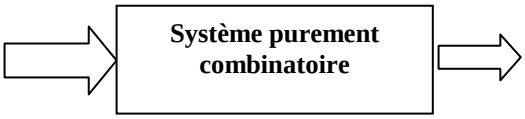
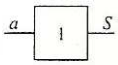
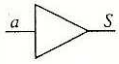
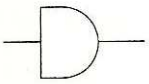
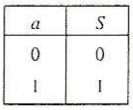
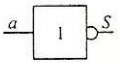
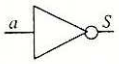
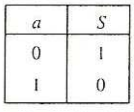
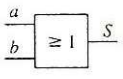
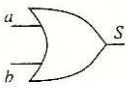
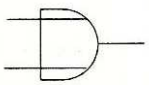
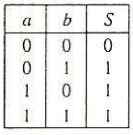
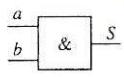
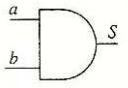
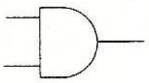
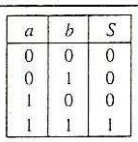
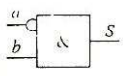
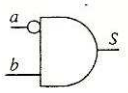
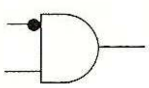
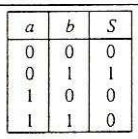
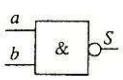
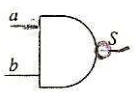
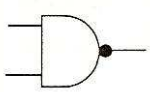
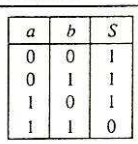
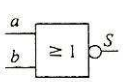
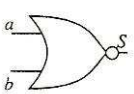
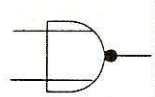
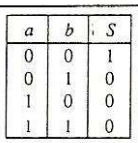
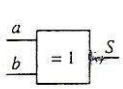
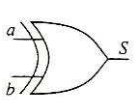
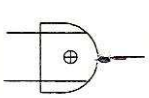
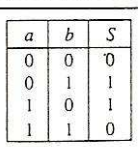
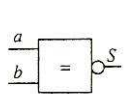
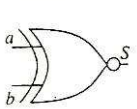
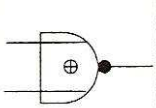
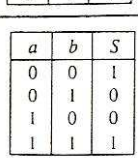
A. Méthodes d'Analyse, de Représentation et de Conception des Automatismes Combinatoires.

I. Définition .



II. Rappel de logique combinatoire.

II.1. Les fonctions logiques.

Opérateur ou fonction	Équation logique	Symboles usuels AFNOR	Symboles usuels ASGS	DIN	Table de vérité
OUI	$S = a$				
NON	$S = \bar{a}$				
OU (inclusif)	$S = a + b$				
ET	$S = a \cdot b$				
Inhibition	$S = \bar{a} \cdot b$				
NAND (NON ET)	$S = \overline{a \cdot b} = \bar{a} + \bar{b}$				
NOR (NON OU)	$S = \overline{a + b} = \bar{a} \cdot \bar{b}$				
OU EXCLUSIF (XOR)	$S = a \oplus b$ $S = \bar{a}b + a\bar{b}$				
identité logique coïncidence	$S = \overline{a \oplus b}$ $S = \bar{a}\bar{b} + ab$				

Remarques

II.2. Propriété de l'algèbre de Boole.

Propriétés du **OU:**	Propriétés du **ET:**
a + 0 =	a.0 =
a + 1 =	a.1 =
a + a =	a.a =
$a + \bar{a}$ =	$a.\bar{a}$ =

Associativité :

Commutativité :

Distributivité :

Propriétés et règles particulières :

*** les 2 propriétés d'absorption:**

(1) $a + a.b = a$

(2) $a.(a + b) = a$

*** la règle de redondance:** $a.b + \bar{a}.c + bc = a.b + \bar{a}.c$

*** la règle du multiple du complément:** - $a + \bar{a}.b = a + b$

II.3. Théorème de DE MORGAN.

III. Outils d'analyse d'automatismes logiques de complexité réduite.

III.1. LE CHRONOGRAMME.

Présentation :

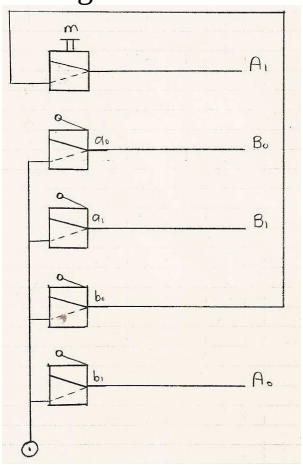
Les diverses étapes de l'analyse sont les suivantes :
- Donnée de départ : Le dossier technique avec le câblage PC et PO.

On en déduit ensuite :

1. Les équations de commande des sorties issues de la PC. *(éventuellement des équations intermédiaires si nécessaires)*
2. Les équations obtenues en 1 et les caractéristiques de la PO permettent **de tracer** le **chronogramme de fonctionnement** du système automatisé.
3. **Du chronogramme on en déduit le cycle de l'automatisme.**

App1 : Soit le système pneumatique décrit ci-dessous :

Câblage PC …. On en déduit ensuite les **équations de commandes**



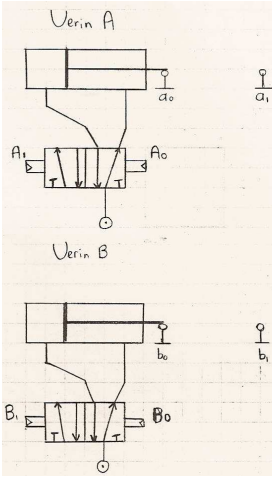
A1 =

A0 =

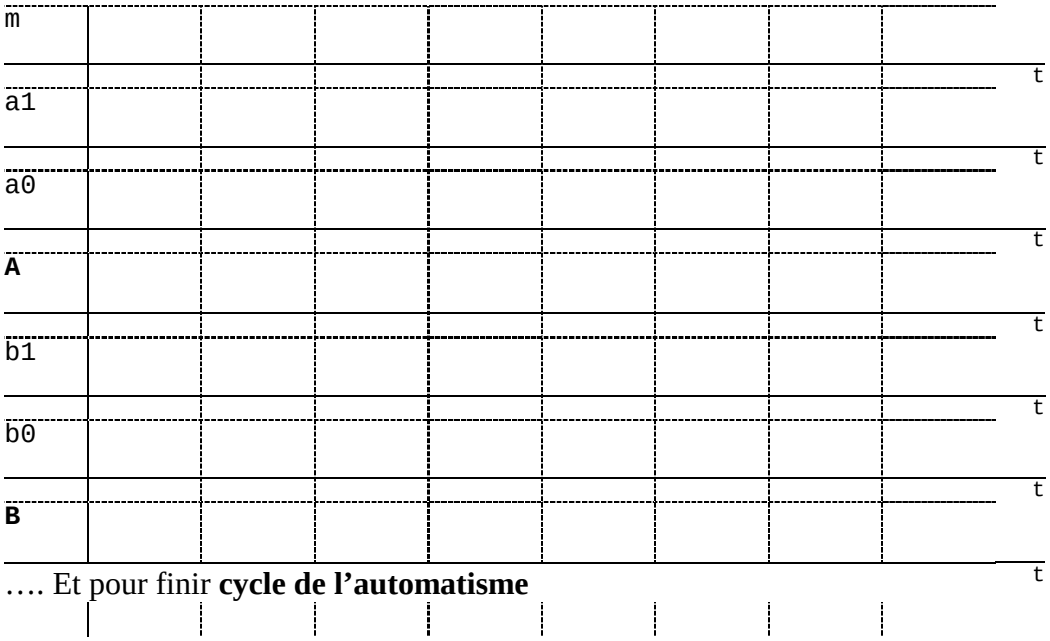
B1 =

B0 =

Câblage PO

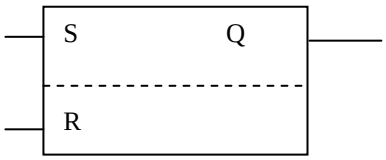


…. Puis le **chronogramme** correspondant :

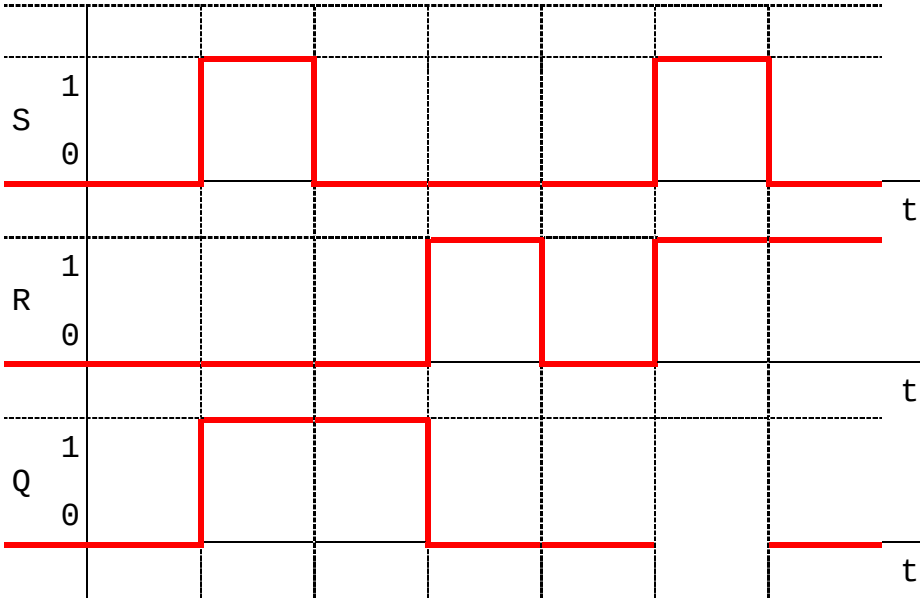


…. Et pour finir **cycle de l'automatisme**

App2 : soit la mémoire suivante



Chronogramme de fonctionnement :



Conclusions :

- Cette représentation présente l'avantage d'être facile à lire, à écrire et à comprendre.
- Cette représentation est le plus souvent utilisée pour analyser en détail une fonction (capteurs, compteur, temporisateur, monostable …) ou un sous ensemble d'un automatisme plus complexe.

