT Vi[4]) AND (NOT Vi[1])
Vi[6] := (NOT Vi[5]) AND (NOT Vi[0])
Vi[7] := (NOT Vi[3] AND NOTb[2] ANDa[2]
Vi[8] := Vi[4] AND NOTb[1] ANDa[1]
Vi[9] := Vi[5] AND NOTb[0] ANDa[0]
Vi[10] := (NOT Vi[3] AND NOTa[2] ANDb[2]
Vi[11] := Vi[4] AND NOTa[1] ANDb[1]
Vi[12] := Vi[5] AND NOTa[0] ANDb[0]

Sup := a[3] AND NOTb[3] OR Vi[7] OR Vi[8] OR Vi[9]
Inf := b[3] AND NOTa[3] OR Vi[10] OR Vi[11] OR Vi[12]
Egal := NOT ( Sup OR Inf )
```

END_FONCTION

La Norme CEI 1131

La programmation

Les blocs fonctionnels

Les blocs fonctionnels Bistables standards

Les blocs fonctionnels standards de détection de front

Les blocs fonctionnels standards de compteurs

Les blocs fonctionnels standards de temporisateurs

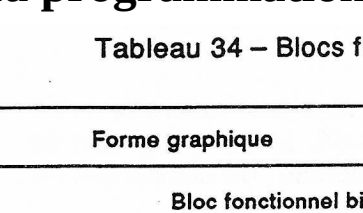
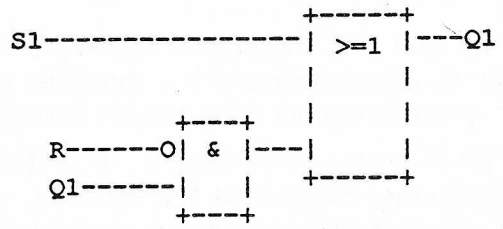
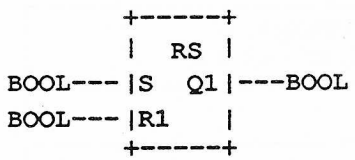
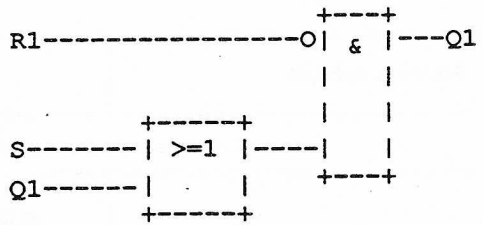
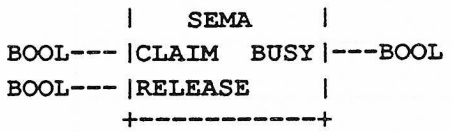
La Norme CEI 1131

La programmation

Les blocs fonctionnels

Bistables standards

Tableau 34 – Blocs fonctionnels bistables standards

N°	Forme graphique	Corps de bloc fonctionnel
1	Bloc fonctionnel bistable (forcé dominant) (notes 1 et 2)	
		
2	Bloc fonctionnel bistable (réinitialisé dominant) (notes 1 et 2)	
		
3	Sémaphore avec "Essai et Positionnement" en continu (notes 3, 4, 5 et 6)	
		VAR X : BOOL := 0 ; END_VAR BUSY := X ; IF CLAIM THEN X := 1 ; ELSIF RELEASE THEN BUSY := 0 ; X := 0 ; END_IF

La Norme CEI 1131

La programmation

Les blocs fonctionnels

Les blocs fonctionnels Bistables standards

Les blocs fonctionnels standards de détection de front

Les blocs fonctionnels standards de compteurs

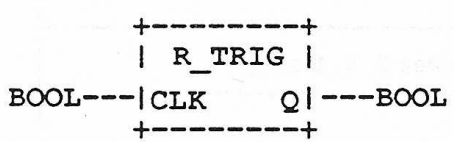
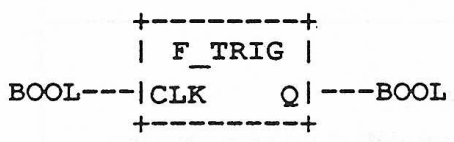
Les blocs fonctionnels standards de temporisateurs

La Norme CEI 1131

La programmation

Les blocs fonctionnels standards de détection de front

Tableau 35 – Blocs fonctionnels standards de détection de fronts

N°	Forme graphique	Définition (langage ST – voir 3.3)
1	Détecteur de front montant	
		FUNCTION_BLOCK R_TRIG VAR_INPUT CLK : BOOL ; END_VAR VAR_OUTPUT Q : BOOL ; END_VAR VAR M : BOOL := 0 ; END_VAR Q := CLK AND NOT M ; M := CLK ; END_FUNCTION_BLOCK
2	Détecteur de front descendant	
		FUNCTION_BLOCK F_TRIG VAR_INPUT CLK : BOOL ; END_VAR VAR_OUTPUT Q : BOOL ; END_VAR VAR M : BOOL := 1 ; END_VAR Q := NOT CLK AND NOT M ; M := NOT CLK ; END_FUNCTION_BLOCK

La Norme CEI 1131

La programmation

Les blocs fonctionnels

Les blocs fonctionnels Bistables standards

Les blocs fonctionnels standards de détection de front

Les blocs fonctionnels standards de compteurs

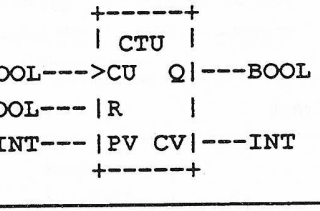
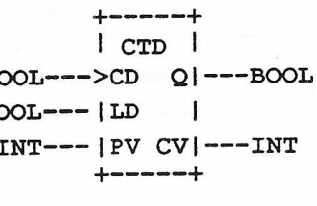
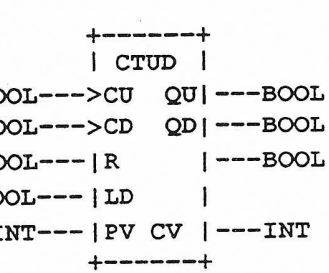
Les blocs fonctionnels standards de temporisateurs

La Norme CEI 1131

La programmation

Les blocs fonctionnels standards de compteurs

Tableau 36 – Blocs fonctionnels standards de pteurs

N°	Forme graphique	Corps de bloc fonctionnel (langage ST – voir 3.3)
1	Compteur (comptage)	
		IF R THEN CV := 0 ; ELSIF CU AND (CV < PVmax) THEN CV := CV+1 ; END_IF ; Q := (CV >= PV) ;
2	Compteur (décomptage)	
		IF LD THEN CV := PV ; ELSIF CD AND (CV > PVmin) THEN CV := CV-1 ; END_IF ; Q := (CV <= 0) ;
3	Compteur (comptage-décomptage)	
		IF R THEN CV := 0 ; ELSIF LD THEN CV := PV ; ELSIF CU AND (CV < PVmax) THEN CV := CV+1 ; ELSIF CD AND (CV > PVmin) THEN CV := CV-1 ; END_IF ; QU := (CV >= PV) ; QD := (CV <= 0) ;

NOTE - Les valeurs numériques des variables limites PVmin et PVmax sont propres à l'application concernée.

