

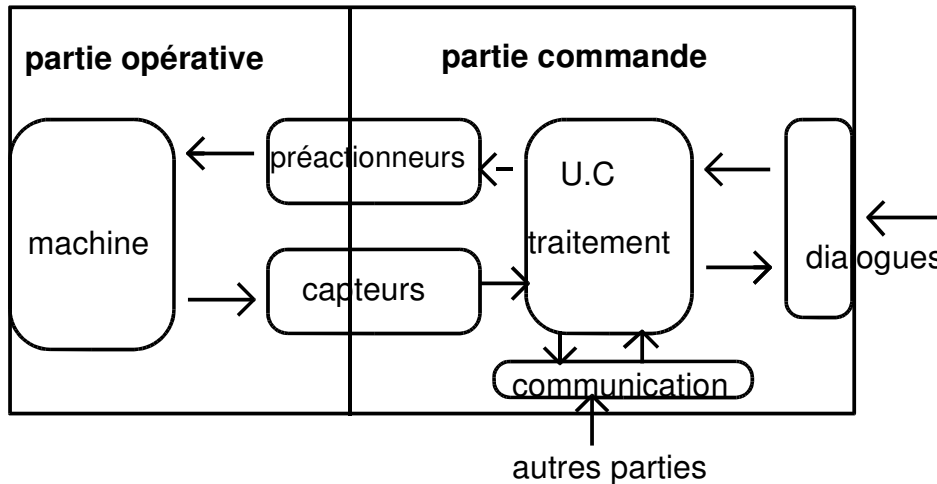
TD1 Le langage FBD de la norme 1131-3

I - Système automatisé

Généralités

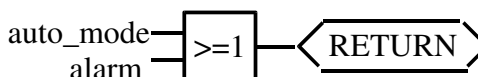
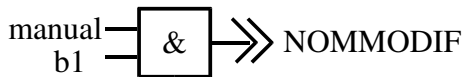
Un système automatisé se décompose en deux parties dépendantes :

- la **partie opérative** est le processus à automatiser
- la **partie commande** est l'automatisme qui en fonction des entrées (information externe venant de la partie opérative, consignes extérieures, etc.....) élabore en sortie des ordres externes destinés à la partie opérative ou à des éléments extérieurs.

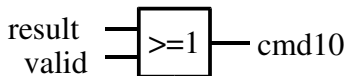


II) Le langage FBD (Function Block Diagram)

Le langage FBD (Function Block Diagram) est un langage graphique qui ressemble à la schématique électronique. Il y a quand même quelques différences :



NOMMODIF:



NOMMODIF:

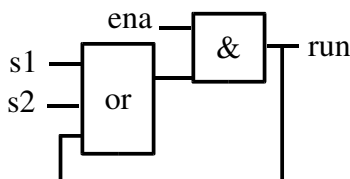


Saut conditionnel

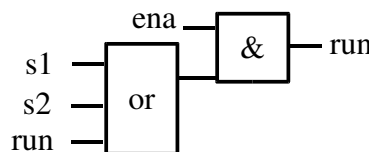
Retour conditionnel

Cela signifie que l'on peut donner un nom à un sous-graphe.

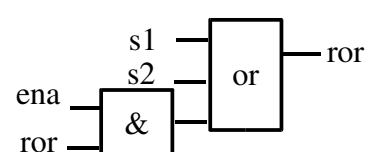
Il faut éviter cependant les boucles car il n'y a aucune information sur l'ordre d'exécution dans le réseau dessiné.



Boucle à éviter



1ère option



2ème option

Il faut donc garder à l'esprit qu'il s'agit d'un programme et non d'un schéma destiné à être implanter matériellement comme en électronique.

III) Ecrire et utiliser son propre package pour implanter FBD en VHDL

Lors du TD2 de a2i11 (assemblage de fonctions) nous avons passé sous silence le fait que pour faire la description structurelle présentée il fallait utiliser notre propre package définissant ce qu'est un un et, un ou et un inverseur. Nous écrivons ci-dessous la version complète du programme.

	<pre> LIBRARY portes; --portes.vif en WARP. Utiliser --library manager pour dire où est ce fichier USE portes.mesportes.ALL; ENTITY Fct IS PORT(e0,e1,e2 : IN BIT; s : OUT BIT); END Fct; ARCHITECTURE truc OF Fct IS SIGNAL e0e1,e2bar : BIT; BEGIN i1:et PORT MAP(e0,e1,e0e1); i2:inverseur PORT MAP(e2,e2bar); i3:ou PORT MAP(e0e1,e2bar,s); END truc; </pre>
--	---

Voici en condensé comment on réalise un package :

PACKAGE mesportes IS		
COMPONENT et PORT(e0,e1 : IN BIT; s : OUT BIT); END COMPONENT;	COMPONENT ou PORT(e0,e1 : IN BIT; s : OUT BIT); END COMPONENT;	COMPONENT inverseur PORT(e : IN BIT; s : OUT BIT); END COMPONENT;
END mesportes;		

```

ENTITY et IS
PORT(e0,e1 : IN BIT;
      s : OUT BIT);
END et;

ARCHITECTURE aet OF et
IS
BEGIN
    s<=e0 AND e1;
END aet;
    
```

```

ENTITY ou IS
PORT(e0,e1 : IN BIT;
      s : OUT BIT);
END ou;

ARCHITECTURE aou OF ou
IS
BEGIN
    s<=e0 OR e1;
END aou;
    
```

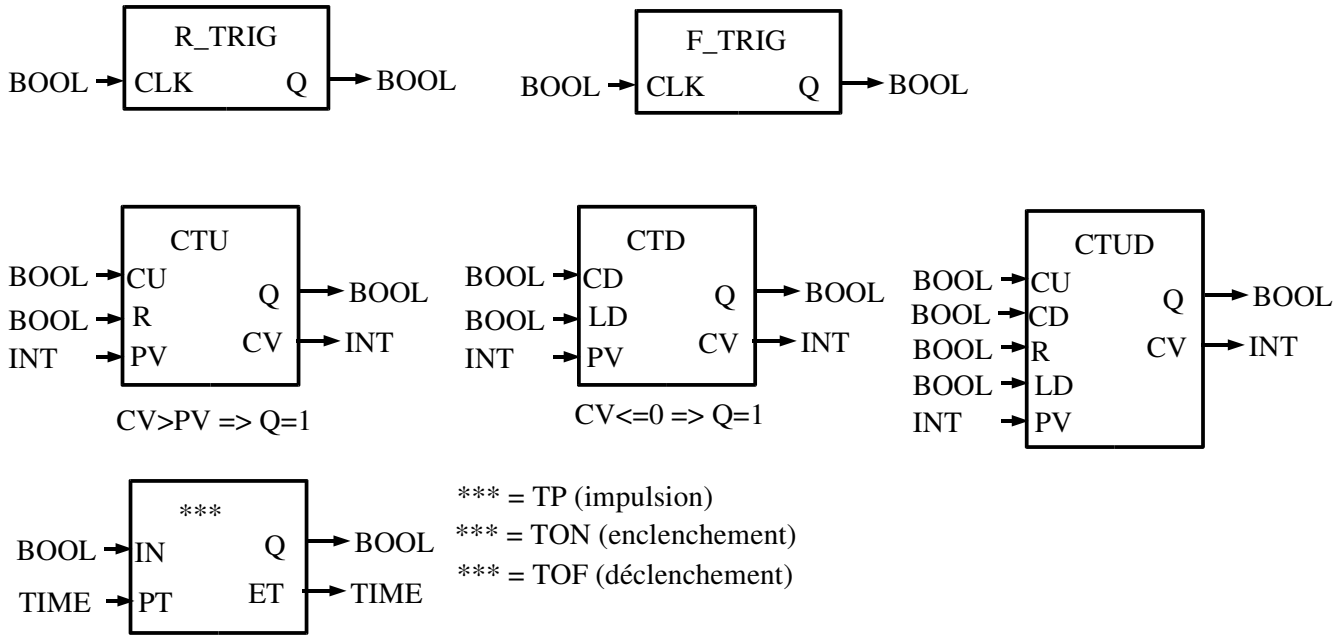
```

ENTITY inverseur IS
PORT(e : IN BIT;
      s : OUT BIT);
END inverseur;

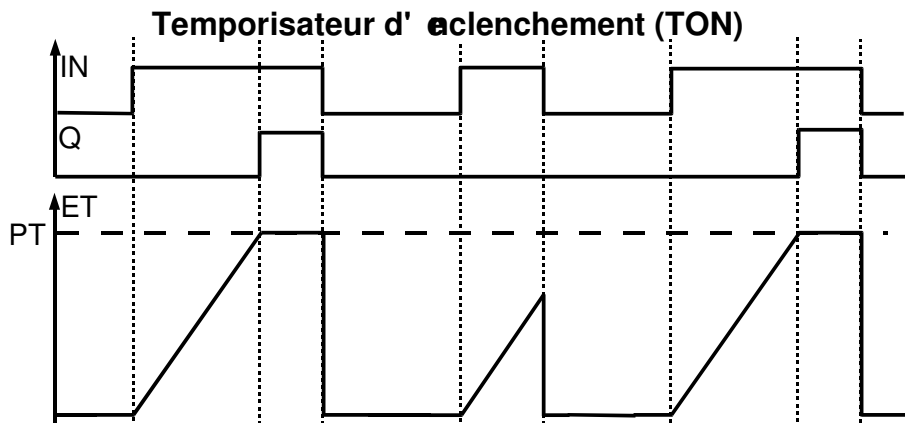
ARCHITECTURE ainv OF
inverseur IS
BEGIN
    s<= NOT e;
END ainv;
    
```

