

# Manuel de Référence

## PL7 Micro/Junior/Pro

### Description du logiciel PL7

fre Mars 2005

---

## Structure de la documentation

---

### Présentation

Ce manuel se compose de trois tomes:

- Tome 1: Description du logiciel PL7
    - Généralités
    - Langage à contacts
    - Langage liste d'instructions
    - Langage littéral structuré
    - Langage Grafcet
    - Blocs fonction DFB
    - Modules Fonctionnels
  - Tome 2: Description détaillée des instructions et des fonctions
    - Instructions de base
    - Instructions avancées
    - Objets bits et mots système
  - Tome 3: Annexes
    - Différences entre PL7-2/3 et PL7-Micro/Junior
    - Aide-mémoire
    - Liste des mots réservés
    - Conformité au standard CEI 1131-3
    - Serveur OLE Automation
    - Performances
-



---

# Table des matières



---

<b>A propos de ce manuel</b> .....	<b>11</b>
<b>Partie I Description du logiciel PL7</b> .....	<b>13</b>
Présentation .....	13
<b>Chapitre 1 Présentation du logiciel PL7</b> .....	<b>15</b>
Présentation .....	15
Présentation des logiciels PL7 .....	16
Présentation des langages PL7 .....	17
Structure logicielle PL7 .....	20
Modules fonctionnels .....	22
<b>Chapitre 2 Description des objets langages PL7</b> .....	<b>25</b>
Présentation .....	25
Définition des principaux objets booléen .....	26
Définition des principaux objets mots .....	27
Adressage des objets bits .....	29
Adressage des objets de modules d'entrées/sorties du TSX 37 .....	31
Adressage des objets de modules d'entrées/sorties en rack .....	34
Adressage des objets langage de modules déportés sur bus FIPIO .....	36
Adressage des objets langage liés au bus AS-i .....	39
Adressage des objets mots .....	41
Règle de recouvrements .....	43
Objets de bloc fonction .....	44
Objets PL7 de type tableau .....	46
Objets indexés .....	48
Objets Grafcet .....	51
Symbolisation .....	52
Objets présymbolisés .....	54
<b>Chapitre 3 Mémoire utilisateur</b> .....	<b>55</b>
Présentation .....	55
Structure mémoire des automates Micro .....	56
Structure mémoire des automates Premium .....	58
Description de la mémoire bits .....	59

---

	Description de la mémoire mots . . . . .	61
	Caractéristiques de la mémoire des automates TSX 37 . . . . .	62
	Caractéristiques de la mémoire des automates TSX/PCX 57 10/15/20/25/26/28 64 . . . . .	64
	Caractéristiques de la mémoire des automates TSX/PCX 57 30/35/36 . . . . .	66
	Caractéristiques de la mémoire des automates TSX 57 453/4823 . . . . .	68
<b>Chapitre 4</b>	<b>Modes de marche . . . . .</b>	<b>71</b>
	Présentation . . . . .	71
	Traitement sur coupure et reprise secteur . . . . .	72
	Traitement sur reprise à chaud . . . . .	74
	Gestion du démarrage à froid . . . . .	76
<b>Chapitre 5</b>	<b>Structure logicielle . . . . .</b>	<b>79</b>
	Présentation . . . . .	79
5.1	Description des tâches . . . . .	80
	Présentation . . . . .	80
	Présentation de la tâche maître . . . . .	81
	Description des sections et des sous-programmes . . . . .	82
	Présentation de la tâche rapide . . . . .	85
	Présentation des traitements événementiels . . . . .	86
5.2	Structure monotâche . . . . .	88
	Présentation . . . . .	88
	Structure logicielle monotâche . . . . .	89
	Exécution cyclique . . . . .	90
	Exécution périodique . . . . .	92
	Contrôle du temps de cycle . . . . .	95
5.3	Structure multitâche . . . . .	96
	Présentation . . . . .	96
	Structure logicielle multitâche . . . . .	97
	Séquencement des tâches dans une structure multitâche . . . . .	99
	Affectation des voies d'entrées/sorties aux tâches maître et rapide . . . . .	100
	Echanges d'entrées/sorties dans les traitements événementiels . . . . .	101
5.4	Modules fonctionnels . . . . .	103
	Structuration en modules fonctionnels . . . . .	103
<b>Partie II</b>	<b>Description des langages PL7 . . . . .</b>	<b>105</b>
	Présentation . . . . .	105
<b>Chapitre 6</b>	<b>Langage à contacts . . . . .</b>	<b>107</b>
	Présentation . . . . .	107
	Présentation générale du langage à contacts . . . . .	108
	Structure d'un réseau de contacts . . . . .	109
	Etiquette d'un réseau de contacts . . . . .	110
	Commentaire d'un réseau de contacts . . . . .	111
	Éléments graphiques du langage à contacts . . . . .	112
	Règles de programmation d'un réseau de contacts . . . . .	115

---

---

	Règle de programmation des blocs fonction . . . . .	116
	Règles de programmation des blocs opération . . . . .	117
	Exécution d'un réseau de contacts . . . . .	118
<b>Chapitre 7</b>	<b>Langage liste d'instructions . . . . .</b>	<b>121</b>
	Présentation . . . . .	121
	Présentation générale du langage liste d'instructions . . . . .	122
	Structure d'un programme liste d'instructions . . . . .	123
	Etiquette d'une phrase en langage liste d'instructions . . . . .	124
	Commentaire d'une phrase en langage liste d'instructions . . . . .	125
	Présentation des instructions en langage liste d'instructions . . . . .	126
	Règle d'utilisation des parenthèses en langage liste d'instructions . . . . .	129
	Description des instructions MPS, MRD et MPP . . . . .	131
	Principes de programmation des blocs fonction prédéfinis . . . . .	133
	Règles d'exécution d'un programme liste d'instructions . . . . .	135
<b>Chapitre 8</b>	<b>Langage littéral structuré . . . . .</b>	<b>137</b>
	Présentation . . . . .	137
	Présentation du langage littéral structuré . . . . .	138
	Structure d'un programme en langage littéral structuré . . . . .	139
	Etiquette d'une phrase en langage littéral structuré . . . . .	140
	Commentaire d'une phrase en langage littéral structuré . . . . .	141
	Instructions sur objets bits . . . . .	142
	Instructions arithmétiques et logiques . . . . .	143
	Instructions sur tableaux et chaîne de caractères . . . . .	145
	Instructions de conversions numériques . . . . .	148
	Instructions sur programme et instructions spécifiques . . . . .	149
	Structure de contrôle conditionnelle IF...THEN . . . . .	151
	Structure de contrôle conditionnelle WHILE...END_WHILE . . . . .	153
	Structure de contrôle conditionnelle REPEAT...END_REPEAT . . . . .	154
	Structure de contrôle conditionnelle FOR...END_FOR . . . . .	155
	Instruction de sortie de boucle EXIT . . . . .	156
	Règles d'exécution d'un programme littéral structuré . . . . .	157
<b>Chapitre 9</b>	<b>Grafcet . . . . .</b>	<b>161</b>
	Présentation . . . . .	161
9.1	Présentation générale du Grafcet . . . . .	162
	Présentation . . . . .	162
	Présentation du Grafcet . . . . .	163
	Description des symboles graphiques du Grafcet . . . . .	164
	Description des objets spécifiques au Grafcet . . . . .	166
	Possibilités du Grafcet . . . . .	168
9.2	Règle de construction du Grafcet . . . . .	169
	Présentation . . . . .	169
	Représentation du Grafcet . . . . .	170
	Utilisation des divergences et convergences OU . . . . .	171

---

	Utilisation des divergences et convergences ET . . . . .	172
	Utilisation des renvois . . . . .	173
	Utilisation des liaisons orientées . . . . .	176
	Commentaire Grafcet . . . . .	177
9.3	Programmation des actions et des conditions . . . . .	178
	Présentation . . . . .	178
	Programmation des actions associées aux étapes . . . . .	179
	Programmation des actions à l'activation ou à la désactivation . . . . .	181
	Programmation des actions continues . . . . .	182
	Programmation des réceptivités associées aux transitions . . . . .	183
	Programmation des réceptivités en langage à contacts . . . . .	184
	Programmation des réceptivités en langage liste d'instructions . . . . .	185
	Programmation des réceptivités en langage littéral structuré . . . . .	186
9.4	Macro-étapes . . . . .	187
	Présentation . . . . .	187
	Présentation des macro-étapes . . . . .	188
	Constitution d'une macro-étape . . . . .	189
	Caractéristiques des macro-étapes . . . . .	190
9.5	Section Grafcet . . . . .	192
	Présentation . . . . .	192
	Structure d'une section Grafcet . . . . .	193
	Description du traitement préliminaire . . . . .	194
	Prépositionnement du Grafcet . . . . .	195
	Initialisation du Grafcet . . . . .	196
	Remise à zéro du Grafcet . . . . .	197
	Figeage du Grafcet . . . . .	198
	Remise à zéro des macro-étapes . . . . .	199
	Fonctionnement du traitement séquentiel . . . . .	201
	Description du traitement postérieur . . . . .	203

<b>Chapitre 10</b>	<b>Blocs fonction DFB . . . . .</b>	<b>205</b>
	Présentation . . . . .	205
	Présentation des blocs fonction DFB . . . . .	206
	Comment mettre en oeuvre un bloc fonction DFB . . . . .	207
	Définition des objets des blocs fonction type DFB . . . . .	209
	Définition des paramètres DFB . . . . .	211
	Définition des variables DFB . . . . .	212
	Règle de codage des Types DFB . . . . .	214
	Création des instances de DFB . . . . .	216
	Règle d'utilisation des DFB dans un programme . . . . .	217
	Utilisation d'un DFB dans un programme en langage à contacts . . . . .	218
	Utilisation d'un DFB dans un programme en langage liste d'instructions ou littéral . . . . .	219
	Exécution d'une instance DFB . . . . .	220
	Exemple de programmation de bloc fonction DFB . . . . .	221



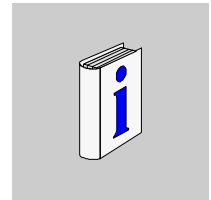
---

<b>Index</b>	.....	<b>225</b>
--------------	-------	------------

---

---

## A propos de ce manuel



---

### Présentation

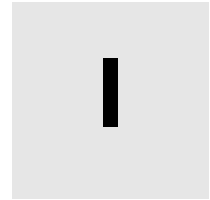
<b>Objectif du document</b>	Ce manuel décrit les langages de programmation des automates Micro, Premium et Atrium.
<b>Champ d'application</b>	La mise à jour de cette publication prend en compte les fonctionnalités de PL7 V4.5. Elle permet néanmoins de mettre en oeuvre les versions antérieures de PL7.
<b>Commentaires utilisateur</b>	Envoyez vos commentaires à l'adresse e-mail <a href="mailto:techpub@schneider-electric.com">techpub@schneider-electric.com</a>

---



---

# Description du logiciel PL7



---

## Présentation

### Objet de cet intercalaire

Cet intercalaire présente le logiciel PL7. Il décrit les notions élémentaires de base nécessaires à la programmation des automates Micro et Premium.

### Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre	Titre du chapitre	Page
1	Présentation du logiciel PL7	15
2	Description des objets langages PL7	25
3	Mémoire utilisateur	55
4	Modes de marche	71
5	Structure logicielle	79

---



---

# Présentation du logiciel PL7



---

## Présentation

### Objet de ce chapitre

Ce chapitre présente les principales caractéristiques du logiciel PL7.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Présentation des logiciels PL7	16
Présentation des langages PL7	17
Structure logicielle PL7	20
Modules fonctionnels	22

---

## Présentation des logiciels PL7

---

### Généralités

La conception et la mise en oeuvre des applications pour automates Micro et Premium se réalisent à l'aide des logiciels PL7.

Il est proposé 3 types de logiciels PL7:

- PL7 Micro,
- PL7 Junior,
- PL7 Pro.

---

### Logiciels PL7

Le tableau suivant montre les différences entre les 3 types de logiciels.

Services		PL7 Micro	PL7 Junior	PL7 Pro
Programmation/Mise au point/ Exploitation		M	M/P/A	M/P/A
Blocs fonction utilisateur	Création	-	-	P/A
	Utilisation	-	P/A	P/A
Ecrans d'exploitation	Création	-	-	M/P/A
	Utilisation	-	M/P/A	M/P/A
Modules fonctionnels		-	-	P/A
Bloc fonction DFB de diagnostic		-	-	P/A

#### Légende :

A = automates Atrium

M = automates Micro

P = automates Premium

- = non disponible

---

### Conventions d'écriture

Dans la suite du document :

- la notation PL7 ou logiciel PL7 est utilisée pour désigner indifféremment les 3 types de logiciels PL7 Micro, PL7 Junior et PL7 Pro,
  - la notation Premium est utilisée pour désigner indifféremment les processeurs TSX 57, PMX 57, et PCX 57.
-



## Présentation des langages PL7

### Généralités

Le logiciel PL7 propose 4 langages de programmation :

- langage à contacts
- liste d'instructions
- littéral structuré
- Grafcet

Le tableau suivant donne l'utilisation possible des langages en fonction des types d'automates.

Langage	Automates Micro	Automates Premium
Langage à contacts	X	X
Liste d'instructions	X	X
Littéral structuré	X	X
Grafcet	X (excepté les macro-étapes)	X

### Légende :

X = disponible

- = non disponible

Ces langages peuvent être mixés au sein d'une même application. Une section de programme peut être écrite en langage à contacts, une autre en littéral ...

Ces langages mettent en oeuvre :

- des blocs fonction pré-définis (Temporisations, Compteurs,...),
- des fonctions métiers (analogique, communication, comptage...),
- des fonctions spécifiques (gestion du temps, chaîne de caractères...).

Les objets du langage sont symbolisables à l'aide de l'éditeur de variables ou en ligne dans les éditeurs de programme.

Le logiciel PL7 est conforme à la norme IEC 1131-3 (voir (Voir Manuel de référence, Tomes 2 et 3) ).

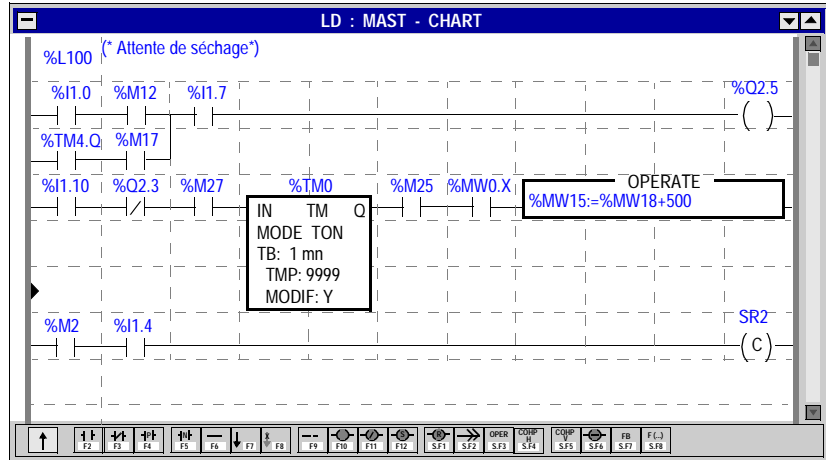
## Langage à contacts

Le langage à contacts (LD) est un langage graphique. Il permet la transcription de schémas à relais, il est adapté au traitement combinatoire.

Il offre les symboles graphiques de base : contacts, bobines, blocs.

L'écriture de calculs numériques est possible à l'intérieur de blocs opérations.

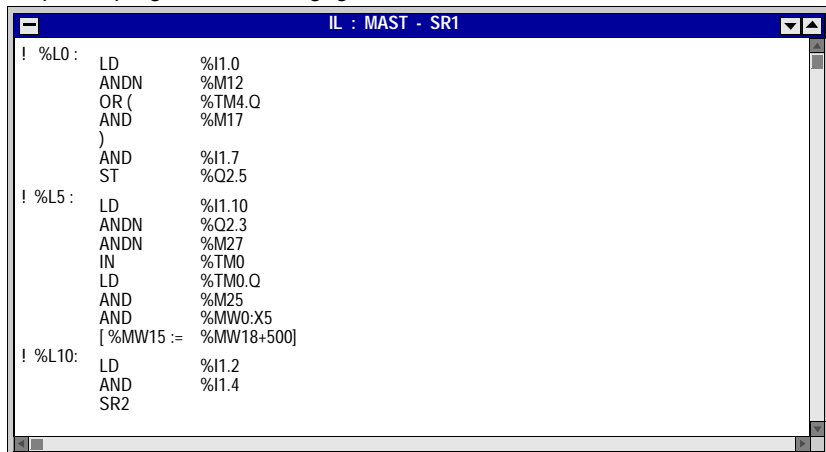
Exemple de réseau de contacts



## Langage liste d'instructions

Le langage liste d'instructions (IL) est un langage "machine" booléen qui permet l'écriture de traitements logiques et numériques.

Exemple de programme en langage liste d'instructions



## Langage littéral structuré

Le langage littéral structuré (ST) est un langage de type "informatique" permettant l'écriture structurée de traitements logiques et numériques.

Exemple de programme en langage littéral structuré

```

ST : MAST - SR10
!
(* Recherche du premier élément non nul dans un tableau de 32 mots
Détermination de sa valeur (%MW10), de son rang (%MW11)
Cette recherche s'effectue si %M0 est à 1
%M1est mis à 1 si un élément non nul existe, sinon il est mis à 0 *)

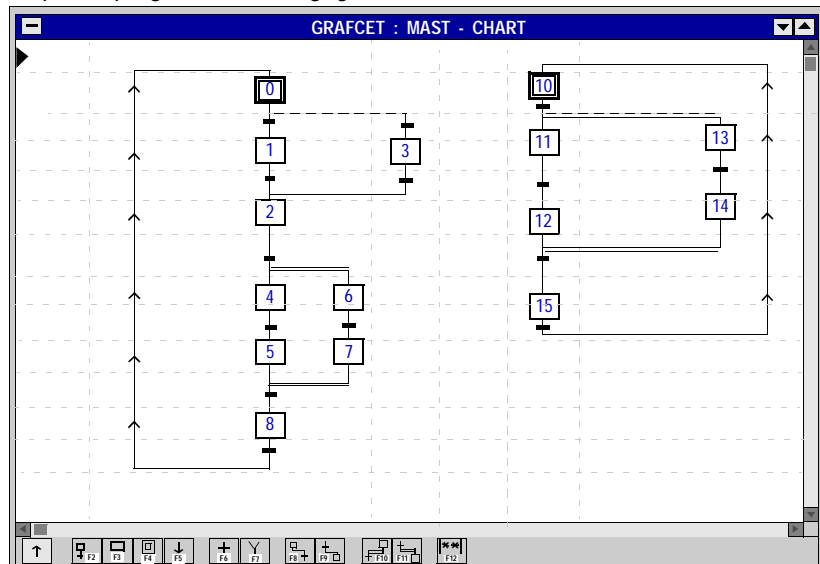
IF %M0 THEN
  FOR %MW 99 := 0 TO 31 DO
    IF %MW100 [%MW99]<> 0 THEN
      %MW 10 := %MW100 [%MW99];
      %MW 11 := %MW 99;
      %M1 := TRUE;
      EXIT;      (*Sortie de la boucle FOR*)
    ELSE
      %M1 := FALSE;
    END_IF;
  END_FOR;
ELSE
  %M1 := FALSE;
END_IF;

```

## Langage Grafcet

Le langage Grafcet permet de représenter graphiquement et de façon structurée le fonctionnement d'un automate séquentiel.

Exemple de programme en langage Grafcet.



## Structure logicielle PL7

---

### Généralités

Le logiciel PL7 propose deux types de structure :

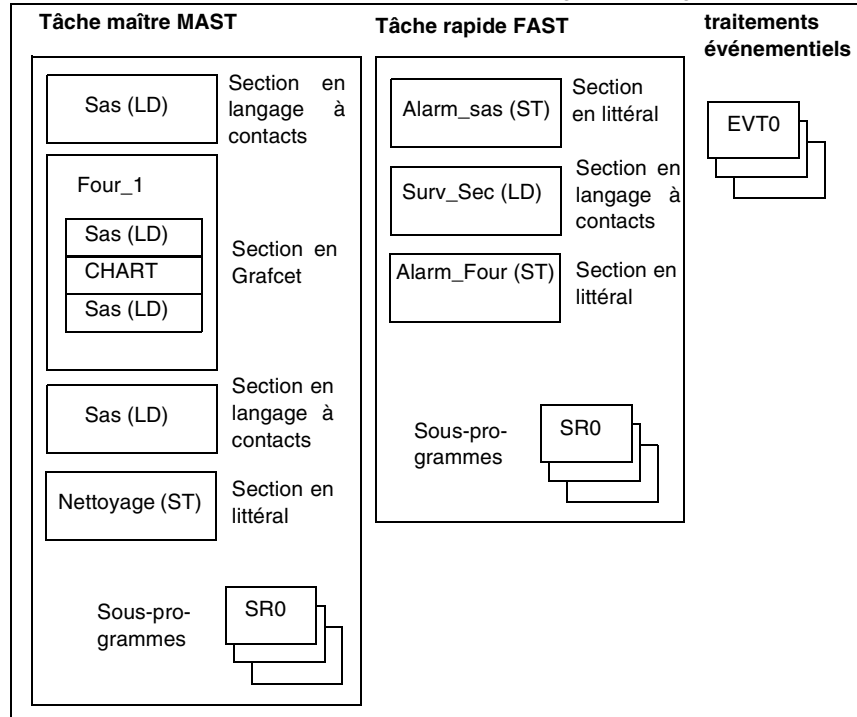
- **Monotâche** : c'est la structure simplifiée proposée par défaut, où une seule tâche maître composée d'un programme, constitué de plusieurs sections et de sous-programmes, est exécutée.
  - **Multitâche** : cette structure, mieux adaptée pour des applications temps réel performantes, se compose d'une tâche maître, d'une tâche rapide et de traitements événementiels prioritaires.
-

## Principe

Les tâches maître et rapide d'un programme PL7 se composent de plusieurs parties appelées sections et de sous-programmes.

Chacune de ces sections peut être programmée dans le langage approprié au traitement à réaliser.

L'illustration suivante montre un exemple de découpage d'un programme PL7.



Ce découpage en sections permet de créer un programme structuré et de générer ou incorporer aisément des modules de programme.

Les sous-programmes peuvent être appelés depuis n'importe quelle section de la tâche à laquelle ils appartiennent ou depuis d'autres sous-programmes de la même tâche.

## Modules fonctionnels

---

### Généralités

Le logiciel PL7 Pro permet de structurer une application pour automate Premium en modules fonctionnels.

Un module fonctionnel est un regroupement d'éléments de programme destinés à réaliser une fonction d'automatisme.

Indépendamment de la structure multitâches des automates, vous pouvez définir une structure arborescente multiniveaux de l'application d'automatisme.

A chaque niveau, vous pouvez rattacher des sections de programme écrites en langage à contacts, littéral, liste d'instructions ou Grafcet, ainsi que des tables d'animation et des écrans d'exploitation.

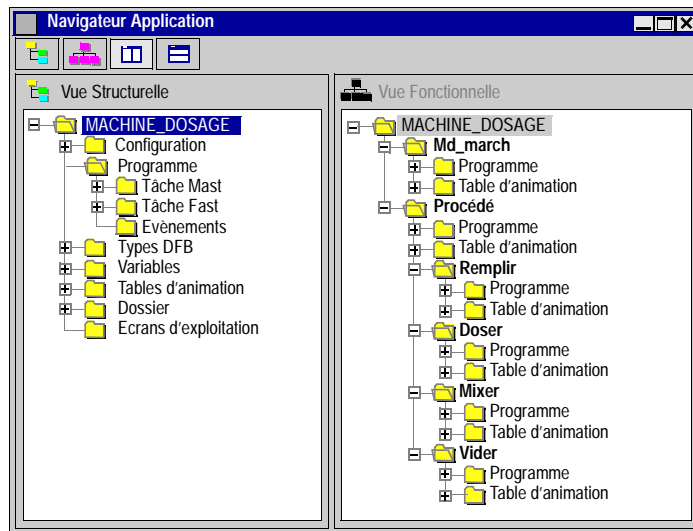
---

### Vue fonctionnelle

La vue fonctionnelle en modules permet d'avoir une découpe par fonctions cohérentes vis-à-vis du procédé à commander.

La vue structurelle donne une vue de l'ordre d'exécution des sections de programme par l'automate.

L'illustration suivante montre les 2 vues possibles d'une application.



**Services  
associés à la vue  
fonctionnelle**

Les services d'exploitation sont disponibles dans l'une ou l'autre vue. En particulier, par une seule commande, il est possible de forcer l'exécution ou non d'un module fonctionnel.

Dans ce cas, toutes les sections rattachées au module fonctionnel sont automatiquement forcées.

---

**Export/import de  
modules  
fonctionnels**

Vous pouvez exporter tout ou partie de la structure arborescente en modules fonctionnels.

Dans ce cas, l'ensemble des sections de programme des différents niveaux de modules est exporté.

---





---

# Description des objets langages PL7

# 2

---

## Présentation

### Objet de ce chapitre

Ce chapitre décrit tous les objets des langages PL7. Ces objets sont utilisés en tant qu'opérandes dans les instructions.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Définition des principaux objets booléen	26
Définition des principaux objets mots	27
Adressage des objets bits	29
Adressage des objets de modules d'entrées/sorties du TSX 37	31
Adressage des objets de modules d'entrées/sorties en rack	34
Adressage des objets langage de modules déportés sur bus FIPIO	36
Adressage des objets langage liés au bus AS-i	39
Adressage des objets mots	41
Règle de recouvrements	43
Objets de bloc fonction	44
Objets PL7 de type tableau	46
Objets indexés	48
Objets Grafcet	51
Symbolisation	52
Objets présymbolisés	54

---

## Définition des principaux objets booléen

**Description** Le tableau suivant décrit les principaux objets booléens.

Bits	Description	Exemples	Accès en écriture
Valeurs immédiates	0 ou 1 (False ou True)	0	–
Entrées/sorties	Ces bits sont les "images logiques" des états électriques des entrées/sorties. Ils sont rangés dans la mémoire de données et sont mis à jour à chaque scrutation de la tâche dans laquelle ils sont configurés.  <b>Note</b> : Les bits d'entrées/sorties non utilisés ne peuvent pas être employés comme bits internes.	%I23.5 %Q51.2	Non Oui
Internes	Les bits internes permettent de mémoriser des états intermédiaires durant l'exécution du programme.	%M200	Oui
Système	Les bits système %S0 à %S127 surveillent le bon fonctionnement de l'automate ainsi que le déroulement du programme application.	%S10	Selon i
Blocs fonction	Les bits de blocs fonction correspondent aux sorties des blocs fonction standard ou instance de DFB. Ces sorties peuvent être soit câblées directement, soit exploitées en tant qu'objet.	%TM8.Q	Non
Extraits de mots	Le logiciel PL7 donne la possibilité d'extraire l'un des 16 bits d'un objet mot.	%MW10:X5	Selon type de mots
Étapes et macro-étapes Grafset	Les bits d'état permettent de connaître l'état d'une étape, d'une macro-étape ou encore d'une étape de macro-étape.	%X21 %X5.9	Oui Oui

## Définition des principaux objets mots

**Description** Le tableau suivant décrit les principaux objets mots.

Mots	Description	Exemples	Accès en écriture
Valeurs immédiates	Ce sont des valeurs algébriques de format homogène avec celui des mots simple et double longueur (16 ou 32 bits), qui permettent d'affecter des valeurs à ces mots.	2542	–
Entrées/sorties	Ce sont les "images logiques" des valeurs électriques des entrées/sorties (exemple : entrées/sorties analogiques). Ils sont rangés dans la mémoire de données et sont mis à jour à chaque scrutation de la tâche dans laquelle ils sont configurés.	%IW23.5 %QW51.1	non oui
Internes	Ils sont destinés à stocker des valeurs en cours du programme. Ils sont rangés à l'intérieur de l'espace Données dans une même zone mémoire.	%MW10 %MD45	oui oui
Constants	Ils mémorisent les constantes ou les messages alphanumériques. Leur contenu ne peut être écrit ou modifié que par le terminal. Ils sont stockés au même endroit que le programme, Ils peuvent donc avoir comme support de la mémoire FLASH EPROM.	%KW30	oui (uniquement par terminal)
Système	Ces mots assurent plusieurs fonctions : <ul style="list-style-type: none"> <li>certains renseignent sur l'état du système (temps de fonctionnement système et application, ...).</li> <li>d'autres permettent d'agir sur l'application (modes de marche, ...).</li> </ul>	%SW5	selon i
Blocs fonction	Ces mots correspondent aux paramètres ou valeurs courantes des blocs fonction standard ou instance de DFB.	%TM2.P	oui
Communs	Ils sont destinés à être échangés automatiquement sur toutes les stations connectées au réseau de communication.	%NW2.3	oui
Grafcet	Les mots Grafcet permettent de connaître les temps d'activité des étapes.	%X5.T	oui

**Format des valeurs** Les valeurs des mots peuvent être codées dans les formats suivants :

Type	Taille	Exemple de valeur	Borne inférieure	Borne supérieure
Entier base 10	Simple longueur	1506	-32768	+32767
	Double longueur	578963	-2 147 483 648	2 147 483 647
Entier base 2	Simple longueur	2#1000111011111011011	2#10...0	2#01...1
	Double longueur	2#100011101111101101111111110111110111111	2#10...0	2#01...1
Entier base 16	Simple longueur	16#AB20	16#0000	16#FFFF
	Double longueur	16#5AC10	16#000000000	16#FFFFFFFF
Flottant		-1.32E12	-3.402824E+38 (1) 1.175494E-38 (1)	-1.175494E-38 (1) 3.402824E+38 (1)
Légende				
(1)	bornes exclues			

## Adressage des objets bits

**Présentation** L'adressage des bits internes, système et étapes suit les règles suivantes :

%	M, S ou X	i
Symbole	Type d'objet	Numéro

**Syntaxe** Le tableau ci-dessous décrit les différents éléments constituant l'adressage.

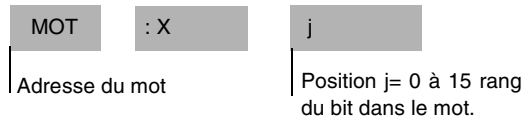
Famille	Élément	Valeurs	Description
Symbole	%	-	-
Type d'objet	M	-	<b>Bits internes</b> destinés à mémoriser les états intermédiaires en cours du programme. Ils sont rangés à l'intérieur de l'espace données dans une même zone mémoire.
	S	-	<b>Bits système</b> (Voir Manuel de référence, Tome 2), ces bits assurent plusieurs fonctions : <ul style="list-style-type: none"> <li>certains renseignent sur l'état du système par lecture des bits %Si (débordement du chien de garde, ...).</li> <li>d'autres permettent d'agir sur l'application (initialisation Grafcet, ...).</li> </ul>
	X	-	<b>Bits d'étape</b> , les bits étapes (Voir <i>Objets Grafcet</i> , p. 51) donnent l'état d'activité des étapes.
Numéro	i	-	La valeur maximum du numéro dépend du nombre d'objets configuré.

### Exemples :

- %M25 = bit interne numéro 25
- %S20 = bit système numéro 20
- %X6 = bit étape numéro 6

### Bits extrait de mots

Le logiciel PL7 permet d'extraire l'un des 16 bits des mots simple longueur. Le repère du mot est alors complété par le rang du bit extrait suivant la syntaxe ci-après:



#### Exemples :

- %MW10:X4 = bit numéro 4 du mot interne %MW10
- %QW5.1:X10 = bit numéro 10 du mot de sortie %QW5.1

<b>Note :</b> L'extraction de bits de mots peut aussi s'effectuer sur mots indexés.
---

---

---

## Adressage des objets de modules d'entrées/sorties du TSX 37

---

- Présentation** L'adressage des principaux objets bit et mot de modules d'entrées/sorties est de type géographique. C'est à dire qu'il dépend :
- du numéro (adresse) du rack,
  - de la position physique du module dans le bac,
  - du numéro de la voie du module.

**Illustration** L'adressage est défini de la manière suivante :

%	I,Q,M,K	X, W, D, F	X	i	r
Symbole	Type d'objet	Format	Position	N° voie	Rang

---

**Syntaxe**

Le tableau ci-dessous décrit les différents éléments constituant l'adressage.

Famille	Élément	Valeurs	Description
<b>Symbole</b>	%	-	-
<b>Type d'objet</b>	I	-	Image de l'entrée physique du module, Image de la sortie physique du module, Ces informations sont échangées de manière implicite à chaque cycle de la tâche à laquelle elles sont attachées.
	Q	-	
	M	-	Variable interne Ces informations de lecture ou d'écriture sont échangées à la demande de l'application.
	K	-	Constante interne Ces informations de configuration sont disponibles en lecture seulement.
Format (taille)	X	-	Booléen Pour les objets de type booléen, le X peut être omis.
	W	16 bits	Simple longueur.
	D	32 bits	Double longueur.
	F	32 bits	Flottant. Le format flottant utilisé est celui de la norme IEEE Std 754-1985 (équivalent IEC 559).
Position module	x	0 à 8 0 à 10	TSX 37-10 TSX 37-21/22 <b>Note</b> : un module au format standard (occupant 2 positions) est adressé comme 2 modules au 1/2 format superposés, (voir les explications ci-après).
N° voie	i	0 à 31 ou MOD	Numéro de voie du module. MOD : voie réservée à la gestion du module et des paramètres communs à toutes les voies.
Rang	r	0 à 127 ou ERR	Position du bit dans le mot. ERR : indique un défaut module ou voie.

**Exemples**

Le tableau ci-dessous présente quelques exemples d'adressage d'objets.

Objet	Description
%I.5	Voie d'entrée numéro 5 du module d'entrées/sorties situé à la position 1.
%MW2.0.3	Mot d'état de rang 3 de la voie 0 du module d'entrées/sorties situé à la position 2.
%I5.MOD.ERR	Information de défaut du module d'entrées/sorties situé à la position 5.



**Cas des modules au format standard** Ils sont adressés comme 2 modules au 1/2 format superposés.

Par exemple, un module de 64 E/S occupant les positions 5 et 6, est vu comme 2 modules 1/2 format :

- un 1/2 module de 32 entrées situé à la position 5,
- un 1/2 module de 32 entrées situé à la position 6,

Le tableau ci-après décrit le codage Position/Numéro de voie en fonction du module.

Module	1/2 format			Format standard			
	4S	8E	12E	28E/S	32E	32S	64E/S
Numéro de voie	0 à 3	0 à 7	0 à 11	0 à 15 (E)	0 à 15 (E)	0 à 15 (S)	0 à 31 (E)
				0 à 11 (S)	0 à 15 (E)	0 à 15 (S)	0 à 31 (S)
Adressage : Position/Numéro de voie (x=position)	x.0	x.0	x.0	x.0	x.0	x.0	x.0
	à	à	à	à	à	à	à
	x.3	x.7	x.11	x.15	x.15	x.15	x.31
				(x+1).0	(x+1).0	(x+1).0	(x+1).0
				à	à	à	à
				(x+1).11	(x+1).15	(x+1).15	(x+1).31

**Exemples** Le tableau ci-dessous présente deux exemples d'adressage d'objets d'un module standard 28E/S occupant les positions 3 et 4.

Objet	Description
%I3.6	Voie d'entrée numéro 6 du module.
%Q4.2	Voie de sortie numéro 2 du module

## Adressage des objets de modules d'entrées/sorties en rack

### Présentation

L'adressage des principaux objets bit et mot de modules d'entrées/sorties est de type géographique. C'est à dire qu'il dépend :

- du numéro (adresse) du rack,
- de la position physique du module dans le rack,
- du numéro de la voie du module.

### Illustration

L'adressage est défini de la manière suivante :

%	I, Q, M, K	X, W, D, F	X	Y	i	r
Symbole	Type d'objet	Format	Rack	Position	N° voie	Rang

### Syntaxe

Le tableau ci-dessous décrit les différents éléments constituant l'adressage.