La Norme CEI 1131

La programmation

Les blocs fonctionnels

Les blocs fonctionnels Bistables standards

Les blocs fonctionnels standards de détection de front

Les blocs fonctionnels standards de compteurs

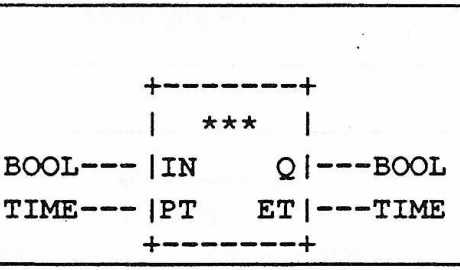
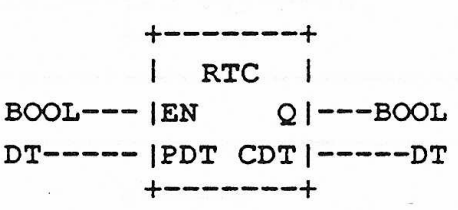
Les blocs fonctionnels standards de temporisateurs

La Norme CEI 1131

La programmation

Les blocs fonctionnels standards de temporisateurs

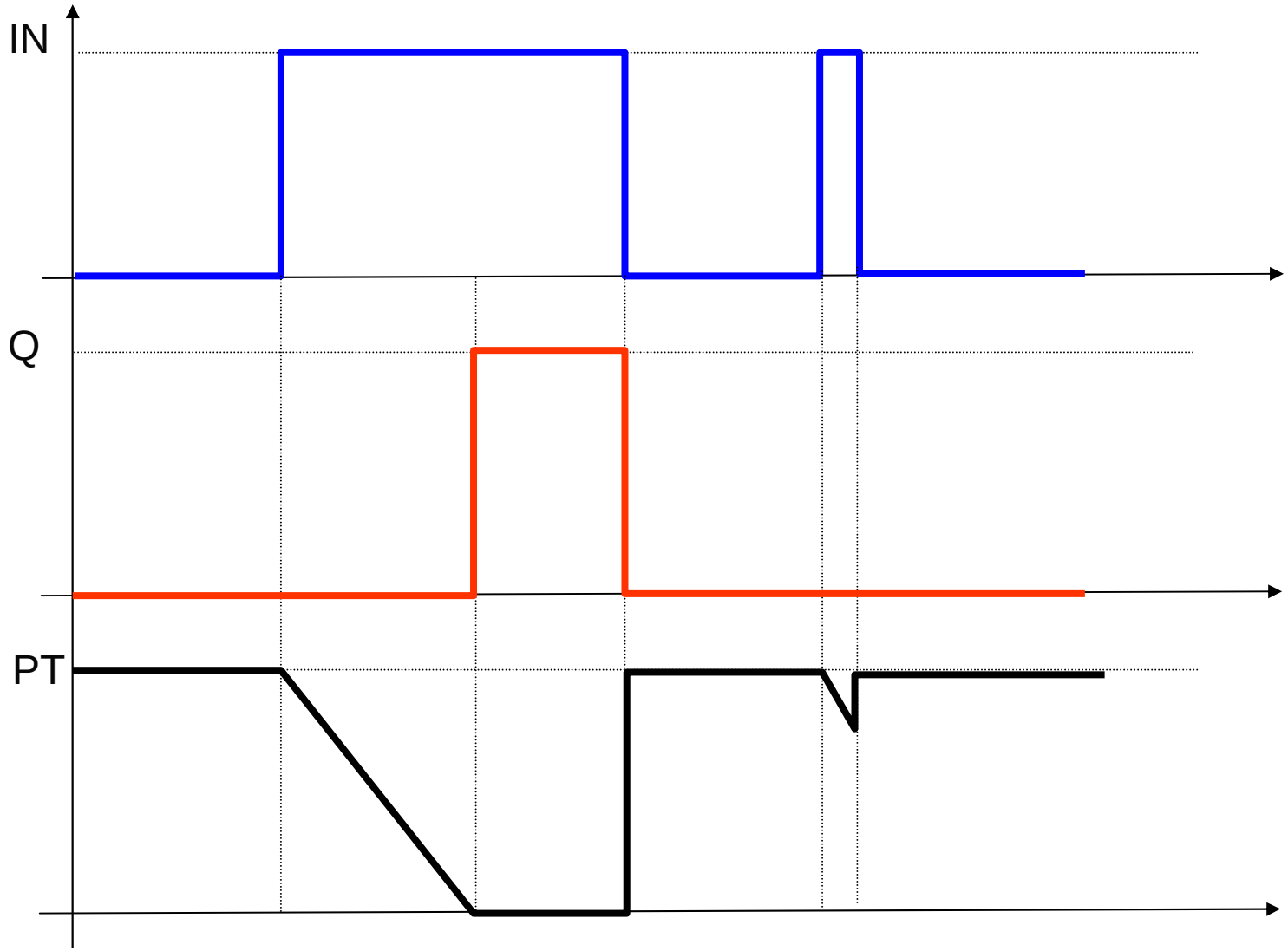
Tableau 37 – Blocs fonctionnels standards de temporisateurs

N°	Description	Forme graphique
1	***est: TP (impulsion)	
2a	TON (Enclenchement)	
2b	T---0 (Enclenchement)	
3a	TOF (Déclenchement)	
3b	0---T (Déclenchement)	
4	Horloge temps réel	
	PDT = Date et heure prédéfinies, chargées sur le front montant de EN CDT = Date et heure du jour, valables lorsque EN=1 Q = copie de EN	
NOTE - Dans les langages littéraux, les caractéristiques 2b et 3b ne doivent pas être utilisées.		

La Norme CEI 1131

La programmation

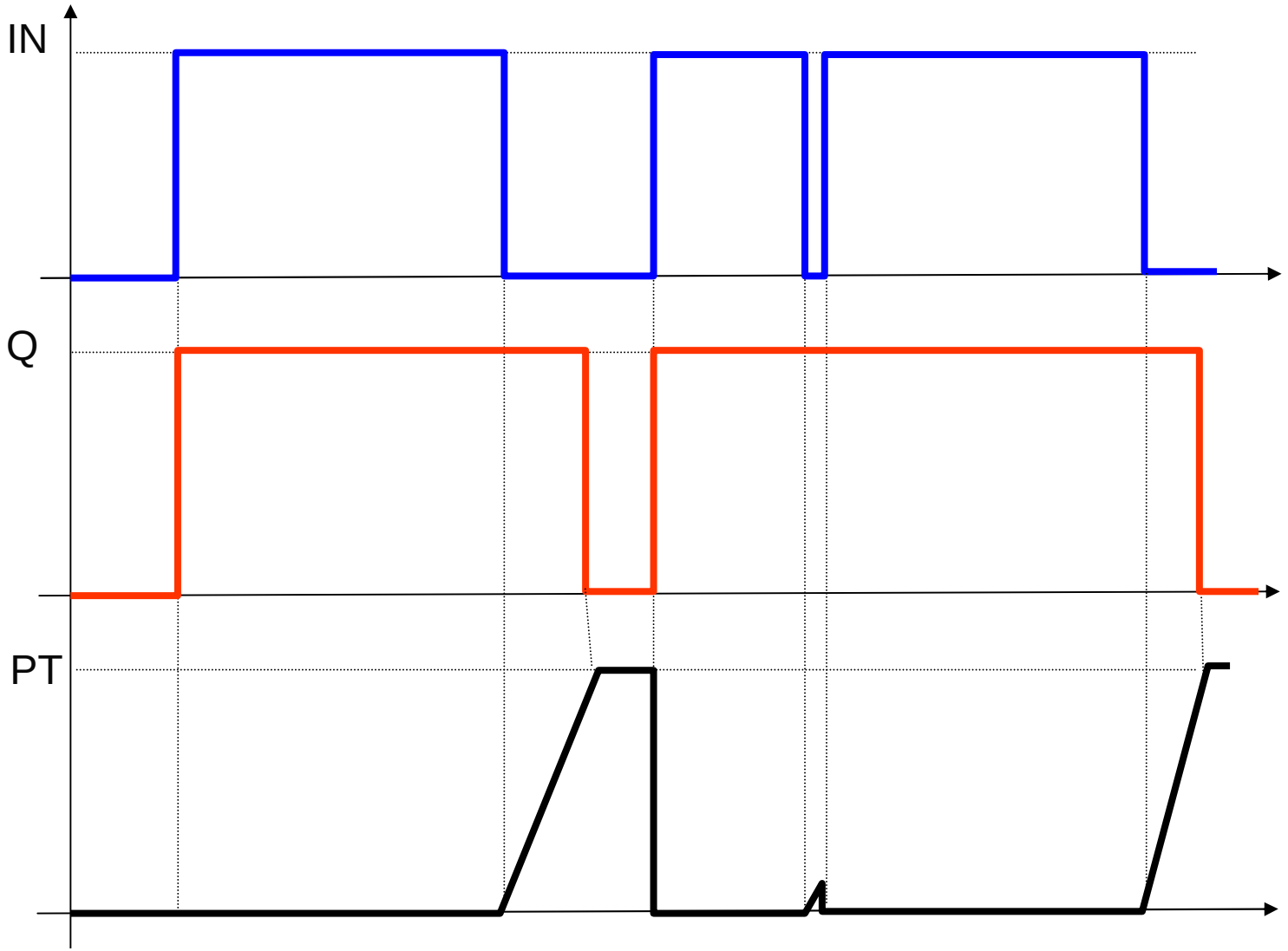
Les blocs fonctionnels standards de temporisateurs (TON)



La Norme CEI 1131

La programmation

Les blocs fonctionnels standards de temporisateurs (TOF)



LA PROGRAMMATION DU GRAFCET

GJC

La Programmation du grafcet

Introduction

Langages d'automatismes	Langage GRAFCET	Langage informatique
Méthode Synchrone		algorithme
Méthode asynchrone		

La Programmation du grafcet

Introduction

Niveaux de synchronisme

L’API est une machine synchrone

L’API est une machine séquentielle et cyclique

L’implémentation du grafcet

REGLE 1
REGLE 2
REGLE 3
REGLE 4
REGLE 5

La Programmation du grafcet

Introduction

Niveaux de synchronisme

L’API est une machine synchrone

L’API est une machine séquentielle et cyclique

L’implémentation du grafcet

REGLE 1

REGLE 2

REGLE 3

REGLE 4

REGLE 5

Famille ou type 1

La Programmation du grafcet

Introduction

Niveaux de synchronisme

L'API est une machine synchrone

L'API est une machine séquentielle et cyclique

L'implémentation du grafcet

REGLE 1

REGLE 2

REGLE 3

REGLE 4

REGLE 5

Famille ou type 3

- Synchronisé

- hiérarchisé

La Programmation du grafcet

Introduction

RAPIDITÉ

- Simplicité du programme
- Taille du programme

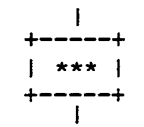
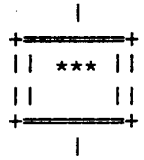
→ Architecture décentralisée

Technique de programmation

- standart
- Améliorée : sauts
- Scrutation situationnelle

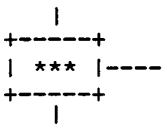
La Programmation du grafcet

Le langage SFC

N°	Représentation	Description
1		Etape - Forme graphique avec liaisons dirigées "*****" = Nom d'étape
		Etape initiale - Forme graphique avec liaisons dirigées "*****" = Nom d'étape initiale (note 2)

2	`STEP *** :`
	`(* Corps d'étape *)`
	`END_STEP`
		Etape - Forme littérale sans liaisons dirigées (voir 2.6.3)
		"*****" = Nom d'étape

	`INITIAL_STEP *** :`
	`(* Corps d'étape *)`
	`END_STEP`
		Etape initiale - Forme littérale sans liaisons dirigées (voir 2.6.3)
		"*****" = Nom d'étape

3a	`***.X`	Drapeau d'étape - Forme générale "*****" = Nom d'étape ***.X = 1 booléen quand *** est actif = sinon 0 booléen
3b		Drapeau d'étape - Connexion directe de variable booléenne ***.X à droite de l'étape "*****"
4	`***.T`	Temps écoulé pour une étape - Forme générale "*****" = Nom d'étape ***.T = une variable de type TIME (voir définition 2.6.2)

NOTES

1 Lorsque la caractéristique 3a, 3b, ou 4 est acceptée, elle doit constituer une erreur si le programme utilisateur tente de modifier la variable associée. Par exemple: si S4 est un nom d'étape, alors les énoncés suivants devraient être des erreurs dans le langage ST défini en 3.3:

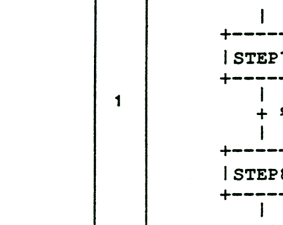
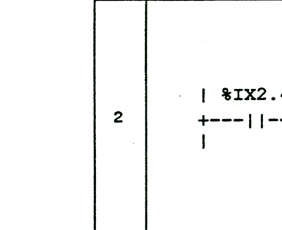
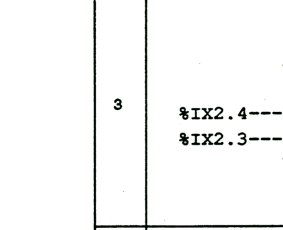
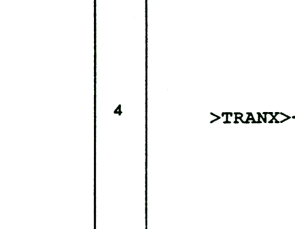
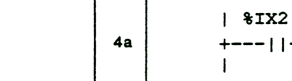
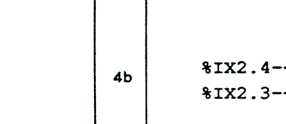
S4.X := 1 ; (* ERROR *)

S4.T := t#100ms ; (* ERROR *)