III.2. LA TABLE DE VERITE.

Présentation :

IV. Représentation de nombres binaires.

IV.1. Les codes Binaires usuels.

code binaire pur	code hexadé-cimal	code décimal codé binaire	code binaire réfléchi

	code binaire pur ↓															
e					d	c	b	a		d	c	b	a		d	c	b	a
0					0	0	0	0						0				
0					0	0	0	1						1				
0					0	0	1	0						2				
0					0	0	1	1						3				
0					0	1	0	0						4				
0					0	1	0	1						5				
0					0	1	1	0						6				
0					0	1	1	1						7				
0					1	0	0	0						8				
0					1	0	0	1						9				
0					1	0	1	0										
0					1	0	1	1										
0					1	1	0	0										
0					1	1	0	1										
0					1	1	1	0										
0					1	1	1	1										
1	0	0	0	0														
1	0	0	0	1														
1	0	0	1	0														
1	0	0	1	1														
1	0	1	0	0														
1	0	1	0	1														
1	0	1	1	0														
1	0	1	1	1														
1	1	0	0	0														
1	1	0	0	1														
1	1	0	1	0														
1	1	0	1	1														
1	1	1	0	0														
1	1	1	0	1														
1	1	1	1	0														
1	1	1	1	1														

IV.2. Numération.

Tout nombre peut se mettre sous la forme d'une *Somme de polynômes*;

a=variable

$C_{(B)} = a_n*B^n+a_{n-1}*B^{n-1}+a_{n-2}*B^{n-2}+....+a_1*B^1+a_0*B^0$; avec n=rang

B=base

(ex: $964 = 9*10^2+6*10^1+4*10^0 = 900+60+4$)

Ceci s'écrit encore:

B=base

k=rang

$$C_{(B)} = \sum_{k=0}^{n} a_k*B^k$$ **avec n=rang le plus élevé**

a= variable de rang k; a ∈ [0,…,B-1]

Opérations sur les nombres:

*** Le CODAGE;**

*** Le DECODAGE;**

*** Le TRANSCODAGE;**

Binaire Pur ou Naturel (base 2)

11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**← n: nombre de variables binaires**
1	*1*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*1*	*← Chiffre en binaire pur*
10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**← r: rang**

1 Octet = 8 bits ; 1 MOT = 16 bits

- **Hexadécimal** (base 16 de 0..9ABCDEF)

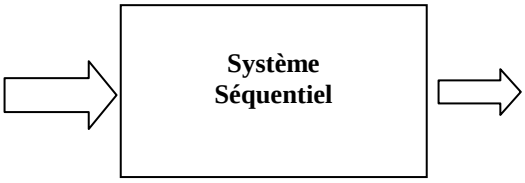
- **BCD** (décimal codé binaire) ex 3215: 3 2 1 5
  0011 0010 0001 0101 (BCD)

- **Code ASCII** 256 caractères alphanumériques et de contrôle codés sur 8 bits / caractère.

Remarque: 1 mot égal 2 octets, donc on pourra coder 2 caractères ASCII dans 1 mot.

B. Méthodes d'Analyse, de Représentation et de Conception des Automatismes Séquentiels.

I. Définition :



II. Graphe Fonctionnel de Commande Etape Transition ; le Grafcet.

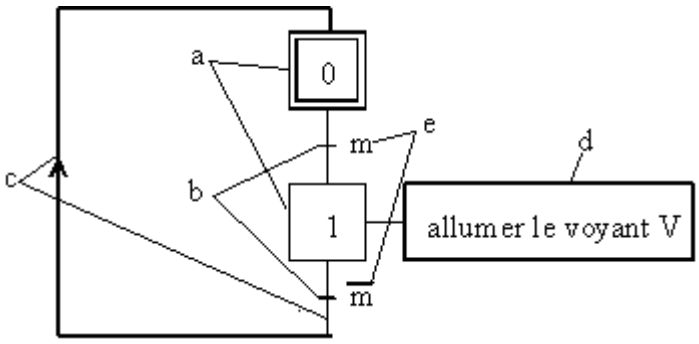
II.1. Généralité :

Cette méthode d'analyse, de représentation et de conception est parfaitement adaptée à l'étude des automatismes séquentiels industriels.
L'intérêt fondamental du Grafcet réside dans le fait que l'on tient compte à la fois de l'action précédente et de la condition caractéristique de la fin de cette action pour enclencher l'action suivante.

II.2. Règle de syntaxe

- ***Eléments graphiques de représentation***

(a) les étapes
(b) les transitions
(c) les liaisons orientées qui relient les précédentes
(d) les actions associées aux étapes
(e) les réceptivités associées aux transitions



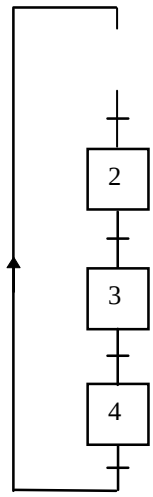
II.3. Les règles d'évolution d'après norme EN60848-2 d'août 2002 (et rappel NF C03-190)

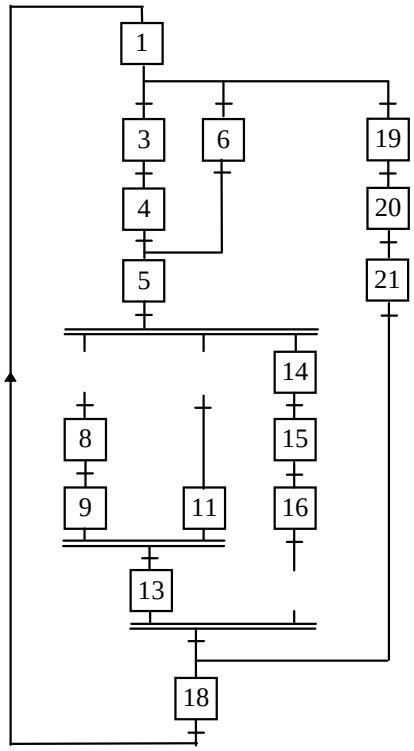
➢ Règle n°1 "situation initiale"

La situation initiale est une situation active à l'instant initial, elle est donc décrite par l'ensemble des étapes actives à cet instant. Le choix de la situation à l'instant initial repose sur des considérations méthodologiques et relatives à la nature de la partie séquentielle du système visé.

- Symbole "situation initiale":

Exemples

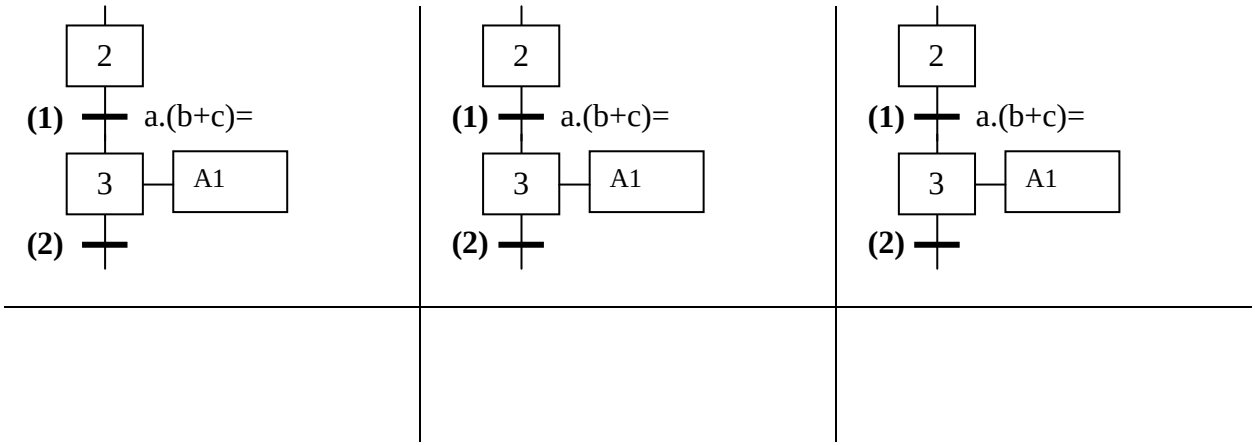




➢ Règle n°2 "franchissement d'une transition"