Famille	Élément	Valeurs	Description
Symbole	%	-	-
Type d'objet	I	-	Image de l'entrée physique du module,
	Q	-	Image de la sortie physique du module, Ces informations sont échangées de manière automatique à chaque cycle de la tâche à laquelle elles sont attachées.
	M	-	Variable interne Ces informations de lecture ou d'écriture sont échangées à la demande de l'application.
	K	-	Constante interne Ces informations de configuration sont disponibles en lecture seulement.
Format (taille)	X	-	Booléen Pour les objets de type booléen, cet élément peut être omis.
	W	16 bits	Simple longueur.
	D	32 bits	Double longueur.
	F	32 bits	Flottant. Le format flottant utilisé est celui de la norme IEEE Std 754-1985 (équivalent IEC 559).
Adresse rack	x	0 ou 1 0 à 7	TSX 5710/102/103/153, PMX 57102, PCX 571012). Autres processeurs.
Position module	y	00 à 14 (1)	Numéro de position dans le rack. Lorsque le numéro de rack (x) est différent de 0, la position (y) est codée sur 2 digits : 00 à 14 ; par contre si le numéro de rack (x) = 0, on élimine les zéros non significatifs (élimination par la gauche) de "y" ("x" n'apparaît pas et "y" est sur 1 digit pour les valeurs inférieures à 9).

**(1) : le nombre d'emplacements maximum nécessite l'utilisation de 2 racks à la même adresse.**

Famille	Elément	Valeurs	Description
N° voie	i	0 à 127 ou MOD	MOD : voie réservée à la gestion du module et des paramètres communs à toutes les voies.
Rang	r	0 à 127 ou ERR	Position du bit dans le mot. ERR : indique un défaut module ou voie.

**(1) : le nombre d'emplacements maximum nécessite l'utilisation de 2 racks à la même adresse.**

### Exemples

Le tableau ci-dessous présente quelques exemples d'adressage d'objets.

Objet	Description	Illustration
%MW2.0.3	Mot d'état de rang 3 de la voie 0 du module d'entrées TOR situé à la position 2 du rack 0.	
%MW103.0.3	Mot d'état de rang 3 de la voie 0 du module de sorties TOR situé à la position 3 du rack 1.	
%I102.MOD.ERR	Information de défaut du module d'entrées analogiques situé à la position 2 du rack 1.	
%I204.3.ERR	Information de défaut de la voie 3 du module de sorties analogiques situé à la position 4 du rack 2.	

## Adressage des objets langage de modules déportés sur bus FIPIO

---

### Présentation

L'adressage des principaux objets bit et mot des modules déportés sur bus FIPIO est de type géographique. C'est à dire qu'il dépend :

- du point de connexion,
  - du type de module (base ou extension),
  - du numéro de la voie.
- 

### Illustration

L'adressage est défini de la manière suivante :

<b>%</b>	<b>I, Q, M, K</b>	<b>X, W, D, F</b> \	<b>p.2.c</b> \	<b>m</b>	<b>i</b>	<b>r</b>
Symbole	Type d'objet	Format	Adresse module/voie et point de connexion	N°de module	N° voie	Rang

---

**Syntaxe**

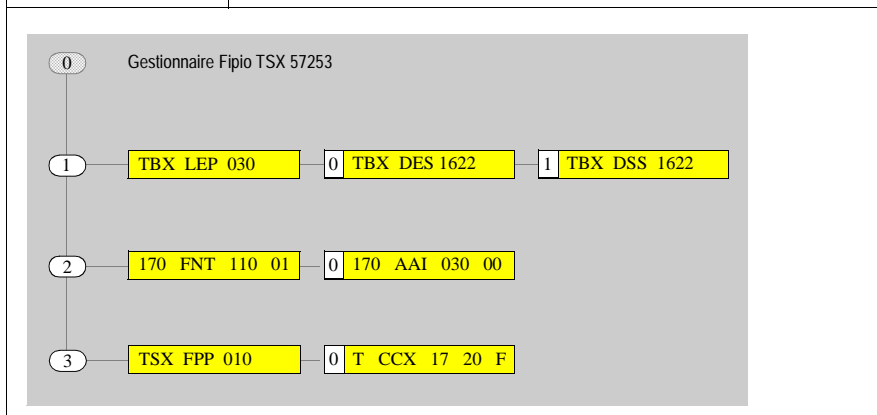
Le tableau ci-dessous décrit les différents éléments constituant l'adressage.

Famille	Élément	Valeurs	Signification
<b>Symbole</b>	%	-	-
<b>Type d'objet</b>	I	-	Image de l'entrée physique du module, Image de la sortie physique du module, Ces informations sont échangées de manière automatique à chaque cycle de la tâche à laquelle elles sont attachées.
	Q	-	
	M	-	
	K	-	Constante interne Ces informations de configuration sont disponibles en lecture seulement.
Format (taille)	X	-	Booléen Pour les objets de type booléen, le X peut être omis.
	W	16 bits	Simple longueur.
	D	32 bits	Double longueur.
	F	32 bits	Flottant. Le format flottant utilisé est celui de la norme IEEE Std 754-1985 (équivalent IEC 559).
Adresse module/ voie et point de connexion	p	0 ou 1	Numéro de position du processeur dans le rack.
	2	-	Numéro de voie de la liaison FIPIO intégrée dans le processeur.
	c	1 à 127	Numéro de point de connexion.
Position module	m	0 ou 1	0 : module de base, 1 : module d'extension.
N° voie	i	0 à 127 ou MOD	MOD : voie réservée à la gestion du module et des paramètres communs à toutes les voies.
Rang	r	0 à 255 ou ERR	ERR : indique un défaut module ou voie.

**Exemples**

Le tableau ci-dessous présente quelques exemples d'adressage d'objets.

Objet	Signification
%MW\0.2.1\0.5.2	Mot d'état de rang 2 du bit image de l'entrée 5 du module de base d'entrées déportées situé au point de connexion 1 du bus FIPIO.
%I\0.2.1\0.7	Bit image de l'entrée 7 du module de base d'entrées déportées situé au point de connexion 1 du bus FIPIO.
%Q\0.2.1\1.2	Bit image de la sortie 2 du module d'extension de sorties déportées situé au point de connexion 1 du bus FIPIO.
%I\0.2.2\0.MOD.ERR	Information de défaut du module Momentum situé au point de connexion 2 du bus FIPIO.
%Q\1.2.3\0.0.ERR	Information de défaut de la voie 0 du module CCX17 situé au point de connexion 3 du bus FIPIO.



## Adressage des objets langage liés au bus AS-i

### Présentation

L'adressage des principaux objets bit et mot associés au bus AS-i est de type géographique. C'est à dire qu'il dépend :

- du numéro (adresse) du rack où est positionné le coupleur,
- de la position physique du coupleur dans le rack,
- du numéro (adresse) de l'équipement esclave sur le bus AS-i.

### Illustration

L'adressage est défini de la manière suivante :

%	I ou Q	\	xy.0	\	n	i
Symbole	Type d'objet		Rack/module/voie du TSX SAY 100		N° esclave	Rang du bit

### Syntaxe

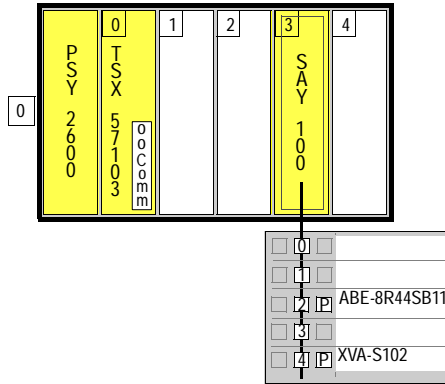
Le tableau ci-dessous décrit les différents éléments constituant l'adressage.

Famille	Elément	Valeurs	Description
Symbole	%	-	-
Type d'objet	I Q	- -	Image de l'entrée physique du module, Image de la sortie physique du module, Ces informations sont échangées de manière automatique à chaque cycle de la tâche à laquelle elles sont attachées.
Adresse rack	x	0 ou 1 0 à 7	TSX 5710/102/103/153, PMX 57102, PCX 571012). Autres processeurs.
Position module	y	00 à 14 (1)	Numéro de position dans le rack. Lorsque le numéro de rack (x) est différent de 0, la position (y) est codée sur 2 digits : 00 à 14 ; par contre si le numéro de rack (x) = 0, on élimine les zéros non significatifs (élimination par la gauche) de "y" ("x" n'apparaît pas et "y" est sur 1 digit pour les valeurs inférieures à 9).
N° voie	0	-	Le coupleur TSX SAY 100 ne possède qu'une seule voie.
N° de l'esclave	n	0 à 31	Adresse physique de l'esclave.
Rang	i	0 à 3	Position du bit image de l'entrée ou de la sortie.
<b>(1) : Le nombre d'emplacements maximum nécessite l'utilisation d'un rack d'extension.</b>			

**Exemple**

Le tableau ci-dessous présente quelques exemples d'adressage d'objets.

Objet	Description
%I3.0\2.1	Entrée 1 de l'esclave 2, le module TSX SAY 100 étant positionné à l'emplacement 3 du rack 0.
%Q3.0\4.3	Sortie 3 de l'esclave 4, le module TSX SAY 100 étant positionné à l'emplacement 3 du rack 0.





## Adressage des objets mots

**Présentation** L'adressage des mots (hors mots de modules d'entrées/sorties, et blocs fonction) suivent une même syntaxe décrite ci-après.

**Illustration** L'adressage des mots internes, constants et système suit les règles suivantes :

%	M, K ou S	B, W, D ou F	i
Symbole	Type d'objet	Format	Numéro

**Syntaxe** Le tableau ci-dessous décrit les différents éléments constituant l'adressage.

Famille	Elément	Valeurs	Description																															
Symbole	%	-	-																															
Type d'objet	M	-	<b>Mots internes</b> destinés à stocker des valeurs en cours du programme. Ils sont rangés à l'intérieur de l'espace donné dans une même zone mémoire.																															
	K	-	<b>Mots constants</b> mémorisent des valeurs constantes ou des messages alphanumériques. Leur contenu ne peut être écrit ou modifié que par le terminal. Ils sont stockés au même endroit que le programme, Ils peuvent donc avoir comme support de la mémoire FLASH EPROM.																															
	S	-	<b>Mots système</b> (Voir Manuel de référence, Tome 2), ces mots assurent plusieurs fonctions : <ul style="list-style-type: none"> <li>certains renseignent sur l'état du système par lecture des mots %SWi (temps de fonctionnement système et application, ...).</li> <li>d'autres permettent d'agir sur l'application (modes de marche, ...).</li> </ul>																															
Format	B	8 bits	<b>Octet</b> , ce format est exclusivement utilisé pour les opérations sur chaîne de caractères.																															
	W	16 bits	<b>Simple longueur</b> : ces mots de 16 bits peuvent contenir une valeur algébrique comprise entre -32 768 et 32 767, <div style="text-align: right; margin-top: 10px;">Rang du bit</div> <table style="margin: 10px auto; border-collapse: collapse;"> <tr> <td style="padding: 2px 5px;">15</td><td style="padding: 2px 5px;">14</td><td style="padding: 2px 5px;">13</td><td style="padding: 2px 5px;">12</td> <td style="padding: 2px 5px;">11</td><td style="padding: 2px 5px;">10</td><td style="padding: 2px 5px;">9</td><td style="padding: 2px 5px;">8</td> <td style="padding: 2px 5px;">7</td><td style="padding: 2px 5px;">6</td><td style="padding: 2px 5px;">5</td><td style="padding: 2px 5px;">4</td> <td style="padding: 2px 5px;">3</td><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px 5px;">0</td><td style="border: 1px solid black; padding: 2px 5px;">1</td><td style="border: 1px solid black; padding: 2px 5px;">1</td><td style="border: 1px solid black; padding: 2px 5px;">1</td> <td style="border: 1px solid black; padding: 2px 5px;">0</td><td style="border: 1px solid black; padding: 2px 5px;">1</td><td style="border: 1px solid black; padding: 2px 5px;">1</td><td style="border: 1px solid black; padding: 2px 5px;">1</td> <td style="border: 1px solid black; padding: 2px 5px;">0</td><td style="border: 1px solid black; padding: 2px 5px;">0</td><td style="border: 1px solid black; padding: 2px 5px;">1</td><td style="border: 1px solid black; padding: 2px 5px;">1</td> <td style="border: 1px solid black; padding: 2px 5px;">0</td><td style="border: 1px solid black; padding: 2px 5px;">1</td><td style="border: 1px solid black; padding: 2px 5px;">0</td><td style="border: 1px solid black; padding: 2px 5px;">0</td> </tr> </table> <div style="display: flex; justify-content: space-between; margin-top: 5px;"> <span>Poids fort</span> <span>Poids faible</span> </div>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																			
0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	0																			

Famille	Élément	Valeurs	Description								
Format	D	32 bits	<p><b>Double longueur</b> : ces mots de 32 bits peuvent contenir une valeur algébrique comprise entre -2 147 483 648 et 2 147 483 647. Ces mots s'implantent en mémoire sur deux mots simple longueur consécutifs.</p> <p style="text-align: right;">Poids faible</p> <p style="text-align: center;">15 14 13 12      11 10 9 8      7 6 5 4      3 2 1 0</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">0 1 1 1</td> <td style="text-align: center;">0 1 1 1</td> <td style="text-align: center;">0 0 1 1</td> <td style="text-align: center;">0 1 0 0</td> </tr> <tr> <td style="text-align: center;">0 0 1 1</td> <td style="text-align: center;">0 1 1 0</td> <td style="text-align: center;">0 1 0 1</td> <td style="text-align: center;">0 0 1 0</td> </tr> </table> <p style="text-align: left;">Poids fort</p>	0 1 1 1	0 1 1 1	0 0 1 1	0 1 0 0	0 0 1 1	0 1 1 0	0 1 0 1	0 0 1 0
	0 1 1 1	0 1 1 1	0 0 1 1	0 1 0 0							
0 0 1 1	0 1 1 0	0 1 0 1	0 0 1 0								
F	32 bits	<p><b>Flottant</b>: le format flottant utilisé est celui de la norme IEEE Std 754-1985 (équivalent IEC 559). La longueur des mots est de 32 bits, ce qui correspond à des nombres flottants simple précision.</p> <p>Exemples de valeurs flottantes :</p> <p>1285.28 12.8528E2</p>									
Numéro	i	-	La valeur maximum du numéro dépend du nombre d'objets configuré.								

**Exemples :**

- %MW15 = mot interne simple longueur numéro 15
- %MF20 = mot interne flottant numéro 20
- %KD26 = double mot constant numéro 26
- %SW30 = mot système numéro 30

**Adressage des mots sur réseau**

L'adressage des mots sur réseau est décrit dans le manuel Métier communication.

D'autre part les réseaux utilisent des objets spécifiques : les mots communs .Ce sont des objets mots simple longueur (16 bits) communs à toutes les stations connectées au réseau de communication.

Adressage : %NW{i.j}k

avec : i = 0 à 127 numéro de réseau, j = 0 à 31 numéro de station et k= 0 à 3 numéro de mot

## Règle de recouvrements

### Principes

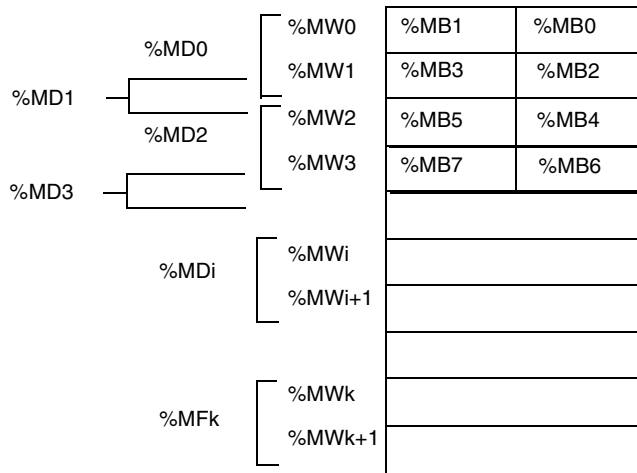
Les octets, mots simples, double longueur et flottant sont rangés à l'intérieur de l'espace donné dans une même zone mémoire.

Ainsi, il y a recouvrement entre :

- le mot double longueur  $\%MD_i$  et les mots simple longueur  $\%MW_i$  et  $\%MW_{i+1}$  (le mot  $\%MW_i$  renfermant les poids faibles et le mot  $\%MW_{i+1}$  les poids forts du mot  $\%MD_i$ ),
- le mot simple longueur  $\%MW_i$  et les octets  $\%MB_j$  et  $\%MB_{j+1}$  (avec  $j=2 \times i$ ),
- le flottant  $\%MF_k$  et les mots simple longueur  $\%MW_k$  et  $\%MW_{k+1}$ .

### Illustration

Cette illustration montre le recouvrement des mots internes.



### Exemples

- $\%MD_0$  correspond à  $\%MW_0$  et  $\%MW_1$  (voir illustration ci-dessus).
- $\%MW_3$  correspond à  $\%MB_7$  et  $\%MB_6$  (voir illustration ci-dessus).
- $\%KD543$  correspond à  $\%KW543$  et  $\%KW544$ .
- $\%MF_{10}$  correspond à  $\%MW_{10}$  et  $\%MW_{11}$ .

## Objets de bloc fonction

---

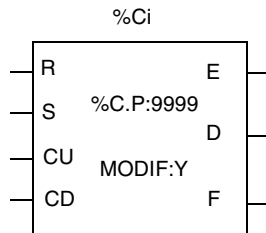
### Généralités

Les blocs fonction mettent en oeuvre des objets bits et des mots spécifiques accessibles par programme.

---

### Exemple de bloc fonction

L'illustration suivante présente un bloc fonction compteur/décompteur.



Bloc compteur/décompteur

---

### Objets bits

Ils correspondent aux sorties des blocs. Ces bits sont accessibles par les instructions booléennes de test.

---

### Objets mots

Ils correspondent :

- aux paramètres de configuration du bloc, ces paramètres peuvent être accessibles ( ex : paramètre de présélection) ou pas (ex: base de temps) par programme,
  - aux valeurs courantes (ex : %Ci.V valeur de comptage en cours).
-

**Liste des objets de blocs fonction accessibles par programme** Le tableau suivant décrit l'ensemble des objets de blocs fonction.

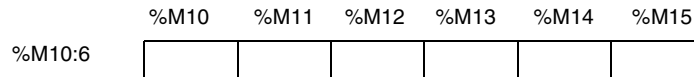
Blocs fonctions	Symbole	Nb Maxi Micro	Nb Maxi Premium	Type d'objets	Description	Adresse	Accès écriture
Temporisateur	%TMi	64	255 (128 par défaut)	Mot	Valeur courante	%TMi.V	non
					Valeur de présélection	%TMi.P	oui
				Bit	Sortie temporisateur	%TMi.Q	non
Compteur/ Décompteur	%Ci	32	255 (64 par défaut)	Mot	Valeur courante	%Ci.V	non
					Valeur de présélection	%Ci.P	oui
				Bit	Sortie débordement (vide)	%Ci.E	non
					Sortie présélection atteinte	%Ci.D	non
Monostable	%MNi	8	255 (32 par défaut)	Mot	Valeur courante	%MNi.V	non
					Valeur de présélection	%MNi.P	oui
				Bit	Sortie débordement (vide)	%MNi.R	non
Registre mot	%Ri	4	255 (4 par défaut)	Mot	Accès au registre	%Ri.I	oui
					Sortie du registre	%Ri.O	oui
				Bit	Sortie registre plein	%Ri.F	non
					Sortie registre vide	%Ri.E	non
Programmateur cyclique	%DRi	8	255 (8 par défaut)	Mot	Numéro de pas en cours	%DRi.S	oui
					Etats du pas j	%DRi.Wj	non
					Temps d'activité du pas	%DRi.V	non
				Bit	Dernier pas défini en cours	%DRi.F	non
Temporisateur série 7	%Ti	64	255 (0 par défaut)	Mot	Valeur courante	%Ti.V	non
					Valeur de présélection	%Ti.P	oui
				Bit	Sortie en cours	%Ti.R	non
					Sortie temporisateur écoulé	%Ti.D	non

**Note :** le nombre total de temporisateurs %TMi + %Ti est limité à 64 pour un Micro, et 255 pour un Premium.

## Objets PL7 de type tableau

**Tableaux de bits** Les tableaux de bits sont des suites d'objets bits adjacents de même type et de longueur définie : L.

Exemple de tableaux de bits : %M10:6



Ce tableau définit les objets bits qui peuvent être mis sous forme de tableau de bits.

Type	Adresse	Exemple	Accès en écriture
Bits d'entrées TOR	%Ix.i:L	%I25.1:8	Non
Bits de sorties TOR	%Qx.i:L	%Q34.0:16	Oui
Bits internes	%Mi:L	%M50:20	Oui
Bits Grafcet	%Xi:L, %Xj.i:L	%X50:30	Non

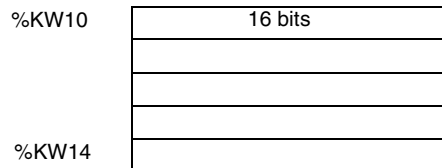
**Note** : Les longueurs maximum des tableaux dépendent des types d'objet

- **Pour les bits d'entrées/sorties TOR** : la taille maximum dépend de la modularité (nombre d'entrées/sorties du module).
- **Pour les bits internes ou Grafcet** : la taille maximum dépend de la taille définie en configuration.

### Tableaux de mots

Les tableaux de mots sont des suites de mots adjacents de même type et de longueur définie : L.

Exemple de tableaux mots : %KW10:5



Ce tableau définit les objets mots qui peuvent être mis sous forme de tableau de mots.

Type	Format	Adresse	Exemple	Accès en écriture
Mots internes	Simple longueur	%MWi:L	%MW50:20	Oui
	Double longueur	%MDi:L	%MD30:10	Oui
	Flottant	%MFi:L	%MF100:20	Oui
Mots constants	Simple longueur	%KWi:L	%KW50:20	Non
	Double longueur	%KDi:L	%KD30:10	Non
	Flottant	%KFi:L	%KF100:20	Non
Mots Grafcet	Mots Grafcet	%Xi.T:L, %Xj.i.T:L	%X12.T:8	Non
Mots système	Mots système	%SWi:L	%SW50:4	Oui

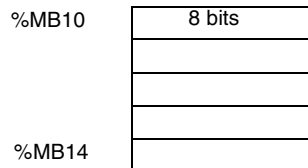
**Note** : Les longueurs maximum des tableaux dépendent des types d'objet.

- **Pour les mots internes, constantes ou Grafcet** : la taille maximum dépend de la taille définie en configuration.
- **Pour les mots système** : seul les mots %SW50 à 53 peuvent être adressés sous forme de tableau.

## Chaînes de caractères

Les chaînes de caractères sont des suites d'octets adjacents de même type et de longueur définie : L.

Exemple de chaîne de caractères : %MB10:5



Ce tableau définit les objets qui peuvent être mis sous forme de chaîne de caractères.

Type	Adresse	Exemple	Accès en écriture
Mots internes	%MBi:L	%MB10:8	Oui
Mots constants	%KBi:L	%KB20:6	Oui

**Note** : l'indice i doit être pair.

## Objets indexés

---

**Adressage direct** L'adressage des objets est dit direct, quand l'adresse de ces objets est fixe et définie à l'écriture du programme.

**Exemple :** %MW26 (mot interne d'adresse 26)

---

**Adressage indexé** En adressage indexé, l'adresse directe de l'objet est complétée d'un index : à l'adresse de l'objet est ajouté le contenu de l'index.

Le tableau suivant présente le type d'index qui peut être utilisé en fonction du type d'automate :

Type d'index	Automates Premium/ Atrium	Automates Micro
un mot interne %MWi	X	X
un mot constant %KWi	X	-
une valeur immédiate	X	-
Légende		
X : utilisable - : non utilisable		

Le nombre de "mots index" n'est pas limité.

Ce type d'adressage permet de parcourir successivement une suite d'objets de même nature (mots internes, mots constants...), : à l'adresse de l'objet est ajoutée le contenu de l'index.

**Exemple :**

MW108[%MW2] : mot d'adresse directe 108 + contenu du mot %MW2.

Si le mot %MW2 a pour contenu la valeur 12, écrire %MW108[%MW2] équivaut donc à écrire %MW120.

---



## Description des objets indexables

Le tableau suivant définit les objets qui peuvent être indexés.

Type	Format	Adresse	Exemple	Accès en écriture
Bit d'entrées	Booléen	%Ixy.i[index]	%I21.3[%MW5]	Non
Bit de sortie	Booléen	%Qxy.i[index]	%Q32.4[%MW5]	Oui
Bit interne	Booléen	%Mi[index]	%M10[%MW5]	Oui
Bit Grafcet	Booléen	%Xi[index]	%X20[%MW5]	Non
		%Xj.i[index]	%X2.3[%MW5]	Non
Mots internes	Simple longueur	%MWi[index]	%MW30[%MW5]	Oui
	Double longueur	%MDi[index]	%MD15[%MW5]	Oui
	Flottant	%MFi[index]	%MF15[%MW5]	Oui
Mots constant	Simple longueur	%KWi[index]	%KW50[%MW5]	Non
	Double longueur	%KDi[index]	%KD50[%MW5]	Non
	Flottant	%KFi[index]	%KF50[%MW5]	Non
Mots Grafcet	Simple longueur	%Xi .T[index]	%X20 .T[%MW5]	Non
		%Xj.i .T[index]	%X2.3 .T[%MW5]	Non
Tableau de mots		%MWi[index]:L	%MW50[%MW5]:10	Oui
		%MDi[index]:L	%MD40[%MW5]:10	Oui
		%KWi[index]:L	%KW70[%MW5]:20	Non
		%KDi[index]:L	%KD80[%MW5]:10	Non

**Note :** Les valeurs maximum des index dépendent des types d'objets indexés.

- **Pour les bits d'entrées/sorties TOR :**  $0 < i + \text{index} < m$  (m étant le nombre maximum d'entrées/sorties du module) .
- **Pour tous les autres objets** (exceptés objet double longueur ou flottant) :  $0 < i + \text{index} < N_{\text{max}}$  ,  $N_{\text{max}}$  = taille maximum dépend de la taille définie en configuration.  
Pour les mots double longueur ou flottant :  $0 < i + \text{index} < N_{\text{max}} - 1$ .

## Indexation des mots doubles

L'adresse réelle = adresse directe du double mot indexé + 2 fois le contenu du mot index.

**Exemple :** %MD6[%MW100]

Si %MW100=10, le mot adressé sera 6 + 2 x 10 -->%MD26.

**Débordement  
d'index**

Il y a débordement d'index dès que l'adresse d'un objet indexé dépasse les limites de la zone incluant ce même type d'objet, c'est-à-dire quand :

- adresse objet + contenu de l'index inférieur à la valeur zéro,
- adresse objet + contenu de l'index supérieur à la limite maximum configurée

En cas de débordement d'index, le système provoque la mise à l'état 1 du bit système %S20 et l'affectation de l'objet s'effectue avec une valeur d'index égale à 0.

Le tableau suivant donne les conditions de mise à 1 et 0 du bit système %S20.

Mis à l'état 1	Remis à l'état 0
● mise à 1 par le système sur débordement d'index	● mise à 0 par l'utilisateur, après modification de l'index

---

## Objets Grafcet

---

### Objets bits

Le tableau suivant récapitule tous les objets bits Grafcet disponibles et décrit leur rôle.

Type	Description
%Xi	état de l'étape i du graphe principal (Chart).
%XMj	état de la macro-étape j du Grafcet.
%Xj.i	état de l'étape i de la macro-étape j du Grafcet.
%Xj.IN	état de l'étape d'entrée de la macro-étape.
%Xj.OUT	état de l'étape de sortie de la macro-étape.

Ces bits sont à 1 lorsque l'étape ou la macro-étape est active, à 0 lorsqu'elle est inactive.

---

### Objets mots

Le tableau suivant récapitule tous les objets mots Grafcet disponibles et décrit leur rôle.

Type	Description
%Xi.Ti	temps d'activité de l'étape i du Grafcet.
%Xj.i.T	temps d'activité de l'étape i de la macro-étape j du Grafcet.
%Xj.IN.T	temps d'activité de l'étape i de la macro-étape j qui lui permettent de connaître l'état de l'étape i de la macro-étape j du Grafcet.
%Xj.OUT.T	temps d'activité de l'étape d'entrée de la macro-étape.
%Xj.OUT	temps d'activité de l'étape de sortie de la macro-étape.

Ces mots incrémentés toutes les 100 ms et prennent une valeur entre 0 et 9999.

---

## Symbolisation

---

### Rôle

Les symboles permettent d'adresser les objets langage PL 7, par des noms ou mnémoniques personnalisés.

---

### Syntaxe

Un symbole est une chaîne de 32 caractères alphanumériques maximum, dont le premier caractère est alphabétique.

Un symbole commence par une lettre majuscule, les autres étant en minuscule (par exemple : `Bruleur_1`).

A la saisie le symbole peut être écrit en majuscules ou en minuscules (par exemple: `BRULEUR_1`), le logiciel met automatiquement le symbole dans sa forme correcte.

---

### Caractères utilisables

Le tableau suivant fournit les caractères utilisables dans la création des symboles.

Type	Description
alphabétiques majuscules	"A à Z" et lettres suivantes "ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞ"
alphabétiques minuscules	"a à z", et lettres accentuées : àáâãäåæçèéêëìíîïðóôõöøùúûüýþÿ
numériques	chiffres de 0 à 9 (ils ne peuvent pas être placés en début de symbole).
le caractère "_"	il ne peut être placé ni en début de symbole, ni en fin de symbole.

Un certain nombre de mots sont réservés par le langage et ne sont pas utilisables comme symboles, voir (Voir Manuel de référence, Tome 3).

---

### Edition des symboles

Les symboles sont définis et associés aux objets langages par l'éditeur de variables, un commentaire de 508 caractères peut être associé à chaque symbole.

Les symboles et leurs commentaires sont sauvegardés sur le disque du terminal et non dans l'automate.

---

## Objets symbolisables

Tous les objets PL7 sont symbolisables exceptés les objets structurés de types tableaux et objets indexés, mais si l'objet de base ou l'index est symbolisé, le symbole est utilisé dans l'objet structuré.

### Exemples :

- si le mot %MW0 a pour symbole "Température", le tableau de mots %MW0:12 est symbolisé par `Température:12`,
- le mot %MW10 a pour symbole `Four_1`, le mot indexé %MW0[%MW10] est symbolisé par `Température[Four_1]`.

Les objets bits extraits de mots, bits ou mots de blocs fonction sont symbolisables mais s'ils ne sont pas symbolisés, ils peuvent hériter du symbole de l'objet de base .

### Exemples :

- si le mot %MW0 a pour symbole `Etat_pompe` et si le bit extrait de mot %MW0:X1 n'est pas symbolisé, il hérite du symbole du mot, %MW0:X1 a pour symbole : `Etat_pompe:X1`,
- si le bloc fonction %TM0 a pour symbole `Tempo_four1` et si la sortie %TM0.D n'est pas symbolisée, elle hérite du symbole du bloc, %TM0.D a pour symbole: `Tempo_four.D`.

## Objets uniquement symboliques

Les paramètres de blocs fonction DFB sont accessibles uniquement sous forme de symboles. Ces objets sont définis par la syntaxe suivante :

### Nom\_DFB.Nom\_paramètre

Les éléments ont la signification et les caractéristiques suivantes.

Élément	Nb de caractères maximum	Description
Nom_DFB	32	nom donné au bloc fonction DFB utilisé.
Nom_paramètre	8	nom donné au paramètre de sorties ou à la variable publique.

**Exemple :** `Controle.Ecart` pour la sortie `Ecart` de l'instance DFB nommée `Controle`.

## Objets présymbolisés

### Rôle

Certains modules métier (exemple : comptage, commande d'axes, ...) permettent une symbolisation automatique des objets qui leur sont associés.

Si vous donnez le symbole générique de la voie %CHxy.i du module, tous les symboles des objets associés à cette voie peuvent être alors sur demande générés automatiquement.

### Syntaxe

Ces objets sont symbolisés avec la syntaxe suivante:

#### Préfixe\_utilisateur\_Suffixe\_constructeur

Les éléments ont la signification et les caractéristiques suivantes :

Élément	Nb de caractères maximum	Description
Préfixe_utilisateur	12	symbole générique donné à la voie par l'utilisateur
Suffixe_constructeur	20	partie du symbole correspondant à l'objet bit ou mot de la voie donnée par le système

**Note :** En plus du symbole, un commentaire constructeur est généré automatiquement, ce commentaire rappelle succinctement le rôle de l'objet.

### Exemple

Cet exemple traite le cas d'un module de comptage situé dans l'emplacement 3 du bac automate.

Si le symbole générique (préfixe-utilisateur) donné à la voie 0 est `Compt_pieces`, les symboles suivants sont générés automatiquement.

Repère	Type	Symbole	Commentaire
%CH3.0	CH		
%ID3.0	DWORD	<code>Compt_pieces_cur_meas</code>	Mesure courante du compteur
%ID3.0.4	DWORD	<code>Compt_pieces_capt</code>	Valeur capturée du compteur
%I3.0	EBOOL	<code>Compt_pieces_enab_activ</code>	Validation active
%I3.0.1	EBOOL	<code>Compt_pieces_pres_done</code>	Présélection effectuée

---

# Mémoire utilisateur

# 3

---

## Présentation

### Objet de ce chapitre

Ce chapitre décrit la structure mémoire des automates Micro et Premium.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Structure mémoire des automates Micro	56
Structure mémoire des automates Premium	58
Description de la mémoire bits	59
Description de la mémoire mots	61
Caractéristiques de la mémoire des automates TSX 37	62
Caractéristiques de la mémoire des automates TSX/PCX 57 10/15/20/25/26/28	64
Caractéristiques de la mémoire des automates TSX/PCX 57 30/35/36	66
Caractéristiques de la mémoire des automates TSX 57 453/4823	68

## Structure mémoire des automates Micro

### Généralités

L'espace mémoire des automates Micro accessible à l'utilisateur est découpé en deux ensembles distincts :

- mémoire bits
- mémoire mots

### Mémoire bits

La mémoire bit est située dans la mémoire RAM intégrée au module processeur. Elle contient l'image des 1280 objets bits.

### Rôle de la mémoire mots

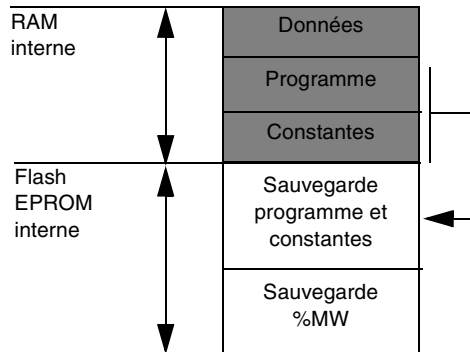
La mémoire mots (16 bits) supporte :

- **les données** : données dynamiques de l'application et données système,
- **le programme** : descripteurs et code exécutable des tâches,
- **les constantes** : mots constants, valeurs initiales et configuration des entrées/sorties.

### Structure sans carte mémoire d'extension

Les données, programme et constantes sont supportés par la mémoire RAM interne au module processeur.

Le schéma suivant décrit la structure mémoire.