2 La liaison dirigée supérieure n'est pas requise si l'étape initiale n'a pas de prédécesseur.

La Programmation du grafcet

Le langage SFC

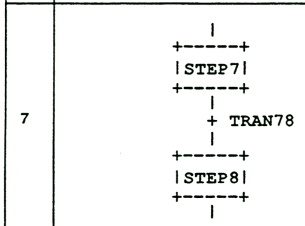
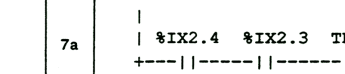
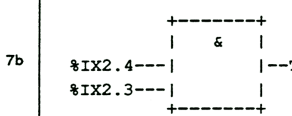
N°	Exemple	Description
1		Etape précédente Condition de transition utilisant le langage ST (voir 3.3) Etape suivante
2		Etape précédente Condition de transition utilisant le langage LD (voir 4.2) Etape suivante
3		Etape précédente Condition de transition utilisant le langage FBD (voir 4.2) Etape suivante
4		Utilisation de connecteur: Etape précédente Connecteur de transition Etape suivante
4a		Condition de transition utilisant le langage LD (voir 4.2)
4b		Utilisant le langage FBD (voir 4.3)

La Programmation du grafcet

Le langage SFC

N°	Exemple	Description
5	STEP STEP7 : END_STEP TRANSITION FROM STEP7 TO STEP 8 := %IX2.4 & %IX2.3 ; END_TRANSITION STEP STEP8 : END_STEP	Equivalent littéral de la caractéristique n° 1 utilisant le langage ST (voir 4.3)

6	STEP STEP7 : END_STEP
	TRANSITION FROM STEP7 TO STEP 8:
	  LD %IX2.4
	  AND %IX2.3
	END_TRANSITION
	STEP STEP8 : END_STEP
		Equivalent littéral de la caractéristique n° 1 utilisant le langage IL (voir 3.2)

7		Utilisation de nom de transition: Etape précédente Nom de transition Etape suivante
7a	TRANSITION TRAN78 :  END_TRANSITION	Condition de transition utilisant le langage LD (voir 4.2)
7b	TRANSITION TRAN78 :  END_TRANSITION	Condition de transition utilisant le langage FBD (voir 4.3)

7c	TRANSITION TRAN78 :
	  LD %IX2.4
	  AND %IX2.3
	END_TRANSITION
		Condition de transition utilisant le langage IL (voir 3.2)

7d	TRANSITION TRAN78 : := %IX2.4 & %IX2.3 ; END_TRANSITION	Condition de transition utilisant le langage ST (voir 3.3)

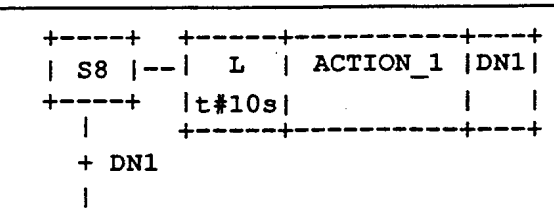
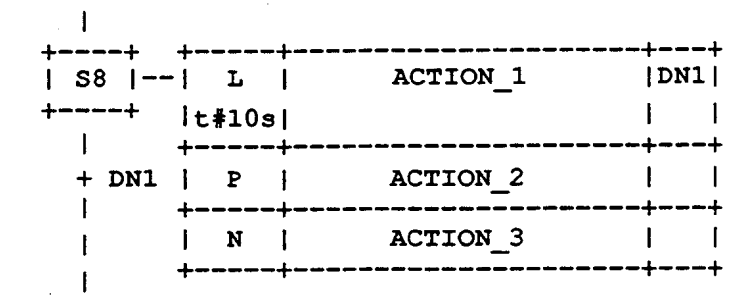
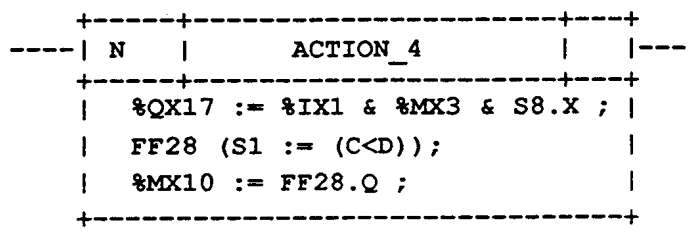
NOTES

1 Si la caractéristique 1 du tableau 40 est acceptée, alors une ou plusieurs des caractéristiques 1, 2, 3, 4 ou 7 du présent tableau doivent être acceptées.

2 Si la caractéristique 2 du tableau 40 est acceptée, alors la caractéristique 5 ou 6 du présent tableau, ou les deux, doivent être acceptées.

La Programmation du grafcet

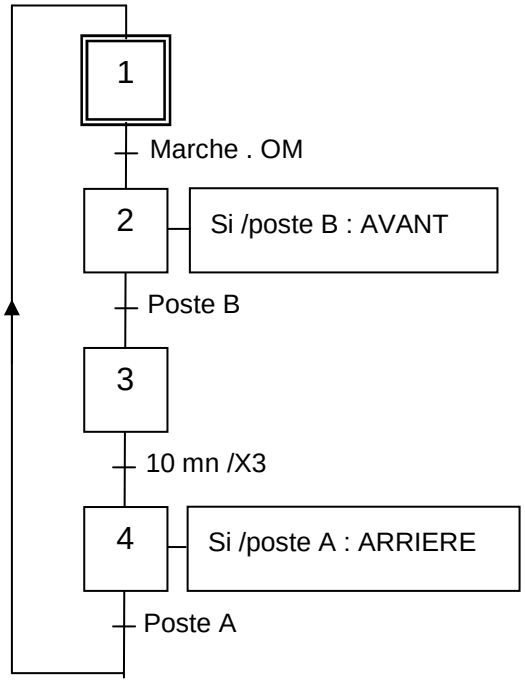
Le langage SFC

N°	Exemple	Caractéristique
1		Bloc d'action (Voir 2.6.4.3)
2		Bloc d'action enchaînés
3	STEP S8 : ACTION_1(L,t#10s,DN1) ; ACTION_2(P) ; ACTION_3(N) ; END_STEP	Corps d'étape littéral
4		Bloc d'action champ "d" (Voir 2.6.4.3)

NOTE - Lorsque la caractéristique 4 est utilisée, le nom d'action correspondant ne peut être utilisé dans aucun autre bloc d'action.

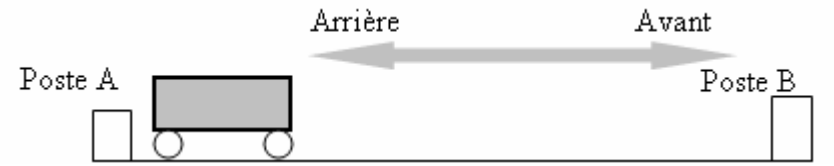
La Programmation du grafcet

Le langage SFC



Commande d'un chariot de transfert.

Un chariot motorisé transfère des produit du poste A au poste B sur demande de l'opérateur par un bouton poussoir "marche".
Le retour du poste B vers A se fait après une temporisation d'attente en B de 10mn (temps de déchargement).