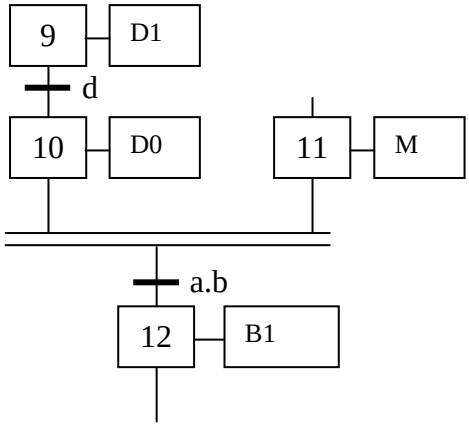
- **Symbole d'une "transition":**

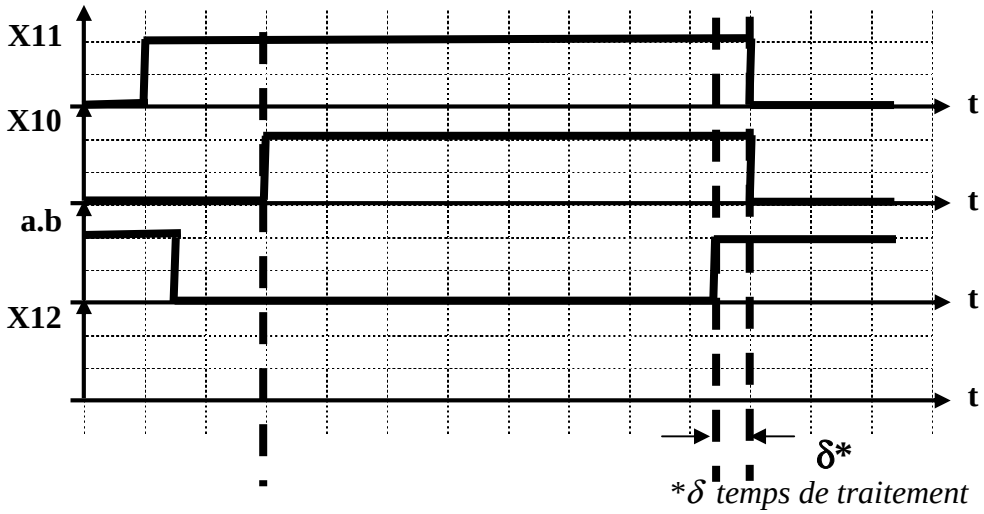
- **Illustration:**



➢ <u>Règle n°3</u> "activation et désactivation d'étape(s)"

- **Chronogramme de synthèse règles 2&3 :**



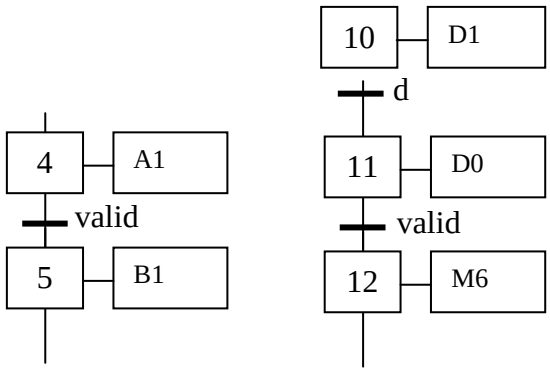


- **D'ou la description « Pas à Pas » ci-dessous :**

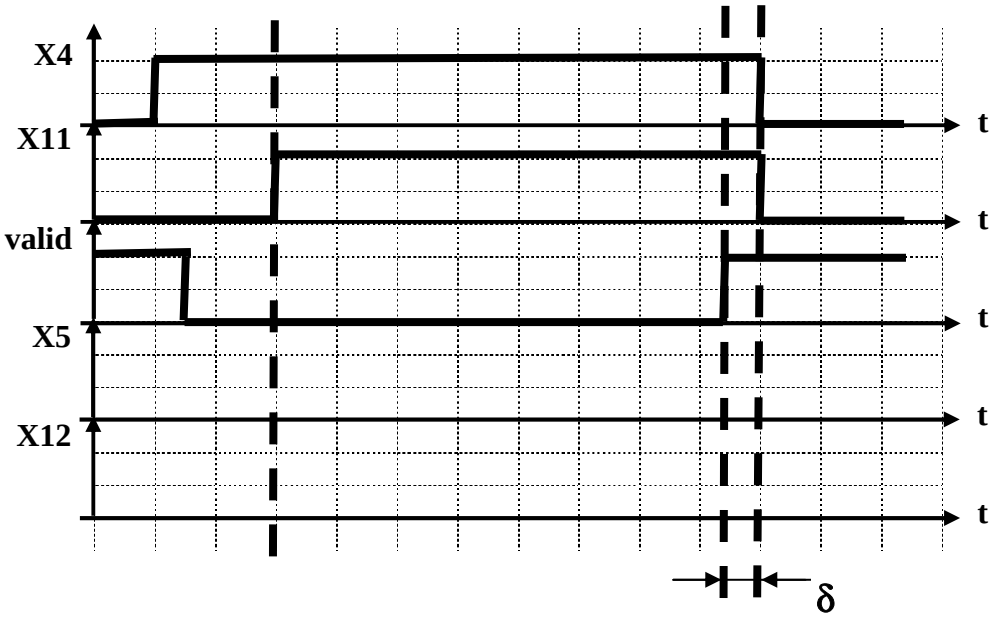
Situation des grafcets	***Actions engendrées***	***Conditions d'évolution***
X9 ; X11	D1 ;M	

➢ <u>Règle n°4</u> "évolutions simultanées"

Exemple :



- **Chronogramme :**

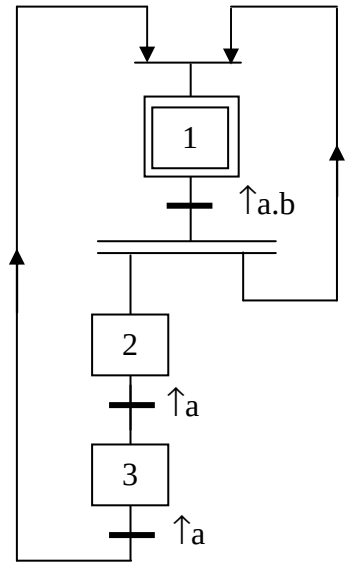


- **<u>Remarque concernant δ</u> :** l'évolution simultanée imposée par les règles du Grafcet n'est technologiquement pas réalisable de manière absolue. Ce temps permet donc d'interpréter «l'évolution simultanée » lors de la réalisation de la PC pour obtenir un comportement conforme au fonctionnement désiré sur la PO.

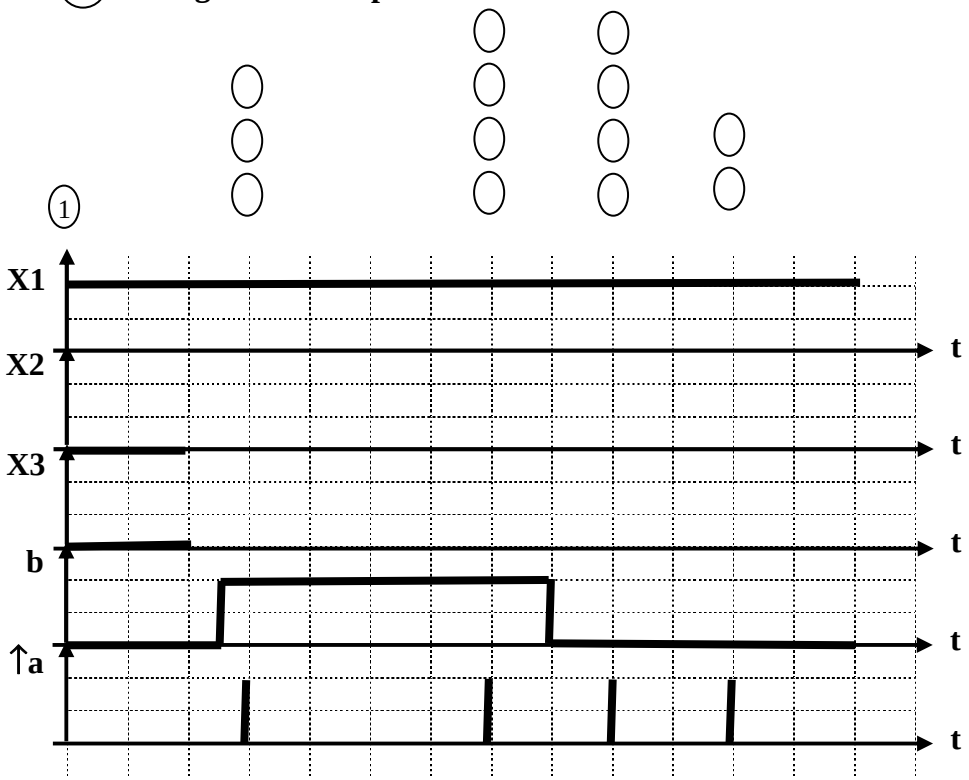
- **D'ou la description « Pas à Pas » ci-dessous :**

Situation des grafcets	***Actions engendrées***	***Conditions d'évolution***
X4 ; X10	A1 ;D1	

➢ <u>Règle n°5</u> "activation et désactivation simultanées"

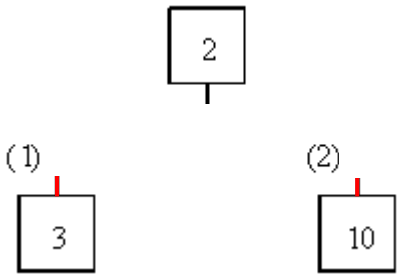


- **Chronogramme : (N°) des règles utilisées pour l'évolution du Grafcet.**



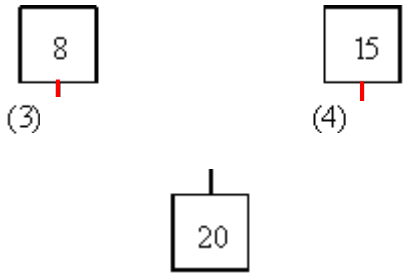
II.4. Les Transitions particulières.

Divergence en OU.



Principe de fonctionnement :
L'étape 2 est active, les transitions (1) et (2) sont donc validées.
Quand r = 1 il y a 2 possibilités :
- Si m = 1 il y a franchissement vers l'étape 3 et l'étape 10 reste inactive.
- Si m = 0 il y a franchissement vers l'étape 10 et l'étape 3 reste inactive.

Convergence en OU.