La mémoire Flash EPROM intégrée au module processeur peut être utilisée pour la sauvegarde :

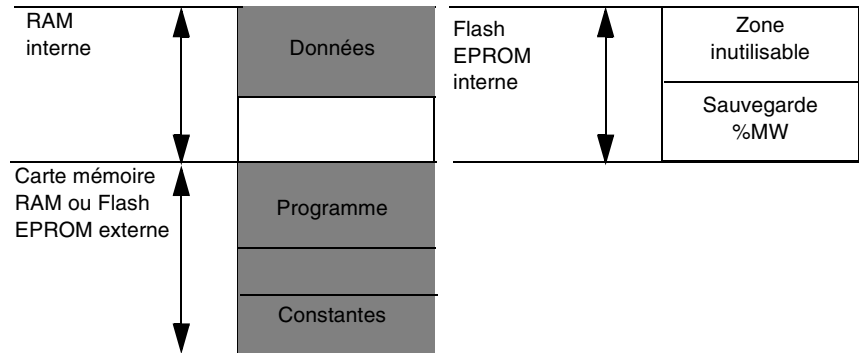
- du programme application (9 ou 15 K mots en fonction du processeur),
- de 1000 mots internes %MWi.



### Structure avec carte mémoire d'extension

Les données sont supportées par la mémoire RAM interne au module processeur. Les programme et constantes sont supportés par la carte mémoire d'extension.

Le schéma suivant décrit la structure mémoire.



Une mémoire de 10/16 K mots Flash EPROM (suivant le processeur) intégrée au module processeur peut être utilisée pour la sauvegarde de 1000 mots internes %MWi.

### Sauvegarde de la mémoire

Les mémoires RAM peuvent être secourues par pile Cadmium-nickel :

- supportée par le module processeur pour la mémoire bit et RAM interne,
- insérée dans carte pour la carte mémoire RAM.

La recopie de l'application dans la mémoire FLASH EPROM interne nécessite que l'automate ne possède pas de carte PCMCIA et que la taille de l'application soit inférieure ou égale à 9/15 K mots (suivant le processeur).

Le transfert d'application depuis la mémoire FLASH EPROM interne vers la mémoire RAM s'effectue automatiquement lorsqu'il y a perte de l'application en RAM (défaut de sauvegarde ou absence de batterie).

Un transfert manuel peut également être demandé, au travers d'un terminal de programmation.

## Structure mémoire des automates Premium

### Généralités

L'espace mémoire des automates Premium ne comporte qu'un seul ensemble. La mémoire bits est intégrée à la mémoire mots (dans la zone des données), elle est limitée à 4096 bits.

### Rôle de la mémoire mots

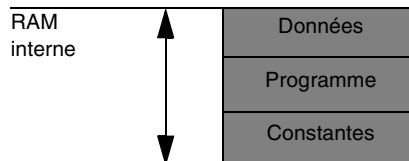
La mémoire mots (16 bits) supporte :

- **les données** : données dynamiques de l'application et données système (le système réserve une zone mémoire RAM de 5 Kmots minimum)
- **le programme** : descripteurs et code exécutable des tâches,
- **les constantes** : mots constants, valeurs initiales et configuration des entrées/sorties.

### Structure sans carte mémoire d'extension

Les programmes, données et constantes sont supportés par la mémoire RAM interne au module processeur.

Le schéma suivant décrit la structure mémoire.

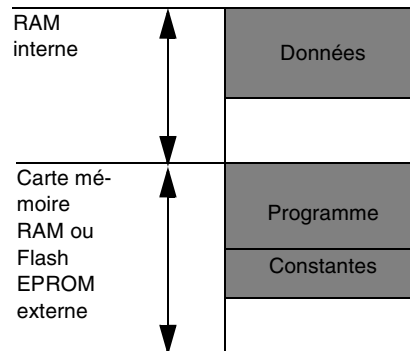


### Structure avec carte mémoire d'extension

Les données sont supportées par la mémoire RAM interne au module processeur.

Les programmes et constantes sont supportés par la carte mémoire d'extension.

Le schéma suivant décrit la structure mémoire.



### Sauvegarde de la mémoire

La mémoire bit et RAM interne est secourue par la pile Cadmium-nickel supportée par le module processeur.

La carte mémoire RAM interne est secourue par pile Cadmium-nickel.

## Description de la mémoire bits

### Généralités

**Pour les automates Micro :** cette mémoire contient 1280 objets bits quel que soit le type d' automate.

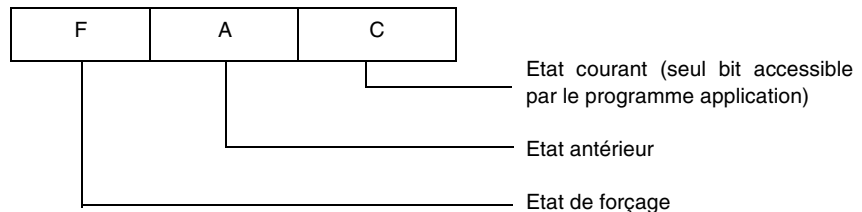
**Pour les automates Premium :** cette mémoire bits n'existe pas et son contenu se trouve dans la mémoire mots dans la zone des données de l'application.

Le codage des objets bits PL7 permet le test de front montant ou descendant sur :

- les bits d'entrées/sorties,
- les bits internes.

### Fonctionnement

Chaque objet bit contenu dans la mémoire bits est mémorisé à l'aide de 3 bits affectés de la façon suivante :



Lors de la mise à jour de la mémoire bits, le système assure :

Phase	Description
1	Le transfert de l'image de l'état courant dans l'état antérieur.
2	La ré-actualisation de l'état courant par le programme, le système ou le terminal (cas du forçage d'un bit).

### Etat de forçage

Sur une demande de forçage par le terminal :

- l'état de forçage F est mis à l'état 1
- l'état courant C est mis à l'état :
  - 1 si le forçage à 1 est demandé
  - 0 si le forçage à 0 est demandé

Ces états n'évoluent plus jusqu'à ce que :

- le forçage soit supprimé et le bit concerné mis à jour,
- le forçage inverse soit demandé, seul l'état courant est modifié.

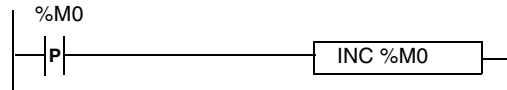
### Conseils d'utilisation des fronts montants ou descendants

Les instructions contacts front montant ou descendant ne fonctionnent correctement que si vous suivez les règles ci-après:

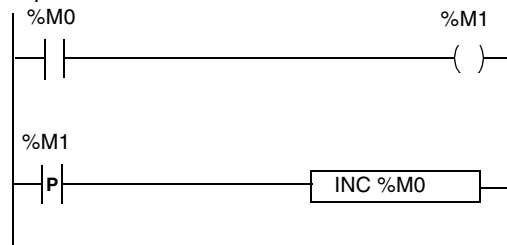
- dans tous les cas, traiter pour un même objet :
  - bit d'entrée : le contact de front dans la tâche où le module d'entrée est échangé,
  - bit de sortie interne : traiter sa lecture et son écriture à l'intérieur d'une même tâche.
- Tout objet bit testé sur front doit être écrit une fois et une seule en utilisant les bobines normale -( )- ou inverse -( / )- (et/ou équivalent en langage liste d'instructions). Ne pas utiliser les bobines -(S)- ou -(R)-. Quand une sortie est déclarée dans une liste d'échange d'un traitement événementiel, elle provoque l'échange du groupe de voies qui lui est associé, ce qui perturbe la gestion des fronts dans la tâche qui gère normalement ce groupe de voies.
- ne pas effectuer de SET ou de RESET d'un objet dont on teste le front car même si le résultat de l'équation conditionnant le SET/RESET vaut 0, l'action SET/RESET n'est pas effectuée mais l'historique de l'objet est mis à jour (perte du front).
- ne pas tester le front d'entrées/sorties utilisées dans une tâche événementielle, dans une tâche maître ou rapide
- pour les bits internes : la détection d'un front est indépendante du cycle de la tâche, un front sur bit interne %Mi est détecté lorsque son état a changé entre 2 lectures.

**Ce front reste détecté tant que ce bit interne n'est pas scruté en zone action.**

**Exemple :** Ainsi dans l'exemple ci-après, si vous forcez à 1 le bit %M0 dans une table d'animation, le front reste permanent.



Pour que le front ne soit détecté qu'une seule fois, il faut utiliser un bit interne intermédiaire. Dans ce cas l'historique de %M1 est mis à jour, donc le front n'est présent qu'une seule fois.



## Description de la mémoire mots

### Généralités

Cette mémoire mots de 16 bits est structurée en 3 espaces logiques :

- Données,
- Programme,
- Constantes.

dont la taille est définie par configuration.

**Note** : Les symboles et les commentaires associés aux objets ne sont pas enregistrés en mémoire automate mais stockés dans l'application locale (disque dur du terminal).

### Mémoire Données application

La mémoire Données comprend les différentes zones suivantes :

Type de mots	Description
Système	Nombre fixe
Blocs fonction	Correspond aux mots et entrées/sorties de ces blocs (valeurs courantes, de réglage...). Le nombre de chaque type de bloc fonction est fixé en configuration
Internes	Taille définie par le nombre déclaré en configuration.
Entrées/sorties	Correspond aux mots associés à chaque module. Leur nombre dépend des modules configurés.
Communs réseau	4 mots communs par station automate (disponibles uniquement si module de communication présent et configuré en échange de mots communs).

### Mémoire Programme application

Cette zone comprend le code programme exécutable, les informations graphiques (réseaux de contacts) et les commentaires programme.

**Note** : Quelque soit le type de langage utilisé ou la taille de la zone programme, le nombre d'éléments de programmation (réseaux de contact, phrases littérales...) est limité. Avant d'atteindre la limite, PL7 vous préviendra à travers une fenêtre d'avertissement.

### Mémoire constante application

Cette zone comprend les paramètres des blocs fonction et des modules d'entrées/sorties définis en configuration et les mots constants %KW.

## Caractéristiques de la mémoire des automates TSX 37

### Taille de la mémoire bits

Le tableau suivant décrit la répartition mémoire des objets bits.

Processeur TSX		37 05/08/10	37 21/22
Taille disponible sur processeur		1280	1280
Type d'objets	bits système %Si	128	128
	bits entrées/sorties %I/Qx.i	(1)	(1)
	bits internes %Mi	256	256
	bits d'étapes %Xi	96	128
Légende			
(1)	dépend de la configuration matérielle déclarée (modules d'entrées/sorties, équipements sur bus AS-i)		

### Taille de la mémoire mots

Le tableau suivant décrit la répartition mémoire des objets mots

Processeur TSX	3705/08	3710	3721			
Cartouche	-	-	-	32 Kmots	64 Kmots	128 Kmots
RAM interne	9 Kmots	14 Kmots	20 Kmots	52 Kmots	84 Kmots	148 Kmots
Données (%MWi)	0,5 Kmots(1)	0,5 Kmots (1)	0,5 Kmots (1)	17,5 Kmots	17,5 Kmots	17,5 Kmots
Programme 100% booléen						
• Langage LD	1,6 Ki	3,8 Ki	6,6 Ki	13,7 Ki	28,5 Ki	58,2 Ki
• Langage IL	2 Ki	4,9 Ki	8,4 Ki	17,5 Ki	36,3 Ki	74,2 Ki
• Langage ST	1,3 Ki	3,3 Ki	5,6 Ki	11,7 Ki	24,2 Ki	49,5 Ki
Programme 90% booléen						
• Langage LD	1,1 Ki	3,1 Ki	5,4 Ki	11,8 Ki	24,7 Ki	50,1 Ki
• Langage IL	1,4 Ki	3,8 Ki	6,6 Ki	14,3 Ki	30,0 Ki	61 Ki
• Langage ST	1,1 Ki	2,9 Ki	5,1 Ki	11,1 Ki	23,3 Ki	47,3 Ki
Programme 65% booléen						
• Langage LD	0,9 Ki	2,2 Ki	4,0 Ki	9,1 Ki	18,9 Ki	38,4 Ki
• Langage IL	1,0 Ki	2,5 Ki	4,6 Ki	10,3 Ki	21,3 Ki	43,4 Ki
• Langage ST	1,0 Ki	2,5 Ki	4,6 Ki	10,3 Ki	21,3 Ki	43,4 Ki

Processeur TSX	3705/08	3710	3721			
Constantes (1)	128 mots	128 mots	128 mots	256 mots	512 mots	512 mots
Légende						
(1)	Taille par défaut, peut être étendue au détriment de la taille programme application.					
Ki	Kinstructions (1024 instructions)					

Le tableau suivant décrit la répartition mémoire des objets mots

Processeur TSX	37 22			
Cartouche	-	32 Kmots	64 Kmots	128 Kmots
RAM interne	20 Kmots	52 Kmots	84 Kmots	148 Kmots
Données (%MWi)	0,5 Kmots (1)	17,5 Kmots	17,5 Kmots	17,5 Kmots
Programme 100% booléen				
• Langage LD	6,3 Ki	13,6 Ki	28,4 Ki	57,6 Ki
• Langage IL	8,1 Ki	17,3 Ki	36,1 Ki	73,5 Ki
• Langage ST	5,4 Ki	11,5 Ki	24,1 Ki	49 Ki
Programme 90% booléen				
• Langage LD	5,2 Ki	11,6 Ki	24,5 Ki	50 Ki
• Langage IL	6,3 Ki	14,2 Ki	29,8 Ki	60,8 Ki
• Langage ST	4,9 Ki	11,0 Ki	23,2 Ki	47,2Ki
Programme 65% booléen				
• Langage LD	3,9 Ki	8,9 Ki	18,8 Ki	38,4 Ki
• Langage IL	4,4 Ki	10,1 Ki	21,2 Ki	43,4 Ki
• Langage ST	4,4 Ki	10,1 Ki	21,2 Ki	43,4 Ki
Constantes (1)	128 mots	256 mots	512 mots	512 mots
Légende				
(1)	Taille par défaut, peut être étendue au détriment de la taille programme application.			
Ki	Kinstructions (1024 instructions)			

**Note :** la commande AP/Bilan mémoire du logiciel PL7 permet de connaître la répartition mémoire de l'application dans la mémoire automate.

---

## Caractéristiques de la mémoire des automates TSX/PCX 57 10/15/20/25/26/28

---

### Taille de la mémoire bits

Ce tableau décrit la répartition mémoire des objets mots des automates TSX 57 1\*\*, TSX 57 2\*\*, TSX 57 2\*23 et PCX 57 203.

Processeur		TSX 57 103/153 et PCX 57 203	TSX57 203/ 253/2623/2823
Type d'objets	bits système %Si	128	128
	bits entrées/sorties %I/Qx.i	(1)	(1)
	bits internes %Mi (nb maxi)	3962	8056
	bits d'étapes %Xi (nb maxi)	1024	1024
Légende			
(1)	dépend de la configuration matérielle déclarée (modules d'entrées/sorties, équipements sur bus AS-i)		

---



## Taille de la mémoire mots

Le tableau suivant décrit la répartition mémoire des objets mots des automates TSX 57 1••, TSX 57 2••, TSX 57 2•23 et PCX 57 203.

Processeur	TSX 57-103 - TSX 57 153			TSX-PCX 57 203/ 2623	TSX- 57 253/2823	TSX-PCX 57 203/ TSX 57 253/2623/ 2823	TSX-PCX 57 203/ TSX 57 253/2623/ 2823	TSX-PCX 57 203/ TSX 57 253/2623/ 2823
Cartouche	-	32K	64K	-	-	32K	64K	128K
RAM interne	32K	32K	32K	48K/64K	48K/64K	48K/64K	48K/64K	48K/64K
Données (%MWi)	0,5 K (1)	26 K	26 K	1K (1)	1K (1)	30,5K	30,5K	30,5K
Programme 100% booléen								
• Langage LD	8,8 Ki	12,3 Ki	26,9 Ki	15,5 Ki	22,8 Ki	12,3 Ki	26,6 Ki	56,5 Ki
• Langage IL	11,2 Ki	15,6 Ki	34,3 Ki	19,7 Ki	29,1 Ki	15,6 Ki	33,9 Ki	71,6 Ki
• Langage ST	7,6 Ki	10,5 Ki	22,9 Ki	13,1 Ki	19,4 Ki	10,4 Ki	22,6 Ki	47,8 Ki
Programme 90% booléen								
• Langage LD	5,2 Ki	8,6 Ki	21,4 Ki	11,0 Ki	17,4 Ki	8,6 Ki	21,1 Ki	46,9 Ki
• Langage IL	6,2 Ki	10,3 Ki	25,6 Ki	13,1 Ki	20,7 Ki	10,3 Ki	25,2 Ki	56,0 Ki
• Langage ST	5,0 Ki	8,3Ki	20,5 Ki	10,5 Ki	16,6 Ki	8,3 Ki	20,2 Ki	44,9 Ki
Programme 65% booléen								
• Langage LD	3,6 Ki	6,7 Ki	16,7 Ki	8,1 Ki	13,1 Ki	6,6 Ki	16,4 Ki	36,6 Ki
• Langage IL	3,7 Ki	6,8 Ki	17,0 Ki	8,3 Ki	13,4 Ki	6,8 Ki	16,8 Ki	37,5 Ki
• Langage ST	4,2 Ki	7,9 Ki	19,7 Ki	9,6 Ki	15,5 Ki	7,8 Ki	19,4 Ki	43,3 Ki
Constantes (1)	128 mots	256 mots	512 mots	256 mots	256 mots	256 mots	512 mots	512 mots
Légende								
(1)	Taille par défaut, peut être étendue au détriment de la taille programme application.							
Ki	Kinstructions							
K	Kmots							

### Note :

- quand ce tableau mentionne pour une caractéristique 2 valeurs séparées par un"/", elles sont associées respectivement à chaque type de processeur (séparés par un "/" dans l'entête du tableau).
- la commande AP/Bilan mémoire du logiciel PL7 permet de connaître la répartition mémoire de l'application dans la mémoire automate.

## Caractéristiques de la mémoire des automates TSX/PCX 57 30/35/36

### Taille de la mémoire bits

Ce tableau décrit la répartition mémoire des objets mots des automates TSX 57 3•3, TSX 57 3623 et PCX 57 353.

Processeur TSX/PCX		57 303/353/3623
Type d'objets	bits système %Si	128
	bits entrées/sorties %I/Qx.i	(1)
	bits internes %Mi (Nb maxi)	16250
	bits d'étapes %Xi (Nb maxi)	1024
Légende		
(1)	dépend de la configuration matérielle déclarée (modules d'entrées/sorties, équipements sur bus AS-i)	

### Taille de la mémoire mots

Le tableau suivant décrit la répartition mémoire des objets mots des automates TSX 57 3•3, TSX 57 3623 et PCX 57 353.

Processeur	TSX 57 303/3623	TSX/PCX57 353	TSX 57 303/3623 / TSX/PCX57 353	TSX 57 303/3623 / TSX/PCX57 353	TSX 57 303/3623 / TSX/PCX57 353	TSX 57 303/3623 / TSX/PCX57 353	TSX 57 303/3623 / TSX/PCX57 353
Cartouche	-	-	32K	64K	128K	256K	384K
RAM interne	64K/80K	64K/80K	80K/96K	80K/96K	80K/96K	80K/96K	80K/96K
Données (%MWi)	1K (1)	1K (1)	30,5K	30,5K	30,5K	30,5K	30,5K
Programme 100% booléen							
• Langage LD	28,8 Ki	30,1 Ki	12,3 Ki	26,6 Ki	56,2 Ki	115,3 Ki	150,5 Ki
• Langage IL	36,7 Ki	38,4 Ki	15,6 Ki	33,9 Ki	71,6 Ki	147,1 Ki	150,5 Ki
• Langage ST	24,5 Ki	25,6 Ki	10,4 Ki	22,6 Ki	47,8 Ki	98,0 Ki	148,3 Ki
Programme 90% booléen							
• Langage LD	22,6 Ki	23,8 Ki	8,6 Ki	21,1 Ki	46,9 Ki	98,4 Ki	149,9 Ki
• Langage IL	27,1 Ki	28,4 Ki	10,3 Ki	25,2 Ki	56,0 Ki	117,5 Ki	157,6 Ki
• Langage ST	21,7 Ki	22,7 Ki	8,3 Ki	20,2 Ki	44,9 Ki	94,2 Ki	142,9 Ki
Programme 65% booléen							
• Langage LD	17,4 Ki	18,2 Ki	6,6 Ki	16,4 Ki	36,6 Ki	77,0 Ki	117,4 Ki
• Langage IL	17,8 Ki	18,6 Ki	6,8 Ki	16,8 Ki	37,5 Ki	78,8 Ki	120,1 Ki
• Langage ST	20,5 Ki	21,5 Ki	7,8 Ki	19,4 Ki	43,3 Ki	91,1 Ki	138,8 Ki
Constantes (1)	256 K	256 K	256 K	1024 K	1024 K	1024 K	1024 K
Légende							

Processeur	TSX 57 303/3623	TSX/ PCX57 353	TSX 57 303/ 3623 / TSX/ PCX57 353	TSX 57 303/ 3623 / TSX/ PCX57 353	TSX 57 303/ 3623 / TSX/ PCX57 353	TSX 57 303/ 3623 / TSX/ PCX57 353	TSX 57 303/ 3623 / TSX/ PCX57 353
(1)	Taille par défaut, peut être étendue au détriment de la taille programme application.						
Ki	Kinstructions						
K	Kmots						

**Note :**

- quand ce tableau mentionne pour une caractéristique 2 valeurs séparées par un "/", elles sont associées respectivement à chaque type de processeur (séparés par un "/" dans l'entête du tableau).
- la commande AP/Bilan mémoire du logiciel PL7 permet de connaître la répartition mémoire de l'application dans la mémoire automate.

## Caractéristiques de la mémoire des automates TSX 57 453/4823

---

### Taille de la mémoire bits

Ce tableau décrit la répartition mémoire des objets mots des automates TSX 57 453/4823.

Processeur TSX		57 453/4823
Type d'objets	bits système %Si	128
	bits entrées/sorties %I/Qx.i	(1)
	bits internes %Mi (Nb maxi)	32634
	bits d'étapes %Xi (Nb maxi)	1024
Légende		
(1)	dépend de la configuration matérielle déclarée (modules d'entrées/sorties, équipements sur bus AS-i)	

---

**Taille de la mémoire mots**

Le tableau suivant décrit la répartition mémoire des objets mots des automates TSX 57 453/4823.

Processeur	TSX 57 453/4823							
Cartouche	-	32K	64K	128K	256K	384K (2)	512K (2)	2*480K (2)
RAM interne	96K	176K	176K	176K	176K	176K	176K	176K
Données (%MWi)	1K (1)	30,5K	30,5K	30,5K	30,5K	30,5K	30,5K	30,5K
Programme 100% booléen								
• Langage LD	37,5 Ki	12,3 Ki	26,6 Ki	56,2 Ki	115,3 Ki	150,5Ki	150,5 Ki	150,5 Ki
• Langage IL	47,8 Ki	15,6 Ki	33,9 Ki	71,6 Ki	147,1 Ki	150,5 Ki	150,5 Ki	150,5 Ki
• Langage ST	31,9 Ki	10,4 Ki	22,6 Ki	47,8 Ki	98,0 Ki	148,3 Ki	150,7 Ki	150,7 Ki
Programme 90% booléen								
• Langage LD	30,2 Ki	8,6 Ki	21,1 Ki	46,9 Ki	98,4 Ki	149,9 Ki	157,6 Ki	157,6 Ki
• Langage IL	36,0 Ki	10,3 Ki	25,2 Ki	56,0 Ki	117,5 Ki	157,6 Ki	157,6 Ki	157,6 Ki
• Langage ST	28,9 Ki	8,3 Ki	20,2 Ki	44,9 Ki	94,2 Ki	142,9 Ki	157,8 Ki	157,8 Ki
Programme 65% booléen								
• Langage LD	23,2 Ki	6,6 Ki	16,4 Ki	36,6 Ki	77,0 Ki	117,4 Ki	157,8 Ki	157,8 Ki
• Langage IL	23,7 Ki	6,8 Ki	16,8 Ki	37,5 Ki	78,8 Ki	120,1 Ki	161,3 Ki	161,3 Ki
• Langage ST	27,4 Ki	7,8 Ki	19,4 Ki	43,3 Ki	91,1 Ki	138,8 Ki	171,3 Ki	171,3 Ki
Constantes (1)	256 mots	256 mots	1024 mots	1024 mots	1024 mots	1024 mots	1024 mots	1024 mots
Légende								
(1)	Taille par défaut, peut être étendue au détriment de la taille programme application.							
(2)	Le nombre d'elements de programmation est atteint							
Ki	Kinstructions							
K	Kmots							

**Note :**

- quand ce tableau mentionne pour une caractéristique 2 valeurs séparées par un"/", elles sont associées respectivement à chaque type de processeur (séparés par un "/" dans l'entête du tableau).
- la commande AP/Bilan mémoire du logiciel PL7 permet de connaître la répartition mémoire de l'application dans la mémoire automate.



---

# Modes de marche



---

## Présentation

### Objet de ce chapitre

Ce chapitre traite du comportement du programme utilisateur sur Reprise à chaud et démarrage à froid.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

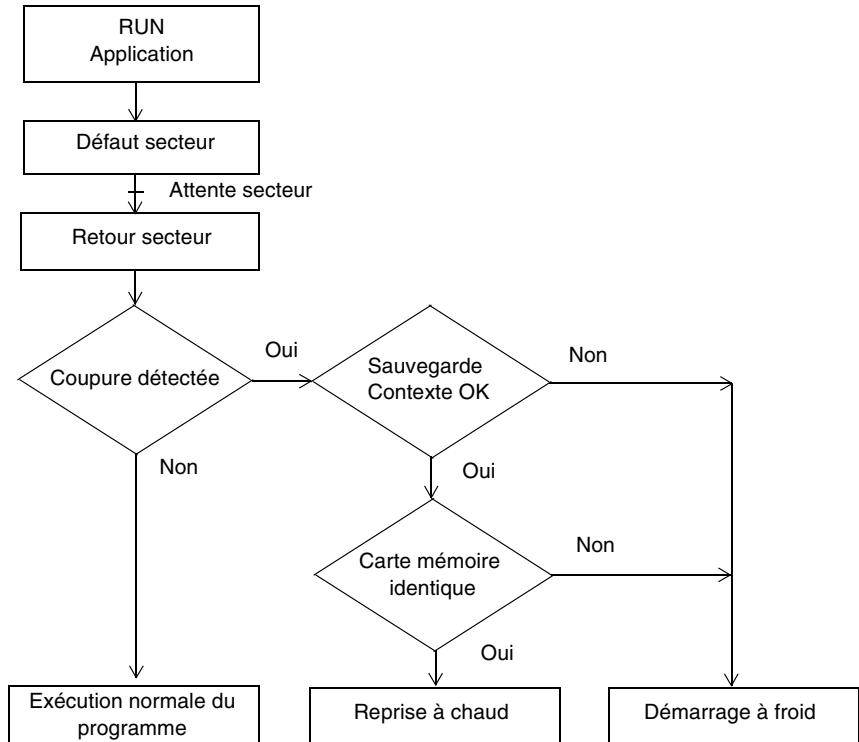
Sujet	Page
Traitement sur coupure et reprise secteur	72
Traitement sur reprise à chaud	74
Gestion du démarrage à froid	76

---

## Traitement sur coupure et reprise secteur

### Illustration

L'illustration présente les différentes reprises secteurs détectées par le système. Si la durée de la coupure est inférieure au temps de filtrage de l'alimentation (environ 10 ms pour les alimentations alternatives ou à 1 ms pour les alimentations continues), celle-ci n'est pas vue par le programme qui s'exécute normalement.





**Fonctionnement** Le tableau ci-après décrit les phases du traitement des coupures secteur.

Phase	Description
1	Lors de la coupure secteur le système mémorise le contexte application et l'heure de la coupure.
2	Il positionne toutes les sorties à l'état repli (état défini par configuration).
3	A la reprise secteur, le contexte sauvegardé est comparé à celui en cours; ce qui définit le type de démarrage à exécuter : <ul style="list-style-type: none"> <li>● si le contexte application a changé (perte du contexte système ou nouvelle application), l'automate effectue l'initialisation de l'application : démarrage à froid,</li> <li>● si le contexte application est identique, l'automate effectue une reprise sans initialisation des données : reprise à chaud.</li> </ul>

**Coupure de l'alimentation sur un rack, autre que le rack 0**

Toutes les voies de ce rack sont vues en erreur par le processeur mais les autres racks ne sont pas perturbés, les valeurs des entrées en erreur ne sont plus rafraîchies dans la mémoire application et sont mises à 0 dans le cas d'un module d'entrée TOR à moins qu'elles aient été forcées auquel cas, elles sont maintenues à la valeur de forçage.

Si la durée de la coupure est inférieure à 10 ms pour les alimentations alternatives ou à 1 ms pour les alimentations continues, celle-ci n'est pas vue par le programme qui s'exécute normalement.

## Traitement sur reprise à chaud

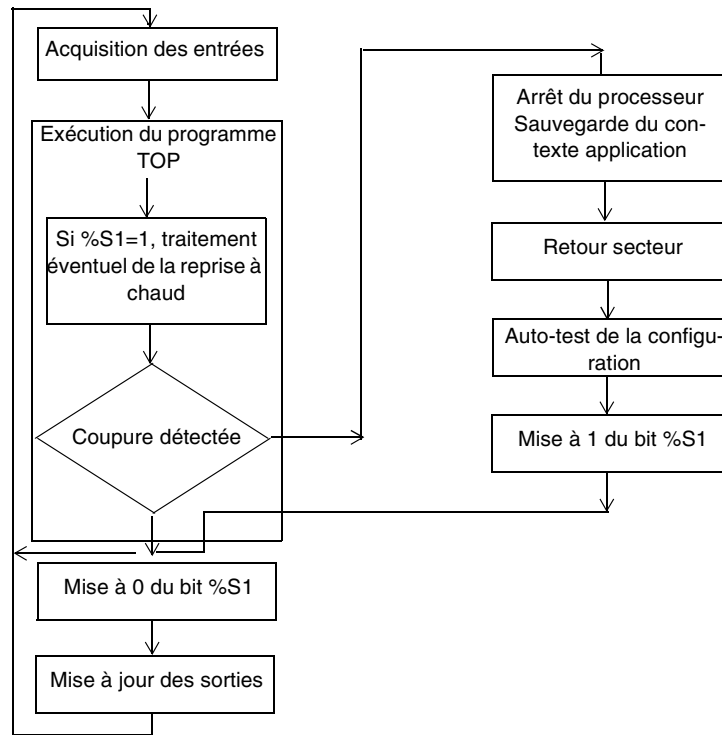
### Cause d'une reprise à chaud

Une reprise à chaud peut être provoquée :

- par une reprise secteur sans perte du contexte,
- par mise à 1 par programme du bit système %S1,
- depuis PL7 par terminal,
- par action sur bouton RESET du module alimentation du rack 0 (excepté sur station équipée d'un processeur PCX 57).

### Illustration

Le dessin ci-après décrit le fonctionnement d'une reprise à chaud.



**Fonctionnement** Le tableau ci-après décrit les phases de reprise de l'exécution du programme sur reprise à chaud.

Phase	Description
1	L'exécution du programme reprend à partir de l'élément où a eu lieu la coupure secteur, sans mise à jour des sorties.
2	A la fin du cycle de reprise, le système effectue : <ul style="list-style-type: none"> <li>● l'initialisation des files de messages et d'événements,</li> <li>● l'envoi des paramètres de configuration à tous les modules d'entrées/sorties TOR et métiers,</li> <li>● la désactivation de la tâche rapide et des traitements événementiels (jusqu'à la fin du premier cycle de la tâche maître).</li> </ul>
3	Le système effectue un cycle de reprise dans lequel il : <ul style="list-style-type: none"> <li>● reprend en compte l'ensemble des modules d'entrées,</li> <li>● relance la tâche maître avec le bit %S1 (reprise à chaud) positionné à 1,</li> <li>● remet à l'état 0 le bit %S1 à la fin de ce premier cycle de la tâche maître,</li> <li>● réactive la tâche rapide et les traitements événementiels à la fin de ce premier cycle de la tâche maître.</li> </ul>

**Traitement par programme de la reprise à chaud**

En cas de reprise à chaud, si vous désirez un traitement particulier vis-à-vis de l'application, vous devez écrire le programme correspondant sur test de %S1 à 1 en début de programme de la tâche maître.

**Evolution des sorties**

Dès la détection de la coupure secteur, les sorties sont mises en position de repli :

- soit elles prennent la valeur de repli,
- soit il y a maintien de la valeur en cours, suivant le choix effectué en configuration.

A la reprise secteur, les sorties sont à zéro jusqu'à ce qu'elles soient remises à jour par la tâche.

## Gestion du démarrage à froid

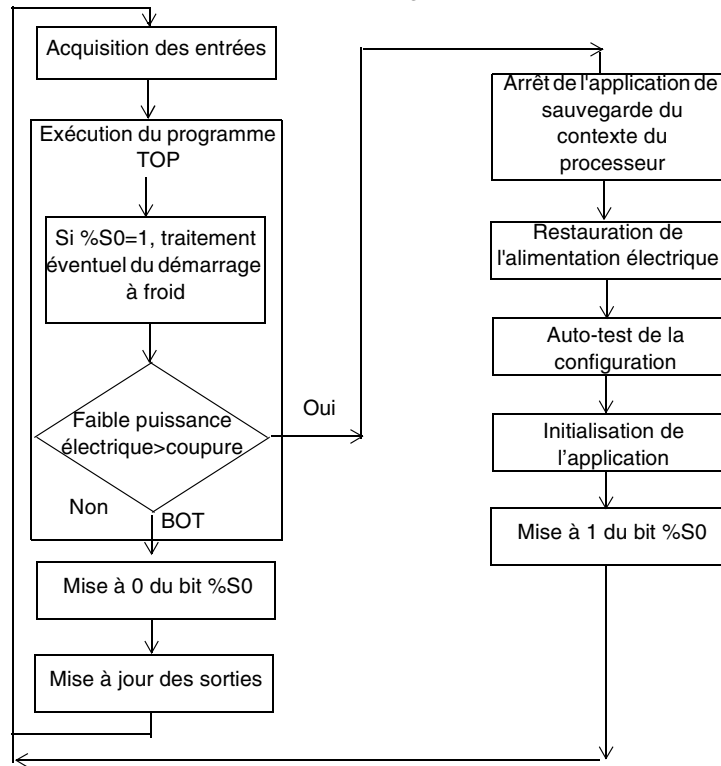
### Cause d'un démarrage à froid

Le tableau suivant décrit les différentes causes possibles d'un démarrage à froid.

Causes	Caractéristiques du démarrage
Chargement d'une application	Démarrage à froid forcé à STOP
Action sur le bouton RESET du processeur	Démarrage à froid forcé à STOP ou à RUN selon la définition de la configuration
Action sur le bouton RESET du processeur suite à un défaut bloquant	Démarrage à froid forcé à STOP
Manipulation du préhenseur ou insertion/extraction d'une carte mémoire PCMCIA	Démarrage à froid forcé à STOP ou à RUN selon la définition de la configuration
Initialisation depuis un PL7 Junior ou Pro Forçage du bit système %S0	Démarrage à froid forcé à STOP ou à RUN selon la définition de la configuration, sans initialisation des modules d'entrées/sorties TOR et métier
Redémarrage après une coupure électrique avec perte de contexte	Démarrage à froid forcé à STOP ou à RUN selon la définition de la configuration

**Illustration**

Le dessin ci-après décrit un redémarrage à froid.

**Opération**

Le tableau ci-après décrit les phases de redémarrage pour exécuter un programme après redémarrage à froid.

Phase	Description
1	Le démarrage s'effectue en RUN ou en STOP suivant l'état du paramètre Démarrage automatique en RUN défini dans la configuration ou si l'utilisation dépend de de l'état de l'entrée RUN/STOP. Le programme reprend en début de cycle.