Il n'y a aucun standard sur les packages du type ci-dessus. Chaque constructeur propose son propre package pour programmer ses composants ce qui pose un problème de portabilité. A noter quand même une initiative avec LPM (Library of Parameterized Modules) (voir plus loin).

Quelques blocs fonctionnels standards (de la norme 1131-3)



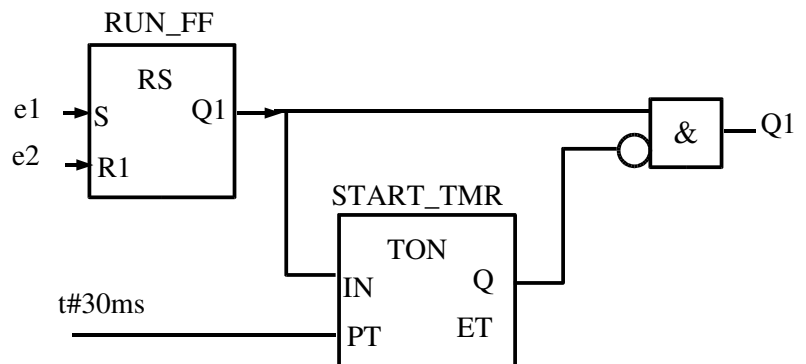
Dans l'esprit de la norme, les compteurs comptent ou décomptent toujours : PV n'est pas un max mais un seuil qui quand il est dépassé passe la sortie à 1. Mais l'envoi d'un front augmentera toujours CV pour CTU.



Citons d'autres blocs fonctionnels : bistable RS et SR, sémaphore (claim:BOOL,release:Bool;Busy:BOOL).

Exercice

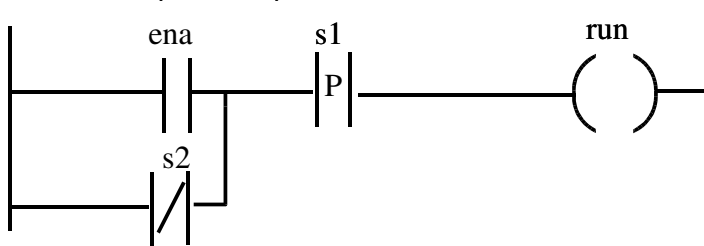
Implanter l'exemple ci-contre en VHDL. Remarquez l'instanciation des blocs fonctionnels. On dispose d'une horloge 10 kHz.



TD2 Retour sur le graphe d'état – Langage LD de la norme

I) Contacts et bobines

Le langage LD (Ladder Diagram) est plus connu chez nous comme langage à contacts. Comment implanter une fonction booléenne : en série = ET en parallèle = OU. Bien distinguer les interrupteurs qui sont des entrées et les bobines qui sont des sorties.



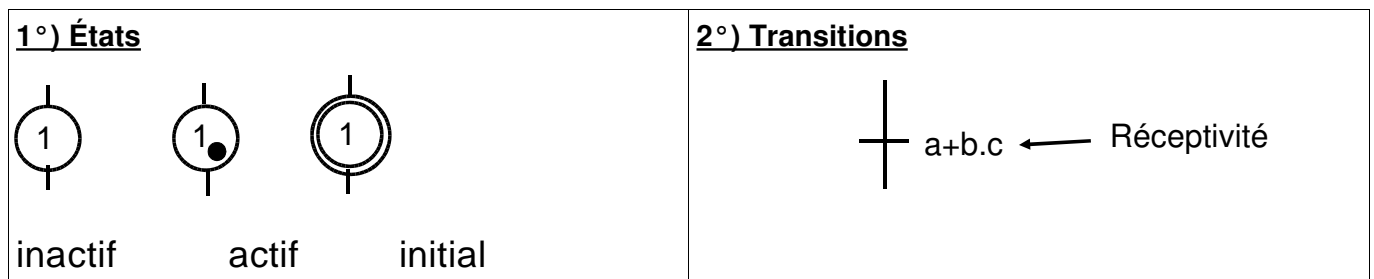
/ = NOT N=Negative pulse, P=Positive Pulse, S=Set, R=Reset. Les trois premiers modificateurs peuvent se trouver dans les interrupteurs et tous peuvent se trouver dans les bobines.



(voir TD 1) peuvent se trouver dans la partie bobine d'un programme LD. Cela signifie que l'on peut donner un nom à un sous-graphe.

II) Graphe d'états

Un graphe d'état est une suite d'états et de transitions réceptives.



III) Équations de récurrence

On cherche pour chacun des états i les conditions d'activations AC_i et les déactivations D_i puis on écrit :

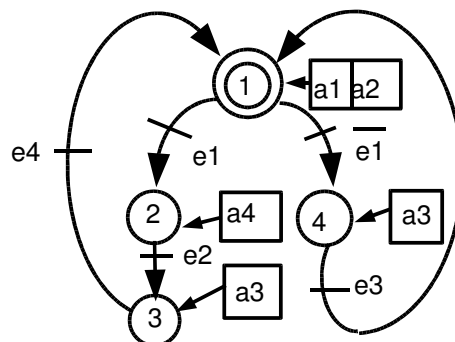
$$x_i^+ = AC_i + \overline{D_i} \cdot x_i + Init$$

pour un état initial et

$$x_i^+ = (AC_i + \overline{D_i} \cdot x_i) \cdot \overline{Init}$$

pour un état normal :

Exemple :



$AC1 = x3.e4+x4.e3$ $AC2 = x1.e1$ $AC3 = x2.e2$ $AC4 = x1./e1$	$D1 = e1+/e1=1$ $D2 = e2$ $D3 = e4$ $D4 = e3$
---	--

Équations de récurrences	Équations de sorties
$x1^+ = x3.e4+x4.e3 + \text{Init}$ $x2^+ = (x1.e1+x2./e2)./ \text{Init}$ $x3^+ = (x2.e2+x3./e4)./ \text{Init}$ $x4^+ = (x1./e1+x4./e3)./ \text{Init}$	$a1 = x1 \quad a2 = x1$ $a3 = x3 + x4$ $a4 = x2$

Exercice 1

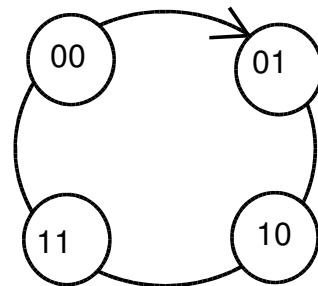
Implanter les équations de cet exemple en langage à contact.