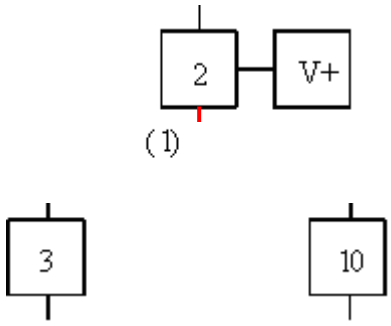


Principe de fonctionnement.
Si l'étape 8 est active la transition (3) est validée.
Quand a0 = 1 la réceptivité associée à (3) est vraie.
L'étape 20 devient active et l'étape 8 est désactivée.

Si l'étape 15 est active la transition (4) est validée.
Quand p0 = 1 la réceptivité associée à (4) est vraie.
L'étape 20 devient active et l'étape 15 est désactivée.

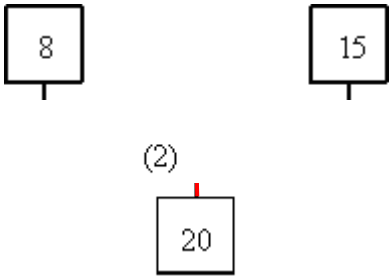
Structure généralement utilisée pour exprimer

Divergence en ET.



Principe de fonctionnement :
Si l'étape 2 est active, la transition (1) validée.
Quand m = 1 alors il y a franchissement et les étapes 3 et 10 sont activées tandis que l'étape 2 est désactivée.

Convergence en ET.

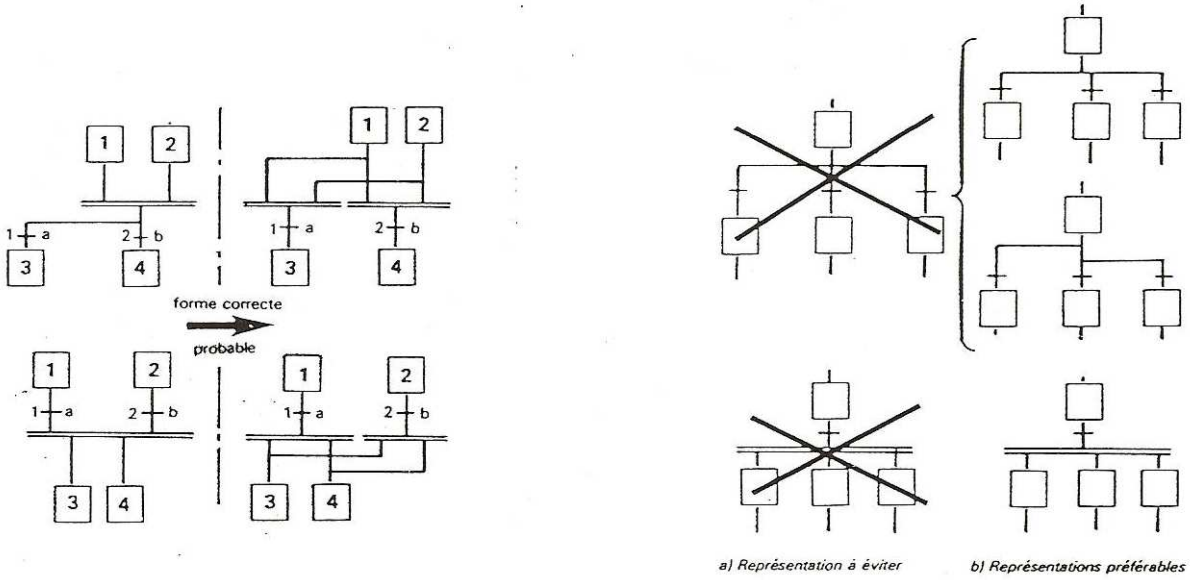


Principe de fonctionnement :
Si les étapes 8 et 15 sont actives alors la transition (2) validée.
Quand a = 1 alors il y a franchissement et l'étape 20 est activée tandis que les étapes 8 et 15 sont désactivées.

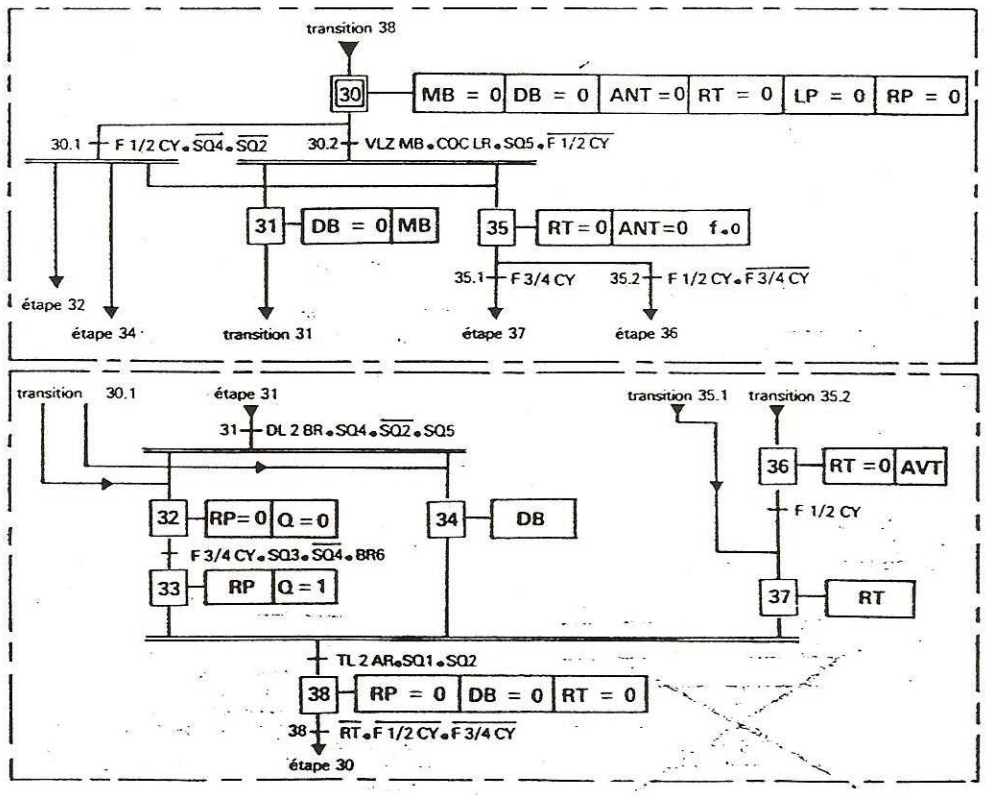
Structure utilisée pour exprimer

- **Complément sur les règles de syntaxe.**

  - les liaisons:

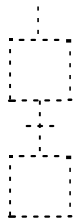


  - Numérotation et renvois:



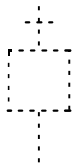
- **Structures particulières**

Transition Source



Remarque : L'activation de l'étape aval d'une transition source est effective aussi longtemps que sa réceptivité associée reste vraie, indépendamment de l'état des réceptivités des transitions validées par cette étape (voir règle d'évolution N°5). Pour éviter une activation continue de l'étape aval de la transition source, il est souhaitable que la réceptivité associée ne soit vraie que lorsqu'un événement d'entrée ou un événement interne se produit. Pour cela l'expression logique formant la réceptivité doit toujours comporter un front de variables.

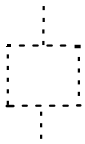
Etape Source



Remarque : Pour permettre l'activation de l'étape source il faut satisfaire au moins l'une des conditions suivantes :

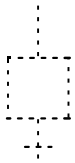
-

Transition Puits



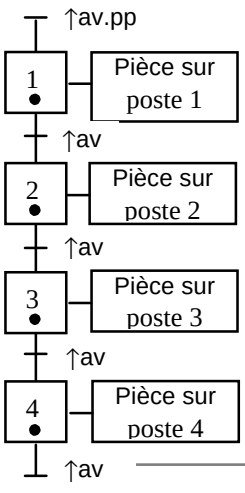
Remarque : Lorsque la transition puits est validée et que sa réceptivité associée * est vraie, le franchissement de cette transition a pour unique conséquence de désactiver les étapes amonts. Attention, si le grafcet se trouve alors désactivé, pour le réactiver, faut impérativement passer par :

Etape Puits



La désactivation de l'étape puits n'est possible que par l'un des deux moyens suivants :

-



EXEMPLE : structure de registre à décalage :
La structure d'un registre à décalage est une utilisation pertinente d'une transition source et d'une transition puits. Dans l'exemple ci-contre, chaque étape active représente la présence d'une pièce sur le poste correspondant. La présence d'une pièce (pp) à l'entrée et l'avance du transfert entre les postes (↑av) active l'étape 1 par le franchissement de la transition source. Puis à chaque (↑av) les transitions validées sont simultanément franchies, y compris la transition puits en aval de l'étape 4.
Remarque : La représentation correspond au cas fréquent où toutes les étapes sont actives simultanément

II.5. Mesure du temps dans un Grafcet.

- Définition formelle d'un opérateur « Temporisation » :

Cet opérateur, interne à la PC pour ces point de vue description est défini formellement dans le Grafcet par les propriétés suivantes :

- **La notation**

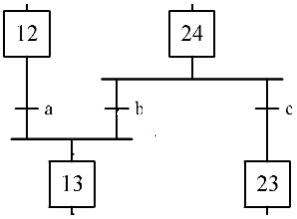
ou

NOTE 1 : L'astérisque (*****) doit être remplacé par la variable que l'on désire temporiser, par exemple une variable d'étape ou une variable d'entrée.
NOTE 2 : t1 et t2 doivent être remplacés par leur valeur réelle exprimée dans l'unité de temps choisie.
NOTE 3 : **La variable temporisée**

II.6. Les actions.