Phase	Description
2	<p>Le système effectue :</p> <ul style="list-style-type: none"> <li>● la réinitialisation des bits, la remise à zéro de l'affectation des E/S et des mots internes (si l'option RAZ de %MW sur redémarrage à froid est cochée dans l'écran Configuration du processeur) ;</li> </ul> <p>Si la réinitialisation de %MW n'est pas active et si des mots internes %MWi sont enregistrés dans la mémoire interne Flash EPROM (TSX 37), ceux-ci sont restaurés lors d'un démarrage à froid ;</p> <ul style="list-style-type: none"> <li>● l'initialisation des bits et des mots système ;</li> <li>● l'initialisation des blocs fonction à partir des données de configuration ;</li> <li>● la désactivation des tâches, autres que la tâche maître, jusqu'à la fin du premier cycle de la tâche maître ;</li> <li>● le réglage de Grafcet sur les étapes initiales ;</li> <li>● l'annulation des forçages ;</li> <li>● l'initialisation des données déclarées dans les DFB : soit à 0, soit à la valeur initiale déclarée dans le code, c'est-à-dire avec la valeur enregistrée via la fonction SAVE ;</li> <li>● l'initialisation des fichiers de messages et d'événements ;</li> <li>● l'envoi des paramètres de configuration à tous les modules d'entrées/sorties TOR et métier.</li> </ul>
3	<p>Pour ce premier cycle de redémarrage, le système effectue :</p> <ul style="list-style-type: none"> <li>● la relance de la tâche maître avec le bit %S0 (redémarrage à froid) défini à 1, le mot %SW10 (détection d'un redémarrage à froid lors de la première révolution d'une tâche) étant réglé à 0 ;</li> <li>● la remise à zéro du bit %S0 et la remise à 1 de chaque bit de mot %SW10 à la fin de ce premier cycle de la tâche maître ;</li> <li>● active la tâche rapide et les traitements événementiels à la fin de ce premier cycle de la tâche maître.</li> </ul>

**Gestion du démarrage à froid pour chaque programme**

Pour effectuer un traitement applicatif après un démarrage à froid de l'automate PL7, il est possible de tester par programme le bit %SW10:X0 (si %SW10:X0=0, il s'agit d'un redémarrage à froid) .

**Evolution des sorties**

Dès la détection de la coupure électrique, les sorties sont mises en position de repli :

- soit elles prennent la valeur de repli,
- soit elles conservent la valeur courante,

suivant le choix effectué lors de la configuration.

Lorsque l'alimentation électrique est de nouveau opérationnelle, les sorties sont réglées à zéro jusqu'à ce qu'elles soient remises à jour par la tâche.

---

# Structure logicielle



# 5

---

## Présentation

### Objet de ce chapitre

Ce chapitre décrit les tâches et leur exécution dans l'automate.

### Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre	Sujet	Page
5.1	Description des tâches	80
5.2	Structure monotâche	88
5.3	Structure multitâche	96
5.4	Modules fonctionnels	103

---

## 5.1 Description des tâches

---

### Présentation

---

**Objet de ce sous-chapitre** Ce sous-chapitre décrit la rôle et le contenu de chacune des tâches pouvant constituer un programme PL7.

---

**Contenu de ce sous-chapitre** Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Présentation de la tâche maître	81
Description des sections et des sous-programmes	82
Présentation de la tâche rapide	85
Présentation des traitements événementiels	86

---



---

## Présentation de la tâche maître

---

### Généralités

La tâche maître représente le programme principal, elle est obligatoire quel que soit la structure adoptée monotâche ou multitâche.

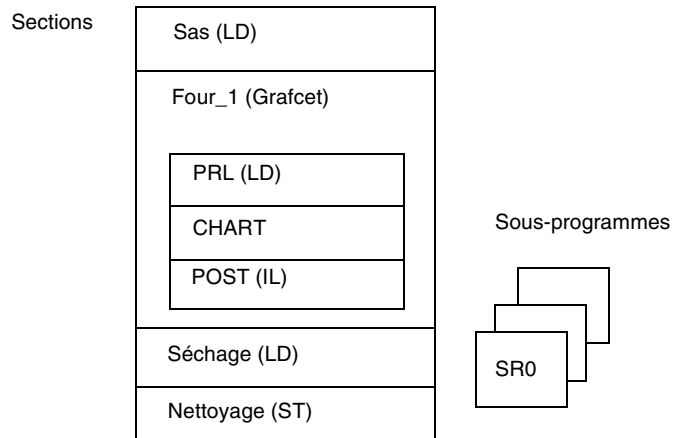
Le programme de la tâche maître (MAST) est constitué de plusieurs modules de programmes appelés sections (Voir *Description des sections et des sous-programmes*, p. 82), et de sous-programmes.

L'exécution de la tâche maître peut être choisie (en configuration) cyclique (Voir *Exécution cyclique*, p. 90) ou périodique (Voir *Exécution périodique*, p. 92).

---

### Illustration

L'illustration suivante montre un exemple de tâche maître comportant 4 sections et 3 sous-programmes.



## Description des sections et des sous-programmes

---

### Présentation des sections

Les sections sont des entités autonomes de programmation. Les étiquettes de repérage des lignes d'instructions, des réseaux de contacts ... sont propres à la section (pas de saut de programme possible vers une autre section).

Elles se programment soit en :

- langage à contacts,
- liste d'instructions,
- littéral structuré,
- Grafcet.

Les sections sont exécutées dans leur ordre de programmation dans la fenêtre du navigateur (vue structurelle).

Les sections sont liées à une tâche, une même section ne peut pas appartenir simultanément à plusieurs tâches.

---

### Présentation des sous-programmes

Les modules sous-programmes se programment aussi comme des entités séparées soit en :

- langage à contacts,
- liste d'instructions,
- littéral structuré,

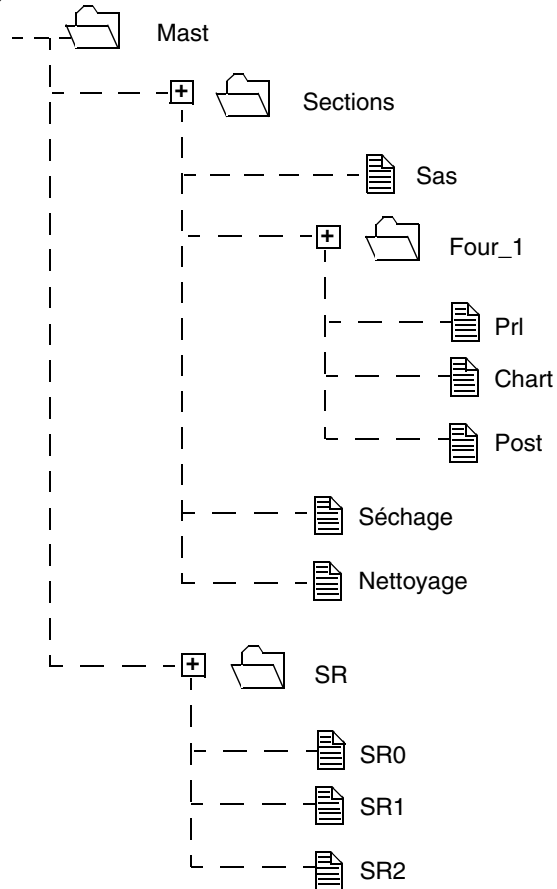
Les appels aux sous-programmes s'effectuent dans les sections ou depuis un autre-sous-programme (8 niveaux d'imbrications maximum).

Les sous-programmes sont aussi liés à une tâche, un même sous-programme ne peut pas être appelé depuis plusieurs tâches.

---

**Exemple**

Le dessin suivant donne l'exemple de structure d'une tâche en sections et sous-programmes.

**Caractéristiques d'une section**

Le tableau suivant décrit les caractéristiques d'une section.

Caractéristique	Description
Nom	24 caractères maximum
Langage	Langage à contacts, Liste d'instructions, Littéral structuré ou Grafcet
Tâche	Maître ou rapide

Caractéristique	Description
Condition (optionnel)	Objets autorisés comme condition : <ul style="list-style-type: none"> <li>● %M,%S,%X</li> <li>● bit indexé,bits extrait de mots</li> <li>● %I , %Q</li> </ul> Tous ces objets sont forçables depuis le terminal excepté les bits %S, bits indexés ,bits extrait , %Ixy.i.ERR,et %I xy.MOD.ERR.  La condition doit être à l'état 1 pour que la section soit exécutée.
Commentaire	250 caractères maximum.
Protection	Protection en écriture, protection en lecture/écriture. La protection peut être globale ou partielle.

**Note :** sur démarrage à froid les conditions d'exécution sont à 0, toutes les sections auxquelles sont associées une condition sont inhibées.

### Section Grafcet

Le tableau suivant décrit les éléments de programme d'une section Grafcet.

Traitement	Désignation	Caractéristiques
Préliminaire	PRL	Programmé en Langage à contacts LD, Liste d'instructions IL ou Littéral structuré ST. Il est exécuté avant le Grafcet.
Grafcet	CHART	Dans les pages Grafcet, sont programmées des réceptivités associées aux transitions et des actions associées aux étapes ou aux étapes de macro-étapes.
Postérieur	POST	Programmé en Langage à contacts LD, Liste d'instructions IL ou Littéral structuré ST. Il est exécuté après le Grafcet.

### Caractéristiques d'un sous-programme

Le tableau suivant décrit les caractéristiques d'un sous-programme SRi.

Caractéristique	Description
Numéro	0 à 253
Langage	Langage à contacts, Liste d'instructions, Littéral structuré.
Tâche	Maître ou rapide
Commentaire	250 caractères maximum.

---

## Présentation de la tâche rapide

---

### Généralités

Cette tâche plus prioritaire que la tâche maître MAST est périodique afin de laisser le temps à la tâche moins prioritaire de s'exécuter.

De plus, les traitements qui lui sont associés doivent donc être courts pour ne pas pénaliser la tâche maître. Comme pour la tâche maître, le programme associé se compose de sections et de sous-programmes.

### Période de la tâche rapide

La période de la tâche rapide FAST est fixée en configuration, de 1 à 255 ms. Celle-ci peut être définie supérieure à celle de la tâche maître MAST pour s'adapter à des traitements périodiques lents mais prioritaires.

Le programme exécuté doit cependant rester court pour éviter le débordement des tâches moins prioritaires.

La tâche rapide est contrôlée par un chien de garde qui permet de détecter une durée anormale du programme application. En cas de débordement, le bit système %S11 est positionné à 1 et l'application est déclarée en défaut bloquant pour l'automate.

### Contrôle de la tâche rapide

Le mot système %SW1 contient la valeur de la période, il est initialisé sur reprise à froid par la valeur définie en configuration, il peut être modifié par l'utilisateur par programme ou terminal.

Des bits et mots système, permettent de contrôler l'exécution de cette tâche :

- %S19 : signale un débordement de période, il est positionné à 1 par le système, lorsque le temps de cycle devient supérieur à la période de la tâche.
- %S31 : permet de valider ou d'inhiber la tâche rapide, il est mis à 0 par le système sur démarrage à froid de l'application, à la fin du premier cycle de la tâche maître. Il est mis à 1 ou à 0 pour valider ou inhiber la tâche rapide.

### Visualisation des temps d'exécution de la tâche rapide

Les mots système suivants permettent d'avoir des informations sur le temps de cycle :

- %SW33 contient le temps d'exécution du dernier cycle,
  - %SW34 contient le temps d'exécution du cycle le plus long,
  - %SW35 contient le temps d'exécution du cycle le plus court.
-

## Présentation des traitements événementiels

---

### Généralités

Les traitements événementiels permettent de réduire le temps de réaction du logiciel sur des événements de commande en provenance de certains modules métiers.

Ces traitements sont exécutés en priorité sur toutes les autres tâches. Elles conviennent donc aux traitements demandant des délais de réactions très courts par rapport à l'arrivée de l'événement.

Le nombre de traitements événementiels programmables dépend du type de processeur.

Type d'automate	Nombre de traitements	Désignation
Micro TSX 37-05/08/10	8	EVT1 à EVT8
Micro TSX 37-21/22	16	EVT0 à EVT15
Premium TSX/PCX 57-1•	32	EVT0 à EVT31
Premium TSX/PCX 57-2•/3•/4•	64	EVT0 à EVT63

### Fonctionnement

L'apparition d'un événement dérouté le programme application vers le traitement qui est associé à la voie d'entrées/sorties qui a provoqué l'événement.

Les entrées (%I, %IW, %ID) associées à la voie d'E/S qui a déclenché l'événement sont mises à jour par le système avant l'appel du traitement événementiel.

L'association entre une voie et un numéro d'événement est réalisée dans l'écran de configuration des voies.

### Evénements de commande

Ce sont des événements externes liés aux fonctions métiers.

Sur automates Micro, les traitements événementiels peuvent être déclenchés par :

- les entrées 0 à 3 du module de position 1, sur front montant ou descendant,
- la ou les voies de comptage des modules de comptage,
- les voies de comptage du module 1 (si celui-ci est configuré en compteur),
- la réception de télégrammes dans un TSX 37-21/22 équipé d'un module TSX FPP20.

---

Sur automates Premium , les traitements événementiels peuvent être déclenchés par:

- les entrées des modules DEY 16 FK, DMY 28 FK, DMY 28 RFK
- les voies des modules de comptage,
- les voies des modules de commande d'axe TSX CAY •,
- les voies des modules de commande pas à pas TSX CFY •,
- les voies de communication "FPP20",
- ...

---

### **Gestion des traitements événementiels**

Les traitements événementiels peuvent être globalement validés ou inhibés par le programme application, au travers du bit système %S38.

Si un ou plusieurs événements interviennent pendant qu'ils sont inhibés, les traitements associés sont perdus.

Deux instructions du langage PL7, `MASKEVT()` et `UNMASKEVT()`, utilisées dans le programme application, permettent également de masquer ou démasquer les traitements événementiels.

Si un ou plusieurs événements interviennent pendant qu'ils sont masqués, ils sont mémorisés par le système et les traitements associés ne seront effectués qu'après démasquage.

---

### **Priorité des traitements**

#### **Automates Micro TSX 37-05/08/10**

Les 8 événements de commande possibles ont tous le même niveau de priorité; de ce fait, un traitement événementiel n'est pas interruptible par un autre.

#### **Automates Micro TSX 37-21/22 ou Premium**

Il existe 2 niveaux de priorité pour les événements de commande : l'événement 0 (EVT0) est plus prioritaire que les autres événements.

---

## 5.2 Structure monotâche

---

### Présentation

---

#### Objet de ce sous-chapitre

Ce sous-chapitre décrit comment s'exécute une application monotâche.

---

#### Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Structure logicielle monotâche	89
Exécution cyclique	90
Exécution périodique	92
Contrôle du temps de cycle	95

---



## Structure logicielle monotâche

---

### Description

Le programme d'une application monotâche est associé à une seule tâche utilisateur : la tâche maître MAST (voir *Présentation de la tâche maître*, p. 81).

Le programme associé à la tâche maître (MAST) est constitué de plusieurs sections, et de sous-programmes.

L'exécution de la tâche maître peut être choisie (en configuration) :

- cyclique (Voir *Exécution cyclique*, p. 90)
  - ou périodique (Voir *Exécution périodique*, p. 92)
-

## Exécution cyclique

### Description

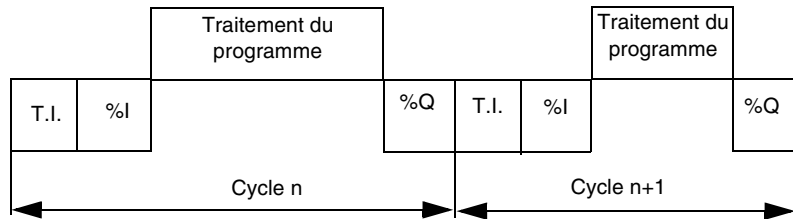
Ce type de fonctionnement correspond à l'exécution normale du cycle automate (choix par défaut).

Il consiste à enchaîner les uns après les autres, les cycles de la tâche maître (MAST).

Après avoir effectué la mise à jour des sorties, le système réalise les traitements qui lui sont propres puis enchaîne un autre cycle de la tâche.

### Fonctionnement

Le dessin suivant montre les phases d'exécution du cycle automate.



### Description des différentes phases

Le tableau ci-après décrit les phases de fonctionnement.

Repère	Phase	Description
T.I.	Traitement interne	Le système réalise implicitement la surveillance de l'automate (gestion des bits et mots système, mise à jour des valeurs courantes de l'horodateur, mise à jour des voyants d'état, détection des passages RUN/STOP, ...) et le traitement des requêtes en provenance du terminal (modifications et animation). Dans le cas du Premium, le traitement interne est réalisé en parallèle avec les traitements des entrées et des sorties.
%I	Acquisition des entrées	Ecriture en mémoire de l'état des informations présentes sur les entrées des modules TOR et métier associées à la tâche,
-	Traitement du programme	Exécution du programme application, écrit par l'utilisateur,
%Q	Mise à jour des sorties	Ecriture des bits ou des mots de sorties associés aux modules TOR et métier associés à la tâche selon l'état défini par le programme application.

**Mode de fonctionnement**

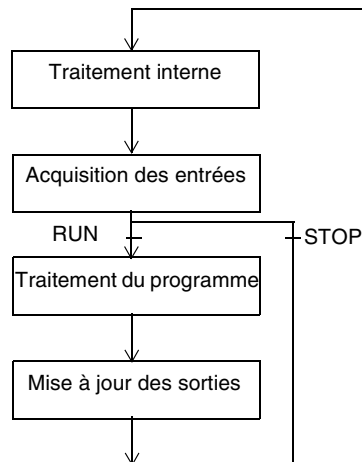
**Automate en RUN**, le processeur effectue dans l'ordre le traitement interne, l'acquisition des entrées, le traitement du programme application et la mise à jour des sorties.

**Automate en STOP**, le processeur effectue :

- le traitement interne,
- l'acquisition des entrées,
- et suivant la configuration choisie :
  - mode repli : les sorties sont mis en position de "repli" ,
  - mode maintien : les sorties sont maintenues à leur dernière valeur.

**Illustration**

L'illustration suivante montre les cycles de fonctionnement.

**Contrôle du cycle**

Le contrôle du cycle s'effectue par chien de garde (Voir *Contrôle du temps de cycle*, p. 95).

## Exécution périodique

### Description

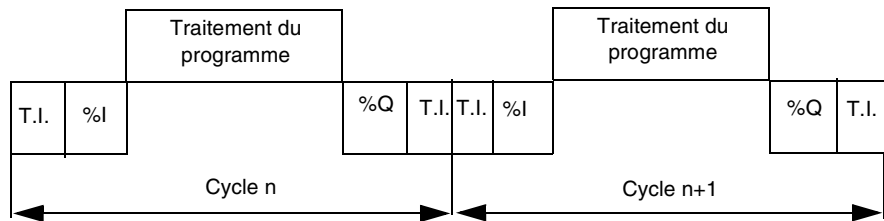
Dans ce mode de fonctionnement, l'acquisition des entrées, le traitement du programme application et la mise à jour des sorties s'effectuent de façon périodique selon un temps défini en configuration (de 1 à 255 ms).

En début de cycle automate, un temporisateur dont la valeur courante est initialisée à la période définie en configuration, commence à décompter.

Le cycle automate doit se terminer avant l'expiration de ce temporisateur qui relance un nouveau cycle.

### Fonctionnement

Le dessin suivant montre les phases d'exécution du cycle automate.



### Description des différentes phases

Le tableau ci-après décrit les phases de fonctionnement.

Repère	Phase	Description
T.I	Traitement interne	Le système réalise implicitement la surveillance de l'automate (gestion des bits et mots système, mise à jour des valeurs courantes de l'horodateur, mise à jour des voyants d'état, détection des passages RUN/STOP, ...) et le traitement des requêtes en provenance du terminal (modifications et animation) Dans le cas du Premium, le traitement interne est réalisé en parallèle avec les traitements des entrées et des sorties.
%I	Acquisition des entrées	Ecriture en mémoire de l'état des informations présentes sur les entrées des modules TOR et métier associés à la tâche,
-	Traitement du programme	Exécution du programme application, écrit par l'utilisateur,
%Q	Mise à jour des sorties	Ecriture des bits ou des mots de sorties associés aux modules TOR et métier associés à la tâche selon l'état défini par le programme application.

**Mode de fonctionnement**

**Automate en RUN**, le processeur effectue dans l'ordre le traitement interne, l'acquisition des entrées, le traitement du programme application et la mise à jour des sorties.

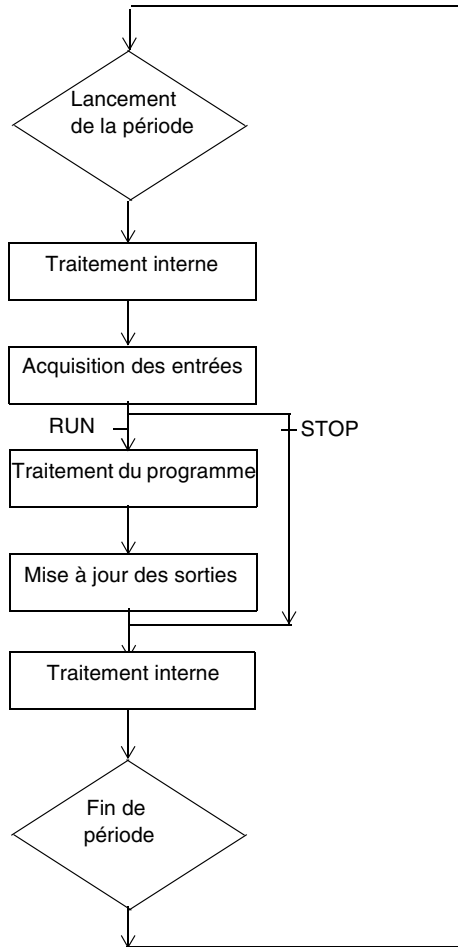
- Si la période n'est pas encore terminée, le processeur complète son cycle de fonctionnement jusqu'à la fin de la période par du traitement interne.
- Si le temps de fonctionnement devient supérieur à celui affecté à la période, l'automate signale un débordement de période par la mise à l'état 1 du bit système %S19 de la tâche, le traitement se poursuit et est exécuté dans sa totalité (il ne doit pas dépasser néanmoins le temps limite du chien de garde). Le cycle suivant est enchaîné après l'écriture implicite des sorties du cycle en cours.

**Automate en STOP**, le processeur effectue :

- le traitement interne,
  - l'acquisition des entrées,
  - et suivant la configuration choisie :
    - mode repli : les sorties sont mis en position de "repli" ,
    - mode maintien : les sorties sont maintenues à leur dernière valeur.
-

**Illustration**

L'illustration suivante montre les cycles de fonctionnement.



**Contrôle du cycle**

Deux contrôles sont effectués :

- débordement de période (Voir *Contrôle du temps de cycle*, p. 95),
  - par chien de garde (Voir *Contrôle du temps de cycle*, p. 95),
-

---

## Contrôle du temps de cycle

---

### Généralités

La durée d'exécution de la tâche maître, en fonctionnement cyclique ou périodique, est contrôlée par l'automate (chien de garde) et ne doit pas dépasser la valeur définie en configuration Tmax (250 ms par défaut, 500 ms maximum).

---

### Chien de garde logiciel (fonctionnement périodique ou cyclique)

Dans le cas de débordement, l'application est déclarée en défaut, ce qui provoque l'arrêt immédiat de l'automate.

- **sur le Micro**, la mise à 0 de la sortie alarme %Q2.0 si elle a été configurée,
- **sur le Premium**, la mise à 0 du relais alarme sur l'alimentation

Le bit %S11 permet de contrôler l'exécution de cette tâche. Il signale un débordement du chien de garde, il est positionné à 1 par le système, lorsque le temps de cycle devient supérieur au chien de garde.

<p><b>Note :</b> Sur le Premium, la valeur du chien de garde doit être supérieure à la période.</p>
---

---

### Contrôle en fonctionnement périodique

En fonctionnement périodique, un contrôle supplémentaire permet de détecter un dépassement de période :

- %S19 : signale un débordement de période, il est positionné à 1 par le système, lorsque le temps de cycle devient supérieur à la période de la tâche.
  - %SW0 : ce mot contient la valeur de la période (en ms), il est initialisé sur reprise à froid par la valeur définie en configuration, il peut être modifié par l'utilisateur.
- 

### Exploitation des temps d'exécution de la tâche maître

Les mots système suivants permettent d'avoir des informations sur le temps de cycle :

- %SW30 contient le temps d'exécution du dernier cycle.
- %SW31 contient le temps d'exécution du cycle le plus long,
- %SW32 contient le temps d'exécution du cycle le plus court.

<p><b>Note :</b> Ces différentes informations sont accessibles aussi depuis l'éditeur de configuration de façon explicite.</p>
--

---

## 5.3 Structure multitâche

---

### Présentation

---

#### Objet de ce sous-chapitre

Ce sous-chapitre décrit comment s'exécute une application multitâche.

---

#### Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Structure logicielle multitâche	97
Séquencement des tâches dans une structure multitâche	99
Affectation des voies d'entrées/sorties aux tâches maître et rapide	100
Echanges d'entrées/sorties dans les traitements événementiels	101

---



## Structure logicielle multitâche

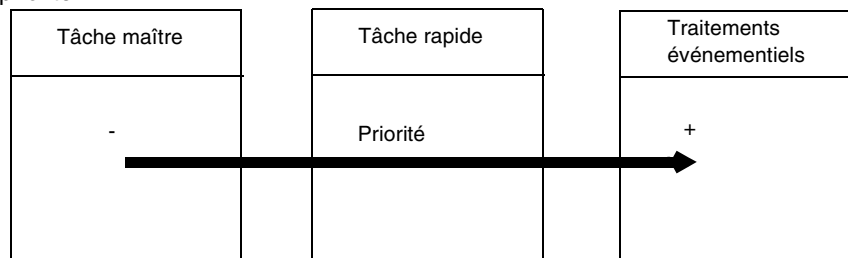
### Description

La structure en tâches d'une telle application est la suivante :

Tâche	Désignation	Description
Maître	MAST	Toujours présente qui peut être cyclique ou périodique.
Rapide	FAST	Optionnelle qui est toujours périodique.
Événementielle	EVTi	Appelés par le système lors de l'apparition d'un événement sur un coupleur d'entrées/sorties. Ces traitements sont optionnels et servent aux applications nécessitant des temps de réponse courts pour agir sur les entrées/sorties.

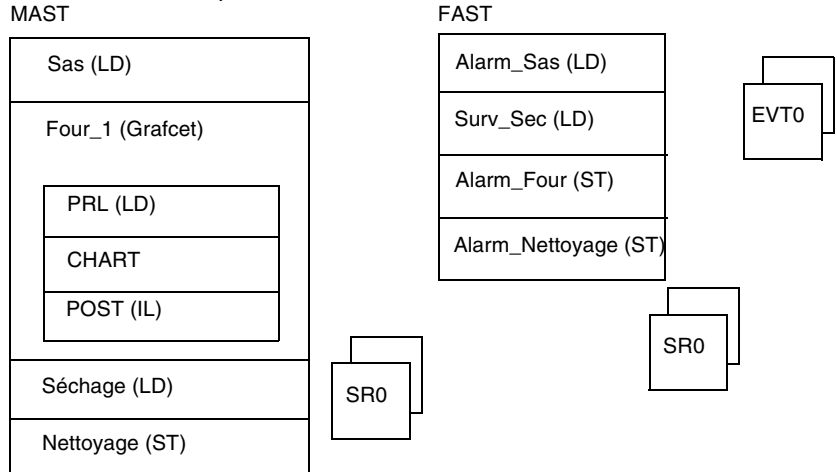
### Illustration

Le dessin suivant montre les tâches d'une structure multitâche et leur niveau de priorité.



**Exemple**

L'exemple suivant présente une structure multitâche composée d'une tâche maître MAST, d'une tâche rapide FAST et de 2 traitements événementiels EVT0 et EVT1.



## Séquencement des tâches dans une structure multitâche

### Généralités

La tâche maître est par défaut active.

La tâche rapide est par défaut active si elle est programmée.

Le traitement événementiel est activé lors d'apparition de l'événement qui lui a été associé.

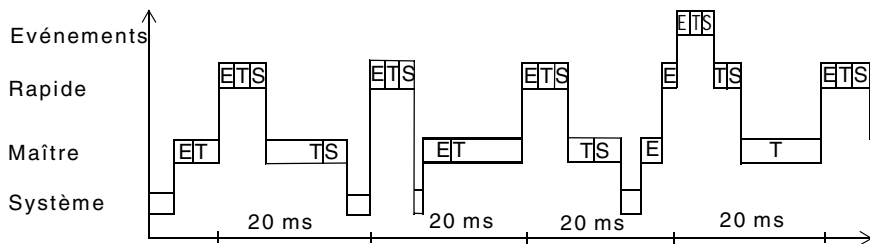
### Fonctionnement

Le tableau suivant décrit l'exécution des tâches prioritaires.

Phase	Description
1	Arrivée d'un événement ou début de cycle de la tâche rapide.
2	Arrêt de l'exécution des tâches en cours moins prioritaires,
3	Exécution de la tâche prioritaire.
4	La tâche interrompue reprend la main lorsque les traitements de la tâche prioritaire se termine.

### Description du séquencement des tâches

Le dessin suivant illustre le séquencement des tâches d'un traitement multitâche comportant une tâche maître cyclique, une tâche rapide de période 20ms et un traitement événementiel.



#### Légende :

E : acquisition des entrées  
 T : traitement du programme  
 S : mise à jour des sorties

### Contrôle des tâches

L'exécution des tâches rapide et événementielles peut être contrôlée par programme à travers l'utilisation des bits système :

- %S30 permet d'activer ou pas la tâche maître MAST.
- %S31 permet d'activer ou pas la tâche rapide FAST.
- %S38 permet d'activer ou pas les traitements événementiels EVTi.

## Affectation des voies d'entrées/sorties aux tâches maître et rapide

---

### Généralités

En plus du programme application, les tâches maître MAST et rapide FAST exécutent des fonctions système liées à la gestion des entrées/sorties implicites qui leur sont associées.

L'association d'une voie ou d'un groupe de voies à une tâche est définie dans l'écran de configuration du coupleur correspondant; la tâche associée par défaut étant la tâche MAST.

---

### Modules TOR

La modularité des modules TOR étant de 8 voies successives (voies 0 à 7, voies 8 à 15, ...), les entrées/sorties peuvent être affectées par groupes de 8 voies, indifféremment à la tâche MAST ou FAST.

**Exemple** : il est possible d'affecter les voies d'un module 28 entrées/sorties de la manière suivante :

- entrées 0 à 7 affectées à la tâche MAST,
  - entrées 8 à 15 affectées à la tâche FAST,
  - sorties 0 à 7 affectées à la tâche MAST,
  - sorties 8 à 15 affectées à la tâche FAST.
- 

### Modules de comptage

Chaque voie d'un module de comptage peut être affectée indifféremment à la tâche MAST ou FAST.

**Exemple** : pour un module de comptage 2 voies, il est possible d'affecter :

- la voie 0 à la tâche MAST,
  - la voie 1 à la tâche FAST.
- 

### Modules analogiques

Les voies des modules d'entrées analogiques du Micro sont obligatoirement affectées à la tâche MAST. Par contre, il est possible d'affecter les voies ou groupes de voies de sorties analogiques indifféremment à la tâche MAST ou FAST, avec une modularité de 2 voies.

**Exemple** : pour un module de 4 sorties analogiques, il est possible d'affecter :

- les voies 0 et 1 à la tâche MAST et,
- les voies 2 et 3 à la tâche FAST.

Les voies des modules d'entrées et sorties analogiques du Premium peuvent être affectées à la tâche MAST ou FAST. Cette affectation est individuelle pour chacune des voies des modules d'entrées ou de sorties analogiques isolées (4 voies isolées) et avec une modularité de 4 voies pour les autres modules.

**Note** : afin d'obtenir des performances optimales, il est préférable de regrouper les voies d'un module dans une même tâche.

## Echanges d'entrées/sorties dans les traitements événementiels

### Généralités

Il est possible d'utiliser à chaque traitement événementiel des voies d'entrées/sorties autres que celle relative à l'événement.

Les échanges sont alors réalisés implicitement par le système avant (%I) et après (%Q) le traitement applicatif.

Ces échanges peuvent être relatifs à une voie (exemple module de comptage) ou à un groupe de voies (module TOR). Dans le second cas, si le traitement modifie par exemple les sorties 2 et 3 d'un module TOR, c'est l'image des sorties 0 à 7 qui sera transférée vers le module.

### Fonctionnement

Le tableau suivant décrit les échanges et les traitements réalisés.

Phase	Description
1	L'apparition d'un événement déroute le programme application vers le traitement qui est associé à la voie d'entrée/sortie qui a provoqué l'événement
2	Toutes les entrées associées à la voie qui a provoqué l'événement sont acquises automatiquement.
3	Toutes les entrées utilisées par l'utilisateur dans le traitement EVTi sont acquises.
4	Le traitement événementiel est exécuté. Il doit être le plus court possible.
5	Toutes les sorties utilisées par l'utilisateur dans le traitement EVTi sont mises à jour. Les sorties associées à la voie qui a provoqué l'événement doivent être également utilisées, pour être mise à jour.

### Règle de programmation

#### Règle générale :

Les entrées échangées (et le groupe de voies associées) lors de l'exécution du traitement événementiel sont remis à jour (perte des valeurs historiques, donc des fronts), il faut donc éviter de tester des fronts sur ces entrées dans les tâches maître (MAST) ou rapide (FAST).

#### Dans le cas des modules TOR TSX DEY 16FK, TSX DMY 28FK ou TSX DMY 28RFK :

L'entrée qui a déclenché l'événement ne doit pas être testée dans le traitement événementiel (la valeur n'est pas rafraîchie).

Le test du front qui a déclenché l'événement doit être effectué sur le mot d'état:

- %IWxy.i:X0 = 1 --> front montant,
- %IWxy.i:X1 = 1 --> front descendant.

**Sur les automates Micro :**

- les modules d'entrées analogiques qui ne peuvent être utilisés que dans la tâche MAST, ne doivent pas être échangés dans un traitement événementiel.
- pour chaque traitement événementiel, il est possible de déclarer au maximum les échanges pour 2 modules en entrée (avant le traitement de l'événement) et 2 modules en sortie (après le traitement de l'événement).

**Performances**

Sur les automates Premium, suivant le processeur utilisé, le nombre d'échanges utilisés est limité:

<b>Nombre d'échanges utilisables dans les traitements événementiels par processeur</b>	<b>P57-1•</b>	<b>P57-2• /3• /4•</b>
Entrées/sorties TOR	32 échanges	128 échanges
Entrées/sorties analogiques	8 échanges	16 échanges
Autres métier	4 échanges	16 échanges

Pour les entrées/sorties TOR, un échange concerne un groupe de 8 voies, il est généré lors de l'utilisation des entrées d'un groupe de 8 voies (autre que le groupe de voies qui génère l'événement) et lors de l'écriture des sorties d'un groupe de 8 voies.

Pour les entrées/sorties analogiques ou d'un autre métier, un échange est généré lors de l'utilisation des entrées d'une voie (autre que la voie qui génère l'événement) et lors de l'écriture des sorties d'une voie).

**Note :**

- Les échanges des entrées/sorties de la tâche EVTi, étant réalisés par voie (pour certains modules analogiques et métiers) ou par groupe de voies (pour les modules TOR et certains modules analogiques), si le traitement modifie par exemple les sorties 2 et 3 d'un module TOR, c'est l'image (mémoire automate) des sorties de 0 à 7 qui sera transférée vers le module.
- Tout échange d'une entrée/sortie dans une tâche événementielle peut provoquer la perte de l'information de front vis à vis des traitements effectués sur cette voie (ou groupe de voie), dans la tâche où elle a été déclarée : MAST ou FAST.

**Visualisation du nombre d'événements traités**

Le mot système %SW48 donne le nombre d'événements traités.

Ce mot est initialisé à 0 sur démarrage à froid puis incrémenté par le système lors du lancement d'un événement.

Ce mot est modifiable par l'utilisateur.

Le bit système %S39 signale la perte d'événement.

