



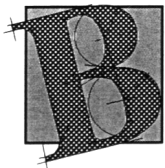
*La norme IEC 1131-3*  
*Concepts et notions de base*  
*Langages de programmation*

**INTERVENTION DE**

**M. DUMÉRY**

**LYCEE Baggio**  
**LILLE**

# NORMALISATION DES LANGAGES DE PROGRAMMATION D'A.P.I.



Lycée Baggio  
Lille

Jean- Jacques DUMÉRY

## CEI 61131-3

# Plan de la présentation

- **Présentation générale de la norme**
- **Les objectifs de la norme CEI 61131**
- **Les notions de base et les concepts importants**
- **Les éléments communs aux différents langages**
- **Les langages de programmation**  
**Quelques exemples d'utilisation**
- **Conclusion**

# **Les différentes parties de la norme**

**Elles s'appliquent aux automates programmables et aux périphériques associés tel que :**

- **Les outils de programmation et de mise au point**
- **Les équipements de test**
- **Les interfaces homme- machine**

# Les cinq parties de la CEI 61131

- 1ère partie : Informations générales
- 2ème partie : Spécifications et essais des équipements
- 3ème partie : Langages de programmation
- 4ème partie : Guide pour l'utilisateur
- 5ème partie : Communications

# Les références internationales

**CEI 61131- 1 et - 2 : octobre**

**CEI 61131- 3 : mars 1993**

**CEI/TR3 61131- 4 : mars 1995**

**CEI 61131- 5 : août 1999**

version préliminaire avant publication

Voir <http://www.iec.ch>

# Les références européennes

**NF EN 61131- 1:        septembre 1994**

**NF EN 61131- 2:        octobre 1996**

**NF EN 61131- 3:        novembre 1993**

Elles comportent en plus des CEI une annexe normative  
(correspondances normes européennes et internationales)

Voir <http://www.afnor.fr>

# Les objectifs de la norme

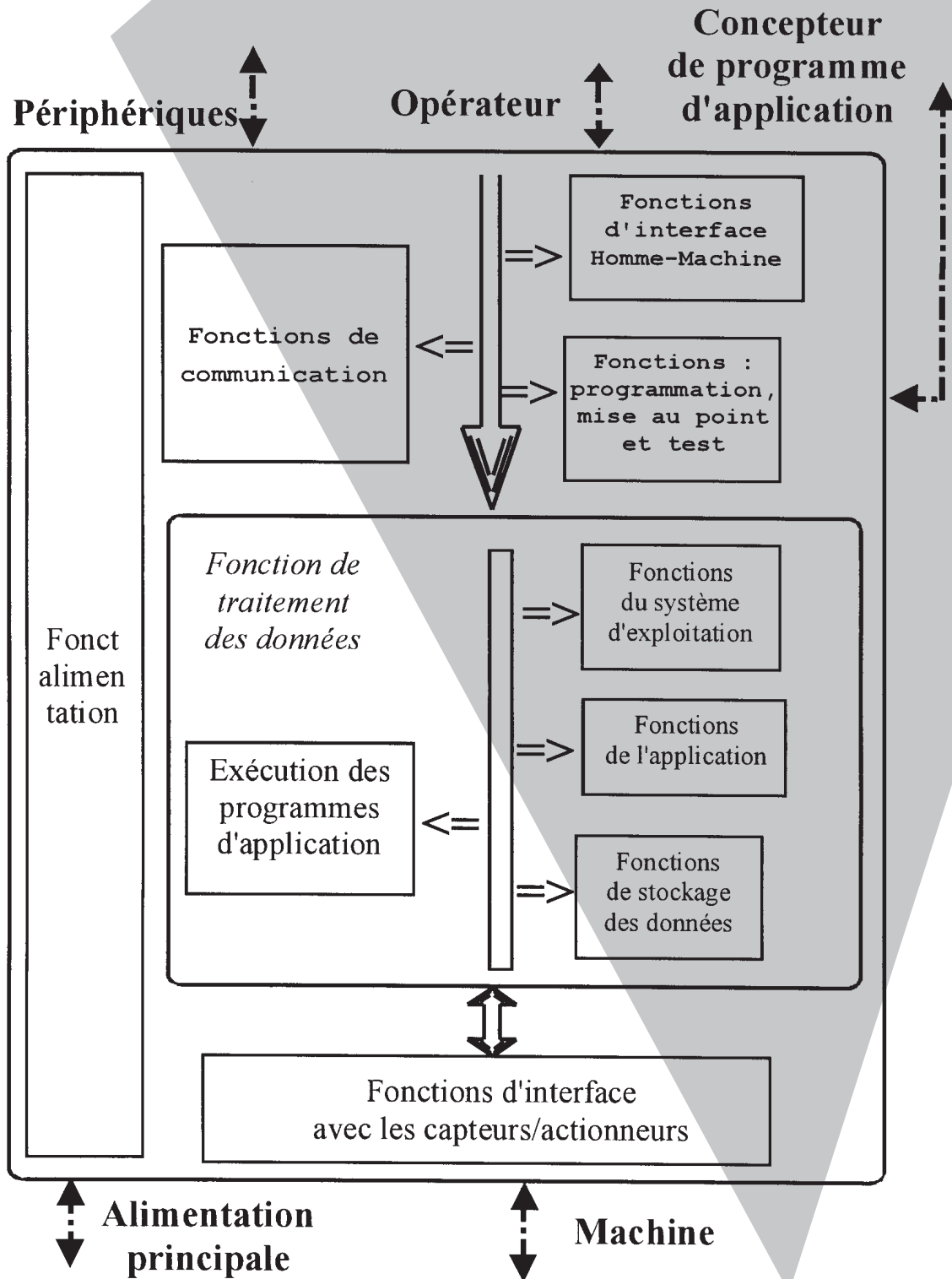
(trois première parties)