IV) Un nouveau style de programmation VHDL

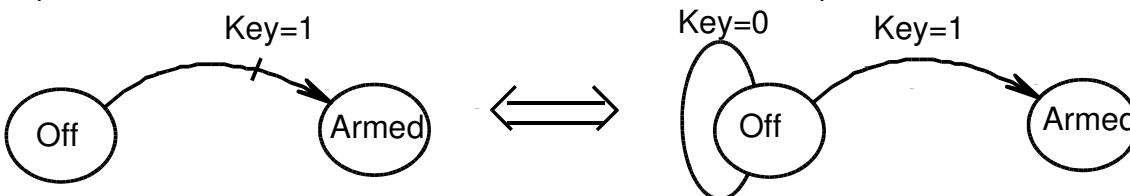
Le séquentiel simple (diagramme d'évolution) sans équations de récurrence

```
ENTITY demo IS PORT(
  clock : IN BIT;
  q : INOUT BIT_VECTOR(0 TO 1));
END demo;
```

```
ARCHITECTURE mydemo OF demo IS
  BEGIN
    PROCESS(clock) BEGIN
      IF clock'EVENT AND clock='1' THEN
        CASE q IS --style case when
          WHEN "00" => q <="01";
          WHEN "01" => q <="10";
          WHEN "10" => q <="11";
          WHEN OTHERS => q <="00" ;
        END CASE;
      END IF;
    END PROCESS;
  END mydemo;
```



Nous avons déjà eu l'occasion de parler de ce problème en a2i11 (voir Tds 15) avec des équations de récurrence. Présentons d'abord la technique avec et sans initialisation.



```
-- sans initialisation
BEGIN
  PROCESS (clock) BEGIN
```

```

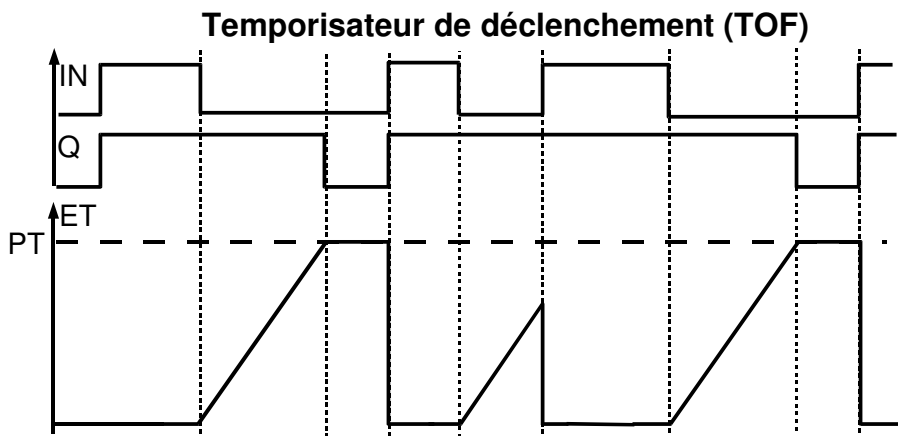
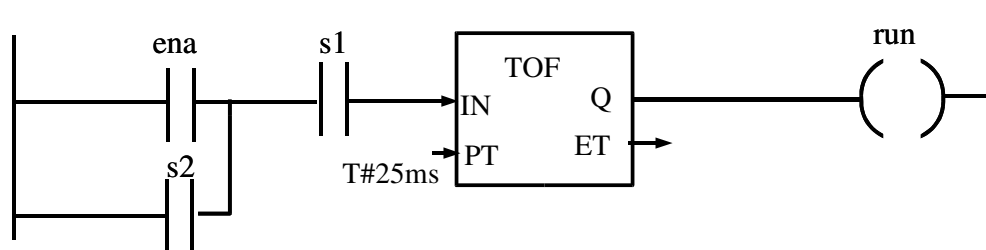
IF clock'EVENT AND clock='1' THEN
  CASE etat IS
    WHEN Off => IF key ='1' THEN etat <= Armed;
                ELSE etat <= Off;
                END IF;
    ....
  END CASE;
END IF;
END PROCESS;
....

-- avec initialisation synchrone
BEGIN
PROCESS (clock) BEGIN
  IF clock'EVENT AND clock='1' THEN
    IF Init='1' THEN etat <=Off; --initialisation synchrone
    ELSE
      CASE etat IS
        WHEN Off => IF key ='1' THEN etat <= Armed;
                    ELSE etat <= Off;
                    END IF;
        ....
      END CASE;
    END IF
  END IF;
END PROCESS;
....

```

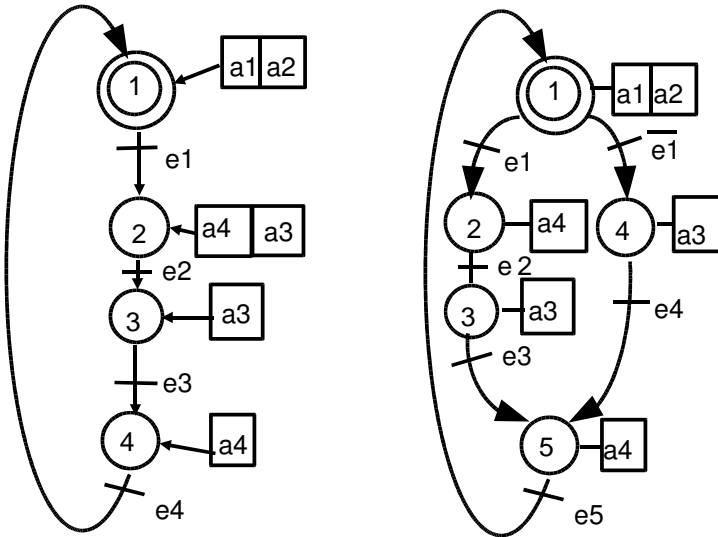
Exercice 2

- 1) Déclarer en FBD le programme ci-dessous.
- 2) Réaliser en VHDL le cahier des charges ci-dessous exprimé en langage à contact. On dispose pour cela d'une horloge à 25 Mhz. Réaliser d'abord un component combinatoire.



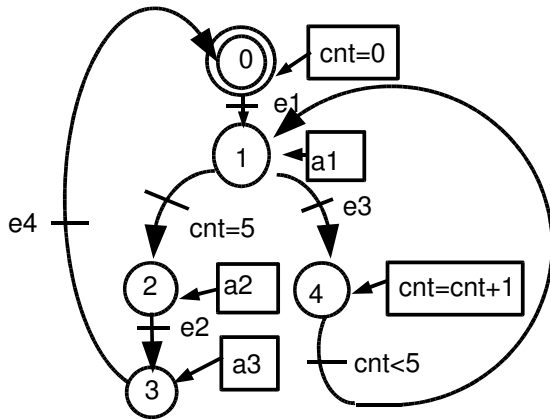
Exercice 3

- 1) Déclarer en FBD le programme ci-dessous.
- 2) Donner les équations de récurrence, en langage LD, de ces graphes d'états :
- 3) Ecrire le programme VHDL pour le premier graphe d'état en utilisant un case.



Exercice 4

Déclarer en FBD le programme ci-dessous. Réaliser ce graphe d'état en VHDL en prenant soin de séparer le compteur et la partie séquentielle. On utilisera donc un composant pour la partie séquentielle et un autre pour le compteur.



TD3 Le langage SFC de la norme 1131-3

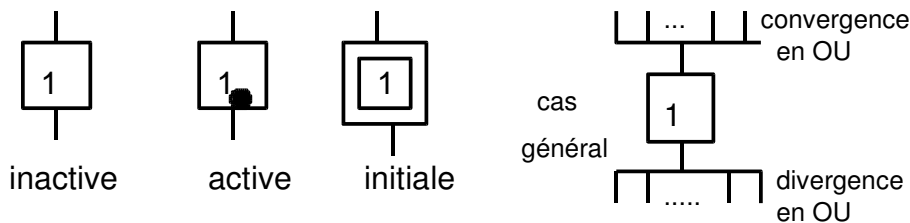
I) Le GRAFCET (règles d'établissement)

I-1) Etapes

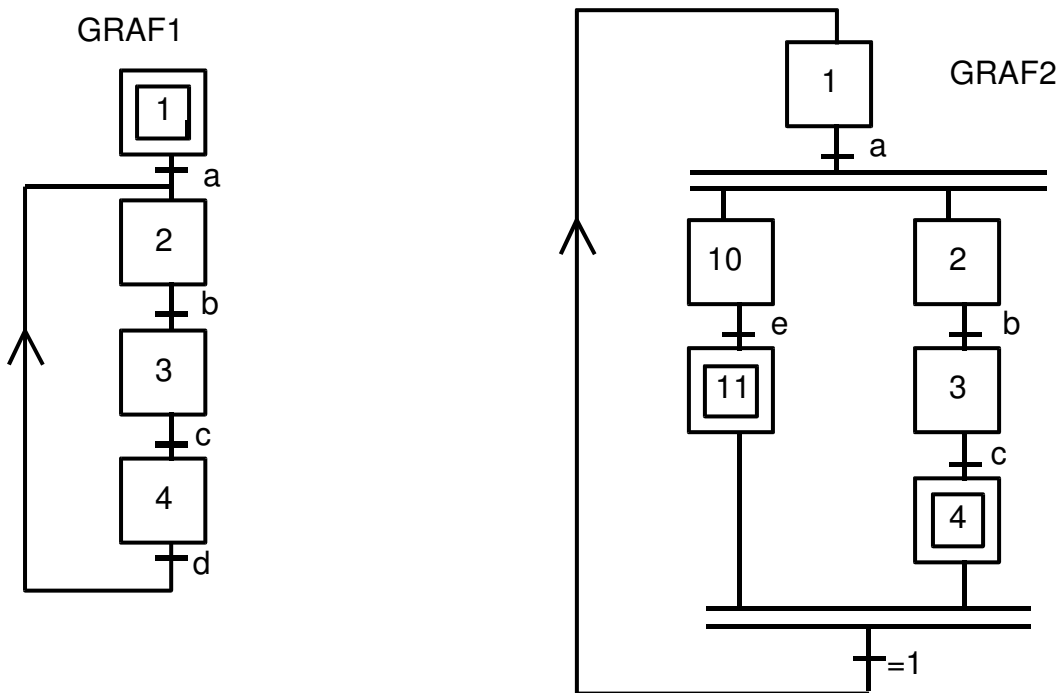
Une étape est représentée par un carré. Elle peut avoir deux états :

- état actif (une marque à l'intérieur du carré)
- état inactif

Règle 1 : La situation initiale d'un grafcet caractérise le comportement de la partie commande vis à vis de la partie opérative et correspond aux étapes actives au début du fonctionnement de la partie commande.