## 5.4 Modules fonctionnels

### Structuration en modules fonctionnels

#### Généralités

Un module fonctionnel est un regroupement d'éléments de programme destinés à réaliser une fonction d'automatisme.

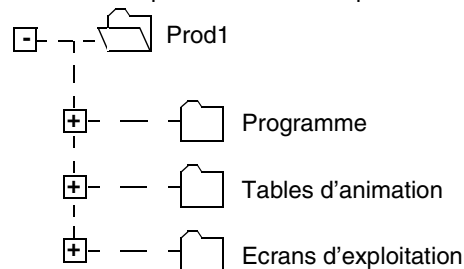
#### Structure

Un module fonctionnel est défini par les attributs suivants :

- nom court : 8 caractères (exemple : TR371)
- nom long : 16 caractères (exemple : Avance/Reculé pour BT371)
- une fiche descriptive (sans limitation du nombre de caractères) non mémorisée dans l'automate mais mémorisée dans le fichier .STX de l'application.

#### Illustration

L'illustration ci-après montre la composition d'un module fonctionnel :



#### Description des éléments d'un module fonctionnel

Le tableau décrit le rôle de chacun des éléments :

Élément	Composition
Programme	Un ou plusieurs modules de code : <ul style="list-style-type: none"> <li>● sections</li> <li>● événements</li> <li>● macro-étapes</li> <li>● tables d'animation</li> <li>● ...</li> </ul>
Tables d'animation	Une ou plusieurs tables d'animations.
Mdm1	Modules fonctionnels de niveau inférieur, ces modules assumant, par rapport à la fonction principale, une ou plusieurs sous-fonctions d'automatisme.

**Limites  
d'utilisation**

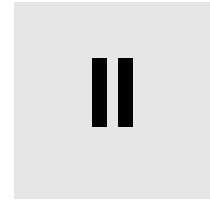
Seul le produit PL7 PRO permet la mise en oeuvre des modules fonctionnels sur les automates Premium.

---



---

# Description des langages PL7



---

## Présentation

### Objet de cet intercalaire

Cet intercalaire décrit les langages de programmation des automates Micro et Premium.

### Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre	Titre du chapitre	Page
6	Langage à contacts	107
7	Langage liste d'instructions	121
8	Langage littéral structuré	137
9	Grafcet	161
10	Blocs fonction DFB	205

---



---

# Langage à contacts



---

## Présentation

### Contenu de ce chapitre

Ce chapitre décrit la programmation en langage à contacts.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Présentation générale du langage à contacts	108
Structure d'un réseau de contacts	109
Etiquette d'un réseau de contacts	110
Commentaire d'un réseau de contacts	111
Éléments graphiques du langage à contacts	112
Règles de programmation d'un réseau de contacts	115
Règle de programmation des blocs fonction	116
Règles de programmation des blocs opération	117
Exécution d'un réseau de contacts	118

---

## Présentation générale du langage à contacts

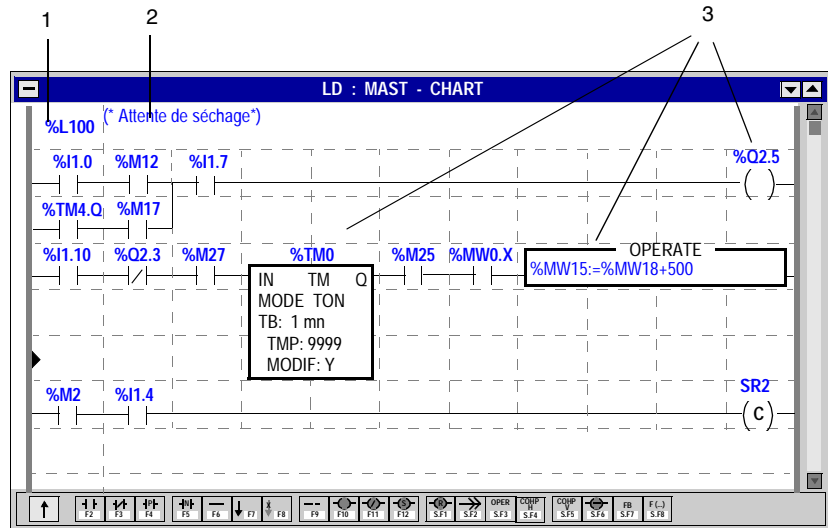
### Généralités

Une section de programme écrite en langage à contacts se compose d'une suite de réseaux de contacts exécutés séquentiellement par l'automate.

La représentation d'un réseau de contacts est proche de celle d'un schéma électrique.

### Illustration d'un réseau de contacts

L'écran suivant présente un réseau de contacts PL7.



### Composition d'un réseau de contacts

Ce tableau décrit les constituants d'un réseau de contacts.

Repère	Élément	Fonction
1	Étiquette	Repère un réseau de contacts (optionnel).
2	Commentaire	Renseigne un réseau de contacts (optionnel).
3	Éléments graphiques	<p>Ils représentent :</p> <ul style="list-style-type: none"> <li>● les entrées/sorties de l'automate (boutons-poussoirs, détecteurs, relais, voyants...)</li> <li>● les fonctions d'automatismes (temporisateur, compteurs...),</li> <li>● les opérations arithmétiques, logiques et spécifiques,</li> <li>● les variables internes de l'automate.</li> </ul>

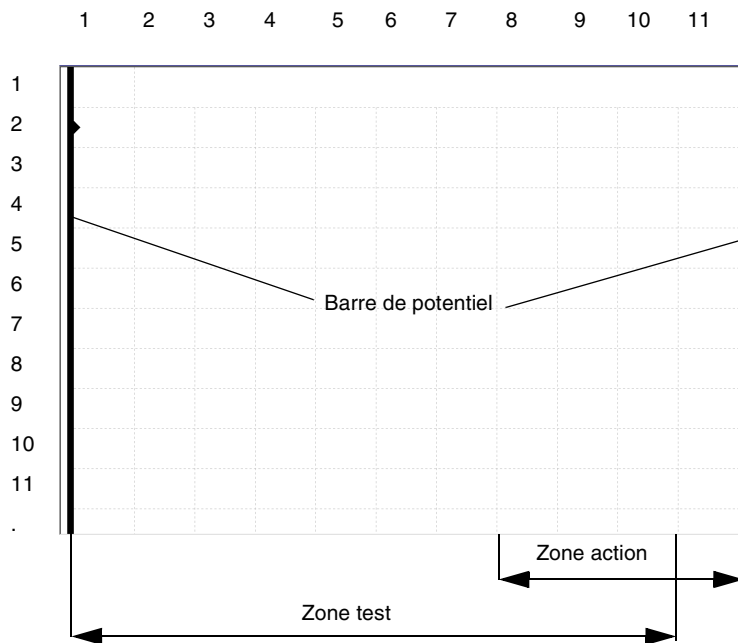
## Structure d'un réseau de contacts

### Introduction

Un réseau s'inscrit entre deux barres de potentiel. Le sens de circulation du courant s'établit de la barre de potentiel gauche vers la barre de potentiel droite.

### Illustration

Le dessin ci-après décrit la structure d'un réseau de contacts.



### Description d'un réseau de contacts

Un réseau de contacts est composé d'un ensemble d'éléments graphiques disposés sur une grille de :

- 16 lignes maximum et 11 colonnes (pour automates Premium),
- 7 lignes maximum et 11 colonnes (pour automates Micro).

Il est réparti en deux zones :

- la zone test, dans laquelle figurent les conditions nécessaires à une action
- la zone action, qui applique le résultat consécutif à un enchaînement de test.

## Étiquette d'un réseau de contacts

---

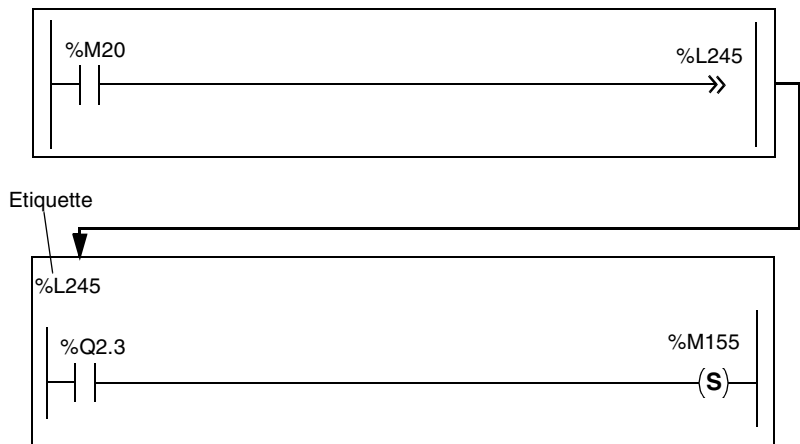
**Généralités** L'étiquette permet de repérer un réseau dans une entité de programme (programme principal, sous-programme, ...). Elle est optionnelle.

---

**Syntaxe** Cette étiquette a la syntaxe suivante : %Li avec i compris entre 0 et 999. Elle se positionne à la partie supérieure gauche devant la barre de potentiel.

---

**Illustration** Les réseaux de contacts suivant illustrent l'utilisation d'une étiquette.



**Règles** Un repère d'étiquette ne peut être affecté qu'à un seul réseau au sein d'une même entité de programme.

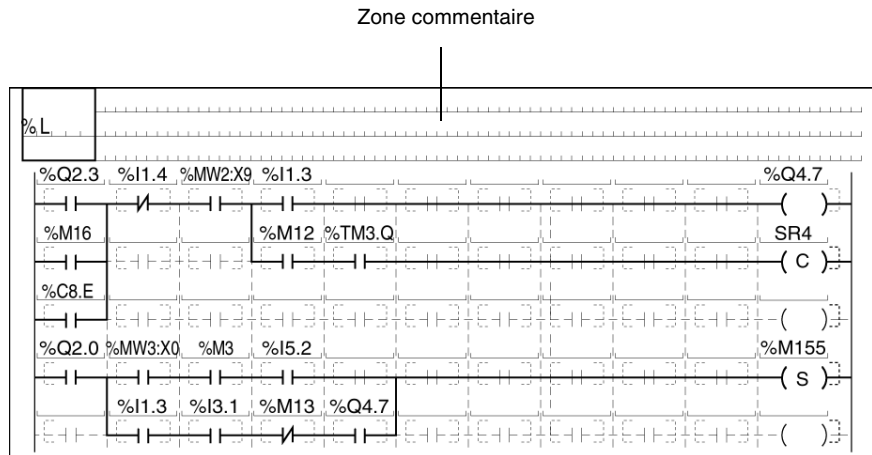
Il est nécessaire d'étiqueter un réseau afin de permettre un branchement après un saut de programme (voir illustration ci-dessus).

L'ordre des repères des étiquettes est quelconque, (c'est l'ordre de saisie des réseaux qui est pris en compte par le système lors de la scrutation).

---

## Commentaire d'un réseau de contacts

- Généralités** Le commentaire facilite l'interprétation du réseau auquel il est affecté, mais n'est pas obligatoire.
- Syntaxe** Le commentaire est intégré au réseau et comprend 222 caractères alphanumériques au maximum, encadrés de part et d'autre par les caractères (\* et \*).
- Illustration** Le dessin ci-dessous repère la position du commentaire.



- Règles** Les commentaires s'affichent dans la zone réservée dans la partie supérieure du réseau de contacts.
- En cas de suppression d'un réseau, le commentaire qui lui est associé est également supprimé.
- Les commentaires sont mémorisés dans l'automate et sont accessibles à tout moment par l'utilisateur. A ce titre, ils consomment de la mémoire programme





## Éléments graphiques du langage à contacts

### Généralités

Les éléments graphiques sont les instructions du langage à contacts.




### Contacts

Les éléments graphiques des contacts se programment en zone test et occupent une cellule (1 ligne de hauteur et 1 colonne de largeur).

Désignation	Graphisme	Fonctions
Contact à fermeture		Contact passant quand l'objet bit qui le pilote est à l'état 1.
Contact à ouverture		Contact passant quand l'objet bit qui le pilote est à l'état 0.
Contact à détection de front montant		Front montant : détection du passage de 0 à 1 de l'objet bit qui le pilote.
Contact à détection de front descendant		Front descendant : détection du passage de 1 à 0 de l'objet bit qui le pilote.

### Éléments de liaison

Les éléments graphiques de liaison permettent de relier les éléments graphiques de test et d'action.

Désignation	Graphisme	Fonctions
Connexion horizontale		Permet de relier en série les éléments graphiques de test et d'action entre les deux barres de potentiel.
Connexion verticale de potentiel		Permet de relier en parallèle les éléments graphiques de test et d'action.
Dérivation court-circuit		Permet de relier 2 objets au travers de plusieurs connexions.




**Bobines**

Les éléments graphiques des bobines se programment en zone action et occupent une cellule (1 ligne de hauteur et une colonne de largeur).

Désignation	Graphisme	Fonctions
Bobine directe	—( )—	L'objet bit associé prend la valeur du résultat de la zone test.
Bobine inverse	—( / )—	L'objet bit associé prend la valeur inverse du résultat de la zone test.
Bobine d'enclenchement	—(S)—	L'objet bit associé est mis à 1 lorsque le résultat de la zone test est à 1.
Bobine de déclenchement	—(R)—	L'objet bit associé est mis à 0 lorsque le résultat de la zone test est à 1.
Saut conditionnel à un autre réseau (JUMP)	->>%Li	Permet un branchement à un réseau étiqueté, amont ou aval. Les sauts ne sont effectifs qu'au sein d'une même entité de programmation (programme principal, sous-programme,...). L'exécution d'un saut provoque : <ul style="list-style-type: none"> <li>● l'arrêt de la scrutation du réseau en cours,</li> <li>● l'exécution du réseau étiqueté demandé,</li> <li>● la non scrutation de la partie du programme située entre l'action de saut et le réseau désigné.</li> </ul>
Bobine dièse	—(#)—	Proposée en langage Grafcet, utilisée lors de dièse la programmation des réceptivités associées aux transistions provoque le passage à l'étape suivante.
Bobine appel à un sous-programme (CALL)	—(C)—	Permet un branchement en début de sous-programme lorsque le résultat de la zone de test sous-programme est à 1. L'exécution d'un appel à un sous-programme provoque : <ul style="list-style-type: none"> <li>● l'arrêt de la scrutation du réseau en cours,</li> <li>● l'exécution du sous-programme,</li> <li>● la reprise de la scrutation du réseau interrompu.</li> </ul>
Retour de sous-programme	<RETURN>	Réservée au sous-programme SR, permet le retour au module appelant lorsque le résultat de la zone de test est à 1.
Arrêt programme	<HALT>	Provoque l'arrêt de l'exécution du programme lorsque le résultat de la zone de test est à 1.


**Blocs fonction standard**

Les éléments graphiques des blocs fonction standard se programment en zone test et occupent une dimension d'une hauteur de 16 lignes maximum et une largeur 3 colonnes.

Désignation	Graphisme	Fonctions
Blocs Temporisateur, Compteur, Monostable, Registre, Programmeur cyclique		Chacun des blocs fonctions standards utilise des entrées, des sorties, des entrées/sorties permettant de les relier aux autres éléments graphiques.




**Blocs fonction DFB**

Les éléments graphiques des blocs fonction DFB se programment en zone test et occupent une dimension d'une hauteur de 16 lignes maximum et une largeur 3 colonnes.

Désignation	Graphisme	Fonctions
Blocs programmables		Chacun des blocs fonctions DFB utilise des entrées, des sorties, des entrées/sorties permettant de les relier aux autres éléments graphiques pour les objets de type bits ou pouvant être affectés à des objets numériques ou tableaux

**Blocs opération**

Les éléments graphiques des blocs opération se programment en zone test et occupent les dimensions mentionnées ci-dessous.

Désignation	Graphisme	Fonctions
Bloc comparaison vertical		Permet la comparaison de 2 opérands, suivant le résultat, la sortie correspondante passe à 1. Dimension : 2 colonnes/4 lignes
Bloc comparaison horizontal		Permet la comparaison de 2 opérands, la sortie passe à 1 lorsque le résultat est vérifié (un bloc peut contenir jusqu'à 4096 caractères). Dimension : 2 colonnes/1 ligne
Bloc Opération		Réalise les opérations arithmétiques, logiques... fait appel à la syntaxe du langage littéral structuré. (Un bloc peut contenir jusqu'à 4096 caractères). Dimension : 4 colonnes/1 ligne

## Règles de programmation d'un réseau de contacts

### Généralités

La programmation d'un réseau de contacts s'effectue à l'aide des éléments graphiques, en respectant les règles de programmation ci-après.

### Règles de programmation

Les éléments graphiques simples de test et d'action occupent chacun une cellule au sein d'un réseau.

Toute ligne de contacts commence sur la ligne de potentiel gauche et doit se terminer sur la ligne de potentiel droite.

Les tests sont toujours situés sur les colonnes 1 à 10.

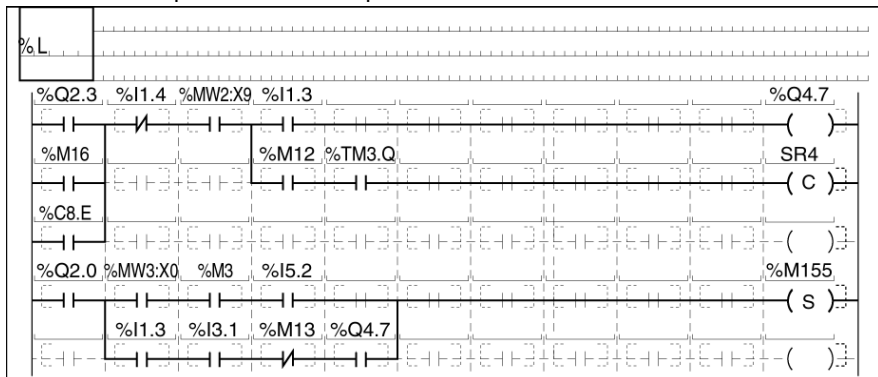
Les actions sont toujours situées sur la colonne 11.

Le sens de circulation du courant est le suivant :

- pour les liaisons horizontales, de la **gauche vers la droite**,
- pour les liaisons verticales, dans les deux sens.

### Exemple de réseau de contacts

L'écran suivant présente un exemple de réseau de contacts.



## Règle de programmation des blocs fonction

### Généralités

Les blocs fonction standard se positionnent dans la zone test des réseaux de contacts.

### Règles de programmation des blocs fonction

Quel que soit le type de bloc fonction utilisé, il doit obligatoirement être relié en entrée à la barre de potentiel gauche, en direct ou à travers d'autres éléments graphiques.

- **sorties "en l'air"** : il n'est pas nécessaire de relier à d'autres éléments graphiques les sorties des blocs fonction,
- **sorties testables** : les sorties des blocs fonction sont accessibles à l'utilisateur sous forme d'objet bit.

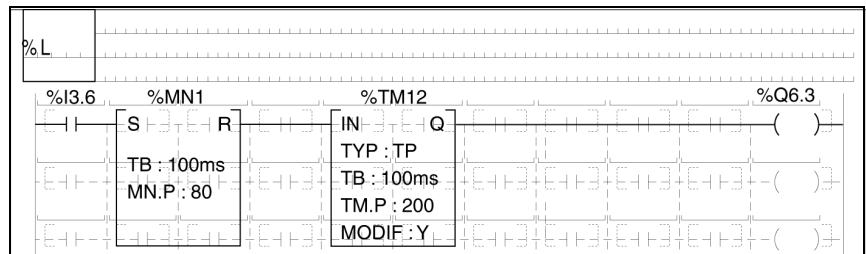
Les variables internes de blocs et les sorties graphiques sont des objets exploitables à distance depuis une autre partie du programme.

Les entrées non câblées des blocs fonction standard sont mises à 0.

Comme pour les éléments graphiques du type contacts, il est possible de réaliser des combinaisons de blocs fonction.

### Exemple d'un réseau de contacts

L'illustration suivante présente un exemple d'un réseau de contacts contenant 2 blocs fonction.



## Règles de programmation des blocs opération

### Généralités

Les blocs comparaison se positionnent dans la zone test et les blocs opération se positionnent dans la zone action.

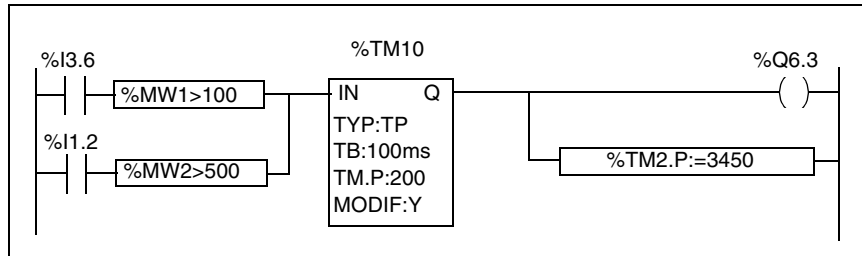
### Règles de programmation des blocs opération

Quel que soit le type de bloc opération utilisé, il doit obligatoirement être relié en entrée à la barre de potentiel gauche, en direct ou à travers d'autres éléments graphiques.

Comme pour les éléments graphiques du type contacts, il est possible de réaliser des combinaisons de blocs fonction et opération.

### Exemple de blocs opération

L'illustration suivante présente un exemple d'un réseau de contacts contenant 2 blocs de comparaison et un bloc opération.

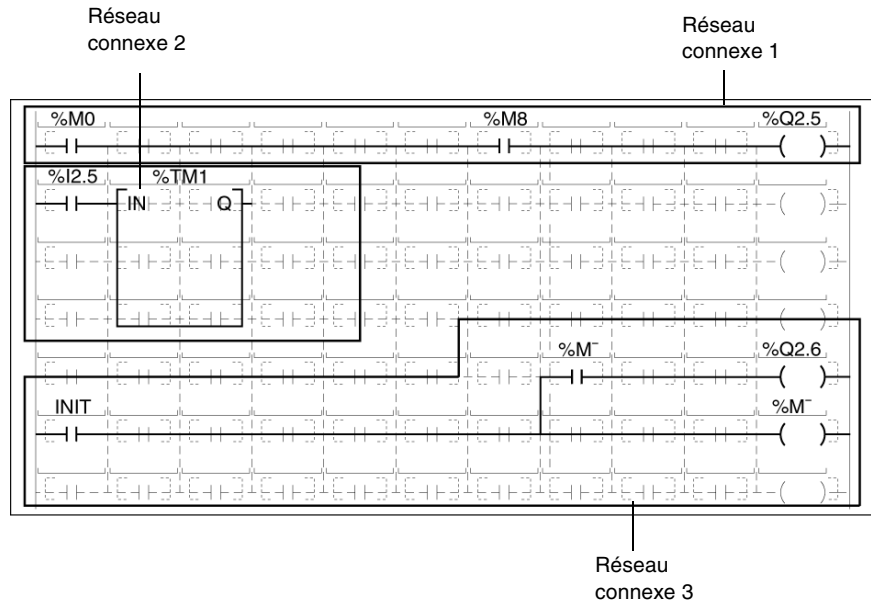


## Exécution d'un réseau de contacts

**Réseau connexe** Un réseau connexe contient des éléments graphiques tous reliés entre eux par des éléments de liaison (hors barre de potentiel), mais indépendants des autres éléments graphiques du réseau (pas de liaisons verticales vers le haut ou vers le bas en limite de réseau connexe).

### Illustration de réseaux connexes

Le réseau de contacts suivant est composé de 3 réseaux connexes.



### Règle d'exécution de réseaux connexes

Le premier réseau connexe évalué est celui dont le coin gauche est situé le plus en haut à gauche.

Un réseau connexe est évalué dans le sens de l'équation : évaluation du réseau de haut en bas, ligne par ligne, et dans chaque ligne de gauche à droite.

Dans le cas où une liaison verticale de convergence est rencontrée, le sous réseau qui lui est associé est évalué (selon la même logique) avant de continuer l'évaluation du réseau qui l'englobe.

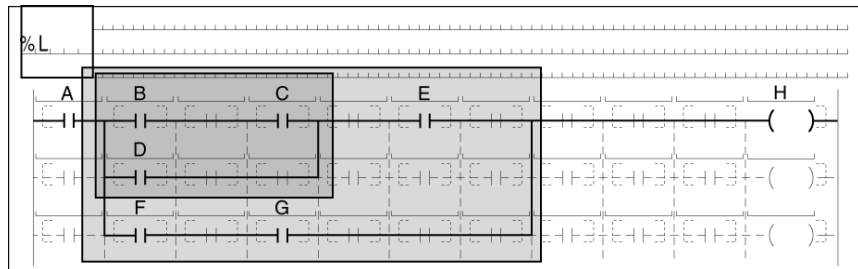
### Exécution des éléments dans un réseau connexe

Le tableau suivant décrit l'ordre d'exécution des éléments dans un réseau connexe.

Phase	Description
1	Le système évalue l'état logique de chaque contact, en fonction de : <ul style="list-style-type: none"> <li>la valeur courante des objets internes de l'application,</li> <li>l'état des entrées des modules d'entrées/sorties acquis en début de cycle</li> </ul>
2	Le système exécute les traitements associés aux fonctions, aux blocs fonctions, et aux sous-programmes,
3	Le système met à jour les objets bits associés aux bobines (la mise à jour des sorties des modules d'entrées/sorties s'effectue en fin de cycle),
4	Le système débranche vers un autre réseau étiqueté du même module programme (saut à un autre réseau ->>%Li), retour au même appelant <RETURN>, ou arrêt du programme <HALT>.

### Exemple 1 : illustration

Le dessin suivant visualise l'ordre d'exécution des éléments graphiques.



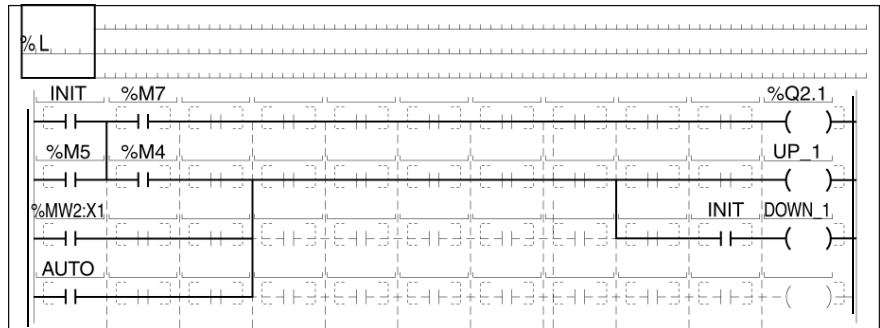
### Exemple 1 : fonctionnement

Le tableau suivant décrit l'exécution des éléments graphiques dans le réseau illustré ci-dessus.

Phase	Description
1	Evaluation du réseau jusqu'à rencontre de la 1ère liaison verticale de convergence : contacts A, B, C.
2	Evaluation du premier sous réseau : contact D,
3	Poursuite de l'évaluation du réseau jusqu'à la rencontre de la deuxième liaison verticale de convergence: contact E,
4	Evaluation du 2ème sous réseau : contacts F et G,
5	Evaluation de la bobine H.

**Exemple 2 :  
illustration**

Le dessin suivant visualise l'ordre d'exécution des éléments graphiques.



**Exemple 2 :  
fonctionnement**

Le tableau suivant décrit l'exécution des éléments graphiques dans le réseau illustré ci-dessus.

Phase	Description
1	bobine 1 : INIT, %M5, %M7, %Q2.1,
2	bobine 2 : %M4, %MW2:X1,AUTO, UP_1,
3	Bloc opération.



---

# Langage liste d'instructions

# 7

---

## Présentation

### Généralités

Ce chapitre décrit les règles de programmation en langage liste d'instructions.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Présentation générale du langage liste d'instructions	122
Structure d'un programme liste d'instructions	123
Etiquette d'une phrase en langage liste d'instructions	124
Commentaire d'une phrase en langage liste d'instructions	125
Présentation des instructions en langage liste d'instructions	126
Règle d'utilisation des parenthèses en langage liste d'instructions	129
Description des instructions MPS, MRD et MPP	131
Principes de programmation des blocs fonction prédéfinis	133
Règles d'exécution d'un programme liste d'instructions	135

---

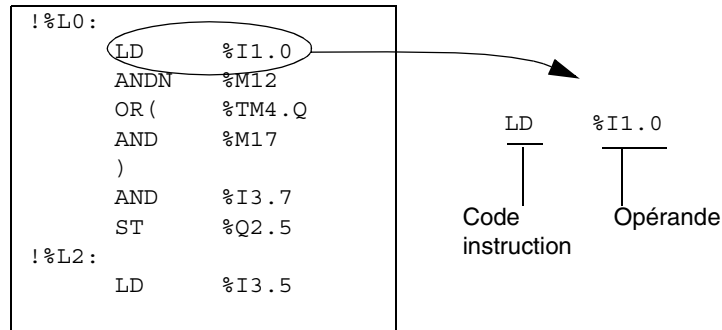
## Présentation générale du langage liste d'instructions

### Généralités

Une section écrite en langage liste d'instructions se compose d'une suite d'instructions exécutées séquentiellement par l'automate.

### Illustration d'un programme

L'illustration suivant présente un programme liste d'instructions PL7 et le détail d'une instruction.



### Composition d'une instruction

Ce tableau décrit les constituants d'une instruction.

Élément	Fonction
Code instruction	<p>Le code instruction détermine l'opération à exécuter. Il existe 2 types de codes instructions :</p> <ul style="list-style-type: none"> <li>● <b>test</b>, dans laquelle figurent les conditions nécessaires à une action (ex : LD, AND, OR...),</li> <li>● <b>action</b>, qui sanctionne le résultat consécutif à un enchaînement de test. (ex : ST, STN, R, ...).</li> </ul>
Opérande	<p>Une instruction agit sur un opérande. Cet opérande peut être :</p> <ul style="list-style-type: none"> <li>● une entrée/sortie de l'automate (boutons-poussoirs, détecteurs, relais, voyants...),</li> <li>● une fonctions d'automatisme (temporisateurs, compteurs...),</li> <li>● une opération arithmétique et logique ou une opération de transfert,</li> <li>● une variable interne de l'automate.</li> </ul>

## Structure d'un programme liste d'instructions

### Généralités

Comme en langage à contacts, les instructions sont organisées en séquence d'instructions (équivalent à un réseau de contacts) appelée phrase.

### Exemple de phrase

L'illustration suivante présente une phrase liste d'instructions PL7.

```

!(*Attente de séchage*)_____ 1
%L2 : _____ 2
      LD      %I1.0
      AND     %M10 _____ 3
      ST      %Q2.5
  
```

### Description d'une phrase

Chaque phrase commence par un point d'exclamation (généralisé automatiquement), elle comporte les éléments suivants.

Repère	Elément	Fonction
1	Commentaire	Renseigne une phrase (optionnel).
2	Etiquette	Repère une phrase (optionnel).
3	Instructions	Une à plusieurs instructions de test, le résultat de ces instructions étant appliqué à une ou plusieurs instructions d'action. Une instruction occupe une ligne maximum

## Étiquette d'une phrase en langage liste d'instructions

---

**Généralités** L'étiquette permet de repérer une phrase dans une entité de programme (programme principal, sous-programme, ...). Elle est optionnelle.

---

**Syntaxe** Cette étiquette a la syntaxe suivante : %Li avec i compris entre 0 et 999. Elle se positionne en début d'une phrase.

---

**Illustration** Le programme suivant illustre l'utilisation d'une étiquette.

```
%L0 :  
    LD      %M40  
    JMPC   %L10  
  
!(*Attente de séchage*)  
%L2 :  
    LD      %I1.0  
    AND    %M10  
    ST     %Q2.5  
...  
%L10 : _____ Etiquette  
    LD      %I3.5  
    ANDN   %Q4.3  
    OR     %M20  
    ST     %Q2.5
```

**Règles** Une même étiquette ne peut être affectée qu'à une seule phrase au sein d'une même entité de programme.

Il est nécessaire d'étiqueter une phrase afin de permettre un branchement après un saut de programme.

L'ordre des repères des étiquettes est quelconque, c'est l'ordre de saisie des phrases qui est prise en compte par le système lors de la scrutation.

---

---

## Commentaire d'une phrase en langage liste d'instructions

---

**Généralités** Le commentaire facilite l'interprétation d'une phrase auquel il est affecté. Il est optionnel.

---

**Syntaxe** Le commentaire peut être intégré au début d'une phrase et peut occuper 3 lignes maximum (soit 222 caractères alphanumériques), encadrés de part et d'autre par les caractères (\* et \*).

---

**Illustration** L'illustration ci-après repère la position du commentaire dans une phrase.

!(*Attente de séchage*)	Commentaire
%L2 :	
LD        %I1.0	
AND       %M10	
ST        %Q2.5	

---

**Règles** Les commentaires s'affichent uniquement à partir de la première ligne de la phrase.

En cas de suppression d'une phrase, le commentaire qui lui est associé est également supprimé.

Les commentaires sont mémorisés dans l'automate et sont accessibles à tout moment par l'utilisateur. A ce titre, ils consomment de la mémoire programme.

---

## Présentation des instructions en langage liste d'instructions

### Généralités



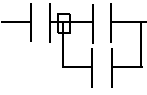
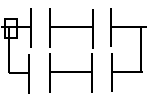
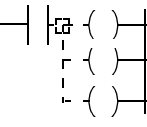
Le langage liste d'instructions comporte des instructions :

- de test,
- d'action,
- sur bloc fonction,
- numériques.

### Instructions de test





Le tableau suivant décrit les instructions de test du langage liste d'instructions.

Désignation	Graphisme équivalent	Fonctions
LD		Le résultat booléen est égal à l'état de l'opérande.
LDN		Le résultat booléen est égal à l'état inverse de l'opérande.
LDR		Le résultat booléen passe à 1 à la détection du passage de 0 à 1 de l'opérande (front montant).
LDF		Le résultat booléen passe à 1 à la détection du passage de 1 à 0 de l'opérande (front descendant).
AND		Le résultat booléen est égal au Et logique entre le résultat booléen de l'instruction précédente et l'état de l'opérande.
ANDN		Le résultat booléen est égal au Et logique entre le résultat booléen de l'instruction précédente et l'état inverse de l'opérande.
ANDR		Le résultat booléen est égal au Et logique entre le résultat booléen de l'instruction précédente, et la détection d'un front montant de l'opérande (1=front montant).
ANDF		Le résultat booléen est égal au Et logique entre le résultat booléen de l'instruction précédente, et la détection d'un front descendant de l'opérande (1=front descendant).
OR		Le résultat booléen est égal au Ou logique entre le résultat booléen de l'instruction précédente et l'état de l'opérande.
ORN		Le résultat booléen est égal au Ou logique entre le résultat booléen de l'instruction précédente et l'état inverse de l'opérande.

Désignation	Graphisme équivalent	Fonctions
ORR		Le résultat booléen est égal au Ou logique entre le résultat booléen de l'instruction précédente, et la détection d'un front montant de l'opérande (1=front montant).
ORF		Le résultat booléen est égal au Et logique entre le résultat booléen de l'instruction précédente et l'état de l'opérande., et la détection d'un front descendant de l'opérande (1=front descendant).
AND(		Et logique (8 niveaux de parenthèses)
OR(		Ou logique (8 niveaux de parenthèses)
XOR, XORN, XORR, XORF	-	Ou exclusif
MPS MRD MPP		Aiguillage vers des bobines.
N	-	Négation

## Instructions d'action

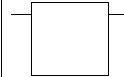
Le tableau suivant décrit les instructions de test du langage liste d'instructions.

Désignation	Graphisme	Fonctions
ST		L'opérande associé prend la valeur du résultat de la zone test.
STN		L'opérande associé prend la valeur inverse du résultat de la zone test.
S		L'opérande associé est mis à 1 lorsque le résultat de la zone test est à 1.
R		L'opérande associé est mis à 0 lorsque le résultat de la zone test est à 1.
JMP	-	Permet un branchement inconditionnel à une phrase étiquetée, amont ou aval.