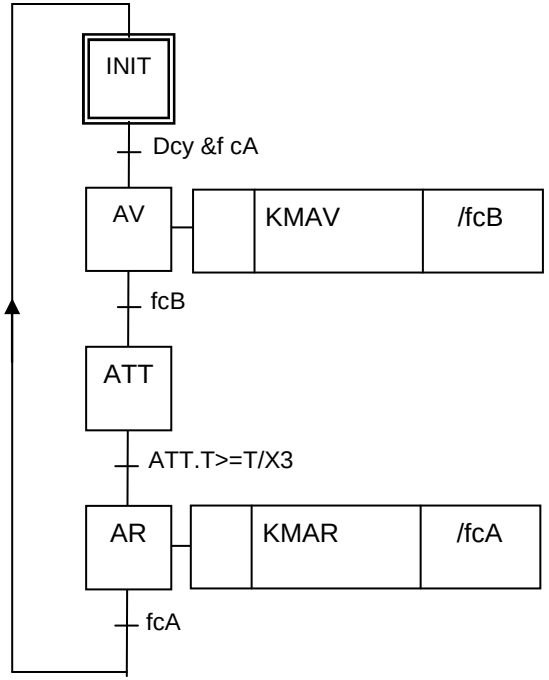
La Programmation du grafcet

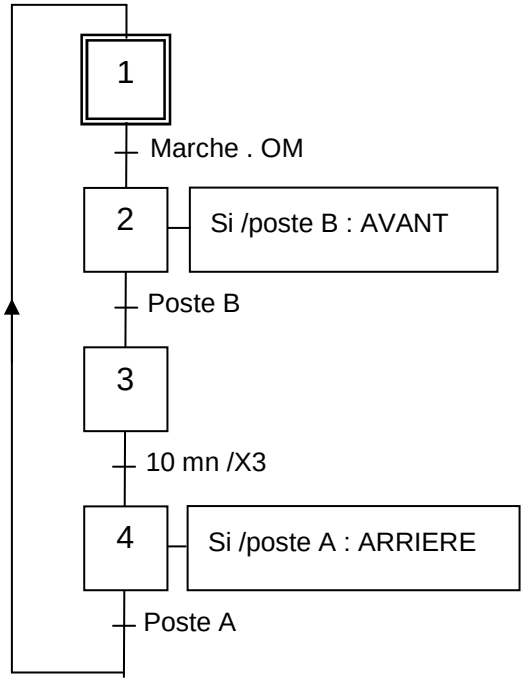
Le langage SFC

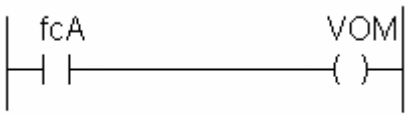
```
PROGRAMME_CHARIOT
        VAR_INPUT
                Dcy : BOOL;              (*BP de marche*)
                FcA : BOOL ;             (*fin de course poste A*)
                FcB : BOOL ;             (*fin de course poste B*)
                T/X3 :TIME :=t#10mn;     (*variable d'attente en B, initialisée à 10mn*)
        END_VAR
        VAR_OUPUT
                KMAV :BOOL ;             (*contacteur marche avant*)
                KMAR :BOOL ;             (*contacteur marche arrière*)
                VOM :BOOL ;              (*voyant origine machine*)
        END_VAR
```

La Programmation du grafcet

Introduction







La Programmation du grafcet

Introduction

corps du programme chariot en langage SFC sous forme littérale.

```
(*commande du chariot*)                    (*commande du voyant OM*)
INITIAL_STEP INIT : END STEP               VOM := fcA
TRANSITION FROM INIT TO AV
        := dcy & fcA ;
END_TRANSITION
STEP AV :
        KMAV(fcB) ;
END_STEP
TRANSITION FROM AV TO ATT
        := fcB ;
END_TRANSITION
STEP ATT : END_STEP
TRANSITION FROM ATD TO AR
        := ATT.T>= t/X12 ;
END_TRANSITION
STEP AR :
        KMAR(fcA) ;
END_STEP
TRANSITION FROM AR TO INIT
        := fcA ;
END_TRANSITION
```

La Programmation du grafcet

Introduction

Toutes les méthodes d'implémentation du GRAFCET devront aborder les 5 points suivants:

1 _ Initialisation :

Peut être assurée par la déclaration des étapes initiales (langage SFC) ou par programmation dans le respect de la règle 1 du grafcet. Généralement exécutée à la mise sous tension.

2 _ Calcul des conditions d'évolutions :

Doit permettre de respecter le règle 2 et éventuellement la règle 4. Associé au franchissement d'une transition, ce calcul peut concerner l'ensemble des transitions ou seulement celles validées.

3 _ Calcul des conditions d'activation des étapes :

Correspond à la règle 3 du grafcet et si nécessaire à la règle 5

4 _ Combinatoire général :

Instruction de blocs fonctionnels standards (tempo, compteur, …)

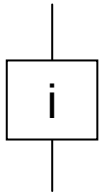
5 _ Equations des sorties :

Calcul des équations logiques des actions associées aux étapes correspondant aux sorties de la PC

La Programmation du grafcet

Introduction

L’etape

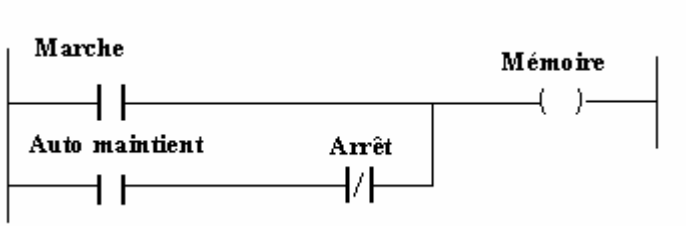


Transition et réceptivité

Les règles 2 et 3 du grafcet précisent ces conditions d’évolution

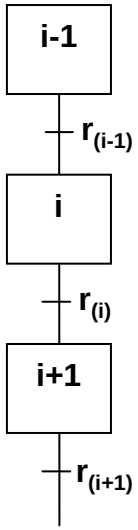
généralisation

Mémoire monostable à marche prioritaire



La Programmation du grafcet

Méthode synchrone



La méthode consiste à l'évaluation préalable de TOUTES les conditions d'évolutions AVANT d'effectuer une seule évolution, de sorte que la situation reste invariante pendant la phase de calcul des activations d'étapes.

Cette méthode de programmation comprend deux blocs distincts, obligatoirement programmés en suivant :

1) le calcul de toutes les conditions d'évolution du grafcet

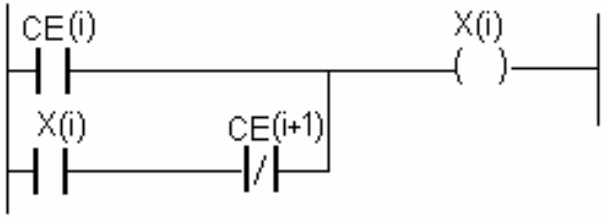
2) le calcul des activations d'étapes du même gra fcet

La Programmation du grafcet

Méthode synchrone

Calcul des CE(i)

Calcul des X(i)



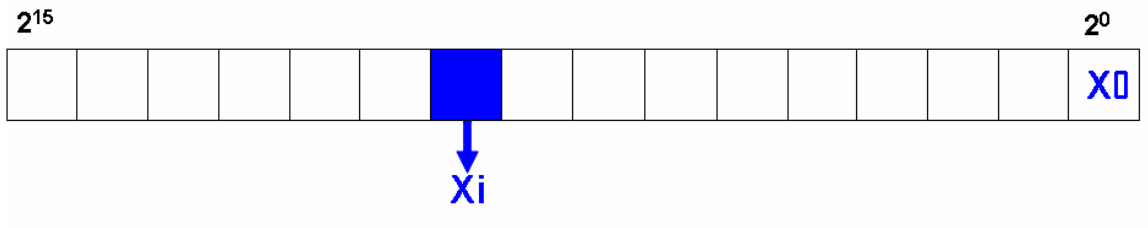
La Programmation du grafcet

Méthode synchrone

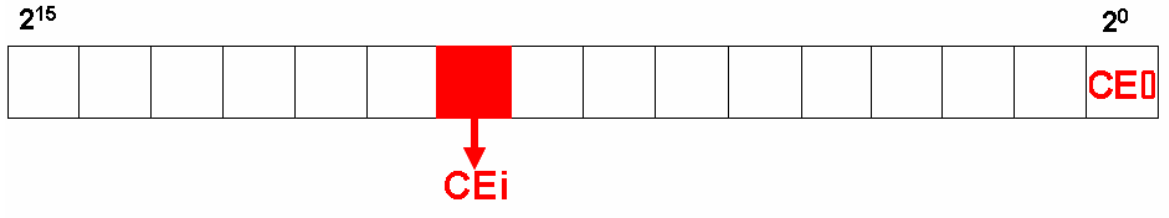
Toutes les étapes du grafcet à programmer seront traduites à partir des équations définies ci-dessus. Ces équations n'utilisent que des données de type booléen, il faut donc utiliser des bits internes pour matérialiser chaque variable d 'étape Xi.

1 étape = 1 bit

Dans la pratique on utilisera les données de type WORD (cordon de 16 bits) pour implémenter un grafcet, ceci nous autorise 16 étapes, mais ne représente pas un nombre limite d'étape pour un grafcet.



Nous appliquerons le même principe à calcul des conditions d'évolutions en utilisant un deuxième mot.