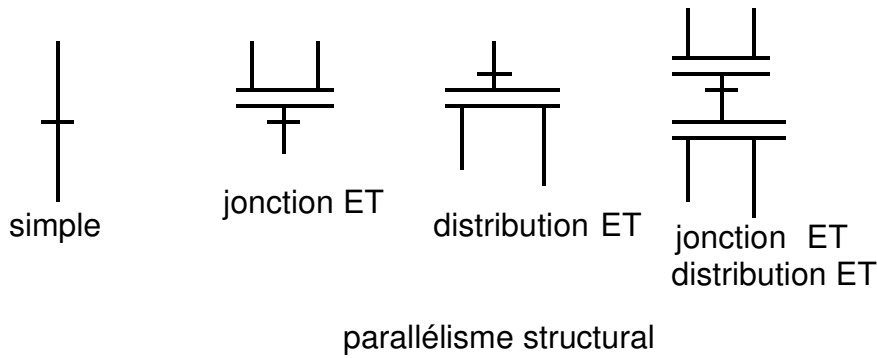


- Remarques :
- la situation initiale peut n'être obtenue qu'une seule fois à la mise sous tension
 - on peut utiliser plusieurs étapes initiales



I-2) Transitions

Une transition est représentée par un trait horizontal. A chaque transition est associée une réceptivité. Une transition est validée lorsque toutes les étapes immédiatement précédentes reliées à cette transition sont actives.



Règle 2 : l'évolution de la situation d'un grafcet correspondant au franchissement d'une transition ne peut se faire :

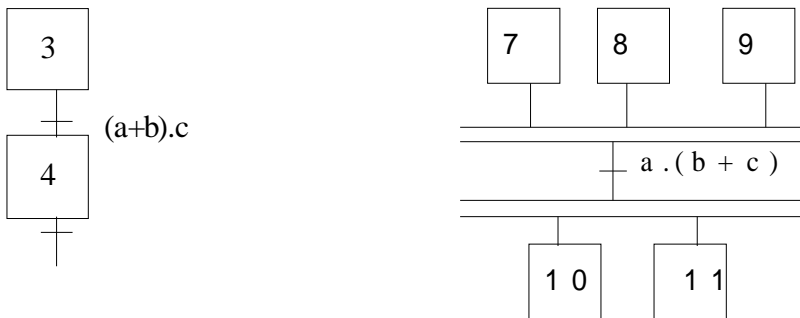
- que lorsque cette transition est validée
- **et que** la réceptivité associée à cette transition est vraie .

Lorsque ces **deux conditions** sont réunies, la transition devient franchissable et elle est obligatoirement franchie.

Règle 3 : Le franchissement d'une transition provoque :

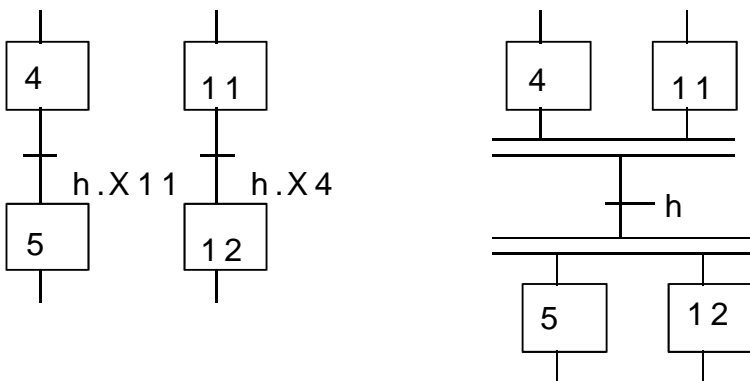
- la désactivation de toutes les étapes immédiatement précédentes reliées à cette transition .
- l'activation de toutes les étapes suivantes reliées à cette transition .

exemples :



Règle 4 : plusieurs transitions simultanément franchissables sont simultanément franchies

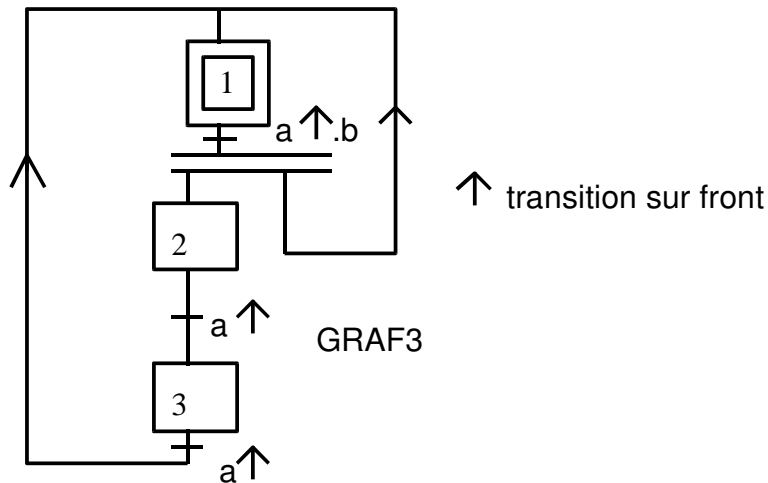
exemple : montrer que ces deux représentations sont équivalentes



Règle 5 : si au cours du fonctionnement de l'automatisme une même étape doit être

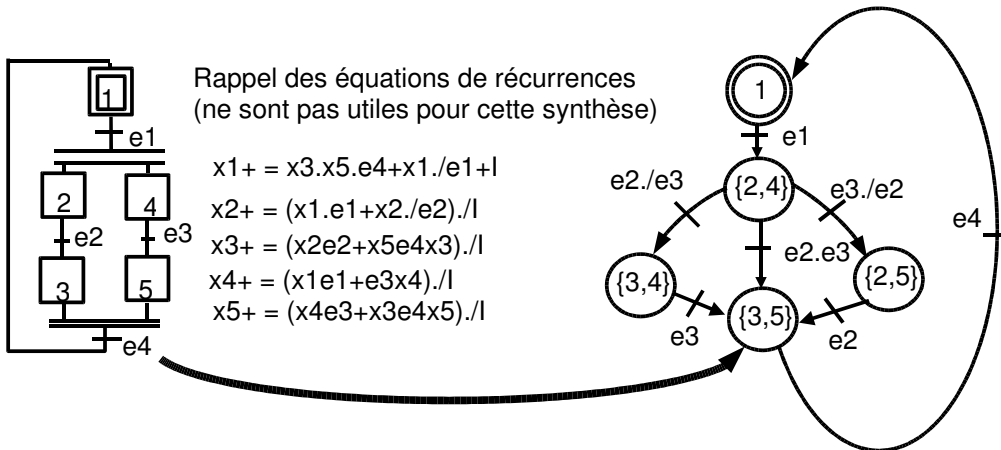
simultanément activée et désactivée, elle reste active.

exemple :



II) Equations de récurrences. Passage GRAFCET graphe d'états

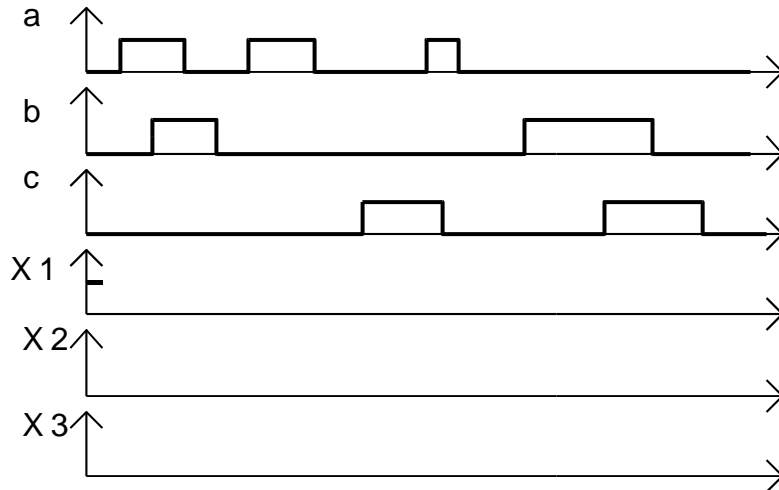
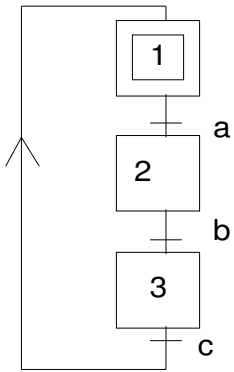
La différence entre un GRAFCET et un graphe d'état est le nombre de jetons : un seul pour le graphe d'état et un ou plusieurs pour le GRAFCET.