- **Donner les définitions et identifier les principales caractéristiques permettant de sélectionner et d'utiliser les A.P.**
- **Spécifier les prescriptions électriques, mécaniques et fonctionnelles ainsi que les méthodes de test et les procédures à suivre pour vérifier la conformité avec ces prescriptions**
- **Spécifier la syntaxe, la sémantique et la représentation des langages de programmation devant être utilisés pour les A.P.**

# Quelques définitions

- **Pour une configuration d'A.P. :**
  - (procédure d') arrêt de sécurité,
  - reprise à froid, à chaud et immédiate ...
- **pour les langages :**
  - un délimiteur,
  - un double mot, un mot long,
  - une instance,
  - une variable globale,
  - un libellé,
  - une donnée non volatile,
  - un champ d'application ...

# Structure fonctionnelle de base d'une configuration d'A.P.



# Partie 3 : langage de programmation

## Notion de base

### • Modules logiciels

( Program organization units)

- le PROGRAMME (PROGRAM)
- le BLOC FONCTIONNEL (FUNCTION BLOCK)
- la FONCTION (Function)

### • LES LANGAGES DE PROGRAMMATION

( dans lesquels les modules peuvent être écrits)

# La fonction

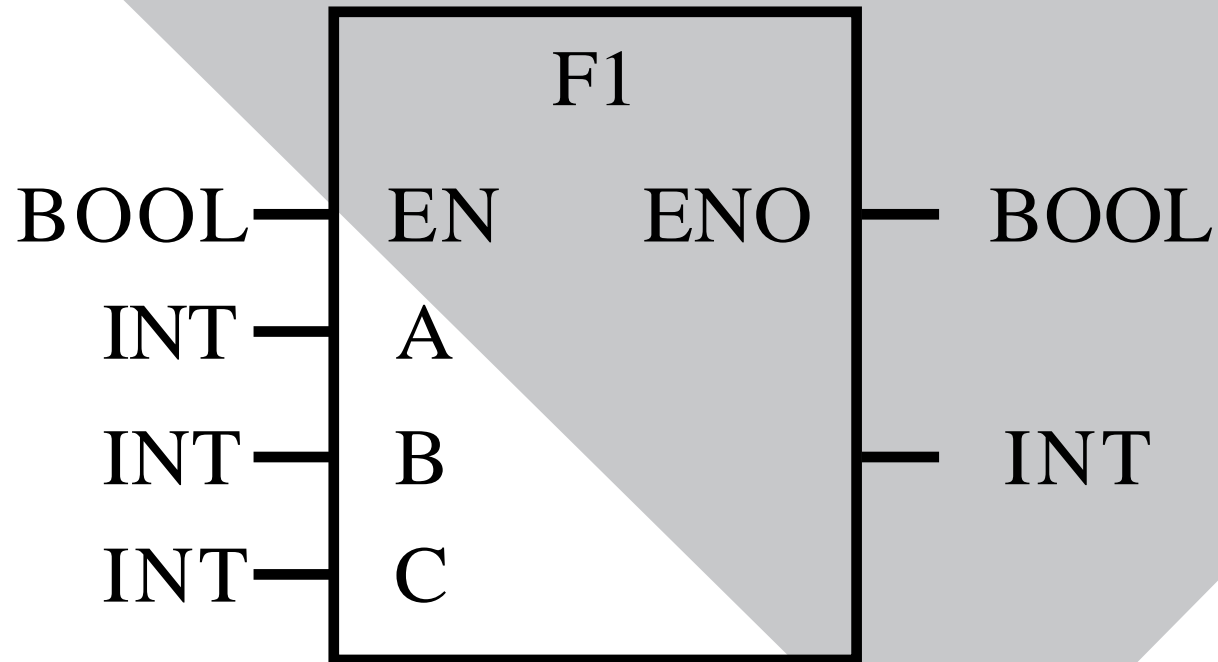
- **Module logiciel ayant :**
  - **plusieurs variables d'entrée possibles,**
  - **une seule variable de sortie,**
  - **pas de mémoire interne,**
  - **parfois une entrée EN (validation) et une sortie ENO (pas d'erreur).**

# Exemple de fonctions

- **fonctions de conversion de type,**
- **fonctions arithmétiques,**
- **fonctions sur chaînes de bits,**
- **fonctions sur chaînes de caractères,**
- **fonctions de sélection et comparaison,**
- **...**