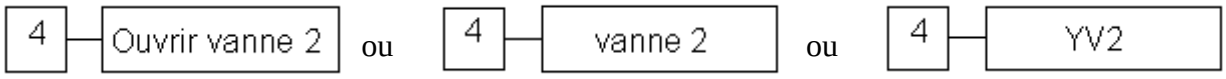
- **Symbole :** 

Cas particulier d'action sur franchissement :

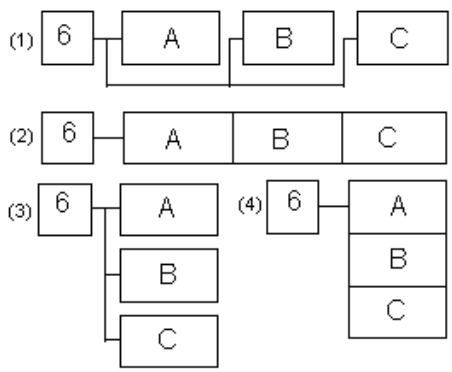


- **Libellé d'une sortie :** Toute action doit posséder un libellé dans le rectangle représentant cette action.

EXEMPLE 1 : Différentes formes, littérales et symboliques, de libellé d'action faisant référence à la sortie dont la valeur vraie doit provoquer l'ouverture de la vanne 2.



EXEMPLE 2 : Différentes représentations (1, 2, 3, 4) de l'association de plusieurs actions à une même étape.

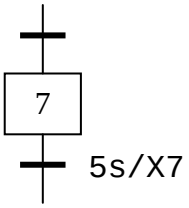


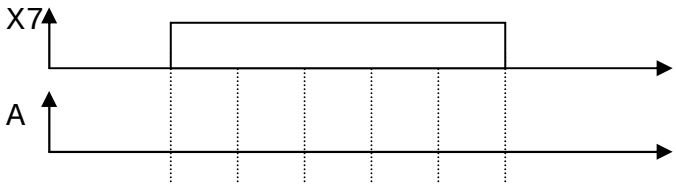
Remarque :

- Les quatre représentations sont strictement équivalentes. Les représentations (2) et (4) peuvent être considérées respectivement comme des simplifications des représentations (1) et (3).

- ***Classification des actions.***

Action



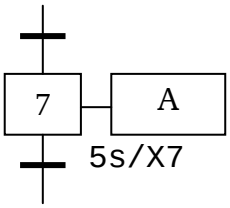


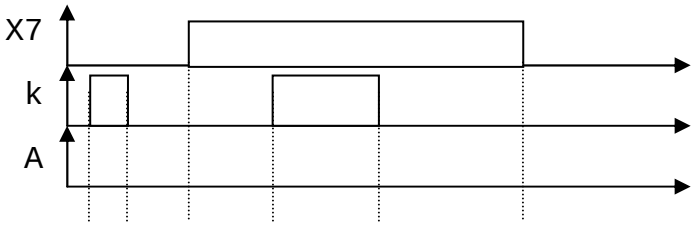
Equation de A : A=

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action



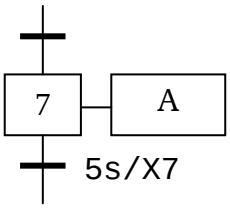


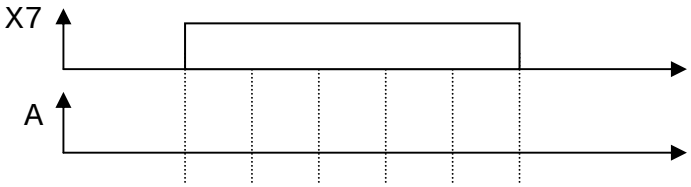
Equation de A : A=

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action



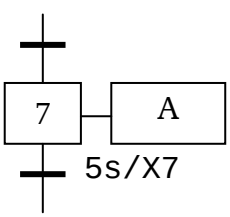


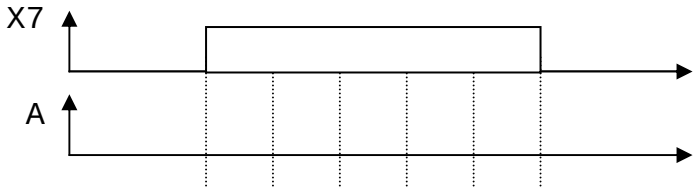
Equation de A : A=

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action



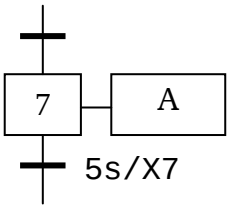


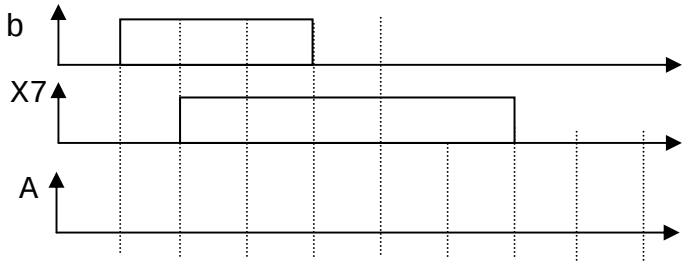
Equation de A : A=

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action



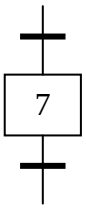


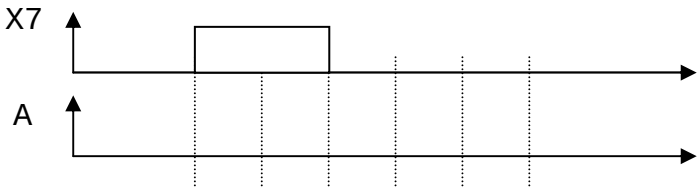
Equation de A : A=

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action



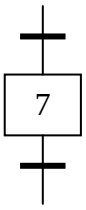


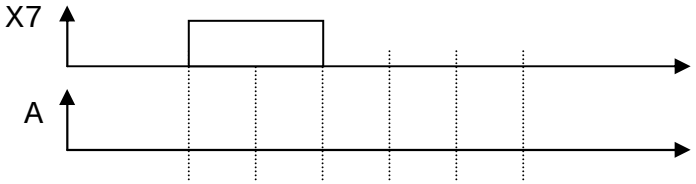
Equation de A :

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action



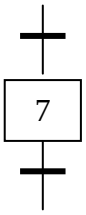


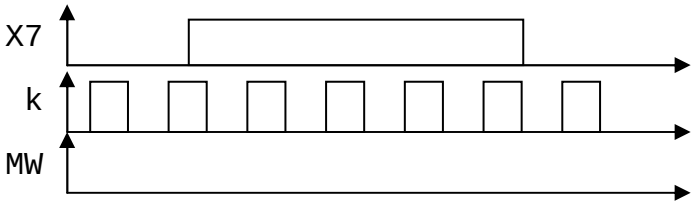
Equation de A :

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

Action





Equation de MW :

Signification: l'action A associée à l'étape 7 est effectuée .....

.....

si MW=12 avant l'activation de X7, donnez la valeur de MW au front descendant de X7 :

II.7. Représentation hiérarchisée de la Partie Commande (PC)

Principe de base :

La Partie commande est structurée en plusieurs niveaux hiérarchisés.
La représentation la plus simple utilise 2 niveaux :

- Un niveau dit LOCAL, définissant la Partie Commande Locale du processus
  On le notera :
- Un niveau dit HIERARCHIQUEMENT SUPERIEUR qui peut recevoir des informations du Pupitre Opérateur, de la Partie Opérative (PO), de la Partie Commande Locale, et émettre des ordres notamment vers la Partie Commande Locale.
  On le notera :

Les Grafcet associés à ces 2 niveaux peuvent être notés :

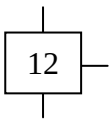
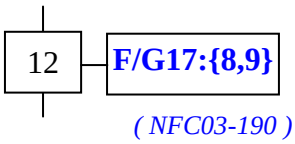
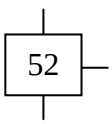
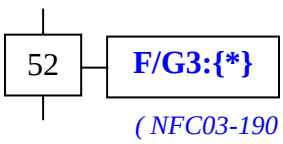
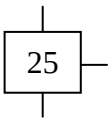
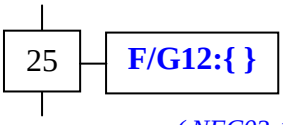
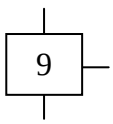
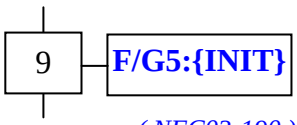
- Pour la PCL { GFNi Grafcet de Fonctionnement Normal i ; GPNi Grafcet de Production Normal i ; Un NOM explicite }
- Pour la PCH → GCH →

II.8. Structuration des Grafcets

a) Forçages

Le forçage est utilisé pour structurer la description d'un automatisme complexe.