III) Exercices

Exercice 1

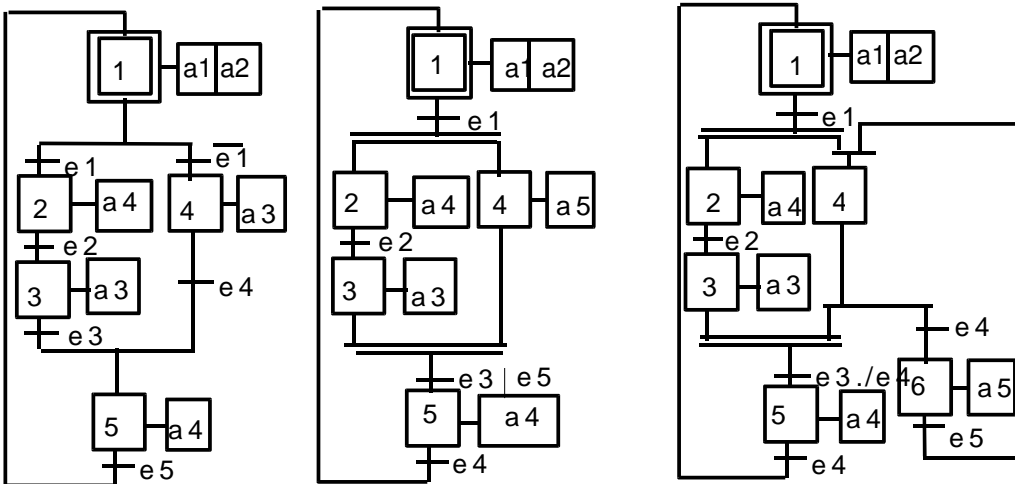
On considère le grafcet suivant, compléter le chronogramme



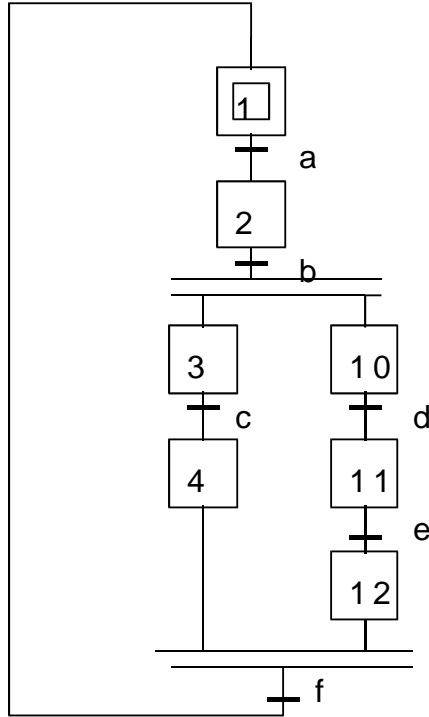
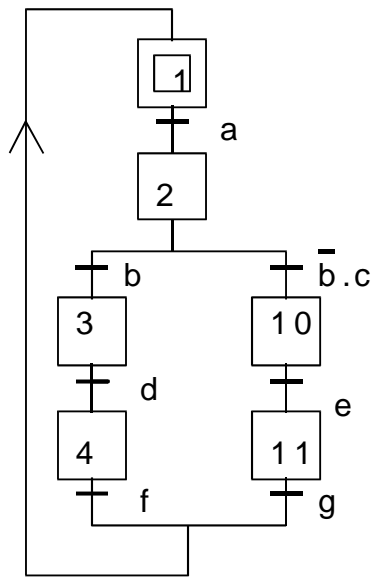
Quel est le nombre d'états possibles ?
Tracer un graphe des états.

Exercice 2

Construire un graphe d'état pour chacun des GRAFCETs ci-dessous. En déduire les équations de récurrence correspondantes puis celles des GRAFCETs.



Exercice 3 : On donne les deux grafjets ci-dessous :



Déclarer en FBD les programmes ci-dessus.

Tracer les graphes des états.

Quel est le nombre d'états possibles dans les deux cas ?

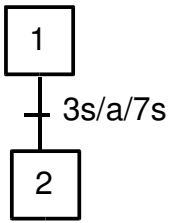
Que se passe-t-il si l'on remplace la transition /b.c par c ? Commenter.

Que se passe-t-il si l'on remplace la convergence en ET par une convergence en OU ?

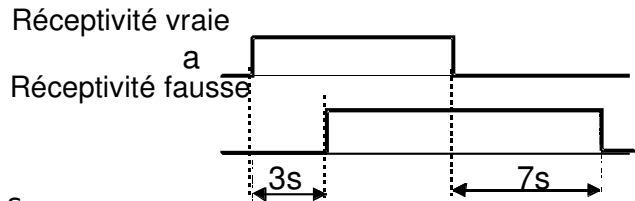
TD4 ARS2 Transitions, Actions et temps

I) Transitions temporisées

Réceptivité dépendante du temps

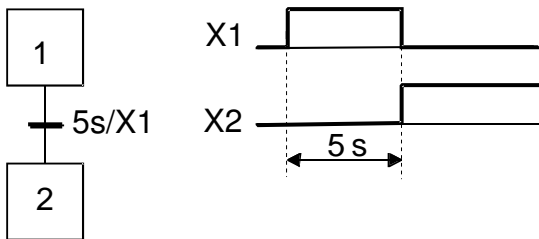


La notation est de la forme «**t1/variable/t2**». Dans l'exemple ci-contre, la réceptivité n'est vraie que 3 s après que « a » passe de l'état 0 à l'état 1, elle ne redevient fausse que 7 s après que « a » passe de l'état 1 à l'état 0.



Simplification usuelle

L'utilisation la plus courante est la temporisation de la variable d'étape avec un temps t2 égal à zéro :



Dans ce cas la durée d'activité de l'étape 1 est de 5 s.

Remarque : Il est possible d'utiliser cette notation lorsque l'étape temporisée n'est pas l'étape amont de la transition. La norme propose la notation :

$X1.t > t\#5s$

II) SFC (Sequentiel Function Chart) : généralités sur les actions

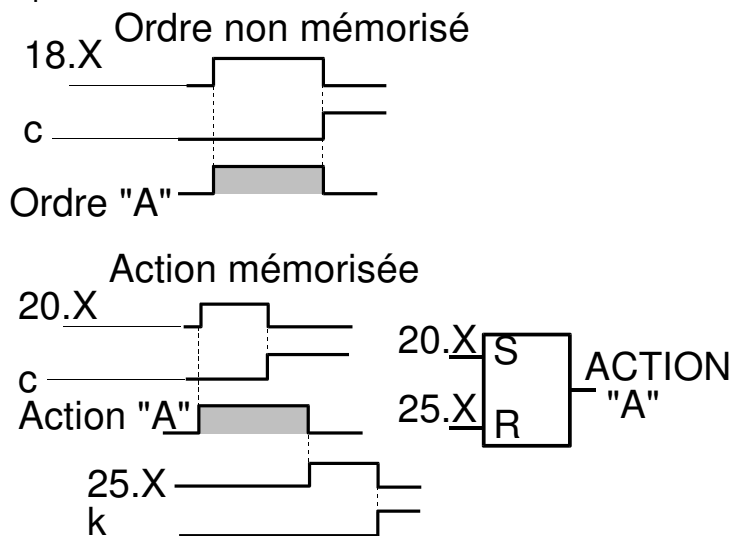
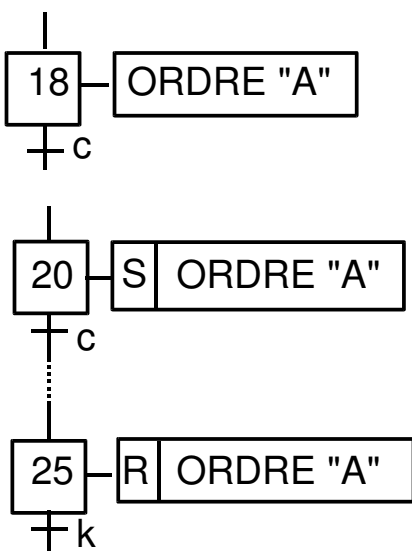
Norme CEI 848 (1988) complétée par 1131-3 (1993)