Désignation	Graphisme	Fonctions
JMPC	-	Permet un branchement conditionné à un résultat booléen à 1 , à une phrase étiquetée amont ou aval.
JMPCN	-	Permet un branchement conditionné à un résultat booléen à 0 , à une phrase étiquetée amont ou aval.
SRn	-	Branchement en début de sous-programme.
RET	-	Retour de sous-programme.
RETC	-	Retour de sous-programme conditionné à un résultat booléen à 1.
RETCN	-	Retour de sous-programme conditionné à un résultat booléen à 0.
END	-	Fin de programme.
ENDC	-	Fin de programme conditionné à un résultat booléen à 1.
ENDCN	-	Fin de programme conditionné à un résultat booléen à 0.

### Instruction sur bloc fonction

Le tableau suivant décrit les instructions de test du langage liste d'instructions..

Désignation	Graphisme	Fonctions
Blocs Temporisateur, Compteur, Monostable, Registre, Programmeur cyclique		Pour chacun des blocs fonction standards, il existe des instructions permettant de piloter le bloc. Une forme structurée permet de câbler directement les entrées/sorties des blocs.

### Instructions numériques

Le tableau suivant décrit les instructions de test du langage liste d'instructions.

Désignation	Instructions	Fonctions
Élément de test	LD[.....] AND[.....] OR[.....]	Permet la comparaison de 2 opérands, la sortie passe à 1 lorsque le résultat est vérifié. <b>Exemple</b> : LD[%MW10<1000] Résultat à 1 lorsque %MW10<1000.
Élément d'action	[.....]	Réalisent les opérations arithmétiques, logiques... Utilisent la syntaxe du langage littéral structuré. <b>Exemple</b> : [%MW10:=%MW0+100] Le résultat de l'opération %MW0+100 est placé dans le mot interne %MW10.



## Règle d'utilisation des parenthèses en langage liste d'instructions

### Généralités

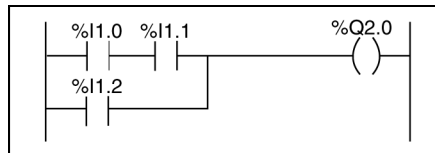
Les instructions AND et OR peuvent utiliser des parenthèses.  
Ces parenthèses permettent de réaliser des schémas à contacts de façon simple.

### Principe

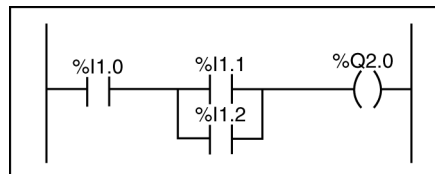
L'ouverture de parenthèses est associée à l'instruction AND ou OR.  
La fermeture de parenthèses est une instruction, elle est obligatoire pour chaque parenthèse ouverte.

### Exemple : AND(

Les deux programmes suivants illustrent l'utilisation de la parenthèse.



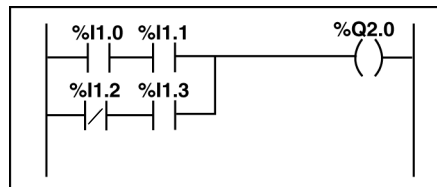
LD	%I1.0
AND	%I1.1
OR	%I1.2
ST	%Q2.0



LD	%I1.0
<b>AND (</b>	<b>%I1.1</b>
OR	%I1.2
)	
ST	%Q2.0

### Exemple : OR(

Le programme suivant illustre l'utilisation de la parenthèse.



LD	%I1.0
AND	%I1.1
<b>OR (N</b>	<b>%I1.2</b>
AND	%I1.3
)	
ST	%Q2.0

### Association des parenthèses à des modificateurs

Les "modificateurs" suivants peuvent être associés aux parenthèses.

Code	Rôle	Exemple
N	Négation	AND (N
F	Front descendant (Falling edge)	AND (F
R	Front montant (Rising edge)	OR (R
[	Comparaison	OR ([%MW0>100]

### Imbrication de parenthèses

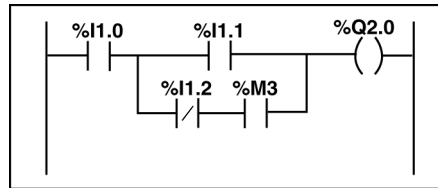
Il est possible d'imbriquer jusqu'à 8 niveaux de parenthèses.

Les règles ci-après doivent être suivies :

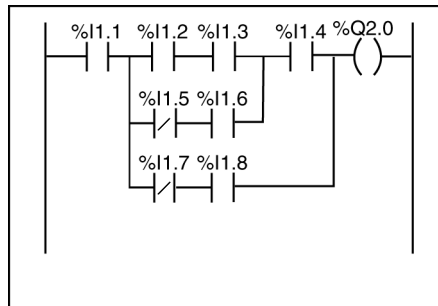
- Chaque parenthèse ouverte doit être impérativement refermée
- Les étiquettes %Li : ne doivent pas être placées dans des expressions entre parenthèses, ainsi que les instructions de saut JMP et d'appel à sous programme SRI,
- Les instructions d'affectation ST, STN, S et R ne doivent pas être programmées entre parenthèses.

#### Exemple :

Les programmes suivants illustrent l'utilisation de l'imbrication des parenthèses.



```
LD      %I1.0
AND (   %I1.1
OR (N   %I1.2
AND     %M3
)
)
ST      %Q2.0
```



```
LD      %I1.1
AND (   %I1.2
AND     %I1.3
OR (N   %I1.5
AND     %I1.6
)
AND     %I1.4
OR (N   %I1.7
AND     %I1.8
)
)
ST      %Q2.0
```

## Description des instructions MPS, MRD et MPP

### Généralités

Les 3 types d'instruction permettent de traiter les aiguillages vers les bobines.

Ces instructions utilisent une mémoire intermédiaire appelée pile pouvant stocker jusqu'à 3 informations booléennes.

**Note** : ces instructions ne peuvent pas être utilisées au sein d'une expression entre parenthèses

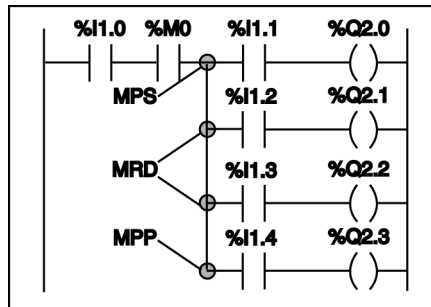
### Rôle

Le tableau suivant décrit le rôle de chacune des instructions

Instruction	Rôle
MPS (Memory PuSh)	Cette instruction a pour effet de stocker le résultat de la dernière instruction de test au sommet de la pile et de décaler les autres valeurs vers le fond de la pile.
MRD (Memory ReaD)	Cette instruction lit le sommet de la pile.
MPP (Memory PoP)	Cette instruction a pour effet de lire, de déstocker le sommet de la pile et de décaler les autres valeurs vers le sommet de la pile.

### Exemple 1

Cet exemple illustre l'utilisation des instructions MPS, MRD, et MPP.



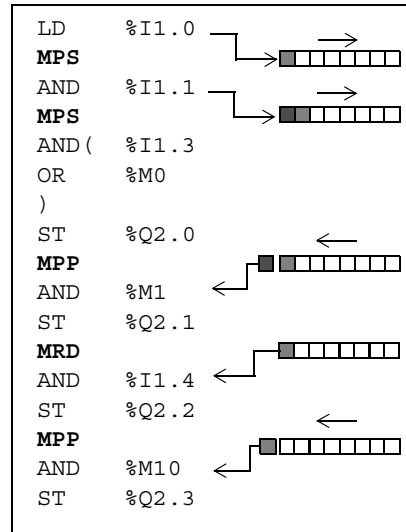
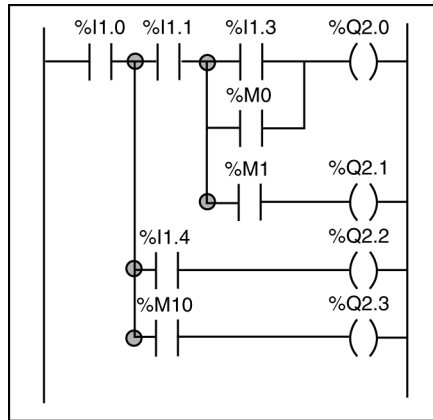
```

LD      %I1.0
AND     %M0
MPS
AND     %I1.1
ST      %Q2.0
MRD
AND     %I1.2
ST      %Q2.1
MRD
AND     %I1.3
ST      %Q2.2
MPP
AND     %I1.4
ST      %Q2.3

```

**Exemple 2**

Cet exemple illustre le fonctionnement des instructions MPS, MRD, et MPP.



## Principes de programmation des blocs fonction prédéfinis

### Généralités

Les blocs fonctions d'automatisme peuvent être programmés de 2 façons différentes :

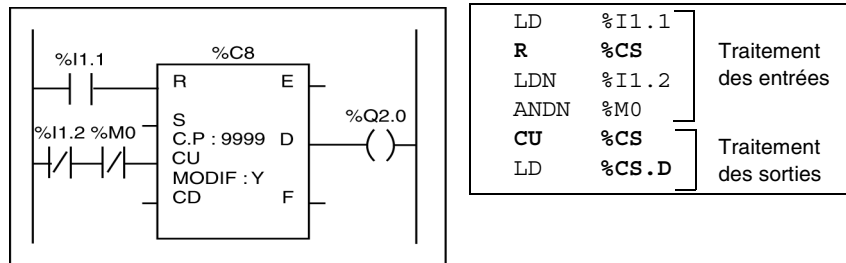
- avec instructions spécifiques à chaque bloc fonction (ex: CU %Ci), cette façon est la plus simple et la plus directe,
- avec instructions de structuration de bloc BLK ,OUT\_BLK, END\_BLK.

### Principe de programmation directe

Les instructions pilotent les entrées des blocs (ex: CU). Les sorties sont accessibles sous forme de bit (ex: %C8 .D).

#### Exemple :

Cet exemple illustre la programmation directe d'un bloc fonction compteur.



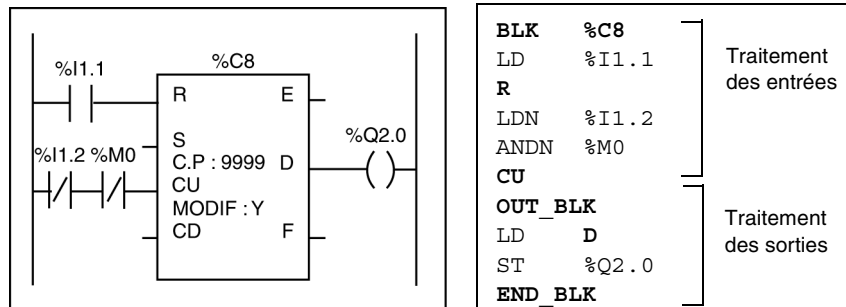
### Principe de programmation structurés

Ce type de programmation utilise une suite d' instructions encadrée par les instructions :

- BLK indique le début du bloc
- OUT\_BLK permet de câbler directement les sorties du bloc
- END\_BLK indique la fin du bloc

#### Exemple :

Cet exemple illustre la programmation structurée d'un bloc fonction compteur.



**Note** : ce principe de programmation structuré nécessitant les instructions supplémentaires BLK, OUT\_BLK et END\_BLK demande des volumes mémoires supérieurs par rapport à la programmation directe. Il est cependant à utiliser, si vous voulez garder une similitude avec des programmes réversibles pour nano-automates TSX 07.

---

## Règles d'exécution d'un programme liste d'instructions

### Principe

L'exécution d'un programme liste d'instructions s'effectue séquentiellement instruction par instruction.

La première instruction d'une séquence d'instructions doit toujours être soit une instruction LD soit une instruction inconditionnelle (ex : JMP).

Chaque instruction (excepté LD et les instructions inconditionnelles) utilise le résultat booléen de l'instruction précédente.

### Exemple 1

Le programme ci-après décrit l'exécution complète d'une phrase.

```
LD    %I1.1  résultat = état du bit %I1.1
AND   %M0    résultat = ET du résultat booléen précédent et de l'état du bit %M0
OR    %M10   résultat = OU du résultat booléen précédent et de l'état du bit %M10
ST    %Q2.0  %Q2.0 prend l'état du résultat booléen précédent
```

### Exemple 2

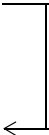
Les parenthèses permettent de modifier l'ordre de prise en compte des résultats booléens :

```
LD    %I1.1  résultat = état du bit %I1.1
AND   %M0    résultat = ET du résultat booléen précédent et de l'état du bit %M0
OR (   %M10   résultat = état du bit %M10
AND   %I1.2  résultat = ET du résultat booléen précédent et de l'état du bit %M10
)
ST    %Q2.0  %Q2.0 prend l'état du résultat booléen précédent
```

### Exemple 3

Le séquençement des instructions peut être modifié par les instructions de saut JMP d'appel à sous-programme.

```
!    LD      %M0
      JMPC   %L10
!    LD      %I1.1
      AND   %M10
      ST    %Q2.0
!    %L10:
      LD      %I1.3
      AND   %M20
      . . . . .
```



Saut à l'étiquette %L10 si %M0=1





---

# Langage littéral structuré



# 8

---

## Présentation

### Objet de ce chapitre

Ce chapitre décrit les règles de programmation en langage littéral structuré.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Présentation du langage littéral structuré	138
Structure d'un programme en langage littéral structuré	139
Etiquette d'une phrase en langage littéral structuré	140
Commentaire d'une phrase en langage littéral structuré	141
Instructions sur objets bits	142
Instructions arithmétiques et logiques	143
Instructions sur tableaux et chaîne de caractères	145
Instructions de conversions numériques	148
Instructions sur programme et instructions spécifiques	149
Structure de contrôle conditionnelle IF...THEN	151
Structure de contrôle conditionnelle WHILE...END_WHILE	153
Structure de contrôle conditionnelle REPEAT...END_REPEAT	154
Structure de contrôle conditionnelle FOR...END_FOR	155
Instruction de sortie de boucle EXIT	156
Règles d'exécution d'un programme littéral structuré	157

---

## Présentation du langage littéral structuré

---

### Généralités

Le langage littéral structuré est un langage évolué de type algorithmique particulièrement adapté à la programmation de fonctions arithmétiques complexes, manipulations de tableaux et gestions de messages.

Il permet la réalisation de programmes par écriture de lignes de programmation, constituées de caractères alphanumériques.

---

### Limite d'utilisation

Ce langage est utilisable avec les logiciels PL7 Micro, PL7 Junior et PL7 Pro sur les automates Premium et Micro.

Dans la version PL7 Pro, ce langage permet la création des blocs fonction utilisateur DFB sur les automates Premium.

---

### Illustration d'un programme

L'illustration suivante présente un programme en langage structuré PL7.

```
!      (* Recherche du premier élément non nul dans un
      tableau de 32 mots, détermination de sa valeur
      (%MW10), de son rang (%MW11). Cette recherche
      s'effectue si %M0 est à 1, %M1 est mis à 1 si
      un élément non nul existe, sinon il est mis à 0*)

      IF %M0 THEN
          FOR %MW99:=0 TO 31 DO
              IF %MW100[%MW99]<>0 THEN
                  %MW10:=%MW100[%MW99];
                  %MW11:=%MW99;
                  %M1:=TRUE;
                  EXIT;      (*Sortie de la boucle*)
              ELSE
                  %M1:=FALSE;
              END_IF;
          END_FOR;
      ELSE
          %M1:=FALSE;
      END_IF;
```

## Structure d'un programme en langage littéral structuré

### Généralités

Une section de programme littéral est organisée en phrases.  
Une phrase littérale est l'équivalent d'un réseau de contacts en langage à contacts.

### Exemple de phrase

L'illustration suivante présente une phrase en langage structurée PL7.

1	<pre>%L20:  (*Attente de séchage*) SET %M0; %MW4 :=%MW2 + %MW9; (*calcul de pression*) %MF12 :=SQRT (%MF14);</pre>	2  3
---	--	------------

### Description d'une phrase

Chaque phrase commence par un point d'exclamation (généré automatiquement), elle comporte les éléments suivants.

Repère	Élément	Fonction
1	Étiquette	Repère une phrase.
2	Commentaire	Renseigne une phrase.
3	Instructions	Une à plusieurs instructions séparées par ";".

**Note :** Chacun de ces éléments est optionnel, c'est-à-dire qu'il est possible d'avoir une phrase vide, une phrase constituée uniquement de commentaires ou uniquement d'une étiquette.

## Étiquette d'une phrase en langage littéral structuré

---

**Rôle** L'étiquette permet de repérer une phrase dans une entité de programme (programme principal, sous-programme, ...). Elle est optionnelle.

---

**Syntaxe** Cette étiquette a la syntaxe suivante : %Li : avec i compris entre 0 et 999. Elle se positionne en début d'une phrase.

---

**Illustration** Le programme suivant illustre l'utilisation d'une étiquette.

```
!      %L20 : _____ Etiquette
      (*Attente de séchage*)
      SET %M0;
      %MW4 := %MW2 + %MW9;
      (*calcul de pression*)
      %MF12 := SQRT (%MF14);
```

**Règles** Une même étiquette ne peut être affectée qu'à une seule phrase au sein d'une même entité de programme.

Il est nécessaire d'étiqueter une phrase afin de permettre un branchement après un saut de programme.

L'ordre des repères des étiquettes est quelconque, c'est l'ordre de saisie des phrases qui est prise en compte par le système lors de la scrutation.

---

---

## Commentaire d'une phrase en langage littéral structuré

---

**Rôle** Le commentaire facilite l'interprétation d'une phrase à laquelle il est affecté. Il est optionnel.

---

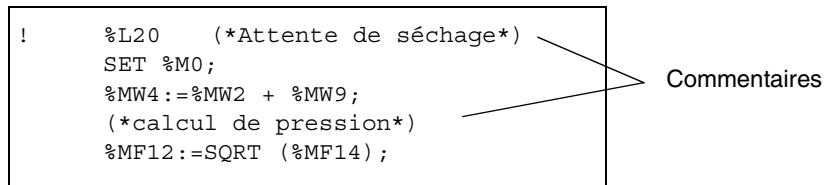
**Syntaxe** Le commentaire peut être intégré n'importe où dans la phrase et le nombre de commentaires par phrase n'est pas limité.

Un commentaire est encadré de part et d'autre par les caractères (\* et \*).

---

**Illustration** L'illustration ci-après repère la position du commentaire dans une phrase.

```
!      %L20      (*Attente de séchage*)
      SET %M0;
      %MW4 := %MW2 + %MW9;
      (*calcul de pression*)
      %MF12 := SQRT (%MF14);
```



Commentaires

---

### Règles

- Tous les caractères sont autorisés dans un commentaire.
- Le nombre de caractères est limité à 256 par commentaire.
- Les commentaires imbriqués sont interdits.
- Un commentaire peut tenir sur plusieurs lignes.

Les commentaires sont mémorisés dans l'automate et sont accessibles à tout moment par l'utilisateur. A ce titre, ils consomment de la mémoire programme.

---

## Instructions sur objets bits

### Instructions sur bits

Les instructions suivantes s'appliquent sur des objets bits.

Désignation	Fonction
:=	Affectation d'un bit
OR	OU booléen
AND	ET booléen
XOR	OU exclusif booléen
NOT	Inversion
RE	Front montant
FE	Front descendant
SET	Mise à 1
RESET	Mise à 0

### Instructions sur tableaux de bits

Les instructions suivantes s'appliquent sur des objets de type tableau de bits.

Désignation	Fonction
Tableau := Tableau	Affectation entre deux tableaux
Tableau := Mot	Affectation d'un mot à un tableau
Mot := Tableau	Affectation d'un tableau à un mot
Tableau := Double mot	Affectation d'un double mot à un tableau
Double mot := Tableau	Affectation d'un tableau à un double mot
COPY_BIT	Copie d'un tableau de bits dans un tableau de bits
AND_ARX	ET entre deux tableaux
OR_ARX	OU entre deux tableaux
XOR_ARX	OU exclusif entre deux tableaux
NOT_ARX	Négation sur un tableau
BIT_W	Copie d'un tableau de bits dans un tableau de mots
BIT_D	Copie d'un tableau de bits dans un tableau de doubles mots
W_BIT	Copie d'un tableau de mots dans un tableau de bits
D_BIT	Copie d'un tableau de doubles mots dans un tableau de bits
LENGHT_ARX	Calcul de la longueur d'un tableau en nombre d'éléments

## Instructions arithmétiques et logiques

### Arithmétique entière sur mots et doubles mots

Les instructions suivantes s'appliquent sur des objets mots et doubles mots.

Désignation	Fonction
+, -, *, /	Addition, Soustraction, Multiplication, Division entière
REM	Reste de la division entière
SQRT	Racine carrée entière
ABS	Valeur absolue
INC	Incrémentatation
DEC	Décrémentatation

### Arithmétique sur flottants

Les instructions suivantes s'appliquent sur des objets flottants.

Désignation	Fonction
+, -, *, /	Addition, Soustraction, Multiplication, Division
SQRT	Racine carrée
ABS	Valeur absolue
TRUNC	Partie entière
LOG	Logarithme base 10
LN	Logarithme népérien
EXP	Exponentielle naturelle
EXPT	Exponentiation d'un réel par un réel
COS	Cosinus d'une valeur en radian
SIN	Sinus d'une valeur en radian
TAN	Tangente d'une valeur en radian
ACOS	Arc cosinus (résultat entre 0 et 2 p)
ASIN	Arc sinus (résultat entre -p/2 et +p/2)
ATAN	Arc tangente (résultat entre -p/2 et +p/2)
DEG_TO_RAD	Conversion degrés en radians
RAD_TO_DEG	Conversion radians en degrés

**Instructions logiques sur mots et doubles mots**

Les instructions suivantes s'appliquent sur des objets mots et doubles mots.

Désignation	Fonction
AND	ET logique
OR	OU logique
XOR	OU logique exclusif
NOT	Complément logique
SHL	Décalage logique à gauche
SHR	Décalage logique à droite
ROL	Décalage logique circulaire à gauche
ROR	Décalage logique circulaire à droite

---

**Comparaisons numériques sur mots, doubles mots et flottants**

Les instructions suivantes s'appliquent sur des objets mots, doubles mots et flottants.

Désignation	Fonction
<	Strictement inférieur à
>	Strictement supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
=	Egal à
<>	Différent de

---



## Instructions sur tableaux et chaîne de caractères

### Instructions sur tableaux de mots et doubles mots

Les instructions suivantes s'appliquent sur des tableaux de mots et doubles mots.

Désignation	Fonction
Tableau := Tableau	Affectation entre deux tableaux
Tableau := Mot	Initialisation d'un tableau
+, -, *, /, REM	Opérations arithmétiques entre tableaux
+, -, *, /, REM	Opérations arithmétiques entre expressions et tableaux
SUM	Sommation des éléments d'un tableau
EQUAL	Comparaison de deux tableaux
NOT	Complément logique d'un tableau
AND, OR, XOR	Opérations logiques entre deux tableaux
AND, OR, XOR	Opérations logiques entre expressions et tableaux
FIND_EQW, FIND_EQD	Recherche du premier élément égal à une valeur
FIND_GTW, FIND_GTD	Recherche du premier élément supérieur à une valeur
FIND_LTW, FIND_LTD	Recherche du premier élément inférieur à une valeur
MAX_ARW, MAX_ARD	Recherche de la valeur maximum dans un tableau
MIN_ARW, MIN_ARD	Recherche de la valeur minimum dans un tableau
OCCUR_ARW, OCCUR_ARD	Nombre d'occurrences d'une valeur dans un tableau
SORT_ARW, SORT_ARD	Tri par ordre croissant ou décroissant d'un tableau
ROL_ARW, ROL_ARD	Décalage circulaire à gauche d'un tableau
ROR_ARW, ROR_ARD	Décalage circulaire à droite d'un tableau
FIND_EQWP, FIND_EQDP	Recherche du premier élément égal à une valeur depuis un rang
LENGTH_ARW, LENGTH_ARD	Calcul de longueur d'un tableau

**Instructions sur tableaux de flottants**

Les instructions suivantes s'appliquent sur des tableaux de flottants.

Désignation	Fonction
Tableau := Tableau	Affectation entre deux tableaux
Tableau := Flottant	Initialisation d'un tableau
SUM_ARR	Sommation des éléments d'un tableau
EQUAL_ARR	Comparaison de deux tableaux
FIND_EQR	Recherche du premier élément égal à une valeur
FIND_GTR	Recherche du premier élément supérieur à une valeur
FIND_LTR	Recherche du premier élément inférieur à une valeur
MAX_ARR	Recherche de la valeur maximum dans un tableau
MIN_ARR	Recherche de la valeur minimum dans un tableau
OCCUR_ARR	Nombre d'occurrences d'une valeur dans un tableau
SORT_ARR	Tri par ordre croissant ou décroissant d'un tableau
ROL_ARR	Décalage circulaire à gauche d'un tableau
ROR_ARR	Décalage circulaire à droite d'un tableau
LENGTH_ARR	Calcul de longueur d'un tableau

---

**Instructions sur chaînes de caractères**

Les instructions suivantes s'appliquent sur des chaînes de caractères.

Désignation	Fonction
STRING_TO_INT	Conversion ASCII í Binaire (mot simple format)
STRING_TO_DINT	Conversion ASCII í Binaire (mot double format)
INT_TO_STRING	Conversion Binaire í (mot simple format) ASCII
DINT_TO_STRING	Conversion Binaire í (mot double format) ASCII
STRING_TO_REAL	Conversion ASCII í Flottant
REAL_TO_STRING	Conversion Flottant í ASCII
<, >, <=, >=, =, <>	Comparaison alphanumérique
FIND	Position d'une sous-chaîne
EQUAL_STR	Position du premier caractère différent
LEN	Longueur d'une chaîne de caractères
MID	Extraction d'une sous-chaîne
INSERT	Insertion d'une sous-chaîne
DELETE	Suppression d'une sous-chaîne
CONCAT	Concaténation de deux chaînes
REPLACE	Remplacement d'une chaîne
LEFT	Début de chaîne
RIGHT	Fin de chaîne

## Instructions de conversions numériques

---

### Instructions de conversions numériques

Les instructions réalisent des conversions de bits, mots, double mots et flottants.

Désignation	Fonction
BCD_TO_INT	Conversion BCD à Binaire
INT_TO_BCD	Conversion Binaire à BCD
GRAY_TO_INT	Conversion Gray à Binaire
INT_TO_REAL	Conversion d'un entier simple format en flottant
DINT_TO_REAL	Conversion d'un entier double format en flottant
REAL_TO_INT	Conversion d'un flottant en entier simple format
REAL_TO_DINT	Conversion d'un flottant en entier double format
DBCD_TO_DINT	Conversion d'un nombre BCD 32 bits en entier 32 bits
DINT_TO_DBCD	Conversion d'un entier 32 bits en nombre BCD 32 bits
DBCD_TO_INT	Conversion d'un nombre BCD 32 bits en entier 16 bits
INT_TO_DBCD	Conversion d'un entier 16 bits en nombre BCD 32 bits
LW	Extraction du mot de poids faible d'un double mot
HW	Extraction du mot de poids fort d'un double mot
CONCATW	Concaténation de 2 mots simples

---

## Instructions sur programme et instructions spécifiques

### Instructions sur programme

Les instructions suivantes n'agissent pas sur des objets du langage mais sur le déroulement du programme.

Désignation	Fonction
HALT	Arrêt de l'exécution du programme
JUMP	Saut à une étiquette
SRi	Appel de sous-programme
RETURN	Retour de sous-programme
MASKEVT	Masquage des événements dans l'automate
UNMASKEVT	Démasquage des événements dans l'automate

### Instructions de gestion du temps

Les instructions suivantes effectuent des opérations sur les dates, heures et sur les durées.

Désignation	Fonction
SCHEDULE	Fonction hordateur
RRTC	Lecture date système
WRTC	Mise à jour de la date système
PTC	Lecture date et code arrêt
ADD_TOD	Ajout d'une durée à une heure du jour
ADD_DT	Ajout d'une durée à une date et heure
DELTA_TOD	Mesure d'écart entre heures du jour
DELTA_D	Mesure d'écart entre dates (sans heure)
DELTA_DT	Mesure d'écart entre dates (avec heure)
SUB_TOD	Remontée dans le temps sur heure
SUB_DT	Remontée dans le temps sur date et heure
DAY_OF_WEEK	Lecture du jour courant de la semaine
TRANS_TIME	Conversion durée en date
DATE_TO_STRING	Conversion d'une date en chaîne de caractères
TOD_TO_STRING	Conversion d'une heure en chaîne de caractères
DT_TO_STRING	Conversion d'une date complète en chaîne de caractères
TIME_TO_STRING	Conversion d'une durée en chaîne de caractères

**Instructions  
"Orphée"**

Les instructions suivantes sont des instructions spécifiques du langage Orphée.

Désignation	Fonction
WSHL_RBIT, DSHL_RBIT	Décalage à gauche sur mot avec récupération des bits décalés
WSHR_RBIT, DSHR_RBIT	Décalage à droite sur mot avec extension de signe et récupération des bits décalés
WSHRZ_C, DSHRZ_C	Décalage à droite sur mot avec remplissage par 0 et récupération des bits décalés
SCOUNT	Comptage/décomptage avec signalisation de dépassement
ROLW, ROLD	Décalage circulaire gauche
RORW, RORD	Décalage circulaire droite

**Instructions de  
temporisation**

Ces instructions sont des fonctions de temporisation destinées à être utilisées pour la programmation du code des DFB.

Désignation	Fonction
FTON	Temporisation à l'enclenchement
FTOF	Temporisation au déclenchement
FTP	Temporisation d'impulsion
FPULSOR	Générateur de signaux rectangulaires

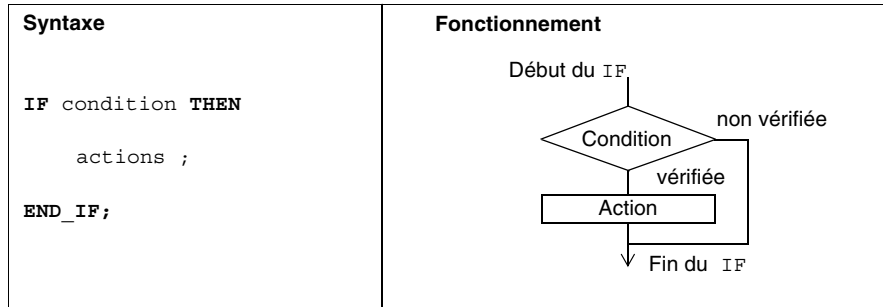
## Structure de contrôle conditionnelle IF...THEN

### Rôle

Cette structure de contrôle réalise une ou plusieurs actions si une condition est vraie. Dans sa forme générale les conditions peuvent être multiples.

### Forme simple

Dans sa forme simple, la structure de contrôle a la syntaxe et le fonctionnement suivant.



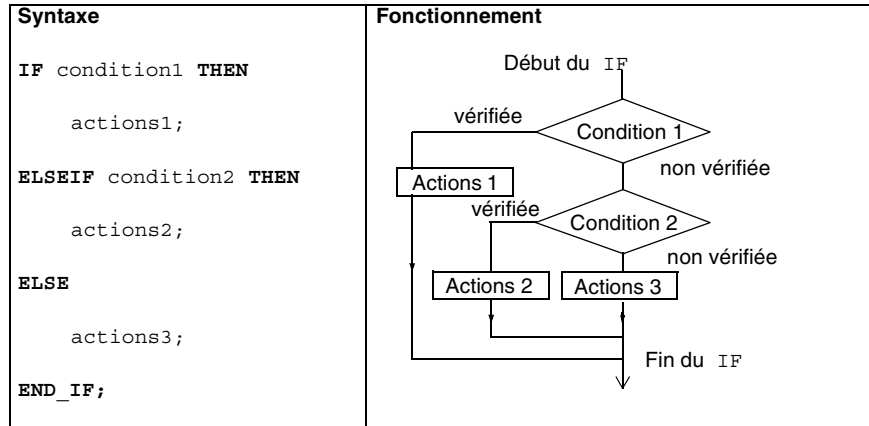
Exemple :

```

! (*Action conditionnelle IF (forme simple)*)
IF %M0 AND %M12 THEN
    RESET %M0;
    INC %MW4;
    %MW10:=%MW8+%MW9;
END_IF;

```

**Forme générale** Dans sa forme générale, la structure de contrôle a la syntaxe et le fonctionnement suivant.



Exemple :

```

! (*Action conditionnelle IF (forme simple)*)
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;

```

### Règle de programmation

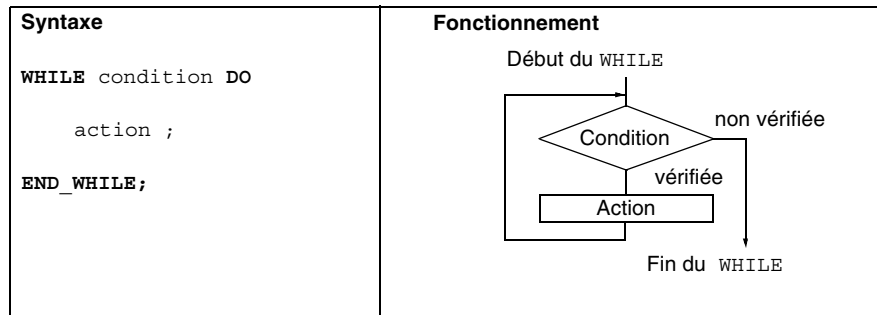
- Les conditions peuvent être multiples.
- Chaque action représente une liste d'instructions.
- Plusieurs structures de contrôle IF peuvent être imbriquées.
- Le nombre de ELSIF est illimité.
- Il y a au maximum une partie ELSE



## Structure de contrôle conditionnelle WHILE...END\_WHILE

**Rôle** Cette structure de contrôle réalise une action répétitive tant qu'une condition est vérifiée.

**Description** La structure de contrôle a la syntaxe et le fonctionnement suivant.



Exemple :

```

! (*Action itérative conditionnelle WHILE*)
WHILE %MW4<12 DO
    INC %MW4;
    SET %M25[%MW4];
END_WHILE;
          
```

### Règle de programmation

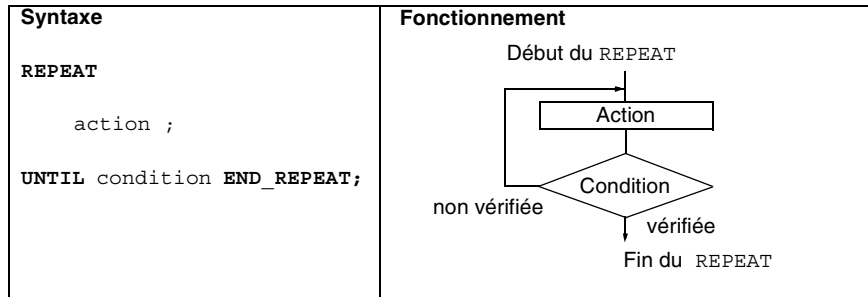
- La condition peut être multiple.
- L' action représente une liste d'instructions.
- Le test sur la condition est effectué avant d'exécuter l'action. Si lors de la première évaluation de la condition, sa valeur est fausse, alors l'action n'est jamais exécutée
- Plusieurs structures de contrôle WHILE peuvent être imbriquées.

**Note** : L'instruction EXIT (Voir *Rôle*, p. 156) permet d'arrêter l'exécution de la boucle et de continuer sur l'instruction suivant le END\_WHILE.