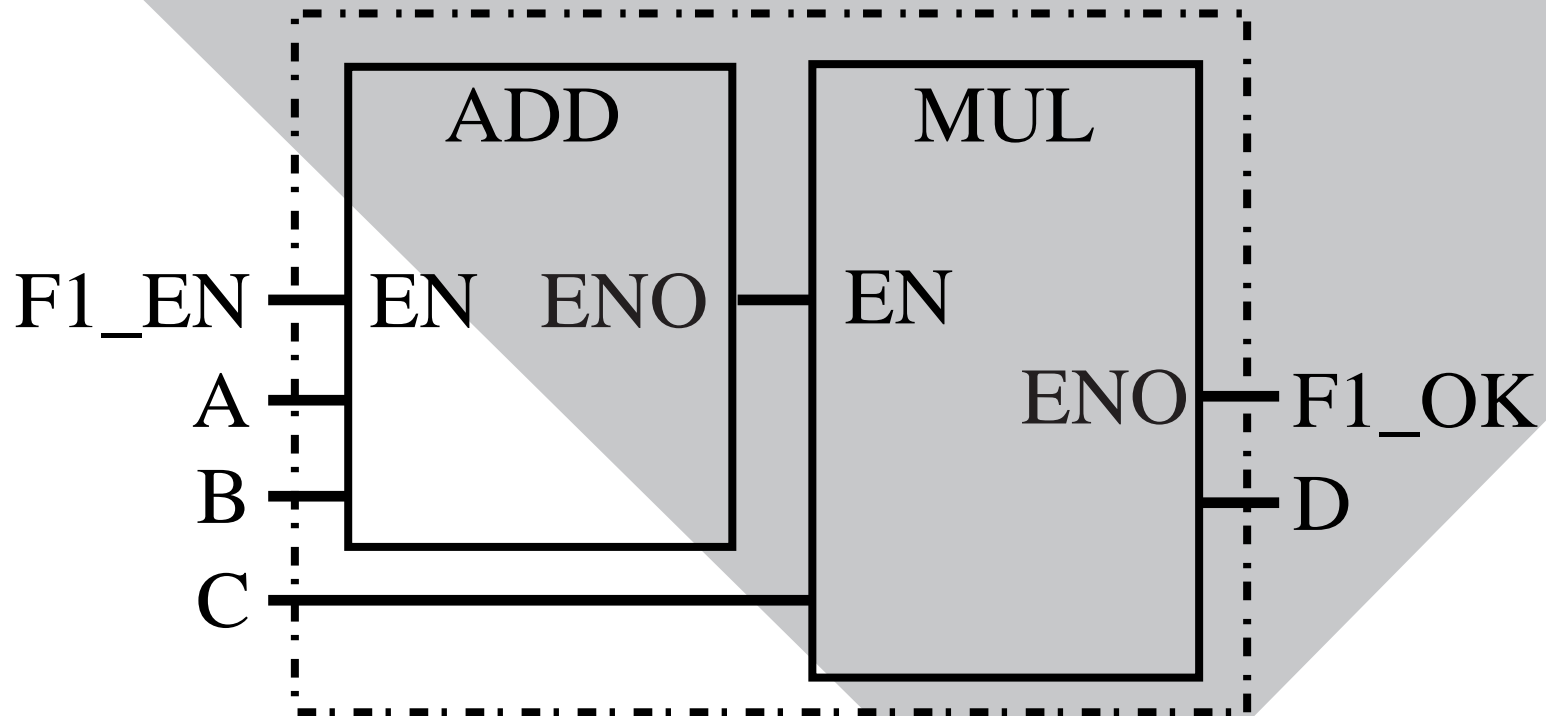
# Exemple de déclaration de fonction

## Spécification externe de F1



# Exemple de déclaration de fonction

## Spécification du corps de F1



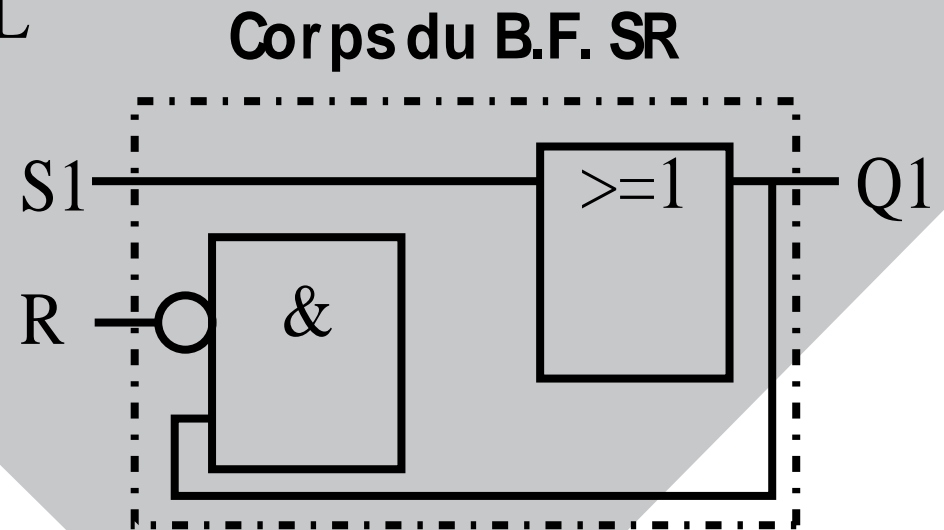
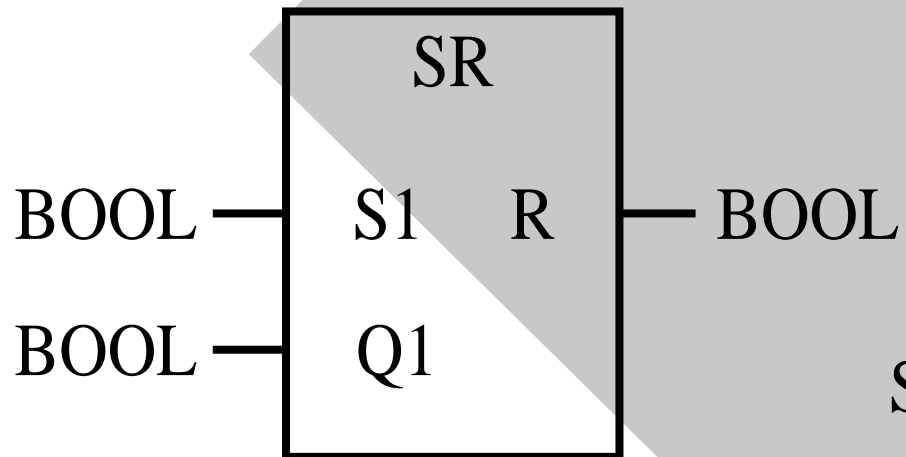
# Le bloc fonctionnel