Types de forçage et formalisme suivant la norme *EN60848 (norme NFC03-190 en rappel)*:

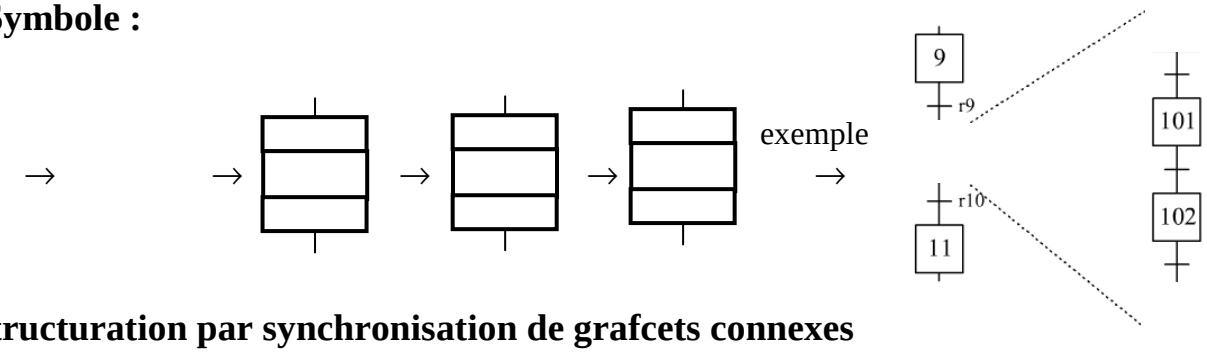
Forçage en situation	**Forçage en situation**	**Forçage en situation**	**Forçage en situation**
 *EN60848*  *(NFC03-190)*	 *EN60848*  *(NFC03-190)*	 *EN60848*  *(NFC03-190)*	 *EN60848*  *(NFC03-190)*
Lorsque l'étape 12 est active, le grafcet G17 est forcé dans la situation {8,9},	**Lorsque l'étape 52 est active, le grafcet G3 est figé dans sa situation courante.**	**Lorsque l'étape 25 est active, le grafcet G12 est forcé dans la situation vide (aucune étape active).**	**Lorsque l'étape 9 est active, le grafcet G5 est forcé dans sa situation initiale.**

b) Macro – étapes ; « ME »

Les macro – étapes permettent de structurer la description du fonctionnement, à chaque macro – étape correspond une expansion unique.

- **Règles d'évolution :**

- **Symbole :**

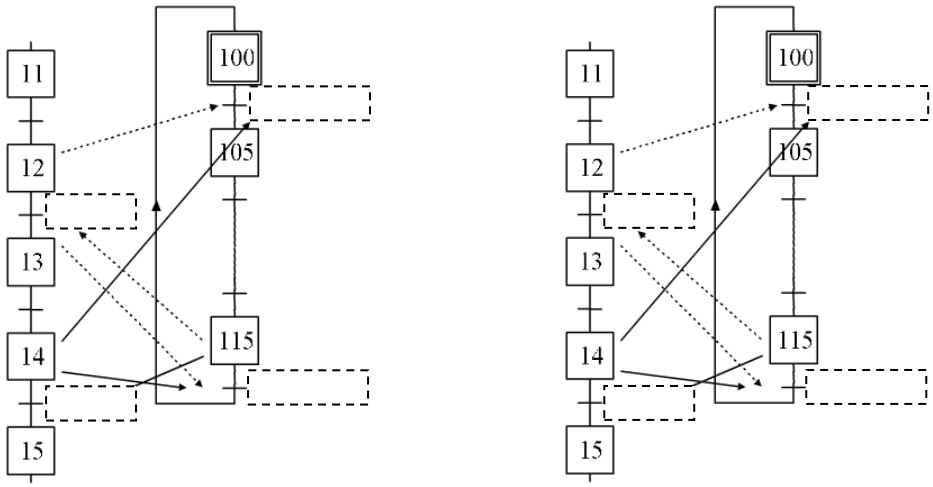


c) Structuration par synchronisation de grafcets connexes

(remplace la notion de « Tache » de la norme *NFC03-190*)

La structuration par synchronisation de grafcets connexes est surtout utile lorsque une même tache (sous – programme) est appelée plusieurs fois par un ou plusieurs grafcets de niveau hiérarchiquement supérieur.

D'après EN60848 *(NFC03-190)*



Dans l'exemple ci-dessus, le grafcet connexe évolue lorsque la réceptivité « » est vraie c'est-à-dire lorsque l'une des deux étapes d'appel est active.

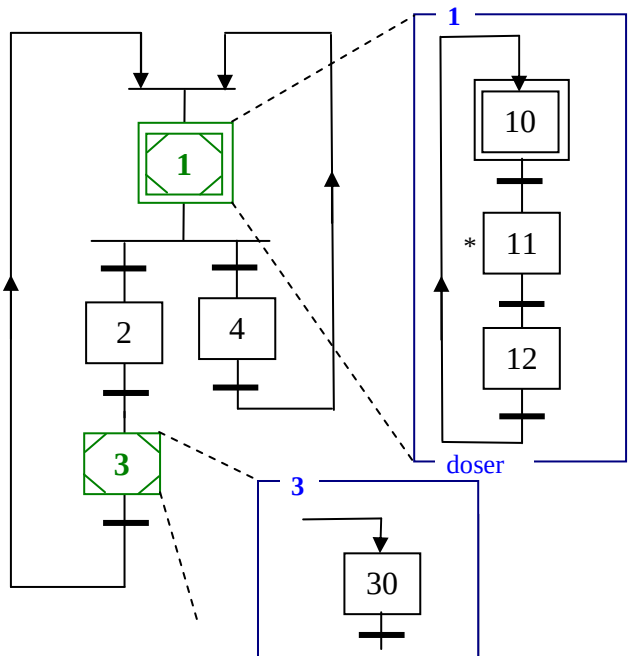
Une étape de synchronisation à la fin du graphe « » permet d'informer le grafcet appelant que la tache est terminée. Cette information est utilisée dans la réceptivité aval à l'appel du sous – programme. le grafcet connexe peut revenir dans son état initial lorsque la réceptivité « » est vraie c'est-à-dire lorsqu'aucune des deux étapes d'appel n'est active.

➢

d) Encapsulation *(n'existait pas dans la norme NFC03-190)*

L'évolution la plus importante de la norme est la notion d'encapsulation, cette notion ajoute un nouvel outil permettant la structuration des systèmes automatisés complexes aux outils précédents (macro – étape , synchronisation de grafcets connexes, forçage).

o Symbole :



Remarque :
On distingue aussi la notion d'étape encapsulante initiale, une étape encapsulante est initiale lorsque le grafcet encapsulé possède au moins une étape initiale.
Ici l'étape 1 est une étape encapsulante initiale.

o Règles d'évolution :

Remarque sur activation d'une étape encapsulante initiale *(ici X1)* **:**

- **Sur reprise énergie ou forçage d'une étape encapsulante initiale on active toutes les étapes initiales associées** *(ici X1 et X1/X10 s'activent)*
- **Dans tous les autres cas on applique la règle générale énoncée ci-dessus** *(ici X1 et X1/X11 s'activent)***.**

o Désignation d'une étape d'un grafcet encapsulé.

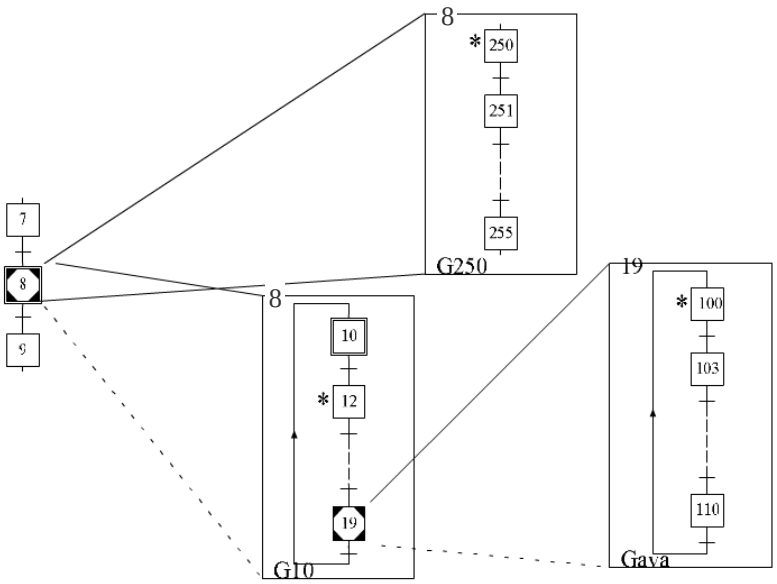
Un grafcet encapsulé est désigné par X*/G# ou

- X* désigne l'étape encapsulante
- et G# le nom du grafcet encapsulé (on peut, s'il n'y a pas d'ambiguïté le designer directement par G#).

Une étape d'un grafcet encapsulé est désigné par X*/X# ou

- X* désigne l'étape encapsulante
- et X# l'étape encapsulée, s'il n'y a pas ambiguïté on peut directement la nommer X#.

➢ Compléter ci-contre la description des Grafcets ci-dessous :



Les grafcets G250 et G10 sont deux grafcets dans l'étape .
Le grafcet G10 comportant une étape initiale, l'étape 8 est une étape .
Ce grafcet est désigné par .
Le grafcet Gava est encapsulé dans l'étape du grafcet .
L'activation de l'étape X8 entraîne l'activation des étapes de et de .
Ces étapes sont désignés par et .
L'étape X100 de Gava est activée par l'activation de de .
cet étape est désignée par (étape X100 du grafcet encapsulé dans X19 du grafcet encapsulé dans X8)

➢ Exemple d'application :

Une encapsulation se comporte comme une macro – étape lors de l'activation, en effet, l'activation de l'étape encapsulante entraîne l'activation des étapes sélectionnées (par l'astérisque) dans le grafcet encapsulé (il peut il y avoir plusieurs étapes activés contrairement à l'expansion d'une macro – étape,).
La désactivation du grafcet encapsulé est équivalent à un forçage dans l'état vide du grafcet encapsulé, cette désactivation intervient dès la désactivation de l'étape encapsulante et une simple évolution du grafcet réalise cette action.
Il devient très facile de programmer un arrêt d'urgence d'un cycle complexe à partir d'une encapsulation (Cf. ci – contre) .