Ordre (action) détaillée :

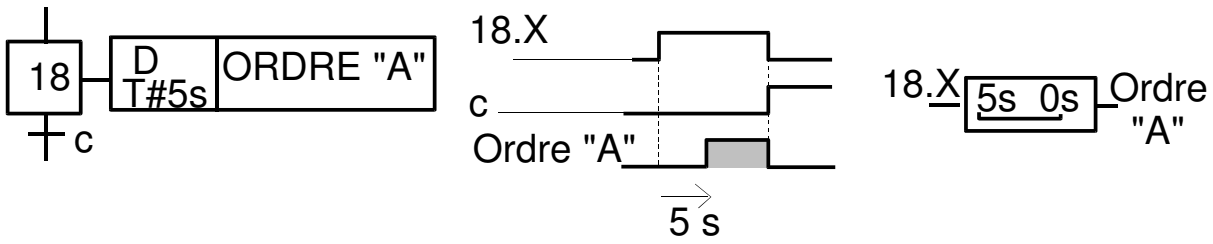
- La section a contient une lettre symbole ou une combinaison de lettres symboles décrivant comment le signal binaire de l'étape sera traité (N=Normal, P=Pulse, S= set R = Reset).
- La section b contient une déclaration symbolique ou littérale décrivant l'ordre ou l'action.
- La section c indique le numéro de référence du signal de fin d'exécution correspondant

NOTES

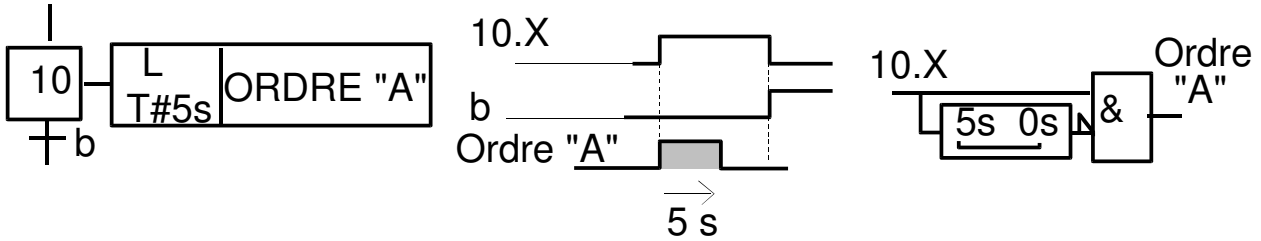
1. La section b doit être au moins deux fois plus grande que les sections a et c.
2. Les sections a et c ne sont spécifiées que si nécessaire.



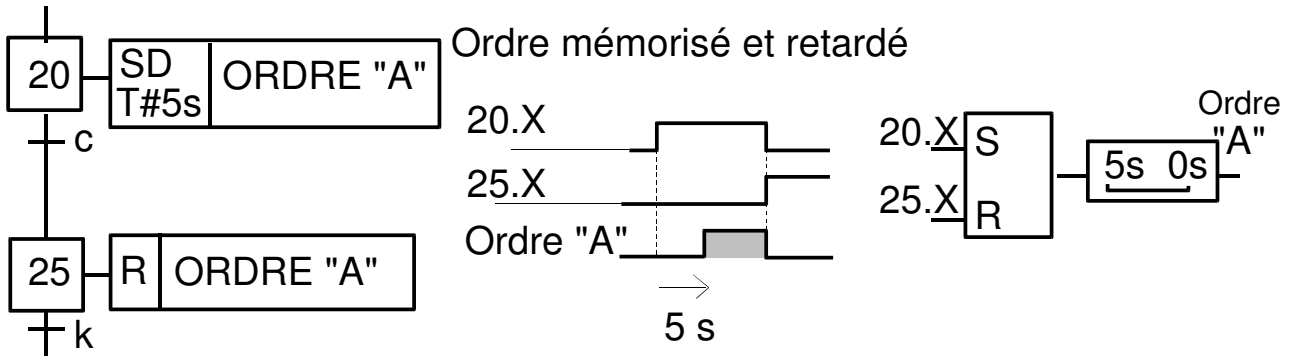
Ordre non mémorisé mais retardé



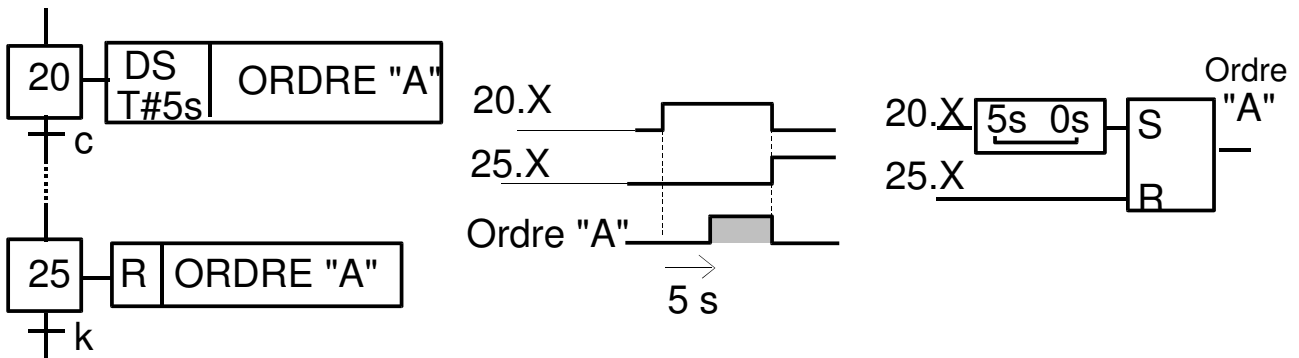
Ordre non mémorisé mais limité dans le temps