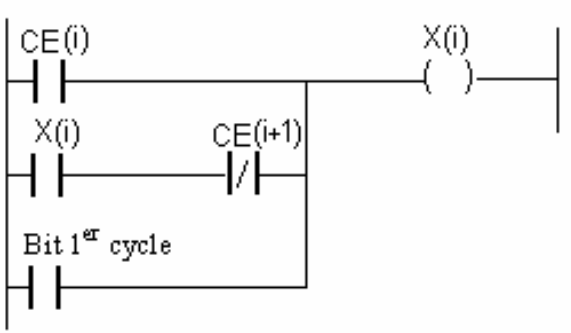
La Programmation du grafcet

Méthode synchrone

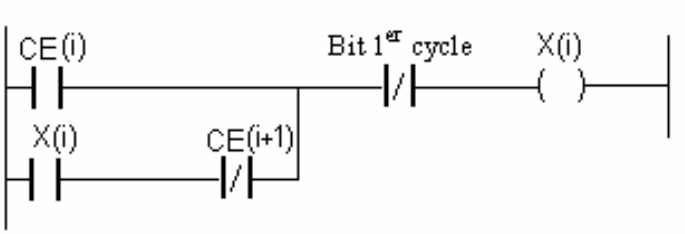
Initialisation

modification des équations

Etape initiale



Etapes non initiales



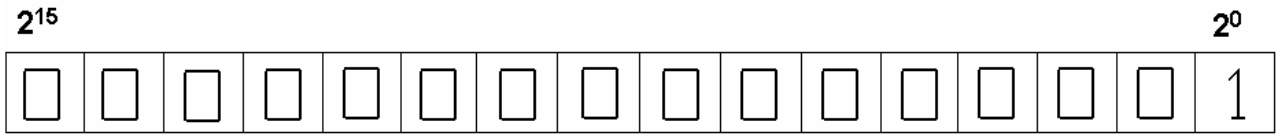
La Programmation du grafcet

Méthode synchrone

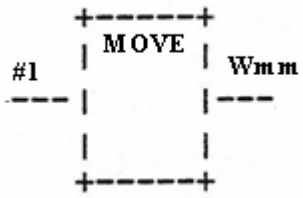
Initialisation

X0 étape initiale

travail sur mot



Xi



La Programmation du grafcet

Méthode synchrone

Structure du programme

INITIALISATION

CALCUL des CONDITION d'ÉVOLUTION

CALCUL d'ACTIVATION d'ÉTAPE

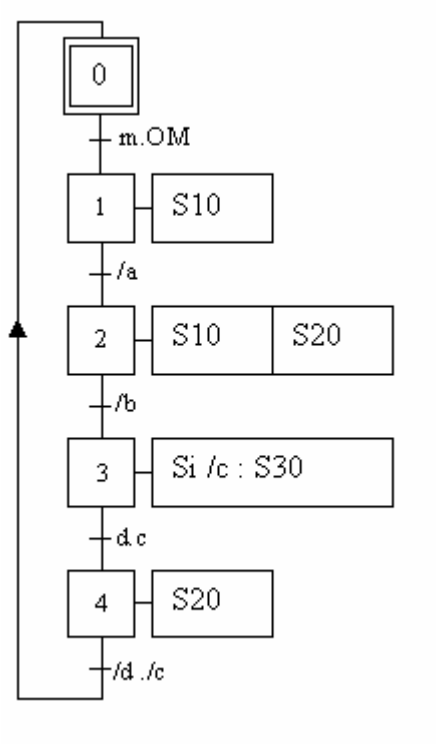
COMBINATOIRE GENERAL

EQUATIONS DES SORTIES

La Programmation du grafcet

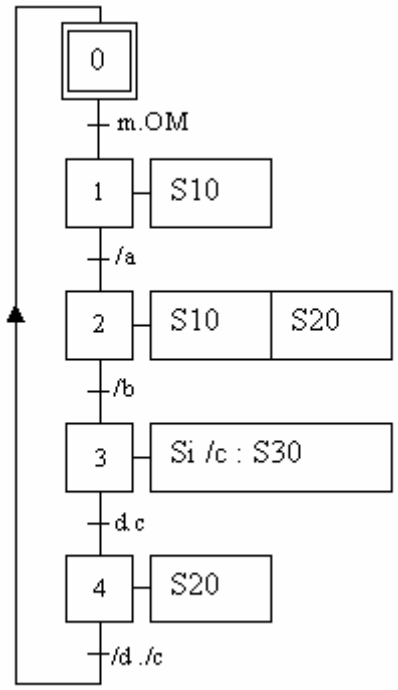
Méthode synchrone

Exemple de programmation : grafcet GRAF_1



La Programmation du grafcet

Méthode synchrone



OM = a ./d

INITIALISATION
Wmm := 1

CALCUL des CONDITIONS d'EVOLUTION
CE1 = X0 . m . OM
CE2 = X1 . /a
CE3 = X2 . /b
CE4 = X3 . d . c
CE0 = X4 . /d . /c

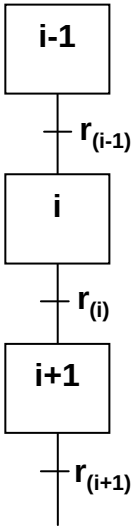
CALCUL D'ACTIVATION D'ETAPE
X0 = CE0 + (X0 . /CE1)
X1 = CE1 + (X1 . /CE2)
X2 = CE2 + (X2 . /CE3)
X3 = CE3 + (X3 . /CE4)
X4 = CE4 + (X4 . /CE0)

COMBINATOIRE GENERAL
OM = a . /d

EQUATIONS DES SORTIES
S10 = X1 + X2
S20 = X2 + X4
S30 = X3 . /c

La Programmation du grafcet

Méthode asynchrone



La méthode de programmation asynchrone ou APPEL/REPONSE du grafcet est une méthode simple à mettre en œuvre et qui ne peut s'utiliser que pour des grafcets de type 1 (famille 1). La condition d'enclenchement de la mémoire d'étape correspond à l'APPEL et le déclenchement à la REPONSE.

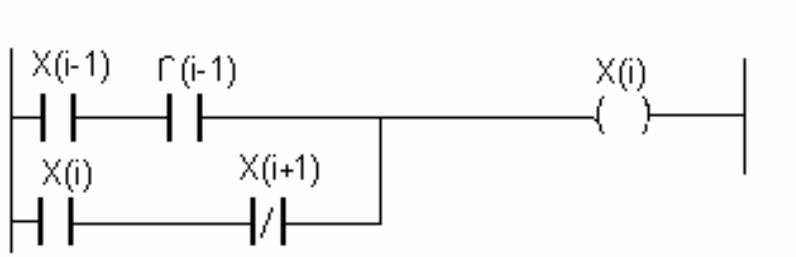
APPEL : mise à 1 de la mémoire d'étape, correspond à la condition d'évolution de la transitions d'entrée de l'étape.

REPONSE : mise à 0 de la mémoire étape, correspond à l'activation de l'étape suivante.

La Programmation du grafcet

Méthode asynchrone

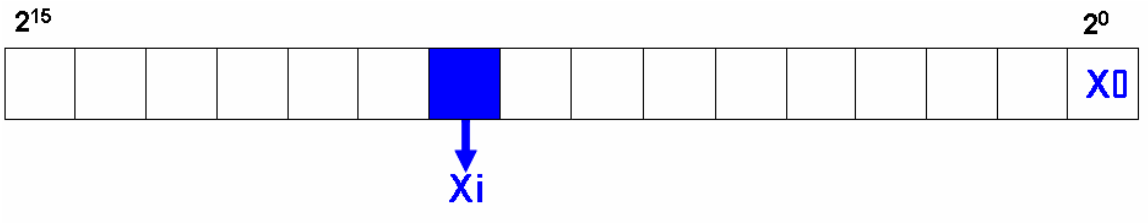
Equation



Toutes les étapes du grafcet à programmer seront traduites à partir de l'équation définie ci-dessus. Cette équation n'utilise que des données de type booléen, il faut donc utiliser des bits internes pour matérialiser chaque variable d 'étape Xi.

1 étape = 1 bit

Comme dan**s** la méthode précédente on utilisera les données de type WORD (cordon de 16 bits) pour implémenter un grafcet, ceci nous autorise 16 étapes, mais ne représente pas un nombre limite d'étape pour un grafcet.



La Programmation du grafcet

Méthode asynchrone

Initialisation

L’initialisation se fait de la même façon que dans la méthode précédente:

-modification de l’équation

- travail sur mot

La Programmation du grafcet

Méthode asynchrone

Structure du programme

INITIALISATION

CALCUL d'ACTIVATION d'ÉTAPE

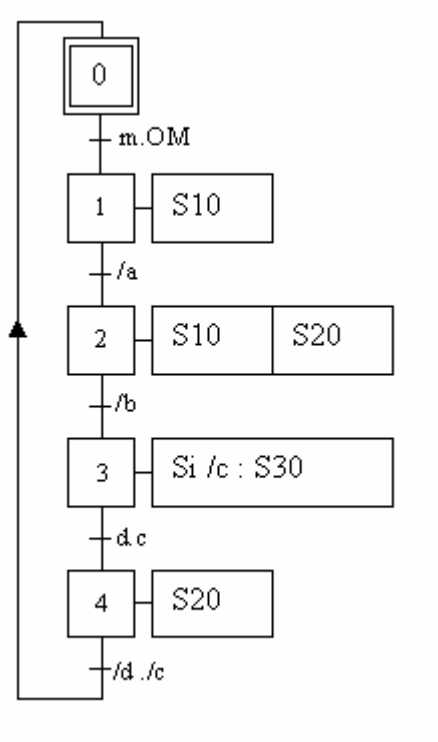
COMBINATOIRE GENERAL

EQUATIONS DES SORTIES

La Programmation du grafcet

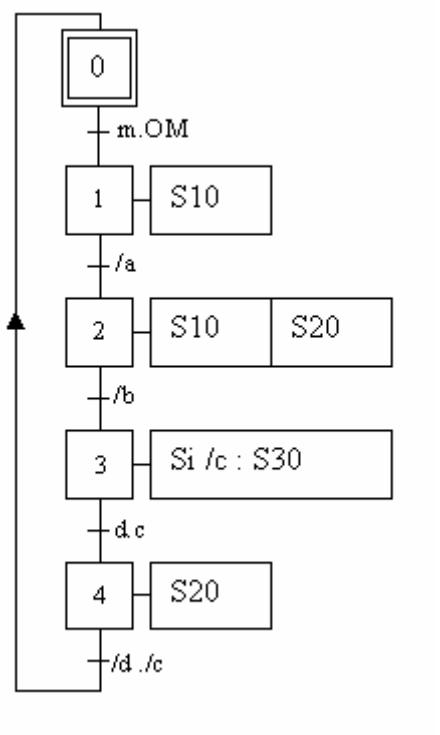
Méthode asynchrone

Exemple de programmation : grafcet GRAF_1



La Programmation du grafcet

Méthode asynchrone



OM = a ./d

INITIALISATION
Wmm := 1

CALCUL D'ACTIVATION D'ETAPE
X0 = (X4 . /d . /c) + (X0 . /X1)
X1 = (X0 . m . OM) + (X1 . /X2)
X2 = (X1 ./a) + (X2 . /X3)
X3 = (X2 ./b) + (X3 . /X4)
X4 = (X3 ./d ./c) + (X4 . /X0)

COMBINATOIRE GENERAL
OM = a . /d

EQUATIONS DES SORTIES
S10 = X1 + X2
S20 = X2 + X4
S30 = X3 . /c

La Programmation du grafcet

Comparaison des méthodes

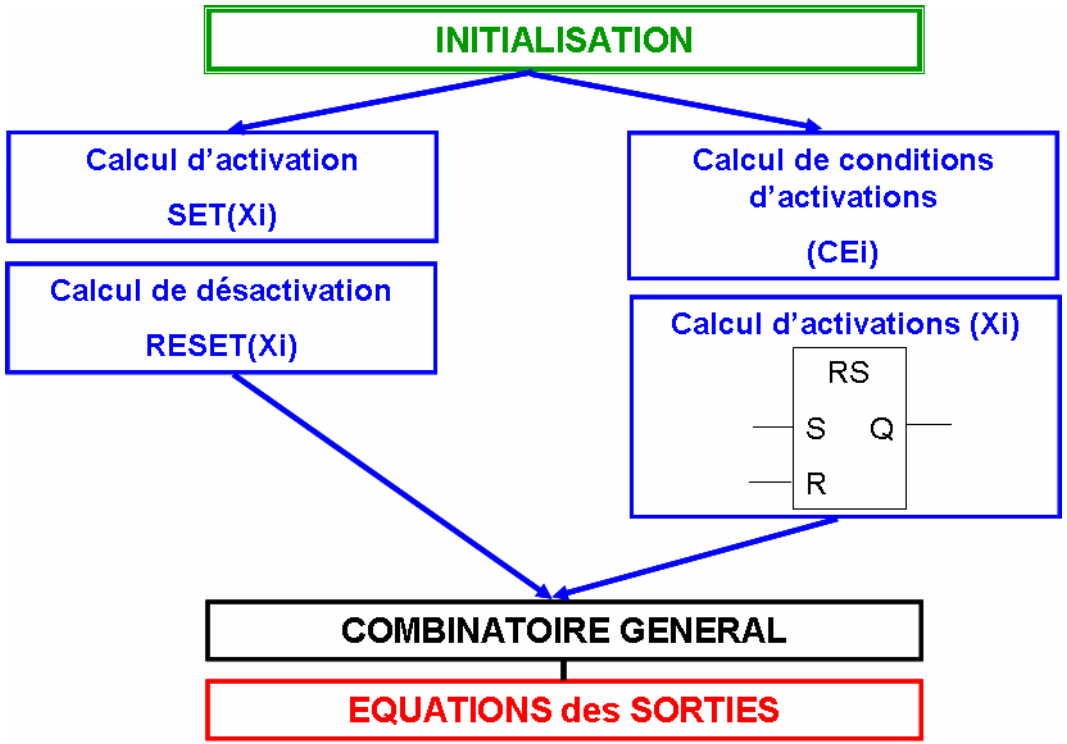
	synchrone	asynchrone
Mémoire de données	2 mots	1 mot
Mémoire programme	15 lignes	10 lignes

La méthode appel/réponse est une méthode simple à implémenter, peu consommatrice de données automate mais qui trouve rapidement ses limites.