## Structure de contrôle conditionnelle REPEAT...END\_REPEAT

**Rôle** Cette structure de contrôle réalise une action répétitive jusqu'à ce qu'une condition soit vérifiée.

**Description** La structure de contrôle a la syntaxe et le fonctionnement suivant :



Exemple :

```

! (*Action itérative conditionnelle REPEAT*)
REPEAT
    INC %MW4;
    SET %M25[%MW4];
UNTIL %MW4>12 END_REPEAT;

```

### Règle de programmation

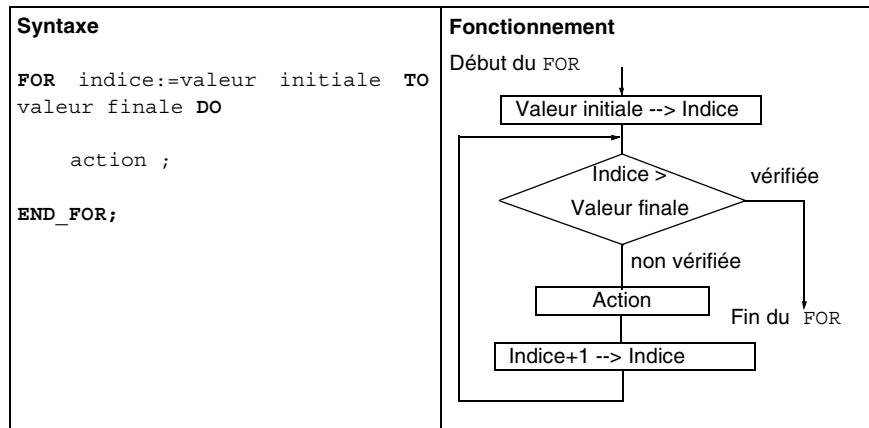
- La condition peut être multiple.
- L' action représente une liste d'instructions.
- Le test sur la condition est effectué après avoir exécuté l'action. Si lors de la première évaluation de la condition, sa valeur est fausse, alors l'action est exécutée une fois.
- Plusieurs structures de contrôle REPEAT peuvent être imbriquées.

**Note :** L'instruction EXIT (Voir *Rôle*, p. 156) permet d'arrêter l'exécution de la boucle et de continuer sur l'instruction suivant le END\_REPEAT.

## Structure de contrôle conditionnelle FOR...END\_FOR

**Rôle** Cette structure de contrôle réalise un traitement un certain nombre de fois en incrémentant de 1 un indice à chaque boucle.

**Description** La structure de contrôle a la syntaxe et le fonctionnement suivant.



Exemple :

```
! (*Action répétitive FOR*)
FOR %MW4=0 TO %MW23+12 DO
    SET %M25[%MW4];
END_FOR;
```

### Règle de programmation

- Lorsque l'indice est strictement supérieur à la valeur finale, l'exécution se poursuit sur l'instruction suivant le END\_FOR.
- L'incrément de l'indice est effectuée automatiquement et n'est donc pas à votre charge.
- L'action représente une liste d'instructions.
- La valeur initiale et la valeur finale sont forcément des expressions numériques de type mot.
- L'indice est forcément un objet de type mot accessible en écriture.
- Plusieurs structures de contrôle FOR peuvent être imbriquées.

**Note :** L'instruction EXIT (Voir *Rôle*, p. 156) permet d'arrêter l'exécution de la boucle et de continuer sur l'instruction suivant le END\_FOR.

---

## Instruction de sortie de boucle EXIT

---

**Rôle** Cette instruction permet d'arrêter l'exécution de la boucle et de continuer sur l'instruction suivant l'instruction de fin de boucle.

---

**Règle de programmation**

- Cette instruction n'est utilisable que dans les actions d'une des 3 boucles WHILE, REPEAT ou FOR.
  - Cette instruction est rattachée à la boucle englobante la plus proche, c'est-à-dire qu'elle n'arrête pas l'exécution de toutes les boucles qui l'englobent.
- 

**Exemple**

Dans cet exemple, l'instruction EXIT permet d'arrêter la boucle REPEAT mais en aucun cas la boucle WHILE.

```
! (*Instruction de sortie de boucle EXIT*)
  WHILE %MW1<124 DO
    %MW2:=0;
    %MW3:=%MW100 [%MW1];
    REPEAT
      %MW500 [%MW2] :=%MW3+%MW500 [%MW2];
      IF (%MW500 [%MW2]>32700) THEN
        EXIT;
      END_IF;
      INC %MW2;
    UNTIL %MW2>25 END_REPEAT;
    INC %MW1;
  END_WHILE;
```

---

## Règles d'exécution d'un programme littéral structuré

### Généralités

L'exécution d'un programme littéral s'effectue séquentiellement, instruction par instruction en respectant les structures de contrôle.

Dans le cas d'expressions arithmétiques ou booléennes composées de plusieurs opérateurs, des règles de priorité sont définies entre les différents opérateurs.

### Règle de priorité des opérateurs

Le tableau ci-dessous donne la priorité pour l'évaluation d'une expression du plus prioritaire ou moins prioritaire.

Opérateur	Symbole	Priorité
Parenthèses	(expression)	Plus forte
Complément logique Inversion - sur opérande + sur opérande	NOT NOT - +	
Multiplication Division Modulo	* / REM	
Addition Soustraction	+ -	
Comparaisons	<,>,<=,>=	
Comparaison égalité Comparaison inégalité	= <>	
ET logique ET booléen	AND AND	
OU exclusif logique OU exclusif booléen	XOR XOR	
OU logique OU booléen	OR OR	Moins forte

**Note :** Lorsqu'il y a conflit entre deux opérateurs de même niveau de priorité, c'est le premier opérateur qui l'emporte (l'évaluation se fait de la gauche vers la droite).

**Exemple 1**

Dans l'exemple ci-après, le NOT est appliqué sur le %MW3 puis le résultat est multiplié par 25. La somme de %MW10 et %MW12 est ensuite calculée, puis le ET logique entre le résultat de la multiplication et de l'addition.

```
NOT %MW3 * 25 AND %MW10 + %MW12
```

---

**Exemple 2**

Dans cet exemple, la multiplication de %MW34 par 2 est d'abord effectuée, puis le résultat est utilisé pour effectuer le modulo.

```
%MW34 * 2 REM 6
```

---

**Utilisation des parenthèses**

Les parenthèses sont utilisées pour modifier l'ordre d'évaluation des opérateurs, pour permettre par exemple de rendre une addition prioritaire sur une multiplication.

Vous pouvez imbriquer les parenthèses et le niveau d'imbrication n'est pas limité.

Les parenthèses peuvent également être utilisées afin d'éviter toute mauvaise interprétation du programme.

---

**Exemple 1**

Dans cet exemple, l'addition est d'abord effectuée puis la multiplication :

```
(%MW10+%MW11)*%MW12
```

---

**Exemple 2**

Cet exemple montre que les parenthèses peuvent être utilisées afin d'éviter toute mauvaise interprétation du programme.

```
NOT %MW2 <> %MW4 + %MW6
```

En utilisant les règles de priorité des opérateurs, l'interprétation est la suivante :

```
((NOT %MW2) <> (%MW4 + %MW6))
```

Mais vous pouvez penser que l'opération est la suivante :

```
NOT (%MW2 <> (%MW4 + %MW6))
```

Les parenthèses permettent donc de clarifier le programme.

---

## Conversions implicites

Les conversions implicites concernent les mots et les doubles mots.

Les opérateurs que vous utilisez dans les expressions arithmétiques, dans les comparaisons et l'opérateur affectation effectuent ces conversions implicites (qui ne sont donc pas à la charge de l'utilisateur).

Pour une instruction de la forme : <opérande 1> <opérateur> <opérande 2>, les cas possibles de conversions sont :

Opérande 1 de type	Opérande 2 de type	Conversion Opérande 1	Conversion Opérande 2	Opération dans le type
Mot	Mot	Non	Non	Mot
Mot	Double mot	Double mot	Non	Double mot
Double mot	Mot	Non	Double mot	Double mot
Double mot	Double mot	Non	Non	Double mot

Pour une affectation de la forme <opérande gauche> := <opérande droite>, c'est l'opérande de gauche qui impose le type attendu pour effectuer l'opération, ce qui signifie que l'opérande de droite doit être convertie si nécessaire suivant le tableau :

Type opérande gauche	Type opérande droite	Conversion opérande droite
Mot	Mot	Non
Mot	Double mot	Mot
Double mot	Mot	Double mot
Double mot	Double mot	Non

**Note** : Toute opération entre deux valeurs immédiates est effectuée en double longueur.





---

## Présentation

### Objet de ce chapitre

Ce chapitre décrit les règles de programmation en Grafcet.

### Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre	Sujet	Page
9.1	Présentation générale du Grafcet	162
9.2	Règle de construction du Grafcet	169
9.3	Programmation des actions et des conditions	178
9.4	Macro-étapes	187
9.5	Section Grafcet	192

---

## 9.1 Présentation générale du Grafcet

---

### Présentation

---

**Objet de ce sous-chapitre**

Ce sous-chapitre décrit les éléments de base d'un Grafcet.

---

**Contenu de ce sous-chapitre**

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Présentation du Grafcet	163
Description des symboles graphiques du Grafcet	164
Description des objets spécifiques au Grafcet	166
Possibilités du Grafcet	168

---

## Présentation du Grafcet

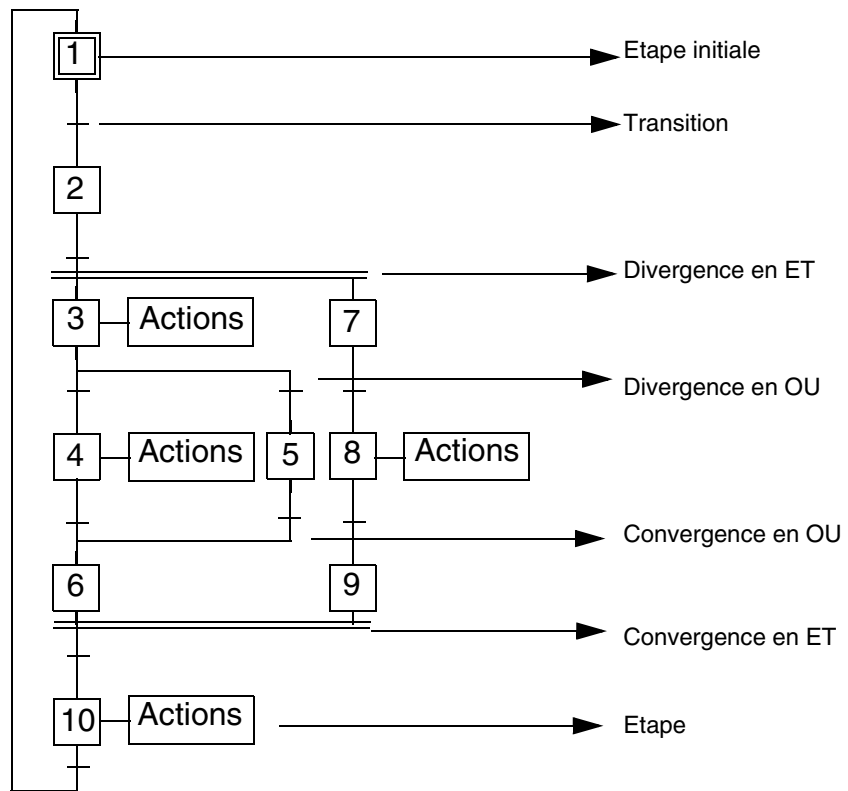
### Généralités

Le langage Grafcet est conforme au langage "Diagramme fonctionnel en séquence" (SFC) de la norme IEC 1131-3.

Le Grafcet permet de représenter graphiquement et de façon structurée le fonctionnement d'un automate séquentiel.

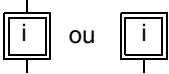
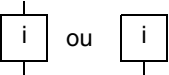

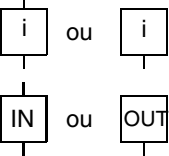

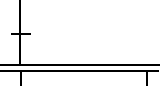
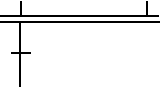
### Présentation

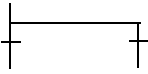
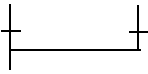



Cette description graphique du comportement séquentiel de l'automatisme et des différentes situations qui en découlent, s'effectue à l'aide de symboles graphiques simples.



## Description des symboles graphiques du Grafcet

**Description** Le tableau suivant décrit les éléments graphiques de base du Grafcet.

Désignation	Symbole	Fonctions
Étapes initiales		Symbolisent les étapes initiales actives en début de cycle après une initialisation ou une reprise à froid.
Étapes simples		Symbolisent un état stable de l'automatisme. Le nombre d'étapes maximum (y compris les étapes initiales) est configurable de : <ul style="list-style-type: none"> <li>• 1 à 96 pour un TSX 37-10,</li> <li>• 1 à 128 pour un TSX 37-20,</li> <li>• 1 à 250 pour un TSX 57.</li> </ul> Le nombre maximum d'étapes actives simultanément est configurable.
Macro-étapes		Symbolise une macro-étape : ensemble unique d'étapes et de transitions. Le nombre de macro-étapes maximum est configurable de 0 à 63 pour TSX 57 uniquement.
Étape de Macro-étapes		Symbolisent les étapes d'une macro-étapes. Le nombre maximum d'étapes pour chaque macro-étapes est configurable de 0 à 250 pour TSX 57.  Chaque macro-étape comporte une étape IN et OUT.
Transitions		Permettent le passage d'une étape à une autre. Une réceptivité associée à cette transition permet de définir les conditions logiques nécessaires au franchissement de cette transition. Le nombre de transitions maximum est de 1024, il n'est pas configurable. Le nombre maximum de transitions valides simultanément est configurable.
Divergences en ET		Transition d'une étape vers plusieurs étapes : permet l'activation simultanée de 11 étapes au maximum.
Convergences en ET		Transition de plusieurs étapes vers une seule : permet la désactivation simultanée de 11 étapes au maximum.

Désignation	Symbole	Fonctions
Divergences en OU		Transition d'une étape vers plusieurs étapes : permet de réaliser un aiguillage vers 11 étapes au maximum.
Convergences en OU		Transition de plusieurs étapes vers une seule : permet de réaliser une fin d'aiguillage venant de 11 étapes au maximum.
Renvois d'origine		"n" est le numéro de l'étape "d'où l'on vient" (étape d'origine).
Renvoi de destination		"n" est le numéro de l'étape "où l'on va" (étape de destination).
Liaisons orientées vers : <ul style="list-style-type: none"> <li>● le haut</li> <li>● le bas</li> <li>● la droite ou la gauche</li> </ul>		Ces liaisons permettent de réaliser un aiguillage, un saut d'étapes, une reprise d'étapes (séquence).

**Note :** le nombre maxi d'étapes (étapes du graphe principal + étapes de macro-étapes) dans la section Grafcet ne doit pas dépasser 1024 sur TSX 57.

## Description des objets spécifiques au Grafcet

### Généralités

Le Grafcet dispose d'objets bits associés aux étapes, de bits système spécifiques, d'objets mots indiquant le temps d'activité des étapes et de mots système spécifiques.

### Objets Grafcet

Le tableau suivant décrit l'ensemble des objets associés au Grafcet.

Désignation		Description
Bits associés aux étapes (1=étape active)	%Xi	Etat de l'étape i du Grafcet principal (i de 0 à n) (n dépend du processeur)
	%XMj	Etat de la macro-étape j (j de 0 à 63 pour TSX /PMX/PCX 57)
	%Xj.i	Etat de l'étape i de la macro-étape j
	%Xj.IN	Etat de l'étape d'entrée de la macro-étape j
	%Xj.OUT	Etat de l'étape de sortie de la macro-étape j
Bits système associés au Grafcet	%S21	Provoque l'initialisation du Grafcet
	%S22	Provoque la remise à zéro générale du Grafcet
	%S23	Provoque le figeage du Grafcet
	%S24	Provoque la remise à 0 de macro-étapes en fonction des mots système %SW22 à %SW25
	%S25	Mis à 1 sur : <ul style="list-style-type: none"> <li>débordement des tables (étapes/transition),</li> <li>exécution d'un graphe incorrect (renvoi de destination sur une étape qui n'appartient pas au graphe).</li> </ul>
Mots associés aux étapes	%Xi.T	Temps d'activité de l'étape i du Grafcet principal
	%Xj.i.T	Temps d'activité de l'étape i de la macro-étape j
	%Xj.IN.T	Temps d'activité de l'étape d'entrée de la macro-étape j
	%Xj.OUT.T	Temps d'activité de de l'étape de sortie de la macro-étape j
Mots système associés au Grafcet	%SW20	Mot permettant de connaître pour le cycle courant le nombre d'étapes actives, à activer et à désactiver.
	%SW21	Mot permettant de connaître pour le cycle courant le nombre de transitions valides, à valider ou à invalider.
	%SW22 à %SW25	Suite de 4 mots permettant de désigner les macro-étapes à remettre à 0 sur mise à 1 du bit %S24.

---

**Bits associés aux étapes**

Les bits associés aux étapes %Xi, aux macro-étapes %XMi, et aux étapes de macro-étapes %Xj.I , %Xj.IN et %Xj.OUT ont les propriétés suivantes :

- Ils sont à 1 lorsque les étapes sont actives.
- Ils peuvent être testés dans toutes les tâches, mais ne peuvent être écrits que dans le traitement préliminaire de la tâche maître (prépositionnement des graphes). Ces tests et actions sont programmés soit en langage à contacts, soit en langage liste d'instructions, soit en langage littéral.
- Ils sont indexables.

---

**Temps d'activité**

Les mots temps d'activité des étapes %Xi.T et des étapes de macro-étapes %Xj.I , %Xj.IN et %Xj.OUT ont les propriétés suivantes :

- Ils sont incrémentés toutes les 100 ms et prennent une valeur de 0 à 9999.
  - Incréméntation du mot : pendant l'activité de l'étape associée.
  - A la désactivation de l'étape, le contenu est figé.
  - A l'activation de l'étape, le contenu est remis à zéro puis incrémenté.
  - Le nombre de mots temps d'activité n'est pas configurable, un mot est réservé pour chaque étape.
  - Ces mots sont indexables.
-

## Possibilités du Grafcet

### Généralités

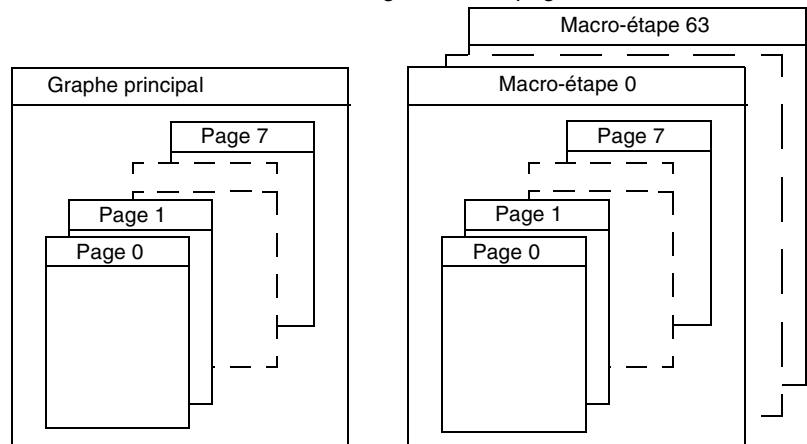
Le traitement séquentiel est structuré en :

- 1 sous ensemble : Graphe principal,
- 64 sous ensembles : Macro-étapes,

Ces sous-ensembles sont eux-mêmes divisés en 8 pages.

### Illustration

L'illustration suivante décrit la structure générale en page du Grafcet.



### Caractéristiques

Elles dépendent du processeur à programmer, elles sont récapitulées dans le tableau ci-dessous.

Nombre	TSX 37-10		TSX 37-20		TSX 57	
	Par défaut	Maxi	Par défaut	Maxi	Par défaut	Maxi
Étapes du Graphe principal	96	96	128	128	128	250
Macro-étapes	0	0	0	0	8	64
Étapes de macro-étapes	0	0	0	0	64	250
Total d'étapes	96	96	128	128	640	1024
Étapes actives simultanément	16	96	20	128	40	250
Transitions valides simultanément	20	192	24	256	48	400

Le nombre de transitions synchrones (ou nombre de convergences en ET) ne doit pas dépasser 64, le nombre total de transitions étant toujours de 1024.



---

## 9.2 Règle de construction du Grafcet

---

### Présentation

#### Objet de ce sous-chapitre

Ce sous- chapitre décrit les règles de base pour construire les graphes du Grafcet.

#### Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Représentation du Grafcet	170
Utilisation des divergences et convergences OU	171
Utilisation des divergences et convergences ET	172
Utilisation des renvois	173
Utilisation des liaisons orientées	176
Commentaire Grafcet	177

---

## Représentation du Grafcet

### Généralités

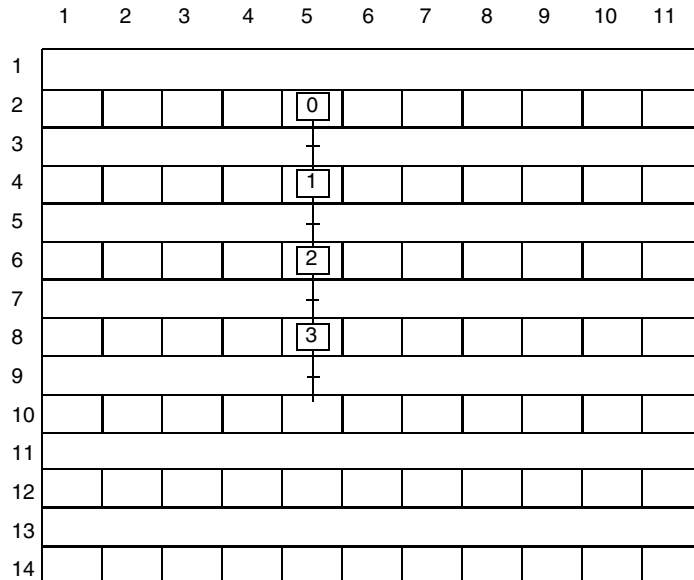
Le graphe principal et chacune des macro-étapes se programment sur 8 pages (page 0 à 7).

Une page Grafcet est constituée de 14 lignes et 11 colonnes qui définissent 154 cellules.

Dans chaque cellule, il est possible de saisir un élément graphique.

### Illustration

Le dessins ci-dessous illustre le découpage d'une page Grafcet.



### Règles d'écriture

- La première ligne permet de saisir des renvois d'origine.
- La dernière ligne permet de saisir des renvois de destination.
- Les lignes paires (de 2 à 12) sont des lignes d'étapes (pour les étapes renvois de destination).
- Les lignes impaires (de 3 à 13) sont des lignes de transitions (pour les transitions et les renvois d'origine).
- Chaque étape est repérée par un numéro différent (0 à 127) dans un ordre quelconque.
- Des graphes différents peuvent être représentés sur une même page.

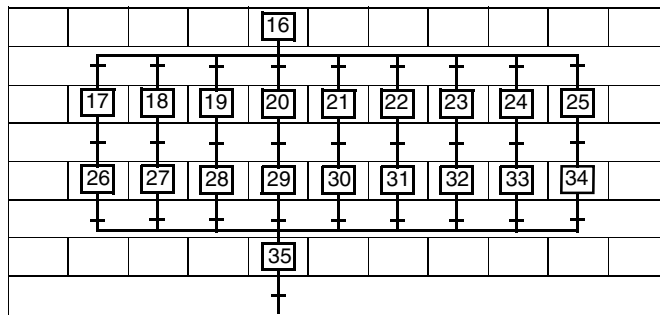
## Utilisation des divergences et convergences OU

**Rôle** Une divergence OU est un aiguillage d'une étape vers plusieurs étapes.

Une convergence OU réalise une fin d'aiguillage.

### Illustration

Le dessin ci-dessous présente une divergence OU de une étape vers 9 étapes et une convergence OU.



### Règle d'utilisation

- Le nombre de transitions en amont d'une fin d'aiguillage (convergence en OU) ou en aval d'un aiguillage (divergence en OU) ne doit pas dépasser 11.
- Un aiguillage peut se tracer vers la gauche ou vers la droite.
- Un aiguillage doit généralement se terminer par une fin d'aiguillage.
- Pour éviter de franchir simultanément plusieurs transitions, les réceptivités associées doivent être exclusives.

---

## Utilisation des divergences et convergences ET

---

**Rôle**

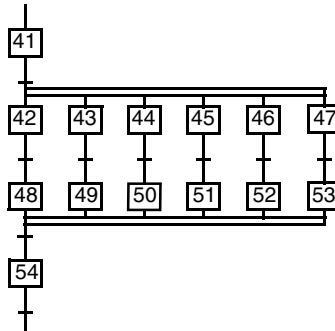
Une divergence ET permet l'activation simultanée de plusieurs étapes.

Une convergence ET permet la désactivation simultanée de plusieurs étapes.

---

**Illustration**

Le dessin ci-dessous présente une divergence et une convergence ET de 6 étapes.

**Règles d'utilisation**

- Le nombre d'étapes en aval d'une activation simultanée (divergence en ET) ou en amont d'une désactivation simultanée (convergence en ET) ne doit pas dépasser 11.
  - Une activation simultanée d'étapes doit généralement se terminer par une désactivation simultanée d'étapes.
  - L'activation simultanée est toujours représentée de la gauche vers la droite.
  - La désactivation simultanée est toujours représentée de la droite vers la gauche.
-

## Utilisation des renvois

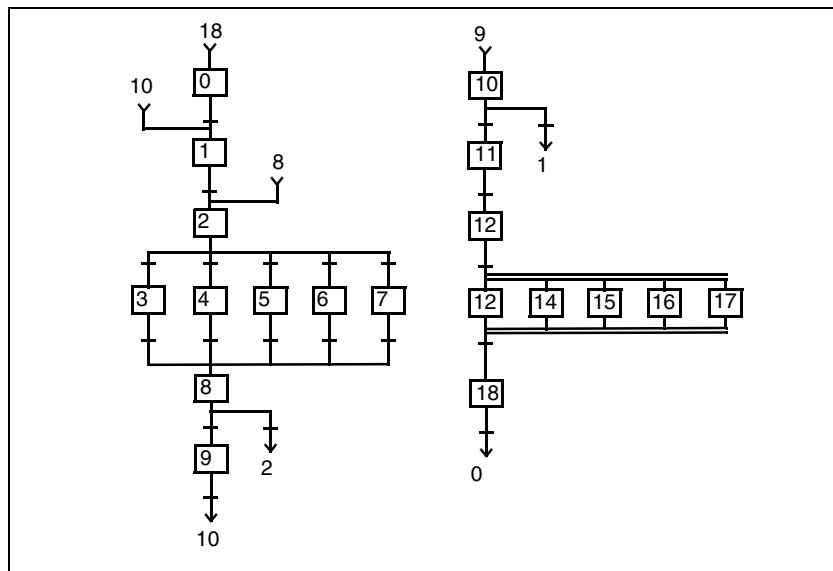
### Rôle

Les renvois assurent la continuité d'un Grafcet lorsque le tracé direct d'une liaison orientée ne peut être fait, soit au sein d'une page, soit entre deux pages consécutives ou non.

Cette continuité est assurée grâce à un renvoi de destination auquel correspond systématiquement un renvoi d'origine.

### Exemple

L'illustration suivante montre des exemples de renvois.

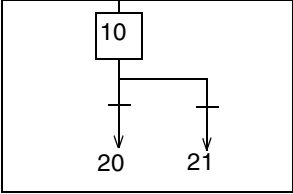
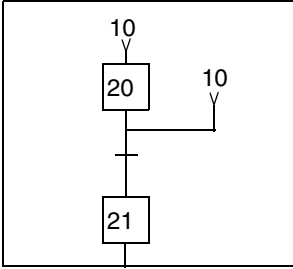
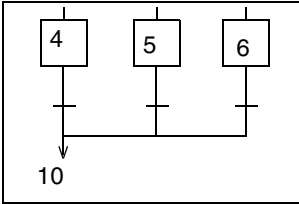
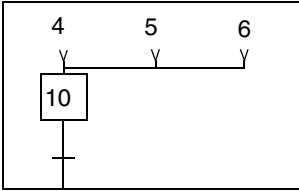


Le tableau ci-après explicite l'utilisation des renvois de l'exemple.

Utilisation	Exemple
Le rebouclage d'un graphe peut s'effectuer à l'aide de renvois.	Rebouclage de l'étape 18 vers l'étape 0.
Une reprise de séquence peut s'effectuer à l'aide de renvois.	Étape 10 vers étape 1 ou étape 8 vers étape 2.
Utilisation des renvois lorsqu'une branche de graphe est plus longue que la page.	Étape 9 vers étape 10.

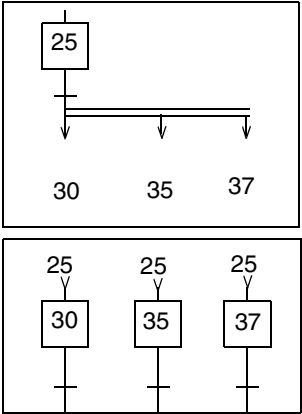
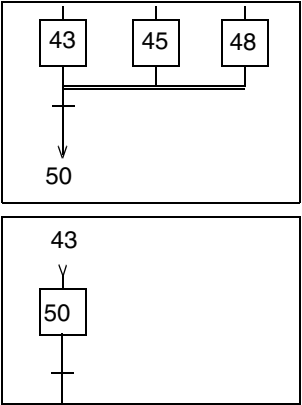
**Renvois utilisés dans les divergences et convergences OU**

Le tableau suivant donne les règles d'utilisation des renvois dans le cas de divergence ou convergence OU.

Règle	Illustration
Pour un aiguillage, les transitions et les renvois de destination doivent être saisis dans la même page.	 <p style="text-align: right;">Page 1</p>
Pour une fin d'aiguillage, les renvois d'origine doivent être saisis dans la même page que l'étape de destination.	 <p style="text-align: right;">Page 2</p>
Pour une fin d'aiguillage suivie d'un renvoi de destination, il doit y avoir autant de renvois d'origine qu'il n'y a d'étapes avant la fin d'aiguillage	<div style="display: flex; flex-direction: column; align-items: flex-end;"> <div style="display: flex; align-items: center; margin-bottom: 10px;">  <p style="margin-left: 10px;">Page 1</p> </div> <div style="display: flex; align-items: center;">  <p style="margin-left: 10px;">Page 2</p> </div> </div>

### Renvois utilisés dans les divergences et convergences ET

Le tableau suivant donne les règles d'utilisation des renvois dans le cas de divergence ou convergence ET.

Règle	Illustration
<p>Pour une activation simultanée d'étapes, les renvois de destination doivent se trouver sur la même page que l'étape et la transition de divergence</p>	 <p>Page 2</p> <p>Page 3</p>
<p>Pour une désactivation simultanée, les étapes et la transition de convergence doivent se trouver sur la même page que le renvoi de destination.</p> <p>Lorsque plusieurs étapes convergent vers une seule transition, le renvoi d'origine porte le numéro de l'étape amont la plus à gauche.</p>	 <p>Page 1</p> <p>Page 2</p>

---

## Utilisation des liaisons orientées

---

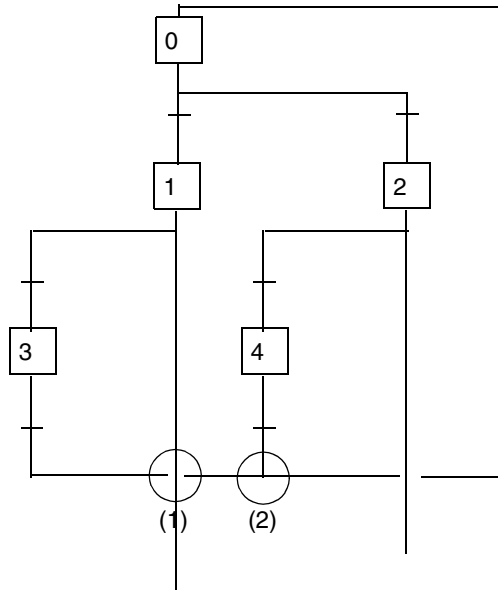
### Rôle

Les liaisons orientées relient une étape à une transition ou une transition à une étape. Elles peuvent être verticales ou horizontales.

---

### Illustration

Le schéma suivant présente un exemple d'utilisation d'une liaison orientée.



---

### Règles

Les liaisons orientées peuvent :

- se croiser (1), elles sont alors de natures différentes,
- se rencontrer (2), elles sont alors de même nature.

Le croisement d'une liaison avec une activation ou une désactivation simultanée d'étapes est impossible.

---



---

## Commentaire Grafcet

---

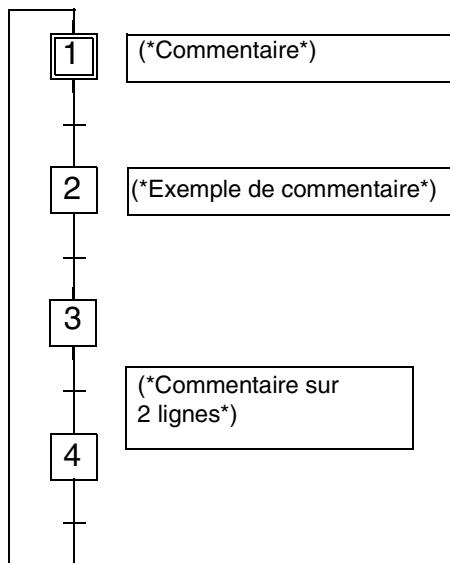
**Généralités** Les commentaires permettent de donner des informations sur les étapes et transitions d'un Grafcet. Ils sont optionnels.

---

**Syntaxe** Le texte du commentaire est encadré par (\*) à gauche et \*) à droite. Sa taille maximale est de 64 caractères.

---

**Illustration** L'illustration suivante présente des exemples de commentaires.



- Règles**
- Dans une page Grafcet, il est possible de saisir un commentaire dans n'importe quelle cellule.
  - Un commentaire occupe deux cellules contiguës sur deux lignes maximum. Si la zone d'affichage est trop petite, le commentaire est tronqué à l'affichage mais lors de l'impression de la documentation, le commentaire est présenté dans son intégralité.
  - Le commentaire saisi dans une page Grafcet est stocké dans les informations graphiques embarquées dans l'automate. A ce titre, ils consomment de la mémoire programme.
-

---

## 9.3 Programmation des actions et des conditions

---

### Présentation

**Objet de ce sous-chapitre**

Ce sous-chapitre décrit les règles de programmation des actions et conditions d'un Grafcet.

**Contenu de ce sous-chapitre**

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Programmation des actions associées aux étapes	179
Programmation des actions à l'activation ou à la désactivation	181
Programmation des actions continues	182
Programmation des réceptivités associées aux transitions	183
Programmation des réceptivités en langage à contacts	184
Programmation des réceptivités en langage liste d'instructions	185
Programmation des réceptivités en langage littéral structuré	186

---

---

## Programmation des actions associées aux étapes

---

### Généralités

Les actions associées aux étapes décrivent les ordres à transmettre à la partie opérative (processus à automatiser) ou à d'autres systèmes automatisés.

Les actions qui peuvent être programmées soit en langage à contacts, soit en langage liste d'instructions, soit en langage littéral structuré.

Ces actions ne sont scrutées que si l'étape à laquelle elles sont associées est active.

---

### 3 types d'actions

Le logiciel PL7 autorise trois types d'action :

- **les actions à l'activation** : actions exécutées une fois lorsque l'étape à laquelle elles sont associées passe de l'état inactif à l'état actif.
- **les actions à la désactivation** : actions exécutées une fois lorsque l'étape à laquelle elles sont associées passe de l'état actif à l'état inactif.
- **les actions continues** : ces actions sont exécutées tant que l'étape à laquelle elles sont associées est active.