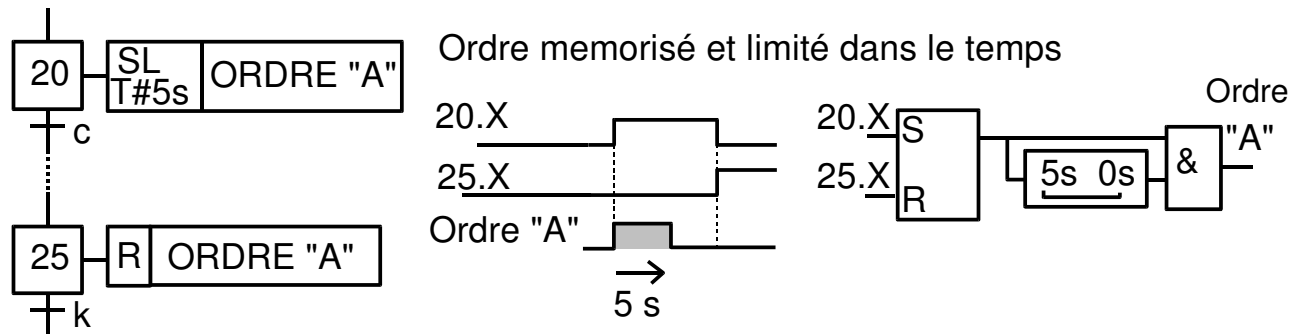
Ordre mémorisé et retardé

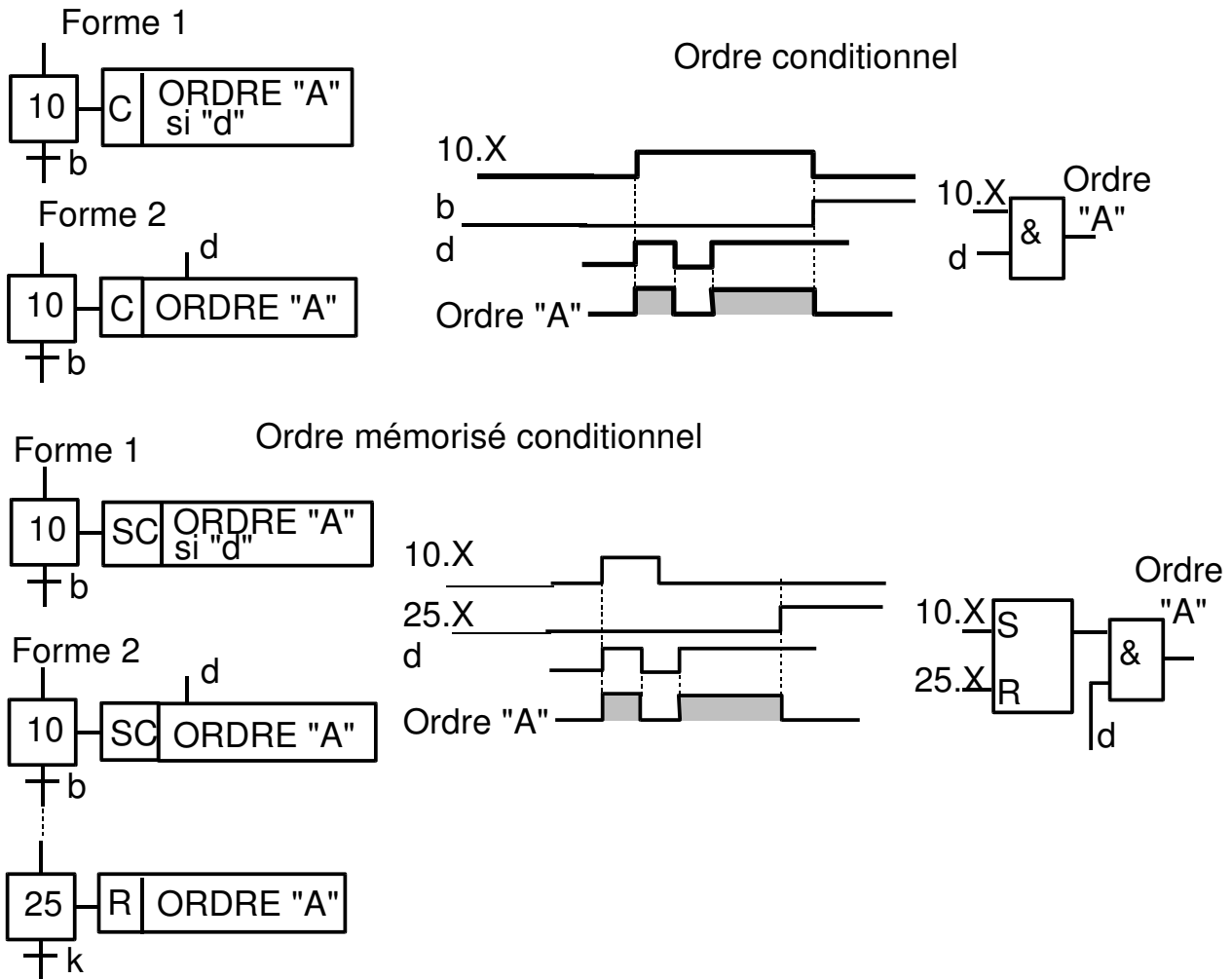


Ordre retardé et mémorisé



Ordre mémorisé et limité dans le temps

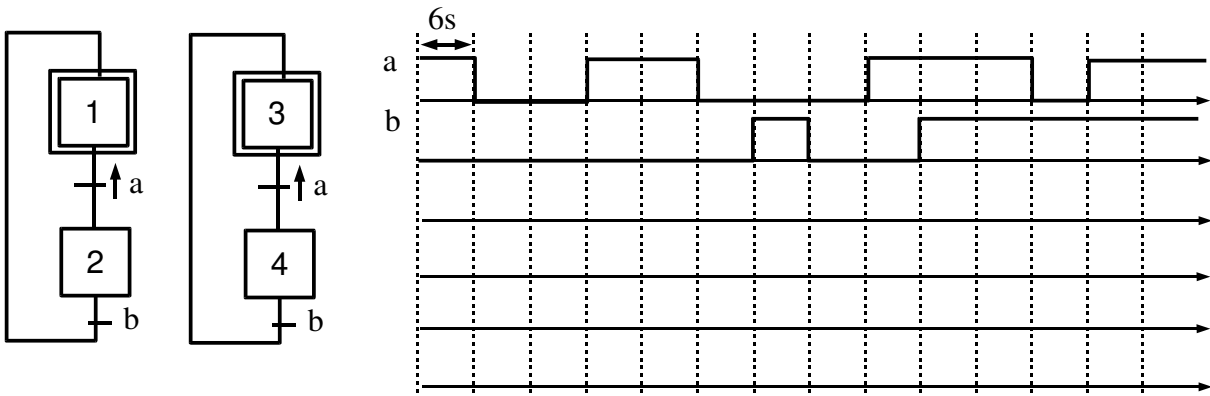




Exercice 1

Pour le chronogramme des variables d'entrées a et b indiquées, donner :

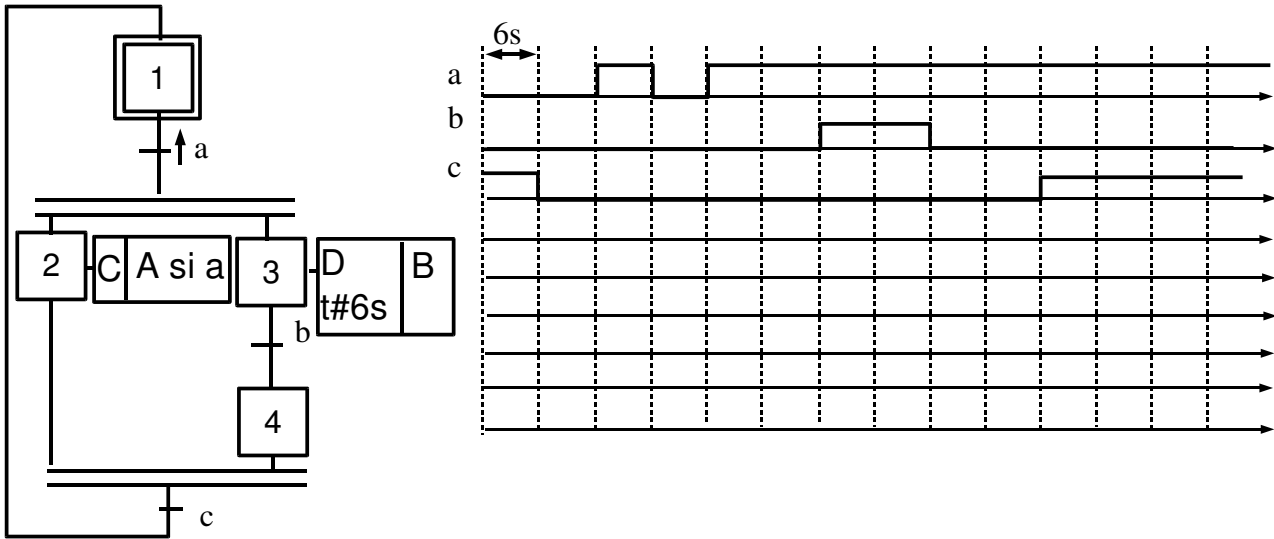
- 1° le chronogramme de 2.X (X2) et 2.t>t#6s (t/2/6s)
- 2° le chronogramme de 3.X (X3) et 3.t>t#6s (t/3/6s)



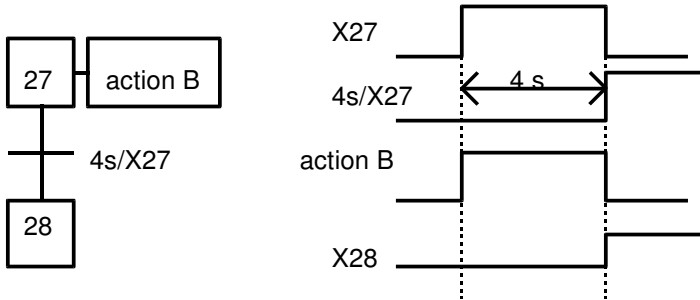
Exercice 2

Pour le chronogramme des variables d'entrées a, b et c indiquées, donner :

- 1° les situations stables successives ;
 - 2° le chronogramme des sorties A et B.
- L'action B s'écrivait autrefois (B si t/3/6s)



Exercice 3 : Utilisation de temporisations.



a) On veut faire la sélection d'une impulsion longue ($> 3s$) : l'action sur un bouton poussoir (BP) doit être maintenue pendant un temps minimum de trois secondes pour provoquer une action B.

b) Même exercice pour une impulsion courte ($< 1s$).