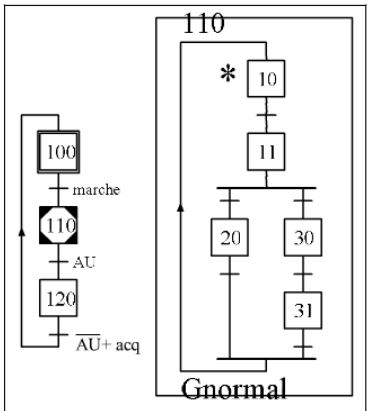


Figure 4 Arrêt d'urgence

III. ALGORITHME, ORGANIGRAMME

Présentation

En automatique, l'algorithme ou l'organigramme est un outils qui permet la description d'un problème posé *(ou d'une partie seulement)* en vue de sa résolution *(généralement programmé).*

III.1. L'Algorithme

1. Définition

Un algorithme est un ensemble fini de règles déterminées servant à résoudre un problème au moyen d'opérations élémentaires.

Exemple : l'algorithme permettant le calcul de (A+B)× C+D est le suivant :

1- Additionner A avec B ;
2- Multiplier la somme obtenue par C ;
3- Additionner le résultat obtenu à D ;

2. Formalisme

- Tout algorithme commence par un <u>début</u> et se ferme par une <u>fin</u>

DEBUT
- Additionner A avec B ;
- Multiplier la somme obtenue par C ;
- Additionner le résultat obtenu à D ;

FIN

- Tout algorithme peut-être mis sous la forme de trois structures fondamentales.

a- structure séquentielle

Les actions se déroulent les unes à la suite des autres.

DEBUT
- **Action-1**
- **Action-2**
- **.....**
- **Action-n**

FIN

b- structure sélective

Elle permet à l'algorithme de prendre une décision entre les actions à effectuer selon une condition de réalisation.

DEBUT
-
-
SI *(IF)* <condition> **ALORS *(THEN)*** <action-1>
SINON *(ELSE)* <action-2>
FIN_SI *(END_IF)*
-
-
FIN

c- Structure répétitive (ou itérative)

Elle est utilisée chaque fois que les actions doivent être répétées. Une condition détermine l'arrêt de la répétition.

Syntaxe 1 : on ne connaît pas le nombre de répétitions.

DEBUT
-
-
TANT QUE *(WHILE)* <condition> **FAIRE *(DO)*** <action(s)>
FIN_TANT QUE *(END_WHILE)*
-
-
FIN

Ou :

DEBUT
-
-
REPETER *(REPEAT)* <action(s)>
JUSQU'A *(UNTIL)* <condition>
FIN_REPETER *(END_ REPEAT)*
-
-
FIN

Syntaxe 2 : on connaît le nombre de répétition au départ.

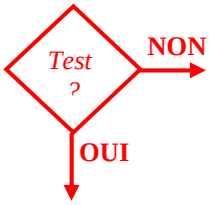
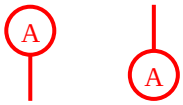
DEBUT
-
-
POUR *(FOR)* <variable=valeur de départ> **A *(TO)*** <valeur finale>
FAIRE *(DO)* <action(s)>
FIN_POUR *(END_FOR)*
-
-
FIN

III.2. L'Organigramme

1. Définition

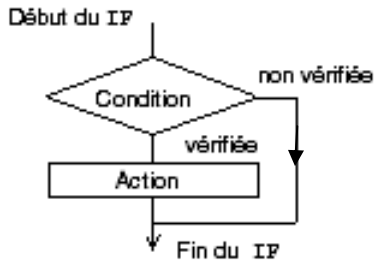
C'est une variante de l'algorithme utilisant une représentation graphique des étapes d'un algorithme à l'aide des symboles normalisés désignant chacun une opération spécifique.

2. Symboles normalisés

Symbole	Désignation
	Une opération de début ou de fin du traitement.
	Une opération d'entrée (lire) ou de sortie (écrire, affichage
	- Si la condition est vérifiée (vraie) l'alternative OUI est exécutée - Si la condition n'est pas vérifiée (fausse), le traitement est poursuivi sur l'alternance Non
	Appel de sous programme
	Pour toute opération de calcul autre que celles citées précédemment.
	Symbole de renvoie et suite.

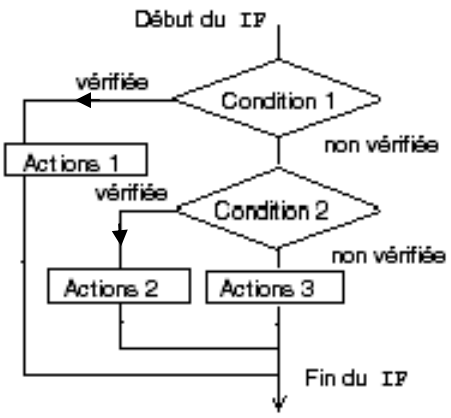
3. Synthèse : algorithme / Organigramme / langage littéral structuré

```
IF condition THEN
     actions ;
END_IF;
```



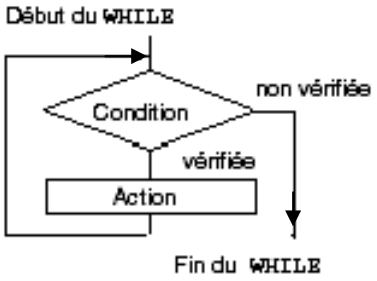

```
IF %M0 AND %M12 THEN
     RESET %M0;
     INC %MW4;
     %MW10:=%MW8+%MW9;
END_IF;
```

```
IF condition1 THEN
     actions1;
ELSEIF condition2 THEN
     actions2;
ELSE
     actions3;
END_IF;
```



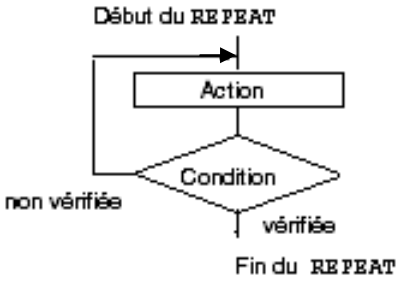

```
IF %M0 AND %M1 THEN
     %MW5:=%MW3+%MW4;
     SET %M10;
ELSEIF %M0 OR %M1 THEN
     %MW5:=%MW3-%MW4;
     SET %M11;
ELSE
     RESET %M10;
     RESET %M11;
END_IF;
```

```
WHILE condition DO
     action ;
END_WHILE;
```



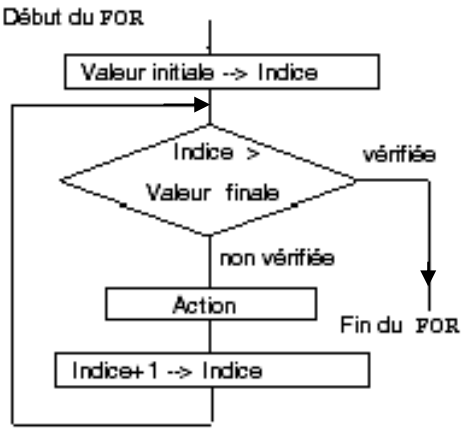

```
WHILE %MW4<12 DO
     INC %MW4;
     SET %M25[%MW4];
END_WHILE;
```

```
REPEAT
     action ;
UNTIL condition END_REPEAT;
```




```
REPEAT
     INC %MW4;
     SET %M25[%MW4];
UNTIL %MW4>12 END_REPEAT;
```

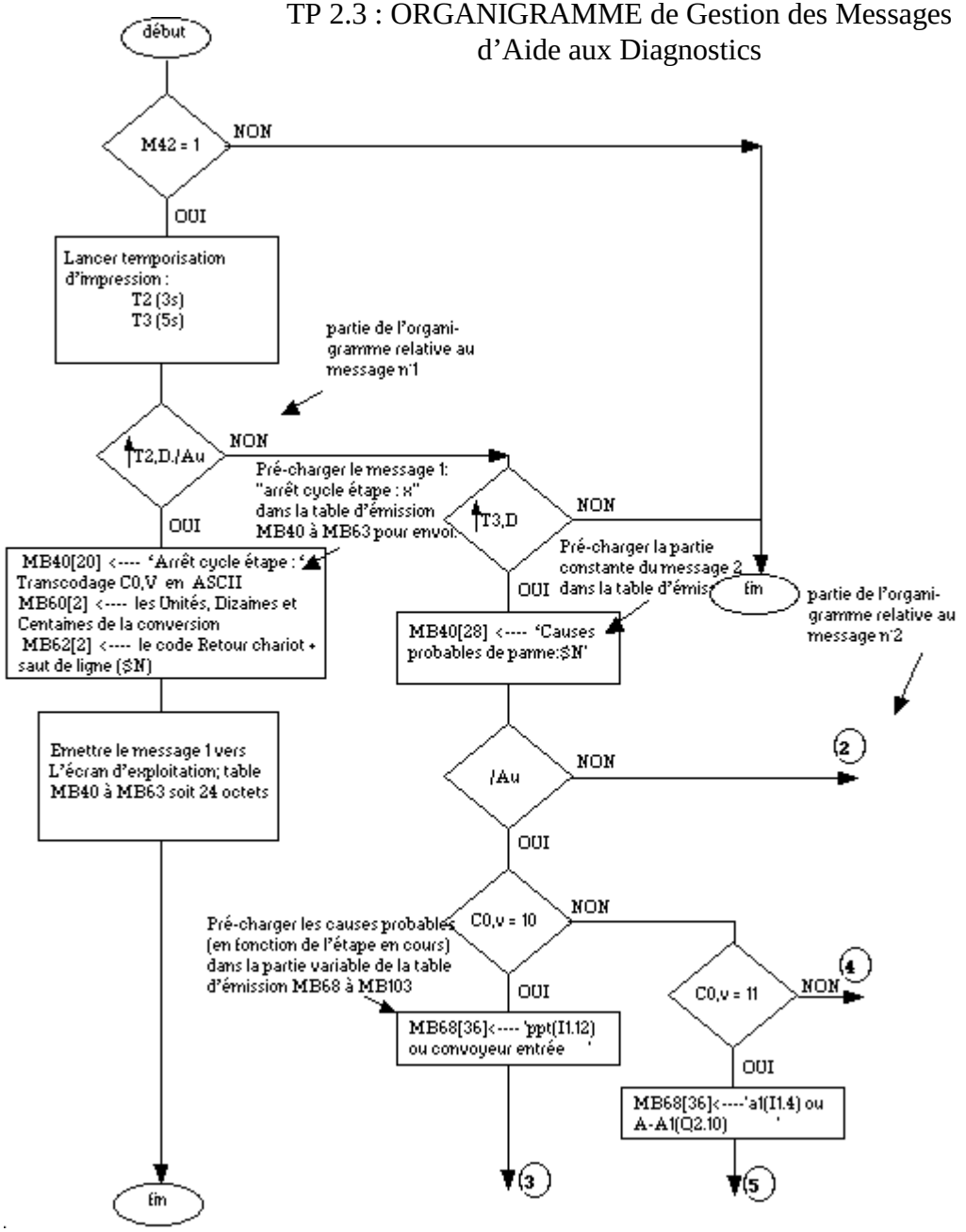
```
FOR indice:=valeur initiale TO valeur finale DO
     action ;
END_FOR;
```