La Programmation du grafcet

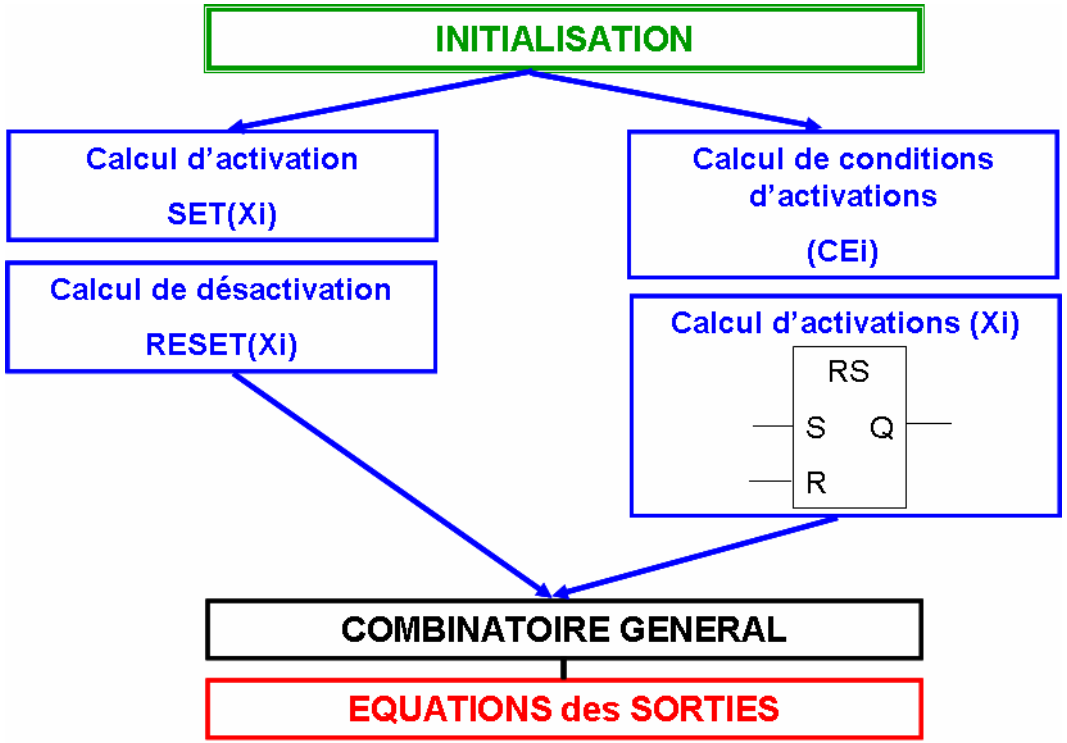
Programmation bistable

La majorité des automates programmables actuels possèdent des adresses de bits bistables dans la mémoire de données. L'accès à ces bits s'accompagne généralement par les instructions "SET" et "RESET", ou par l'utilisation de blocs fonctionnels (norme 1131-3) du type "RS".



La Programmation du grafcet

Programmation bistable



L'utilisation en programmation des bits bistables permet, dans la majorité des automates et pour une grande partie des grafcets, de ne pas avoir à calculer les conditions d'évolutions si l'on prend la précaution de regrouper les activations et les désactivations dans le programme. La structure du programme peut alors avoir deux organisations suivant la technique utilisée.

La Programmation du grafcet

Implémentation du grafcet global

Grafcet structuré

Le concept de macro représentation, utilisé dans le cas de la description du Grafcet de Production Normale (GPN), ne présente pas de problèmes particuliers.
La programmation est fonction de l'outil de développement utilisé.

1°cas :
Le jeu d'instruction de l'automate inclut la programmation des macros-étapes, il suffit dans ce cas de respecter la syntaxe donnée par le constructeur.

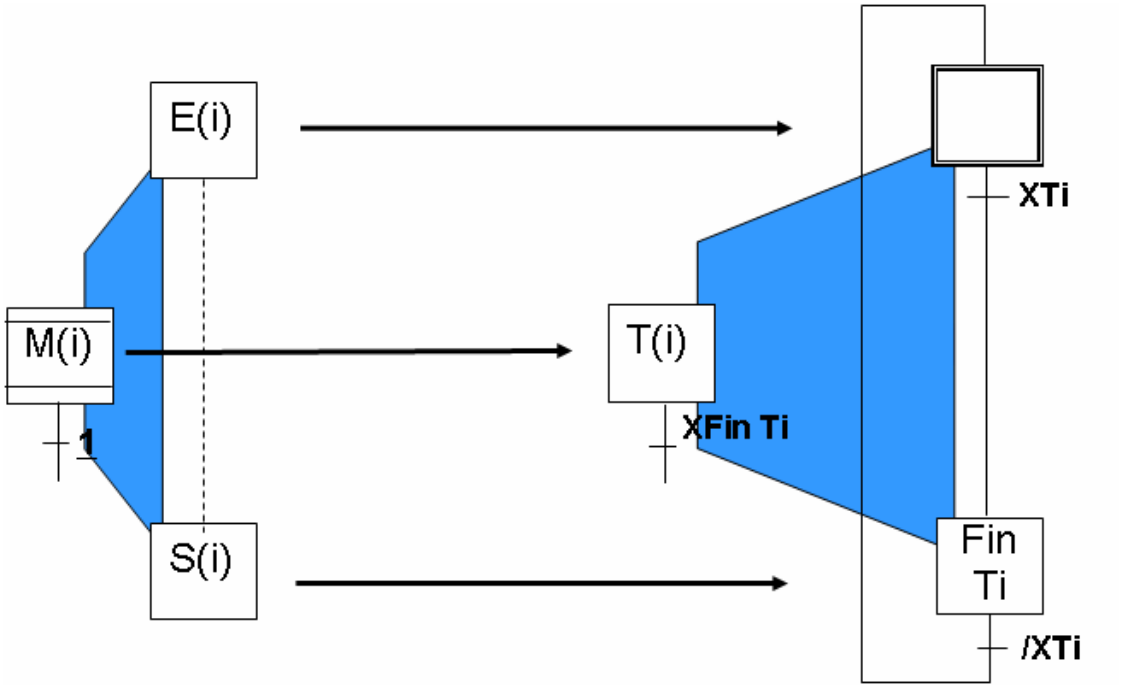
La Programmation du grafcet

Implémentation du grafcet global

Grafcet structuré

Le concept de macro représentation, utilisé dans le cas de la description du Grafcet de Production Normale (GPN), ne présente pas de problèmes particuliers.
La programmation est fonction de l'outil de développement utilisé.

2°cas :
Le passage au point de vue PC du grafcet oblige à interpréter les macros-étapes par des grafcets connexes bouclés.