- **Module logiciel ayant :**
  - **plusieurs variables de sorties possibles,**
  - **une mémoire interne.**

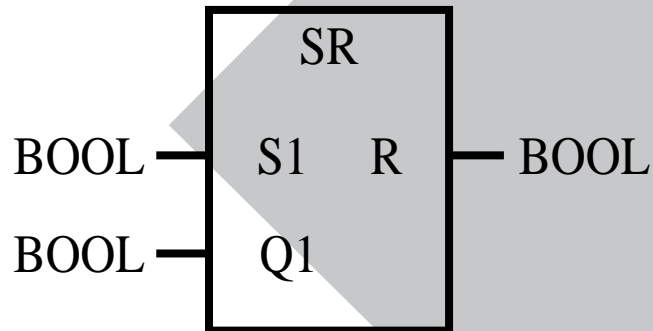
# Exemples de blocs fonctionnels

- mémoires,
- détection de fronts,
- compteurs, temporisations,
- blocs de communications,
- ...

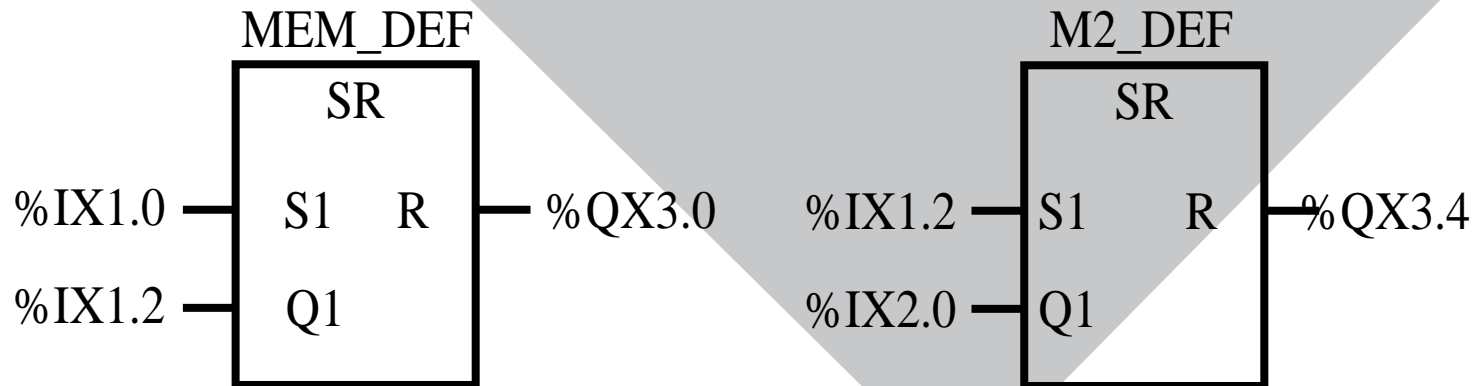
# Exemples de bloc fonctionnel standard



# Bloc fonctionnel instancié



**Il est possible de créer plusieurs instances d'un même B.F. (dans un programme ou un autre B.F.)**

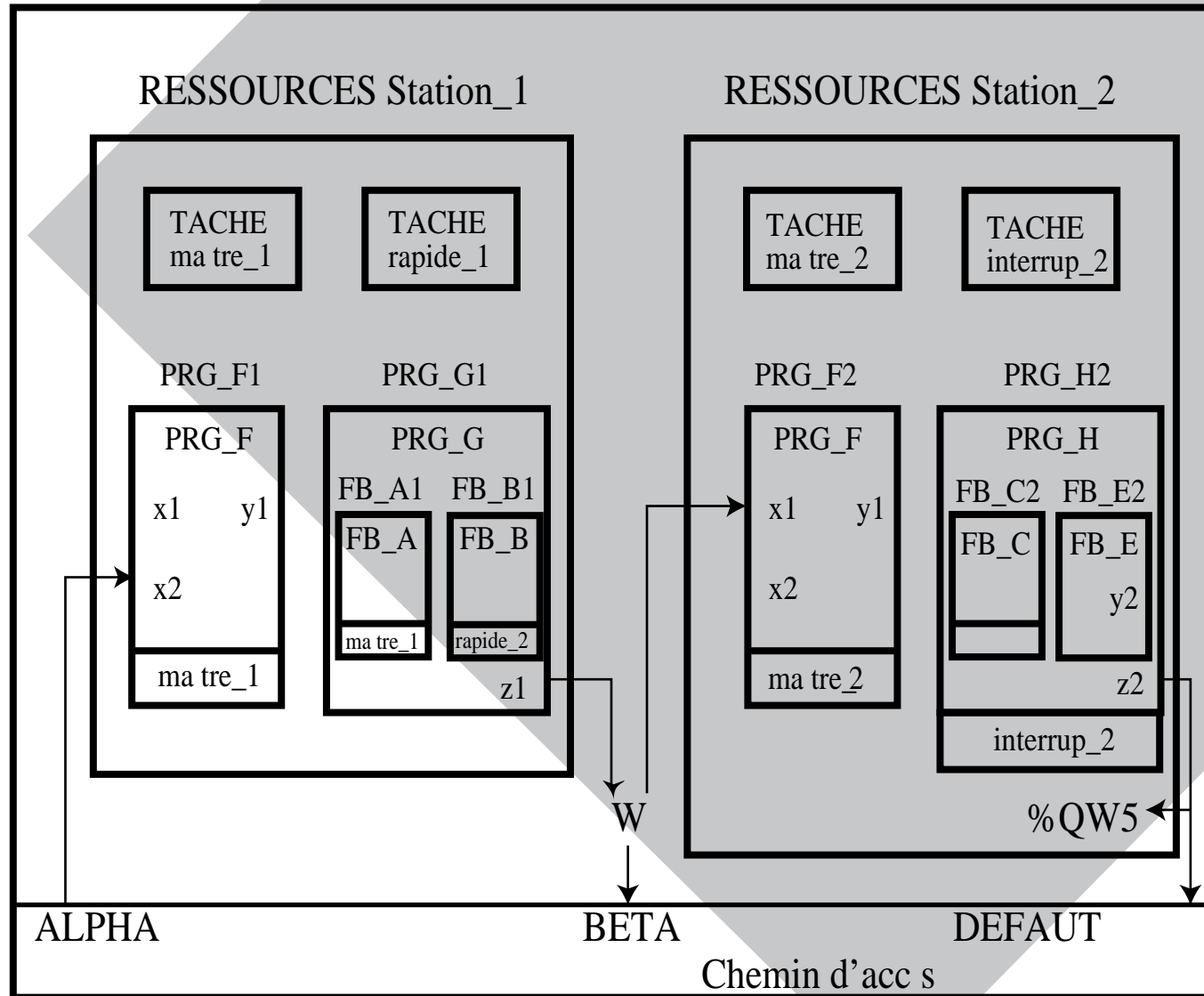


# Le programme

- **Module logiciel construit à l'aide de :**
  - **Fonctions,**
  - **et blocs fonctionnels.**

Les programmes ne peuvent être instanciés que dans des RESSOURCES  
Des VARIABLES GLOBALES pourront être déclarées

# La configuration matérielle



# Les éléments communs aux différents langages

- Les identificateurs ARRET\_TECHN, RETOUR\_OK
- Les mots clés FUNCTION, END\_FUNCTION\_BLOCK
- Les commentaires (\*production normale\*)
- Les libellés : +234, 16#E0, 'ARRET', TIME#2.7s  
numériques, de chaînes de caractères, de datation et de temps,