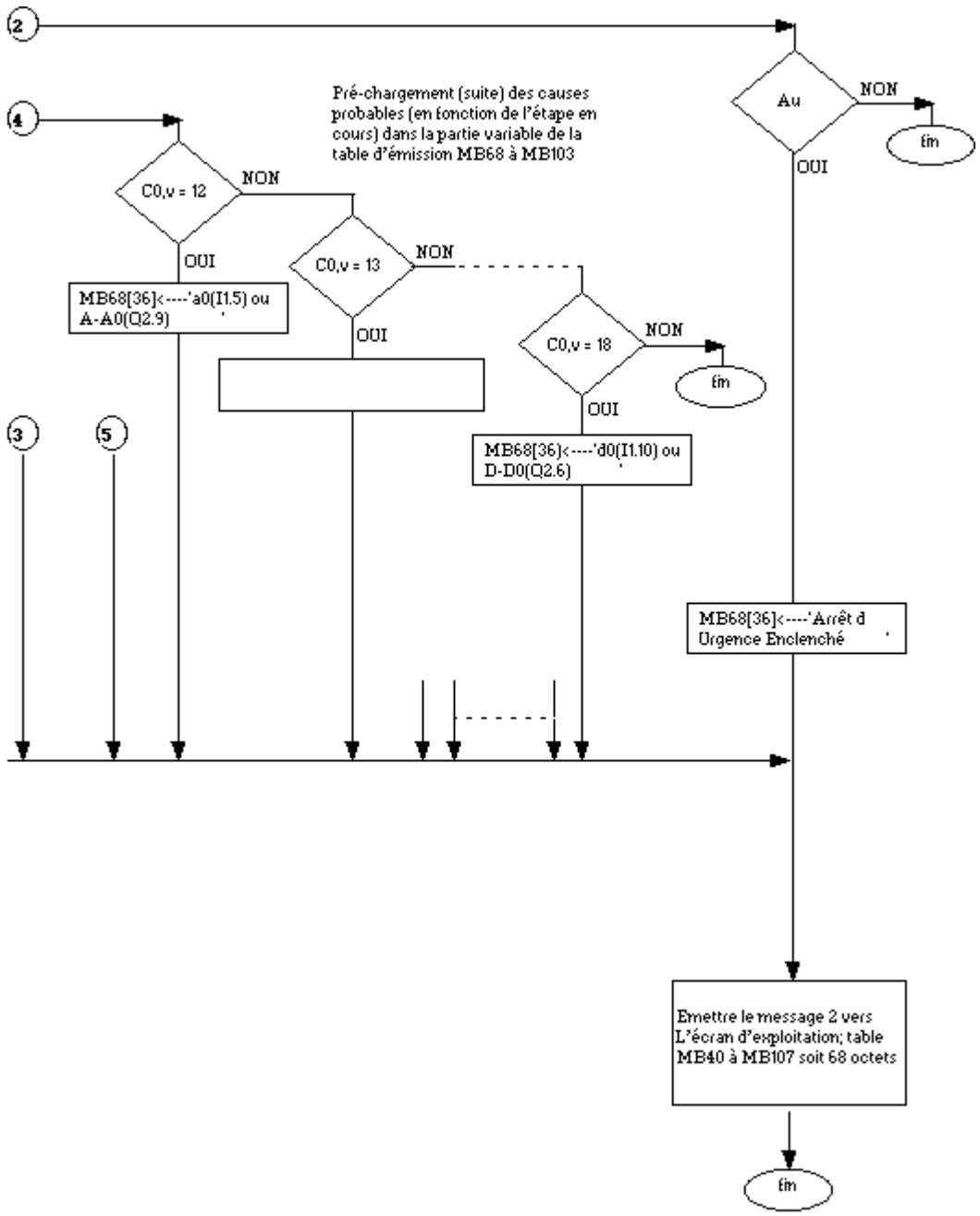


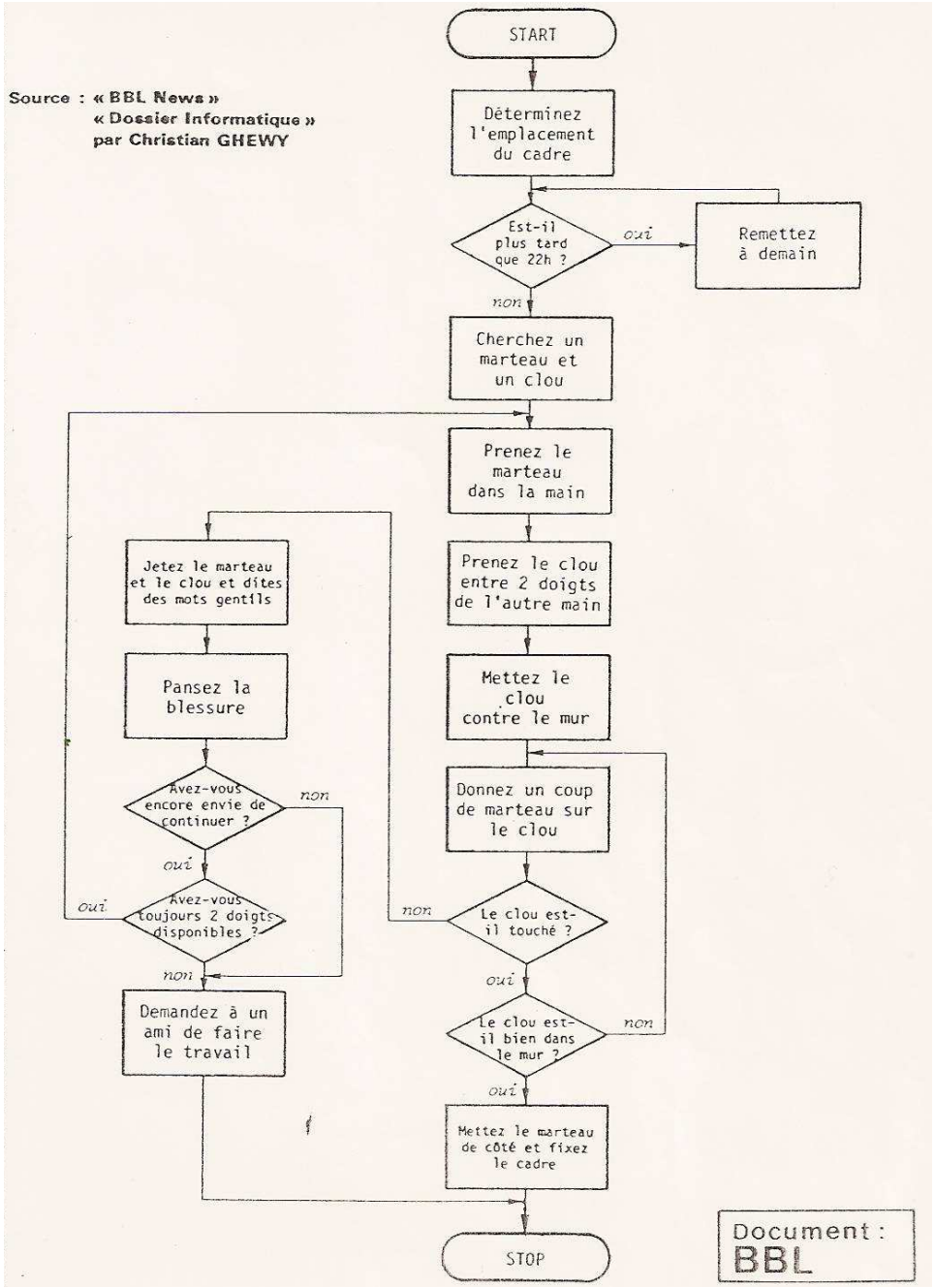

```
FOR %MW4=0 TO %MW23+12 DO
     SET %M25[%MW4];
END_FOR;
```

4. Exemples d'application :

TP 2.3 : ORGANIGRAMME de Gestion des Messages d'Aide aux Diagnostics







5. Conclusion

Cette méthode universelle est surtout utilisée :

- **Pour décrire des procédures à suivre**
- **Pour définir des programmes en informatique**
- **Pour définir des programmes en automatique dans les cas suivants :**
  - **Calculs**
  - **Recherche et diagnostic**
  - **Langage littéral structuré**
  - **….**