La programmation suivant la méthode Activation-Désactivation est applicable dans son intégralité

La Programmation du grafcet

Implémentation du grafcet global

Grafcet hiérarchisé

La programmation d'une structure hiérarchisée nécessite l'application d'une métarègle de stucturation du programme. *Le traitement est structuré en "métamodules", chaque métamodule est associé à un seul niveau de grafcet.*

Règle 1 : Traiter les métamodules dans l'ordre de la hiérarchie décroissante

Règle 2 : Pour chaque métamodule, SI et seulement SI ce niveau n'est pas forcé :

- **effectuer la traitement correspondant au grafcet**
- **exécuter les actions de forçage de niveau inférieur**

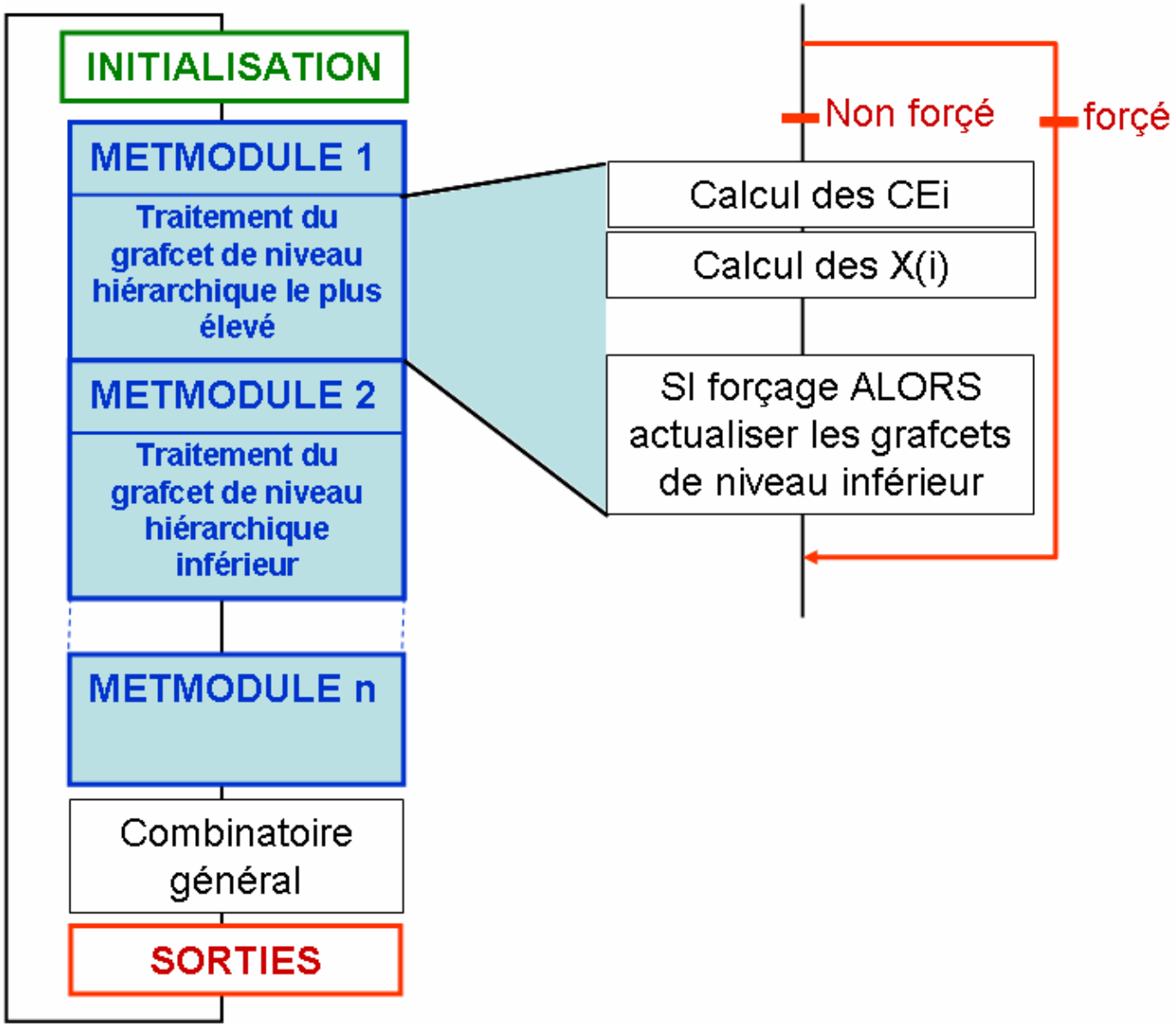
Règle 3 : Sauter le traitement des grafcets partiels forcés

Règle 4 : Effectuer les actions opératives de tous les métamodules en fin de programme

La Programmation du grafcet

Implémentation du grafcet global

Grafcet hiérarchisé



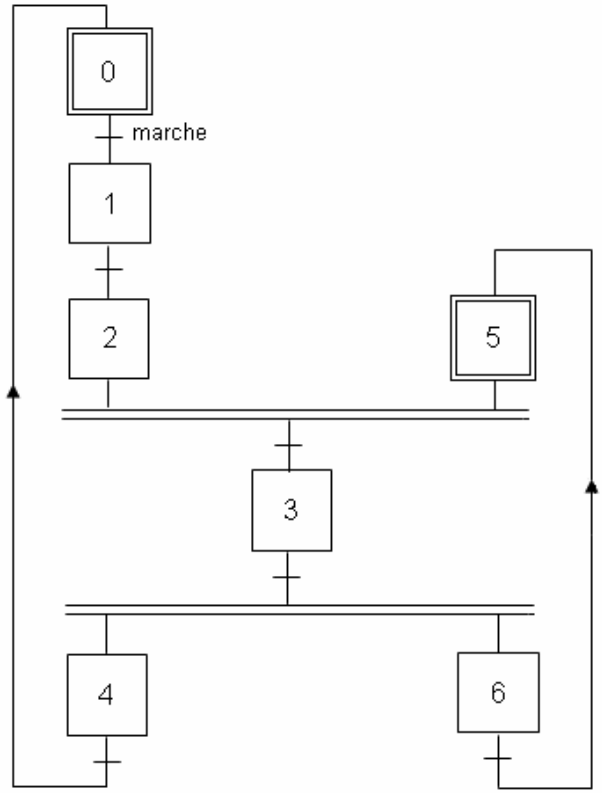
La Programmation du grafcet

Implémentation du grafcet global

Programmation des forçages

Le forçage d'un grafcet , programmé en méthode synchrone en utilisant des mots registres, se fait par modification directe de la valeur du mot en utilisant le bloc fonctionnel de transfert de valeur MOVE.

Mot Registre grafcet W10 # 8
Etape X0 bit 0
Etape X1 bit 1
Etape X2 bit 2
Etape X3 bit 3
Etape X4 bit 4
Etape X5 bit 5
Etape X6 bit 6

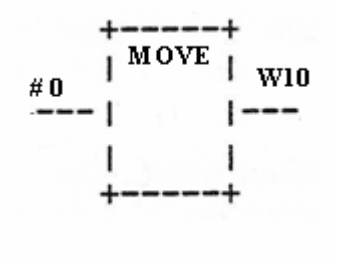


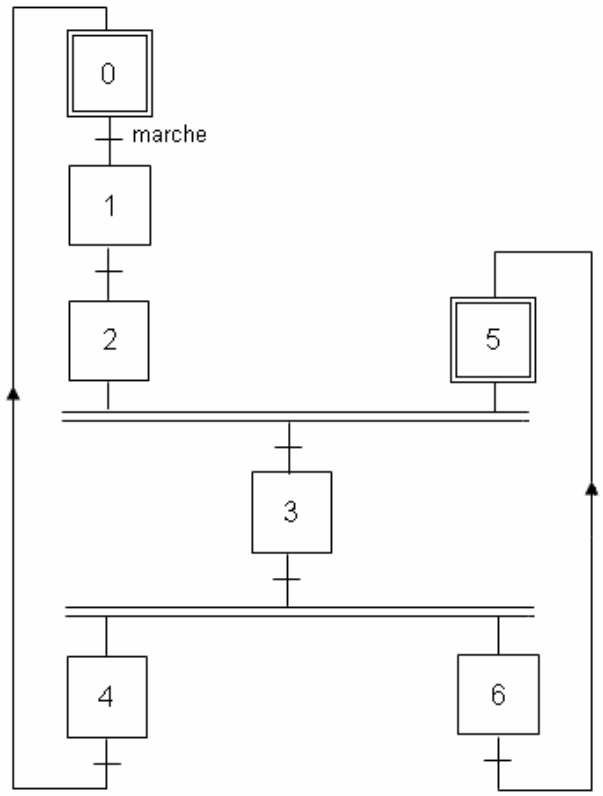
La Programmation du grafcet

Implémentation du grafcet global

Programmation des forçages

Situation vide :
Le forçage en situation vide [F/G# :{}] correspond au forçage à zéro du mot registre utilisé pour le grafcet forcé.