- Les types de données
- Les variables

# Les types de données, exemples

<b>BOOL</b>	<b>Boléen</b>	<b>1 bit</b>
<b>BYTE</b>	<b>Chaîne de bits de longueur 8</b>	<b>8 bits</b>
<b>WORD</b>	<b>Mot</b>	<b>16 bits</b>
<b>DWORD</b>	<b>Mot double</b>	<b>32 bits</b>
<b>LWORD</b>	<b>Mot long</b>	<b>64 bits</b>
<b>INT</b>	<b>Entier</b>	<b>16 bits</b>
<b>UINT</b>	<b>Entier non signé</b>	<b>16 bits</b>
<b>UDINT</b>	<b>Entier double non signé</b>	<b>32 bits</b>

# Les variables à un seul élément

PRÉFIXE	SIGNIFICATION
I	Emplacement d'entrée
Q	Emplacement de sortie
M	Emplacement de mémoire
X	Taille d'un seul bit
Aucun	Taille d'un seul bit
B	Taille d'un octet (8 bits)
W	Taille d'un mot (16 bits)
D	Taille d'un double mot (32 bits)
L	Taille d'un mot long (64 bits)

# Représentation des variables à un seul élément

La représentation directe d'une variable à un seul élément est assurée par l'enchaînement :

du signe "%",  
d'un préfixe d'emplacement,  
d'un préfixe de taille,  
et d'un ou plusieurs entiers non signés  
séparés par le symbole "."