Exercice 4

On désire gérer l'allumage d'une lampe par 4 interrupteurs. C'est un front sur l'un des quatre interrupteurs qui allumera la lampe et c'est un front sur l'un des quatre interrupteurs qui l'éteindra.

TD5 ARS2 Langage IL de la norme 1131-3

I) Langage IL

Le présent paragraphe définit la sémantique du langage IL (Liste d'instructions) dont la syntaxe formelle est donnée dans l'annexe B.2. de la norme.

I.1 Instructions

Comme l'indique le Tableau 1, une liste d'instructions est composée d'une suite d'instructions. Chaque instruction doit débuter sur une nouvelle ligne et doit contenir un opérateur accompagné de modificateurs optionnels et, si cela est nécessaire pour l'opération considérée, un ou plusieurs opérandes séparés par des virgules. Les opérandes peuvent être choisis parmi les représentations des variables en tableau 3.

L'instruction peut être précédée d'une étiquette d'identification suivie de deux points (:). Si un commentaire, il doit constituer le dernier élément d'une ligne. Des lignes vides peuvent être insérées entre les instructions.

Tableau 1 - Exemples de champs d'instructions

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

I.2) Opérateurs, modificateurs et opérandes (p228 de la norme)

Les opérateurs standards ainsi que leurs modificateurs et opérandes autorisés doivent être tels qu'énumérés dans le tableau 3.

Le modificateur "N" indique une négation booléenne de l'opérande.

Par exemple l'instruction ANDN %IX2 est interprétée de la manière suivante :

resultat := resultat AND NOT %IX2

Le modificateur parenthèse gauche "(" indique que l'évaluation de l'opérateur est différée jusqu'à ce qu'un opérateur parenthèse droit ")" soit rencontré ; par exemple la séquence d'instructions suivante:

```
AND( %IX1
OR %IX2
)
```

doit être interprétée de la manière suivante :

resultat := resultat AND (%IX1 OR %IX2)

Le modificateur "C" indique que l'instruction donnée ne doit être exécutée que si le résultat faisant l'objet de l'évaluation en cours a la valeur booléenne 1 (ou la valeur booléenne 0 si l'opérateur est combiné avec le modificateur "N").

Tableau 2 - Opérateurs de liste d'instructions (p 230 de la norme)

N°	Opérateur	Modificateur	Opérande	Sémantique
1	LD	N	Note 2	Rendre le résultat courant égal à l'opérande
2	ST	N	Note 2	Mémoriser le résultat à l'emplacement de l'opérande
3	S	Note 3	BOOL	Positionner l'opérande booléen à 1
	R	Note 3	BOOL	Remettre l'opérande booléen à 0

4	AND	N, (BOOL	AND booléen
5	&	N, (BOOL	AND booléen
6	OR	N, (BOOL	OR booléen
7	XOR	N, (BOOL	OR exclusif booléen
8	ADD	(Note 2	Addition
9	SUB	(Note 2	Soustraction
10	MUL	(Note 2	Multiplication
11	DIV	(Note 2	Division
12	GT	(Note 2	Comparaison : >
13	GE	(Note 2	Comparaison : >=
14	EQ	(Note 2	Comparaison : =
15	NE	(Note 2	Comparaison : <>
16	LE	(Note 2	Comparaison : <=
17	LT	(Note 2	Comparaison : <
18	JMP	C, N	LABEL	Saut vers l'étiquette
19	CAL	C, N	NAME	Appel d'un bloc fonctionnel (Note 4)
20	RET	C, N		Retour d'une fonction appelée ou d'un bloc fonctionnel
21)			Evaluation d'une opération différée

NOTES

1 Se reporter au texte précédent pour toute explication relative aux modificateurs et à l'évaluation des expressions.

2 Ces opérateurs doivent être soit surchargés soit saisis comme défini en 2.5.1.4. Le résultat courant et l'opérande doivent être du même type

3 Ces opérations sont effectuées si et seulement si le résultat courant a la valeur booléenne 1

4 Le nom du bloc fonctionnel est suivi par une liste d'arguments entre parenthèses, tels que définis en 3.2.3.

5 Lorsqu'une instruction JMP est contenue dans une construction ACTION...END_ACTION l'opérande doit être une étiquette à l'intérieur de la même construction

I.3 Opérandes

Les opérandes sont :

Tableau 3

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 (8bits)
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.

II) Les blocs fonctionnels**II.1) Les fonctions standards**

Citons pêle mêle les fonctions standards : ADD, SUB, MUL, DIV, SHL, SHR, SEL, MUX, MAX,

MIN, LIMIT (limiteur, écrêteur), GT, GE, EQ, LE, LT, NE, des fonctions de chaîne de caractères.

II.2 Fonctions et blocs fonctionnels avec IL (p 230 de la norme)

Les blocs fonctionnels, tels que définis en II.2, peuvent être lancés sous condition ou inconditionnellement à l'aide de l'opérateur CAL (Appel). Comme l'illustre le tableau 4, ce lancement peut prendre trois formes.

Tableau 4 - Caractéristiques du lancement de bloc fonctionnel en langage IL (p 232)

N°	Description / exemple
1	CAL avec liste d'entrées CAL C10 (CU := %IX10, PV:=15)
2	CAL avec entrées de charge/mémoire LD 15 ST C10.PV LD %IX10 ST C10.CU CAL C10
3	Utilisation d'opérateurs d'entrée: LD 15 PV C10 LD %IX10 CU C10
NOTE - Une déclaration telle que VAR C10 : CTU; END_VAR est supposée dans les exemples ci-dessus	

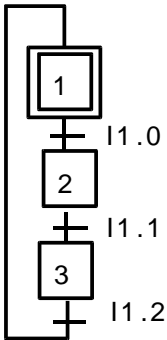
Les blocs fonctionnels doivent être déclarés : c'est la différence avec les fonctions. Les fonctions sont appelées directement en langage IL (voir l'exemple ci-dessous)

Tableau 5 – Appel de fonctions

Etiquette	Opérateur	Opérande	Commentaire
START:	LD	Poids_brut	(* Poids brut *)
	BCD_TO_INT		(* Conversion par exemple*)
	SUB	tare	(* Soustraction de la tare *)
	INT_TO_BCD		

III) Programmation de GRAFCETs en langage IL

Un problème à se poser est comment programmer un GRAFCET à l'aide du langage IL. Première méthode : on utilise les équations de récurrence.



```
(* initialisation *)
LD %S1
S %M0 (* M0 = étape 1 *)
R %M1 (* M1 = étape 2 *)
R %M2 (* M3 = étape 3 *)
(* sequentiel *)
LD %M0
AND %I1.0
S %M1 (* si le resultat courant est vrai *)
R %M0 (* si le resultat courant est vrai *)
LD %M1
AND %I1.1
S %M2
R %M1
....
```

Deuxième méthode : des éléments prédéfinis existent :

- construction STEP ... END_STEP ou INITIAL_STEP ... END_STEP
- construction TRANSITION ... END_TRANSITION (TRANSITION FROM et TO)
- construction ACTION ...END_ACTION

Exemples :

Langage IL	Langage ST
<pre>STEP STEP7 : A(N);END_STEP (*action A normale *) TRANSITION FROM STEP7 TO STEP 8 : LD %IX2.4 AND %IX2.3 END_TRANSITION STEP STEP8 : B(S);END_STEP (* action SET B*) STEP STEP7 : A(N);B(L,t#10s); END_STEP TRANSITION FROM (STEP7,STEP8) TO STEP 9 : LD %IX2.4 AND %IX2.3 END_TRANSITION STEP STEP9 : B(S);A(N);END_STEP (*deux actions*)</pre>	<pre>STEP STEP7 : END_STEP TRANSITION FROM STEP7 TO STEP 8 : := %IX2.4 & %IX2.3; END_TRANSITION STEP STEP8 : END_STEP</pre>

Exercice 1

Reprendre les GRAFCETs de l'exercice 2 du TD3 et les programmer en IL

Exercice 2

Décrire une architecture en VHDL capable d'exécuter un programme IL. Les instructions pouvant être exécutées sont : LD, LDN, AND, ANDN,OR, ST, STN et fin de programme. Le contenu de la mémoire programme sera pour « simplifier » réalisé avec un circuit combinatoire. Les entrées seront au nombre de 4 ainsi que les sorties. Séparer la partie chemin de données et le séquenceur.