Exemple





La Programmation du grafcet

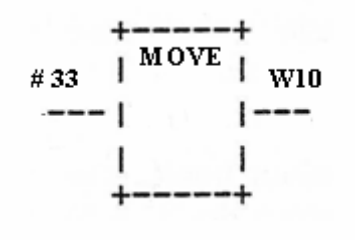
Implémentation du grafcet global

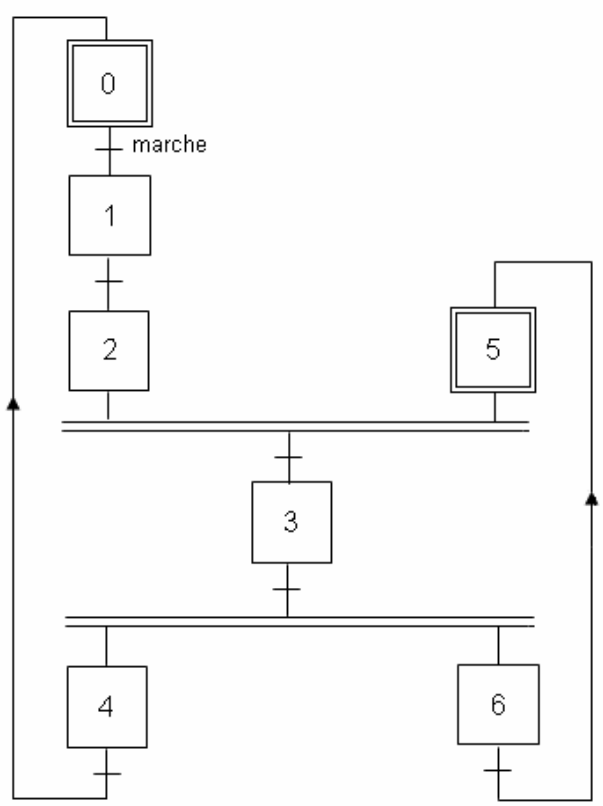
Programmation des forçages

•Situation initiale :

Le forçage en situation initiale [F/G# : (init)] correspond à la mise à 1 dans le mot registre utilisé par le Grafcet des bits correspondant aux étapes initiales

Exemple:





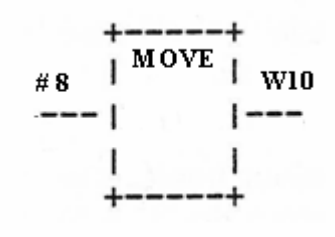
La Programmation du grafcet

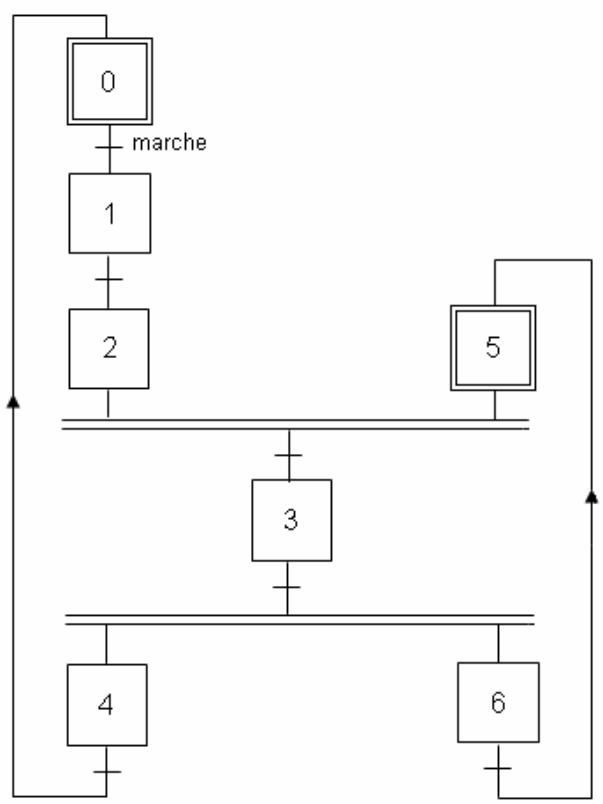
Implémentation du grafcet global

Programmation des forçages

Situation donnée :
Le forçage dans une situation donnée [F/G# : (Xj)] correspond à la mise à 1 dans le mot registre utilisé par le Grafcet du bit correspondant a l' étape Xj

Exemple: [F/G# : (X3)]





La Programmation du grafcet

Implémentation du grafcet global

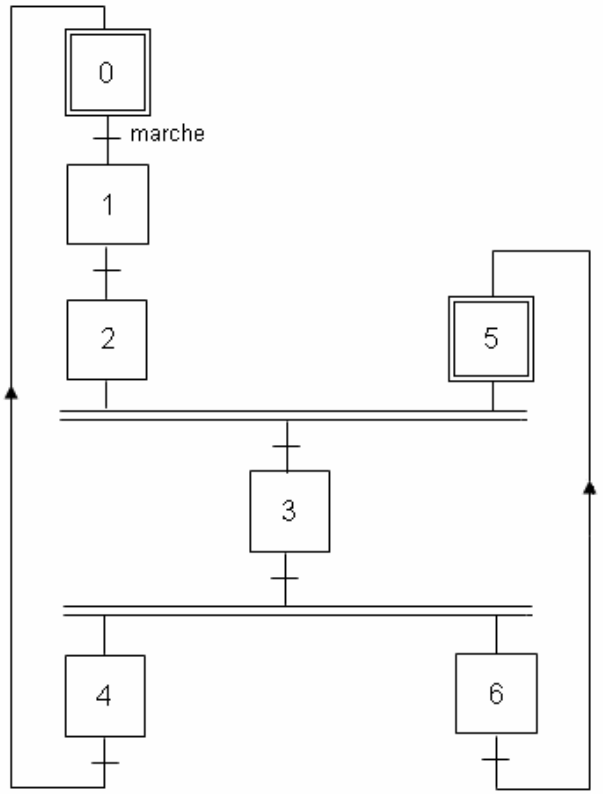
Programmation des forçages

Situation courante :
Le figeage du grafcet [F/G# (*)] peut s'obtenir de plusieurs façons:

1) Par blocage de son évolution en travaillant au niveau des équations des conditions d'évolutions.

2) Par forçage direct et permanent (durée du figeage) du mot registre utilisé par le grafcet figé en deux étapes :
- enregistrement permanent du registre du grafcet forcé dans un registre tampon
- écriture tout le temps du forçage du registre forcé par la valeur du tampon.

3) Par la programmation du saut de traitement du grafcet forcé tout le temps du forçage.

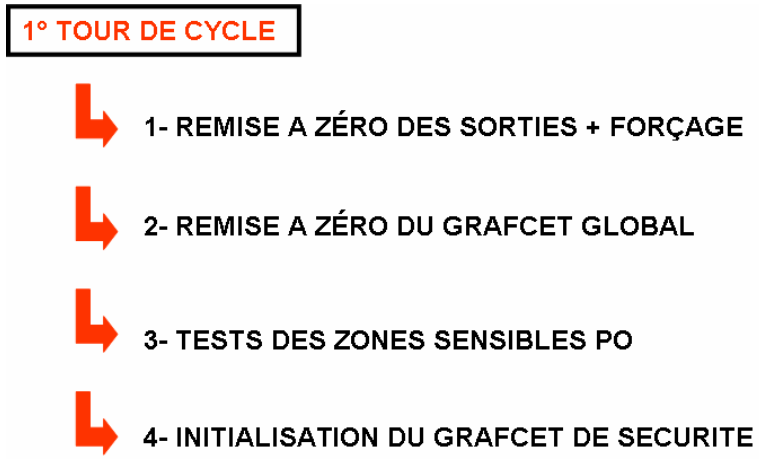


La Programmation du grafcet

Implémentation du grafcet global

Module logiciel de démarrage

A la mise sous tension, l'automate programmable fait un certain nombre de tests internes avant de commencer son cycle de scrutation. Pour une initialisation contrôlée de l'application, il est nécessaire d'effectuer des commandes particulières en début de programme. Nous appellerons cette phase: **« le module d'initialisation ».**
1°tour de cycle : Le premier tour de cycle est signalé par un bit système.



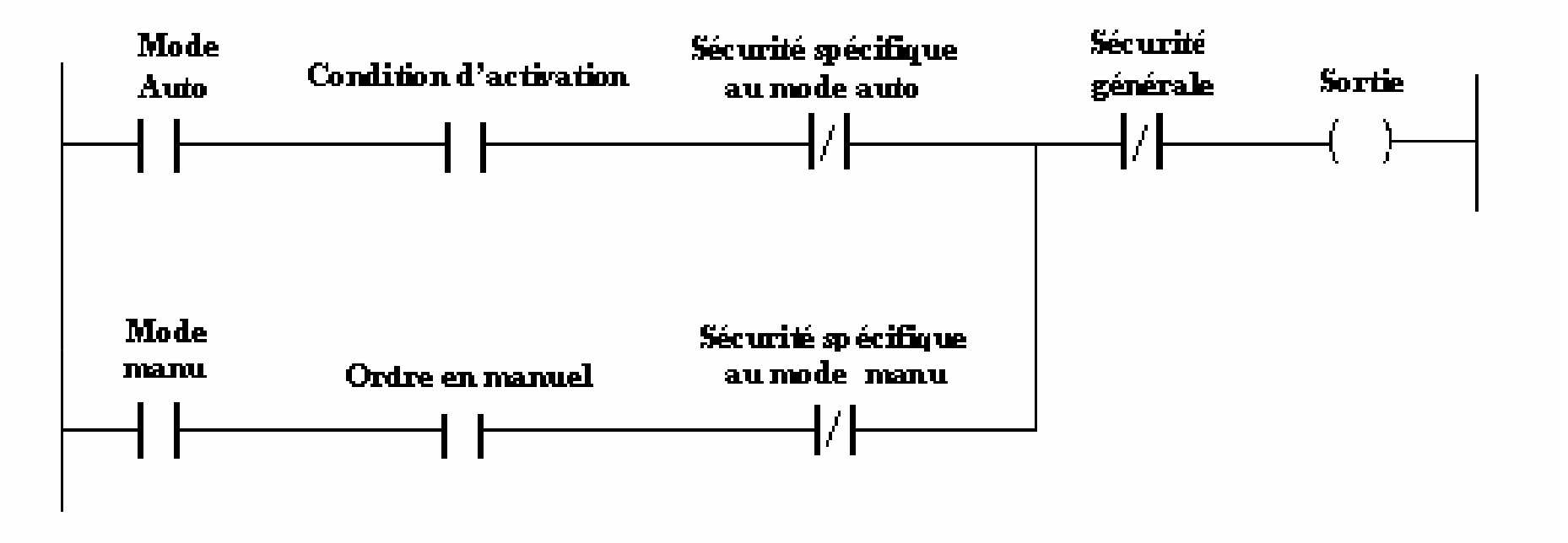
Le déverrouillage des sorties se fait par le Grafcet de sécurité au moment de l'alimentation des sorties.

La Programmation du grafcet

Implémentation du grafcet global

Montage des équations de sorties

Pour le respect de la règle de programmation des sorties (une seule équation par sortie), la forme finale de l'équation d'une sortie aura la structure suivante.



Fin