**Exemples : %I2.0, %Q3.2, %MD25**

# Les langages de programmation

- Les langages littéraux :

- IL liste d'instructions,
- ST langage littéral structuré.

- Les langages graphiques :

- LD langage à contacts,
- FBD langage à blocs fonctionnels.

- Le langage SFC :

# Le langage IL

<b>Etiquette</b>	<b>Opérateur</b>	<b>Opérande</b>	<b>Commentaire</b>
<b>MV1 :</b>	<b>LD</b>	<b>%IX1</b>	<b>(* Etiquette non oblig.*)</b>
	<b>AND N</b>	<b>%MX5</b>	
	<b>ST</b>	<b>%QX2</b>	<b>(* Marche ventilateur*)</b>

*Des Fonctions et des blocs fonctionnels peuvent être lancés en IL*

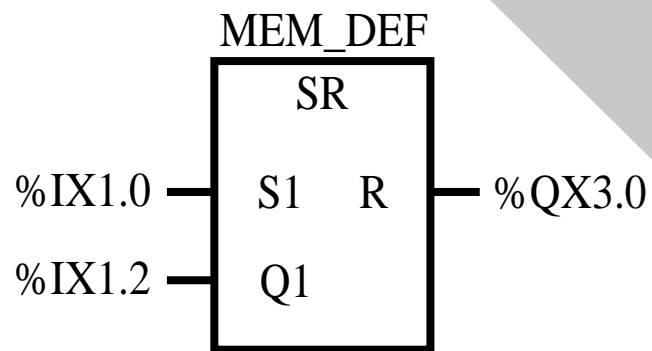
# Le langage ST

Le langage littéral structuré ST utilise :

- des expressions (E<F) AND NOT C
- et des énoncés
  - les énoncés d'affectation, C:=C+1;
  - les énoncés de sélection, IF ... THEN ... ELSE ..., CASE
  - les énoncés d'itération, FOR ... TO ..., WHILE ...REPEAT ...
  - les énoncés de commande.  
de fonctions et B.F.

# Le langage ST

## Exemple d'un énoncé de commande



(\*d clARATION\*)

```
VAR MEM_DEF :SR;END VAR
```

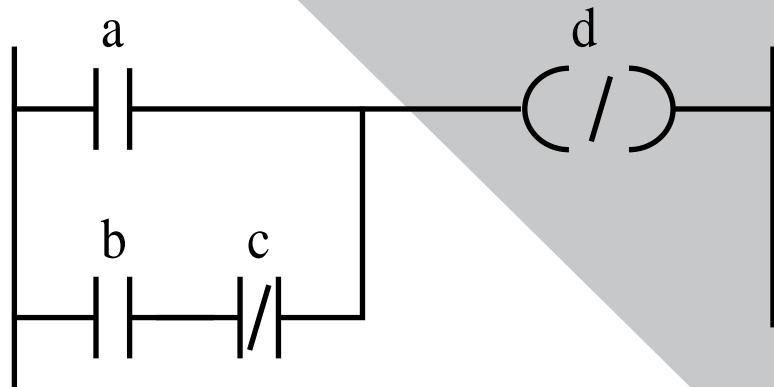
(\*execution\*)

```
MEM_DEF (S1 :=%IX1.0, R :=%IX1.2);
```

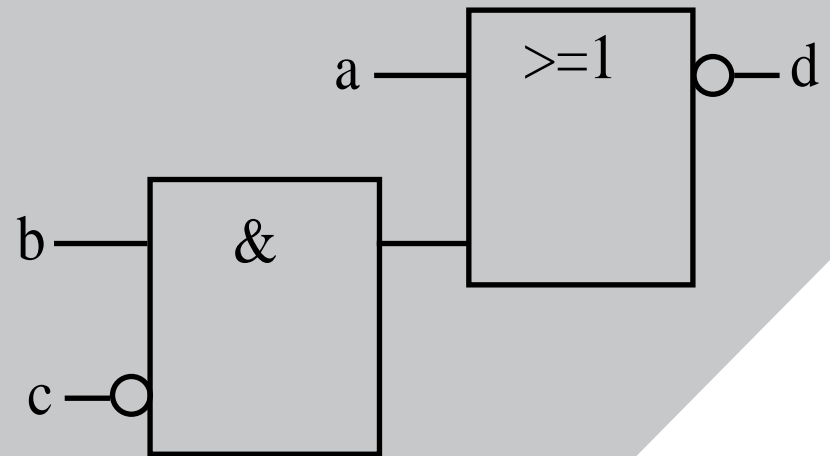
(\*affectation\*)

```
%QX3.0 :=MEM_DEF.Q1;
```

# Les langages graphiques LD et FBD

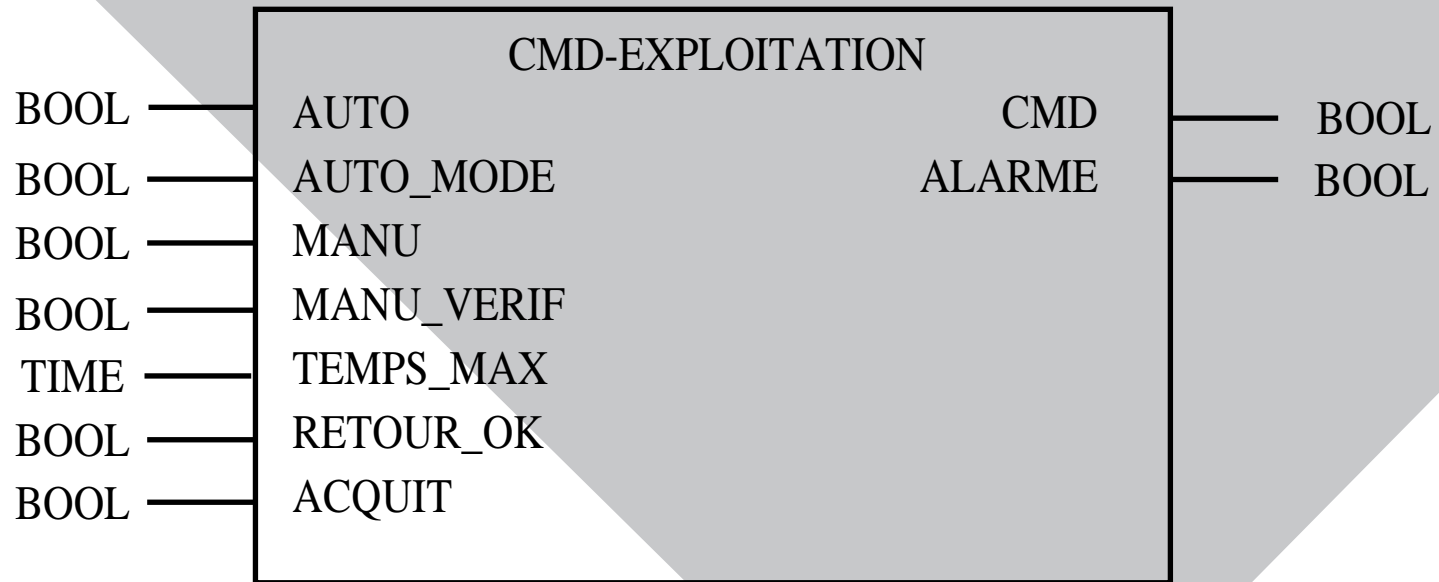


Langage LD



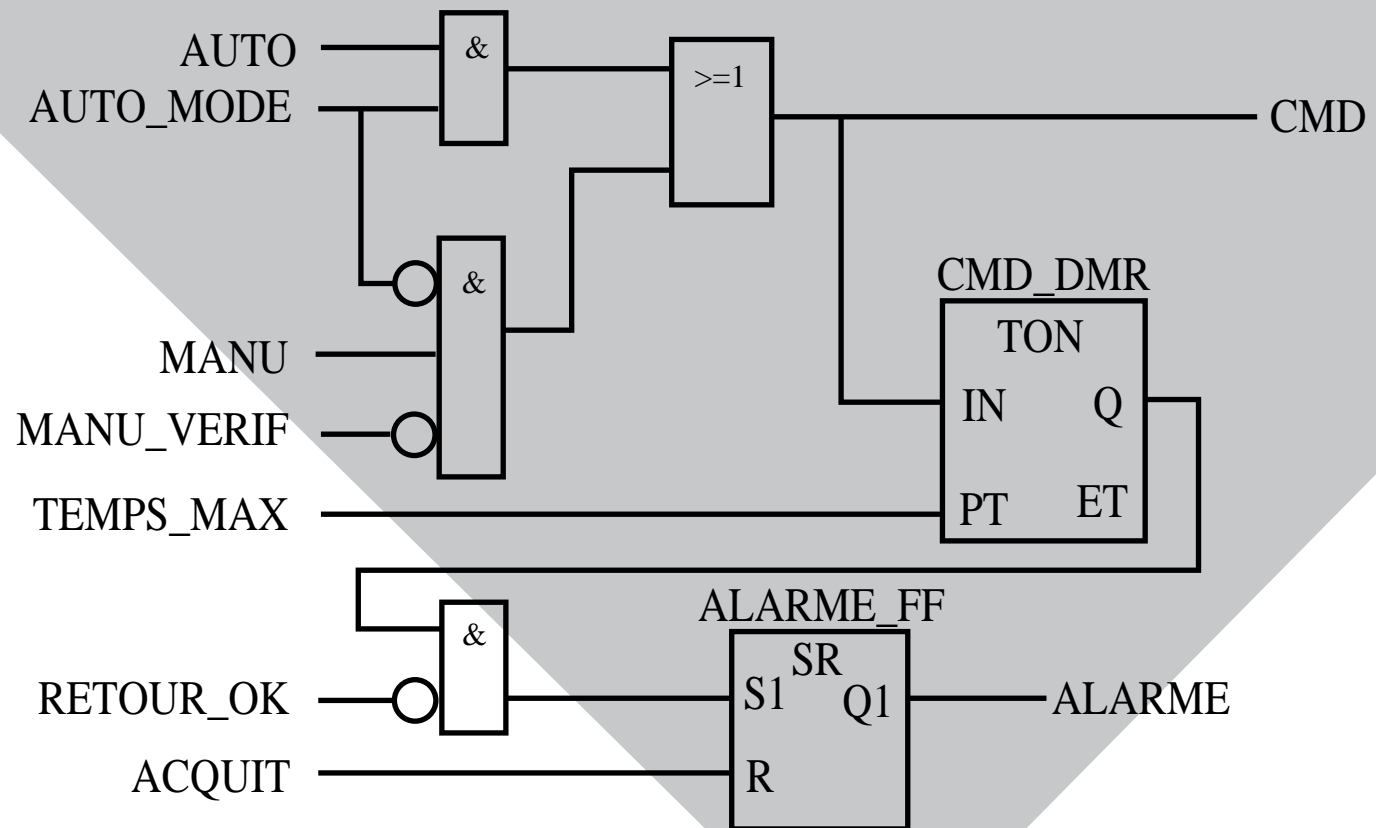
Langage FBD

# Le langage FBD, exemple



Description externe

# Le langage FBD, exemple

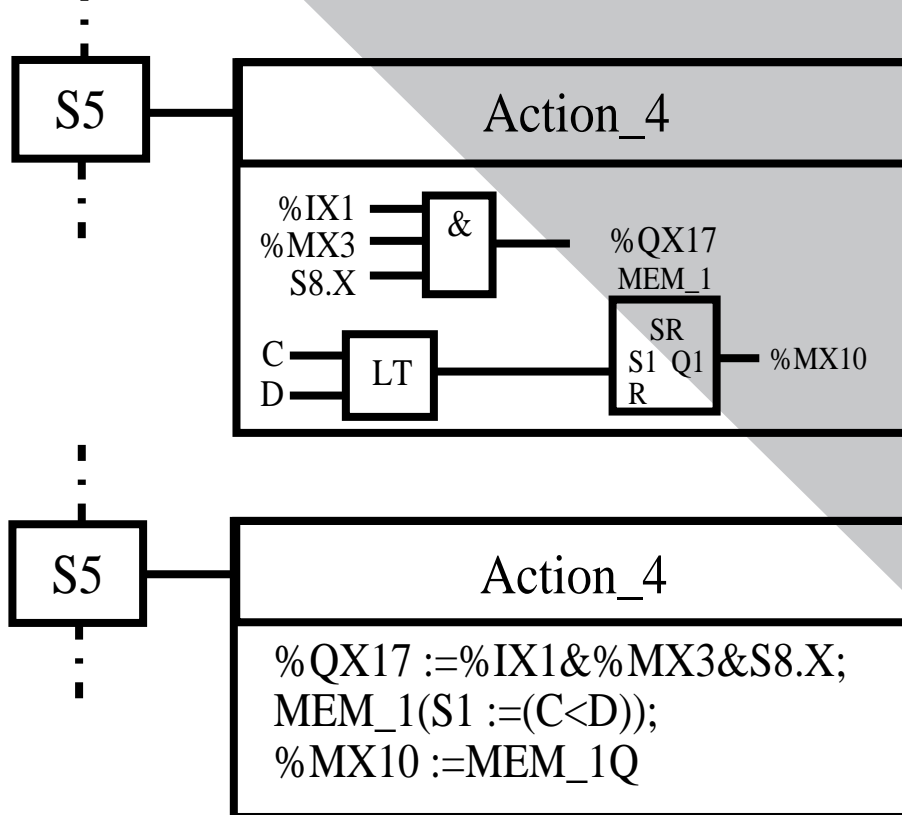


# Le langage SFC

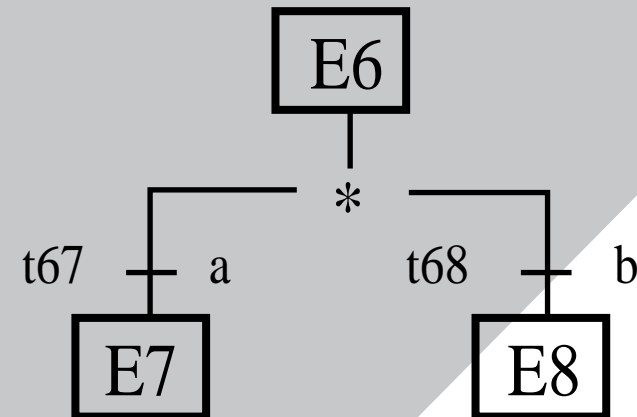
**Il est destiné à être utilisé pour la structuration de l'organisation interne d'un module logiciel dans le but d'assurer :  
des fonctions de commande séquentielle**

# Le langage SFC, remarques

Tous les langages peuvent être utilisés dans les blocs d'action



Le parallélisme interprété est exclu



# CONCLUSION

- **Réponse à une attente des utilisateurs**
- **Mise en oeuvre de principes tels que structuration et modularité**
- **En section de BTS MAI : capacités visées CP44 et CP53**
  - harmonisation des vocabulaires utilisés,
  - notions et concepts de base s'appuyant sur une norme,
  - syntaxe et sémantique indépendants d'une technologie particulière

**Nécessité d'une spécification structurée en amont de la phase de codage**