<p><b>Note</b> : Une même action peut comprendre plusieurs éléments de programmation (phrases ou réseaux de contacts).</p>
--

---

### Repérage des actions

Ces actions sont repérées de la manière suivante :

MAST - <nom section Grafcet> - CHART (ou MACROK)- PAGE n %Xi x  
avec

x = P1 pour Activation, x = N1 Continue, x = P0 Désactivation

n = Numéro de la page

i = Numéro de l'étape

**Exemple** : MAST - Peinture - CHART - PAGE 0 %X1 P1 Action à l'activation de l'étape 1 de la page 0 de la section Peinture

---

**Règles  
d'utilisation**

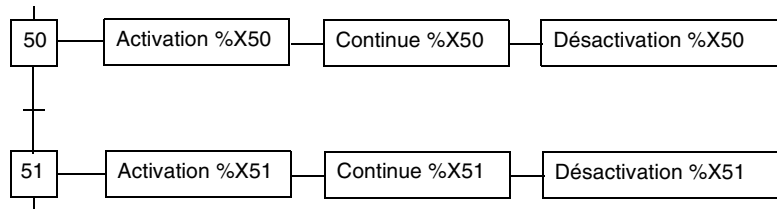
- Toutes les actions sont considérées comme des actions mémorisées, d'où : une action asservie à la durée de l'étape Xn doit être remise à zéro à la désactivation de l'étape Xn ou à l'activation de l'étape Xn+1.  
Une action à effet maintenu sur plusieurs étapes est positionnée à un à l'activation de l'étape Xn et remise à zéro à la désactivation de l'étape Xn+m.
- Toutes les actions peuvent être asservies à des conditions logiques, donc être conditionnelles.
- Les actions asservies à des sécurités indirectes doivent être programmées dans le traitement postérieur (Voir *Description du traitement postérieur*, p. 203) (traitement exécuté à chaque scrutation )

**Ordre  
d'exécution des  
actions**

Pour l'exemple suivant, sur un tour de cycle, l'ordre d'exécution des actions est le suivant. Lorsque l'étape 51 est activée, les actions sont exécutées dans l'ordre suivant :

1. actions à la désactivation de l'étape 50,
2. actions à l'activation de l'étape 51,
3. actions continues de l'étape 51.

Exemple :



Dès la désactivation de l'étape 51, les actions continues associées ne sont plus scrutées.

---

## Programmation des actions à l'activation ou à la désactivation

---

### Règles

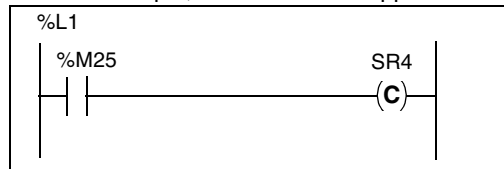
Ces actions sont exécutées une fois lorsque l'étape à laquelle elles sont associées passe de l'état inactif à l'état actif.

Ces actions sont impulsionnelles et sont exécutées sur **un seul tour de scrutation**. Elles permettent l'appel à un sous-programme, l'incrémement d'un compteur, etc.

---

### Exemple 1

Dans cet exemple, cette action fait appel à un sous-programme



### Exemple 2

Dans cet exemple, cette action incrémente le mot %MW10 et remet à 0 les mots %MW0 et %MW25.

```
%L1 :
INC %MW10 ; %MW0 := 0 ; %MW25 := 0 ;
```

---

## Programmation des actions continues

### Règles

Ces actions sont exécutées tant que l'étape à laquelle elles sont associées est active. Elles peuvent être :

- **Actions conditionnelles** : l'action est exécutée si une condition est remplie,
- **Actions temporisées** : c'est un cas particulier, le temps intervenant comme condition logique. Cet asservissement peut être réalisé simplement en testant le temps d'activité associé à l'étape.

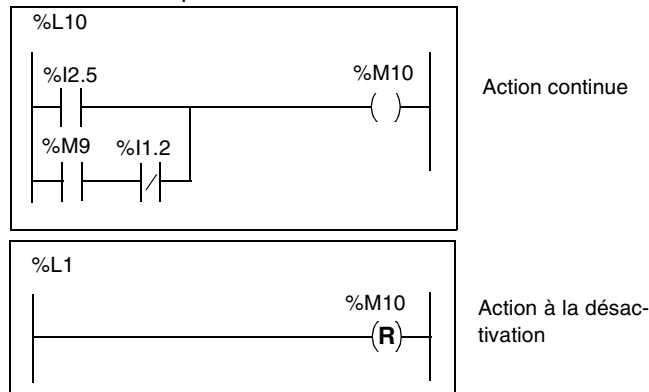
### Exemple d'action conditionnelle

Dans cet exemple, le bit %M10 est asservi à l'entrée %I2.5 ou au bit interne %M9 et à l'entrée %I1.2.

Tant que l'étape 2 est active et que ces conditions sont présentes, %M10 est positionné à 1.

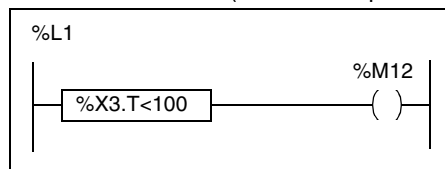
Le dernier état lu à la désactivation est mémorisé puisque les actions associées ne sont plus scrutées. Il est donc nécessaire de remettre à 0 le bit %M10, dans l'action à la désactivation de l'étape par exemple.

Illustration de l'exemple.



### Exemple d'action temporisée

Dans cet exemple, le bit %M12 est piloté tant que le temps d'activité de l'étape 3 est inférieur à 10 secondes (base de temps : 100 ms).



Ces actions peuvent également être inconditionnelles.

## Programmation des réceptivités associées aux transitions

### Généralités

Une réceptivité associée à une transition permet de définir les conditions logiques nécessaires au franchissement de cette transition.

Le nombre de transitions maximum est de 1024, il n'est pas configurable.  
Le nombre maximum de transitions valides simultanément est configurable.

### Règles

- A chaque transition est associée une réceptivité qui peut être programmée soit en langage à contacts, soit en langage liste d'instructions, soit en langage littéral.
- Une réceptivité n'est scrutée que si la transition à laquelle elle est associée est valide.
- Une réceptivité correspond à un réseau de contacts ou à une liste d'instructions ou à une expression littérale, comprenant une série de tests sur bits et/ou sur mot.
- Une réceptivité non programmée est une réceptivité toujours fausse.

### Repérage de la réceptivité

Les réceptivités sont repérées de la manière suivante :

```
MAST - <nom section Grafcet> - CHART (ou MACROk) - PAGE n %X(i)
--> % X(j) avec :
n = Numéro de la page
i = Numéro d'étape amont
j = Numéro d'étape aval
```

**Exemple :** MAST - Peinture - CHART - PAGE 0 %X(0) --> %X(1)  
Réceptivité associée à la transition entre l'étape 0 et l'étape 1 de la page 0 du graphe de la section Peinture.

**Note :** Lors d'une activation simultanée ou d'une désactivation simultanée d'étapes, le repère indiqué est celui de l'étape dans la colonne située le plus à gauche.

### Réceptivité utilisant le temps d'activité

Dans certaines applications, des actions sont pilotées sans contrôle d'information de retour (fin de course, détecteur, ...). La durée de l'étape est conditionnée par un temps, le langage PL7 permet d'utiliser le temps d'activité associé à chaque étape.

**Exemple :** ! X3.T>=150

Cette réceptivité programmée en langage littéral structuré permet de rester dans l'étape 3 pendant 15 secondes.

---

## Programmation des réceptivités en langage à contacts

---

### Règles de programmation

La réceptivité associée à la transition se programme sous la forme d'un réseau de contacts comprenant une zone test et une zone action.

La structure du réseau de contacts est similaire à celle d'un réseau programmé dans un module de programme.

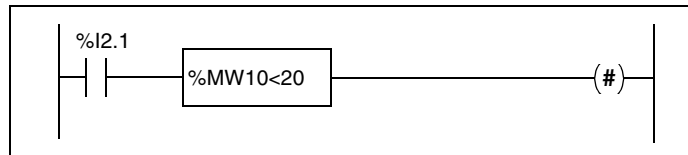
Seuls les éléments suivants peuvent être utilisés :

- **éléments graphiques de test** : contacts (%Mi, %I, %Q, %TMi.D ...), blocs comparaisons,
- **éléments graphiques d'action** : bobine "dièse" uniquement (les autres bobines n'étant pas significatives dans ce cas).

---

### Exemple

Cet exemple illustre la programmation d'une réceptivité en langage à contacts.





---

## Programmation des réceptivités en langage liste d'instructions

---

### Règles de programmation

La réceptivité associée à la transition se programme sous la forme d'une liste d'instructions comportant uniquement des instructions de test.

La liste d'instructions admises pour l'écriture d'une réceptivité diffère d'une liste d'instructions classique par :

- la structure générale : pas d'étiquette (%L).
- la liste des instructions :
  - pas d'instructions d'actions (objets bits, mots ou blocs fonctions),
  - pas de saut, d'appel de sous-programme.

---

### Exemple

Cet exemple illustre la programmation d'une réceptivité en langage liste d'instructions.

!	LD	%I2.1
	AND	[%MW10<20]

---

## Programmation des réceptivités en langage littéral structuré

---

### Règles de programmation

La réceptivité associée à la transition se programme sous la forme d'une expression booléenne ou d'une expression arithmétique ou d'une association des deux.

L'expression admise pour l'écriture d'une réceptivité diffère d'une ligne de programmation en langage littéral par :

- la structure générale :
  - pas d'étiquette (%L)
  - pas de phrase action, de phrase conditionnelle ou de phrase itérative.
- la liste des instructions :
  - pas d'action sur objet bit,
  - pas de saut, d'appel sous-programme,
  - pas de transfert, pas d'instruction d'action sur blocs.

### Exemple

Cet exemple illustre la programmation d'une réceptivité en langage littéral structuré.

```
! %I2.1 AND [%MW10<20]
```

---

---

## 9.4 Macro-étapes

---

### Présentation

---

#### Contenu de ce sous-chapitre

Ce sous-chapitre décrit la programmation des macro-étapes.

---

#### Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Présentation des macro-étapes	188
Constitution d'une macro-étape	189
Caractéristiques des macro-étapes	190

---

## Présentation des macro-étapes

### Généralités

Une macro-étape est une représentation condensée, unique, d'un ensemble d'étapes et de transitions.

Une macro-étape s'insère dans un graphe comme une étape et en respecte les règles d'évolution.

### Macro-représentation

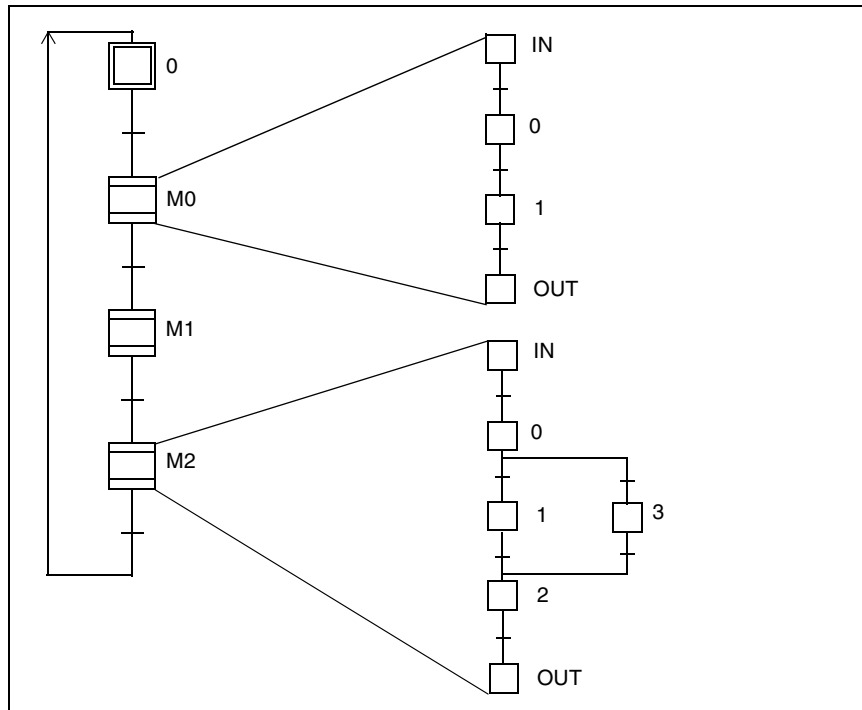
Un Grafcet de premier niveau décrivant l'enchaînement des séquences permet de mieux expliciter la structuration de la partie commande.

Chaque séquence est associée à une symbolisation particulière de l'étape : la macro-étape.

Cette notion de "macro-représentation" permet de hiérarchiser l'analyse. Chaque niveau peut être complété, modifié sans remettre en cause les autres niveaux.

Les macro-étapes sont disponibles pour les automates TSX57.

La figure suivante montre un Grafcet constitué de 3 macro-étapes.

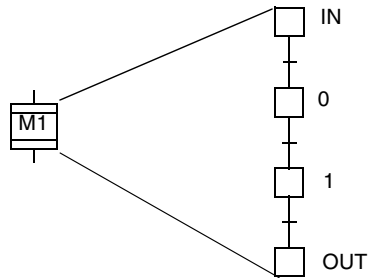


## Constitution d'une macro-étape

### Description

La symbolisation graphique d'une macro-étape se distingue d'une étape par deux traits horizontaux.

L'illustration suivante montre une macro-étape et son expansion.



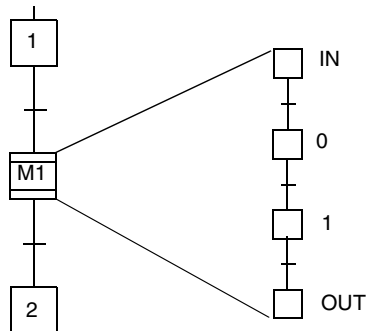
L'expansion d'une macro-étape est caractérisée par deux étapes spécifiques :

- une étape d'entrée répondant aux mêmes règles que les autres étapes,
- une étape de sortie ne pouvant avoir d'actions associées.

### Evolution

Lorsque la macro-étape est active, l'évolution de la macro-étape respecte les règles générales d'évolution d'un Grafcet).

Exemple :



La macro-étape M1 est activée quand l'étape IN est active et que sa réceptivité aval est vraie.

Elle est désactivée quand son étape de sortie est active et que la réceptivité M1>2 est vraie. L'étape 2 est alors activée.

## Caractéristiques des macro-étapes

### Caractéristiques générales

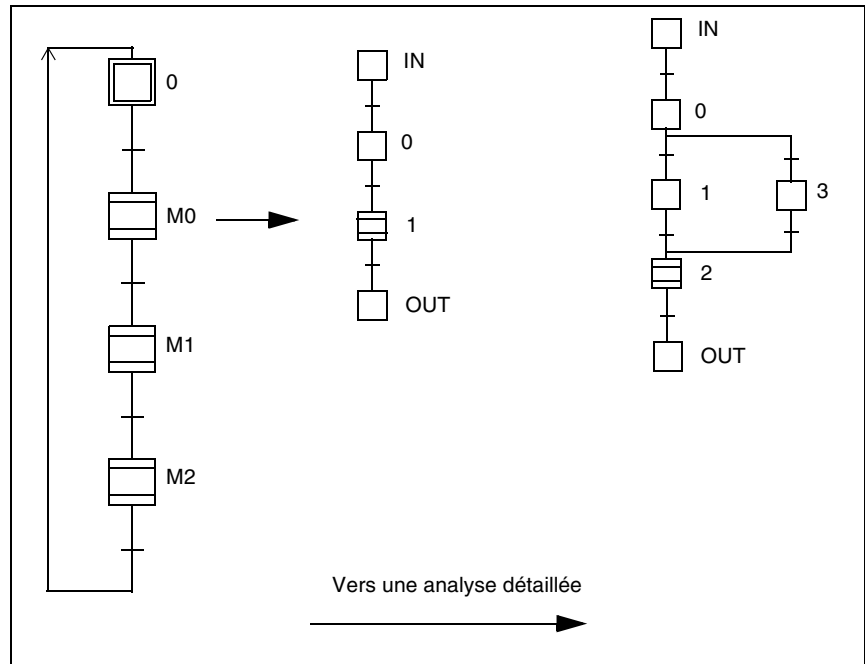
Le langage Grafcet PL7 autorise la programmation de 64 macro-étapes M0 à M63.

L'expansion d'une macro-étape, constituée d'une ou plusieurs séquences, est programmable au plus sur 8 pages et comprend au maximum 250 étapes plus l'étape IN et l'étape OUT.

Une macro-étape peut contenir une ou plusieurs macro-étapes. Cette hiérarchie est possible jusqu'à concurrence de 64 niveaux.

### Illustration

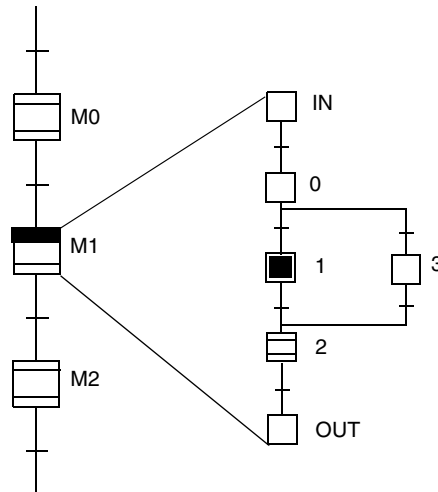
L'analyse d'une application peut être structurée de façon à fournir une approche globale puis détaillée des différentes opérations à réaliser.



**Étapes initiales**

L'expansion d'une macro-étape peut contenir une ou plusieurs étapes initiales. Ces étapes initiales sont activées à la mise sous tension ou lors d'une initialisation par programme. La macro-étape est alors visualisée à l'état actif.

Dans l'exemple ci-après l'étape initiale 1 de l'expansion est activée lors d'une initialisation du programme, la macro-étape est alors à l'état actif.



## 9.5 Section Grafcet

---

### Présentation

#### Contenu de ce sous-chapitre

Ce sous-chapitre présente la constitution d'une section Grafcet. Il décrit l'utilisation et les règles de programmation de chaque traitement.

---

#### Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Structure d'une section Grafcet	193
Description du traitement préliminaire	194
Prépositionnement du Grafcet	195
Initialisation du Grafcet	196
Remise à zéro du Grafcet	197
Figeage du Grafcet	198
Remise à zéro des macro-étapes	199
Fonctionnement du traitement séquentiel	201
Description du traitement postérieur	203

---



## Structure d'une section Grafcet

### Composition d'une section

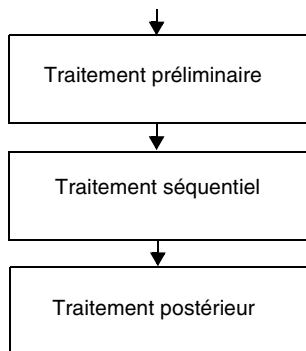
Une section de programme écrit en Grafcet comporte trois traitements consécutifs :

- le traitement préliminaire ,
- le traitement séquentiel,
- le traitement postérieur.

La section Grafcet se programme dans la tâche MAST.

### Illustration

Le dessin suivant illustre l'ordre de scrutation des traitements.



### Rôle des traitements

Le tableau suivant décrit le rôle de chacun des traitements et le langage avec lequel ils peuvent être programmés.

Traitement	Rôle	Langage
Préliminaire	Il permet de traiter : <ul style="list-style-type: none"> <li>● les initialisations sur reprise secteur ou défaillance,</li> <li>● les initialisations sur reprise secteur ou défaillance,</li> <li>● la logique d'entrée.</li> </ul>	Langage à contacts, liste d'instructions ou littéral
Séquentiel	Il permet de traiter l'ossature séquentielle de l'application et donne accès au traitement des réceptivités et des actions directement associées aux étapes.	Grafcet
Postérieur	Il permet de traiter : <ul style="list-style-type: none"> <li>● la logique de sortie,</li> <li>● la surveillance et les sécurités indirectes spécifiques aux sorties.</li> </ul>	Langage à contacts, liste d'instructions ou littéral

**Note :** Les macro-étapes sont exécutées dans leur ordre de scrutation dans le traitement séquentiel.

## Description du traitement préliminaire

---

### Généralités

Saisi en langage à contacts, en langage liste d'instructions ou en langage littéral, le traitement préliminaire est scruté dans sa totalité du haut vers le bas.

Exécuté avant les traitements séquentiel et postérieur, il permet de traiter tous les événements ayant une influence sur ces derniers :

- gestion des reprises secteur et réinitialisations,
- remise à zéro ou prépositionnement des graphes.

C'est donc uniquement dans le traitement préliminaire qu'il faut agir sur les bits associés aux étapes (mise à 0 ou à 1 des bits étapes %Xi ou %Xi.j par les instructions SET et RESET).

---

### Bits système

Les opérations de prépositionnement, initialisation, figeage... s'effectuent à l'aide des bits système %S21 à %S24.

Les bits système associés au Grafcet étant classés numériquement par ordre de priorité (%S21 à %S24), lorsque plusieurs d'entre eux sont simultanément mis à 1 dans le traitement préliminaire, ils sont traités un par un dans un ordre croissant (un seul est effectif par tour de scrutation).

Ces bits sont effectifs au début du traitement séquentiel.

---

### Traitement des reprises à froid

Sur une nouvelle application, ou sur une perte de contexte système, le système effectue un démarrage à froid.

Le bit %S21 est mis à 1 par le système avant l'appel du traitement préliminaire et le Grafcet est positionné sur les étapes initiales.

Si vous désirez un traitement particulier vis-à-vis de l'application en cas de démarrage à froid, il a la possibilité de tester %S0 qui reste à 1 durant le premier cycle de la tâche maître (MAST).

---

### Traitement des reprises à chaud

Suite à une coupure secteur sans changement d'application, le système effectue une reprise à chaud, il repart dans l'état qui précédait la coupure secteur.

Si vous désirez un traitement particulier vis-à-vis de l'application en cas de reprise à chaud, vous avez la possibilité de tester %S1 dans le traitement préliminaire, et d'appeler le programme correspondant.

---

## Prépositionnement du Grafcet

### Rôle

Le prépositionnement du Grafcet peut être utilisé lors du passage d'un fonctionnement en marche normale en marche spécifique ou à l'apparition d'un incident (exemple : défaut provoquant une marche dégradée).

Cette opération intervient sur le déroulement normal du cycle de l'application, elle doit donc être effectuée avec précaution.

### Prépositionnement du Grafcet

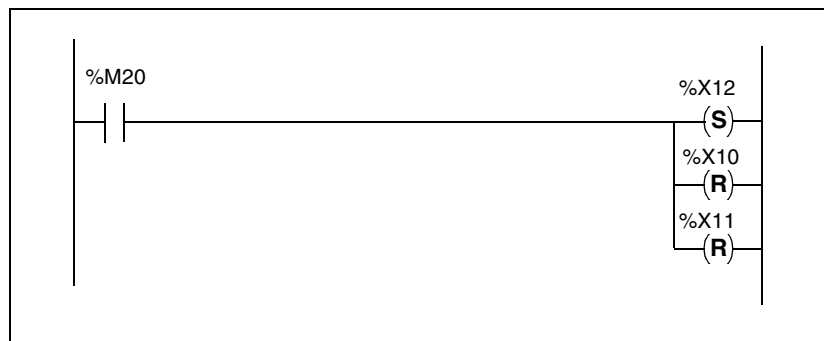
Le positionnement peut porter sur l'ensemble ou sur une partie du traitement séquentiel :

- en utilisant les instructions SET, RESET,
- par remise à zéro générale (%S22) puis, dans le cycle suivant, positionnement à 1 des étapes.

**Note :** Dans le cas de la remise à zéro d'une étape, les actions à la désactivation de celle-ci ne sont pas exécutées.

### Exemple

Dans cet exemple la mise à 1 du bit %M20 provoque le prépositionnement des étapes %X12 à 1, des étapes %X10 et %X11 à 0.



## Initialisation du Grafcet

---

**Rôle**

L'initialisation du Grafcet s'effectue par le bit système %S21.  
Normalement à l'état 0, la mise à l'état 1 de %S21 provoque :

- la désactivation des étapes actives,
- l'activation des étapes initiales.

---

**Initialisation du Grafcet**

Le tableau suivant donne les différentes possibilités de mise à 1 et à 0 du bit système %S21.

Mis à l'état 1	Remis à l'état 0
<ul style="list-style-type: none"><li>● Par mise à l'état 1 de %S0</li><li>● Par le programme utilisateur</li><li>● Par le terminal (en mise au point ou table d'animation)</li></ul>	<ul style="list-style-type: none"><li>● Par le système au début du traitement</li><li>● Par le programme utilisateur</li><li>● Par le terminal (en mise au point ou table d'animation)</li></ul>

---

**Règle d'utilisation**

Lorsqu'il est géré par le programme utilisateur, %S21 doit être positionné à 0 ou 1 dans le traitement préliminaire.

---

## Remise à zéro du Grafcet

---

### Rôle

La remise à 0 du Grafcet s'effectue par le bit système %S22.

Normalement à l'état 0, la mise à l'état 1 de %S22 provoque la désactivation des étapes actives de l'ensemble du traitement séquentiel.

**Note :** la fonction RESET\_XIT permet de réinitialiser par programme les temps d'activation de toutes les étapes du traitement séquentiel .

### Remise à zéro du Grafcet

Le tableau suivant donne les différentes possibilités de mise à 1 et à 0 du bit système %S22.

Mis à l'état 1	Remis à l'état 0
<ul style="list-style-type: none"> <li>● Par le programme utilisateur</li> <li>● Par le terminal (en mise au point ou table d'animation)</li> </ul>	<ul style="list-style-type: none"> <li>● Par le système à la fin du traitement séquentiel</li> </ul>

### Règle d'utilisation

- ce bit doit être écrit à 1 dans le traitement préliminaire,
- la remise à 0 de %S22 est géré par le système ; il est donc inutile de le remettre à 0 par programme ou par le terminal.

Pour redémarrer le traitement séquentiel dans une situation donnée, vous devez prévoir selon l'application une procédure d'initialisation ou de prépositionnement du Grafcet.

---

## Figeage du Grafcet

---

### Rôle

Le figeage du Grafcet s'effectue par le bit système %S23.

Normalement à l'état 0, la mise à l'état 1 de %S23 provoque le maintien en l'état des Grafcet. Quelle que soit la valeur des réceptivités aval aux étapes actives, les Grafcet n'évoluent pas. Le gel est maintenu tant que le bit %S23 est à 1.

---

### Figeage du Grafcet

Le tableau suivant donne les différentes possibilités de mise à 1 et à 0 du bit système %S23.

Mis à l'état 1	Remis à l'état 0
<ul style="list-style-type: none"><li>● Par le programme utilisateur,</li><li>● Par le terminal (en mise au point ou table d'animation).</li></ul>	<ul style="list-style-type: none"><li>● Par le programme utilisateur,</li><li>● Par le terminal (en mise au point ou table d'animation).</li></ul>

---

### Règle d'utilisation

- géré par le programme utilisateur, ce bit doit être positionné à 1 ou 0 dans le traitement préliminaire,
  - le bit %S23 associé aux bits %S21 et %S22 permet de réaliser un figeage du traitement séquentiel à l'état initial ou à l'état 0. De même le Grafcet peut être prépositionné puis figé par %S23.
-

## Remise à zéro des macro-étapes

---

### Rôle

La remise à zéro des macro-étapes s'effectue par le bit système %S24.

Normalement à l'état 0, la mise à l'état 1 de %S24 provoque la mise à zéro des macro-étapes choisies dans une table de 4 mots système (%SW22 à %SW25).

**Note :** la fonction RESET\_XIT permet de réinitialiser par programme les temps d'activation des étapes de macro-étape.

### Remise à zéro des macro-étapes

Le tableau suivant donne les différentes possibilités de mise à 1 et à 0 du bit système %S24.

Mis à l'état 1	Remis à l'état 0
<ul style="list-style-type: none"> <li>Par le programme utilisateur</li> </ul>	<ul style="list-style-type: none"> <li>Par le système au début du traitement</li> </ul>

### Règles d'utilisation

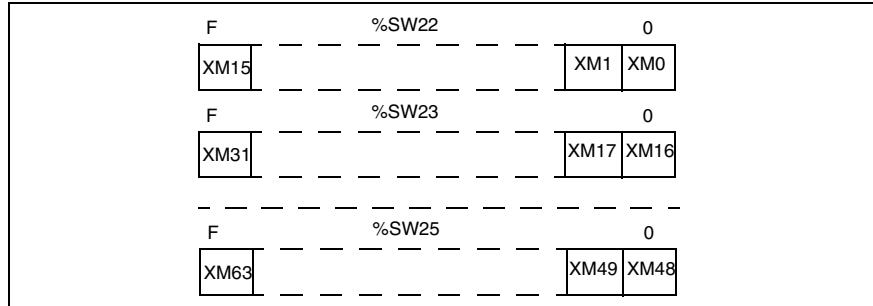
- ce bit doit être écrit à 1 uniquement dans le traitement préliminaire,
  - la mise à 0 de %S24 est gérée par le système, il est donc interdit de le remettre à 0 par programme ou par le terminal.
-

**Table de mots  
%SW22 à %SW25**

A chaque bit de cette table correspond une macro-étape. L'utilisation en est la suivante :

- chargement de la table des mots %SW22 à %SW25 (bit à mettre à 1 lorsque la macro-étape correspondante ne doit pas être mise à zéro),
- validation par %S24.

L'illustration suivante présente le codage des mots %SW22 à %SW25.


**Exemple :**

```
! IF %I4.2 AND %T3.D THEN
%SW22:=16#AF8F;
%SW23:=16#F3FF;
%SW24:=16#FFEF;
%SW25:=16#FFFF;
SET %S24
```

Ces quatre mots sont initialisés à 16#FFFF si %S21 = 1.



## Fonctionnement du traitement séquentiel

### Généralités

Ce traitement permet la programmation de l'ossature séquentielle de l'application. Le traitement séquentiel comprend :

- le graphe principal organisé en 8 pages,
- jusqu'à 64 macro-étapes de 8 pages chacune.

Dans le graphe principal, plusieurs Grafcet non connexes peuvent être programmés et se dérouler simultanément.

L'évolution du Grafcet s'effectue en 3 grandes phases.

### Phase 1

Le tableau suivant décrit les opérations réalisées lors de la première phase.

Phase	Description
1	Evaluation des réceptivités des transitions validées.
2	Demande de désactivation des étapes amont associées.
3	Demande d'activation des étapes aval concernées

### Phase 2

La phase 2 correspond à l'évolution de la situation du Grafcet en fonction des transitions franchies :

Phase	Description
1	Désactivation des étapes en amont des transitions franchies.
2	Activation des étapes en aval des transitions franchies.
3	Invalidation des transitions franchies.
4	Validation des transitions en aval des nouvelles étapes activées.  <b>Résultat</b> : Le système met à jour deux tables dédiées respectivement à l'activité des étapes et à la validité des transitions : <ul style="list-style-type: none"> <li>● la table d'activité des étapes mémorise, pour le cycle courant, les étapes actives, les étapes à activer et les étapes à désactiver,</li> <li>● la table de validité des transitions mémorise, pour le cycle courant, les transitions situées en aval des étapes concernées par la table précédente.</li> </ul>

**Phase3**

Les actions associées aux étapes actives sont exécutées dans l'ordre suivant :

Phase	Description
1	Actions à la désactivation des étapes à désactiver.
2	Actions à l'activation des étapes à activer.
3	Actions continues des étapes actives.

**Dépassement de capacités**

Le nombre d'éléments de la table d'activité des étapes et de la table de validité des transitions est configurable.

Le dépassement de la capacité de l'une ou l'autre entraîne :

- le passage en STOP de l'automate (arrêt de l'exécution de l'application),
- le passage à 1 du bit système %S26 (dépassement de capacité d'une des deux tables),
- le clignotement du voyant ERR de l'automate.

Le système met à disposition de l'utilisateur deux mots système :

- %SW20 : mot permettant de connaître pour le cycle courant, le nombre d'étapes actives, à activer et à désactiver
- %SW21 : mot permettant de connaître pour le cycle courant, le nombre de transitions valides, à valider ou à invalider

**Diagnostic**

En cas de défaut bloquant, les mots système %SW125 à %SW127 permettent de déterminer la nature du défaut.

%SW125	%SW126	%SW127	
DEF7	0	= 0	Dépassement de la table des étapes (étapes/transitions)
DEF7	= 0	0	Dépassement de la table des transitions
DEFE	N° étape	N° macro-étape ou 64 pour le graphe principal	Exécution de graphe incorrect (problème de transition avec renvoi de destination non résolu).

## Description du traitement postérieur

### Généralités

Saisi en langage à contacts, en langage liste d'instructions ou en langage littéral, le traitement postérieur est scruté de haut en bas.

Ce traitement est le dernier exécuté avant l'activation des sorties et permet de programmer la logique de sortie.

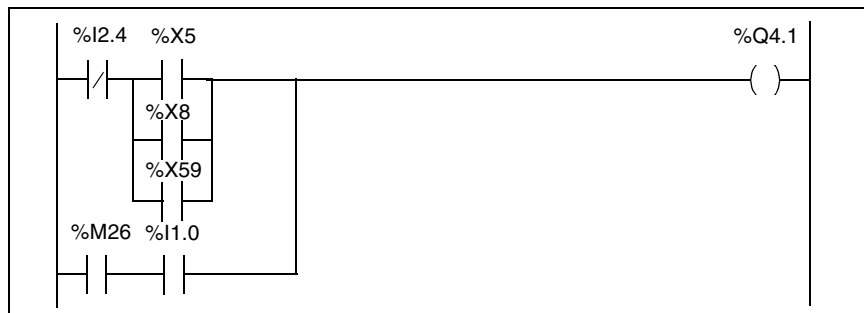
### Actions associées au Grafcet

Le traitement postérieur permet de compléter les consignes émises par le traitement séquentiel en intégrant à l'équation d'une sortie les modes de marche et d'arrêt et les sécurités indirectes spécifiques à l'action.

Il permet également de traiter une sortie activée plusieurs fois dans le traitement séquentiel.

**D'une manière générale, il est recommandé de programmer les actions agissant directement sur le process dans le traitement postérieur.**

### Exemple :



- I2.4 = sécurité indirecte spécifique au pilotage de la sortie %Q4.1.
- %M26 = bit interne résultat de la logique d'entrée traitant des modes de marche et d'arrêt.
- %I1.0 = bouton poussoir.

La sortie %Q4.1 est activée par les étapes 5, 8 et 59 du traitement séquentiel.

### Actions indépendantes du Grafcet

Le traitement postérieur permet également de programmer les sorties indépendantes du traitement séquentiel.

**Contrôle de l'exécution du Grafcet**

Il peut s'avérer nécessaire de contrôler le bon déroulement du Grafcet en testant le temps d'activité de certaines étapes. Le test de ce temps s'effectue par comparaison soit à une valeur minimum soit à une valeur maximum déterminée par l'utilisateur.

L'exploitation du défaut est laissée au choix de l'utilisateur (signalisation, procédure particulière de fonctionnement, édition de message).

Exemple :

```
! IF (%X2.T > 100 AND %X2) THEN SET %Q4.0 ;END_IF ;
```

---

---

# Blocs fonction DFB

# 10

---

## Présentation

### Objet de ce chapitre

Ce chapitre décrit la programmation des blocs fonction utilisateur DFB.

### Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Présentation des blocs fonction DFB	206
Comment mettre en oeuvre un bloc fonction DFB	207
Définition des objets des blocs fonction type DFB	209
Définition des paramètres DFB	211
Définition des variables DFB	212
Règle de codage des Types DFB	214
Création des instances de DFB	216
Règle d'utilisation des DFB dans un programme	217
Utilisation d'un DFB dans un programme en langage à contacts	218
Utilisation d'un DFB dans un programme en langage liste d'instructions ou littéral	219
Exécution d'une instance DFB	220
Exemple de programmation de bloc fonction DFB	221

---

## Présentation des blocs fonction DFB

---

### Rôle

Le logiciel PL7-Pro offre à l'utilisateur la possibilité de créer ses propres blocs fonction répondant aux spécificités de ses applications.

Ces blocs fonction utilisateur permettent de structurer une application. Ils seront utilisés dès qu'une séquence de programme se trouve répétée à plusieurs reprises dans l'application ou pour figer une programmation standard (exemple : algorithme de commande d'un moteur incluant la prise en compte des sécurités locales).

Ils peuvent être transmis à l'ensemble des programmeurs et être utilisés dans la même application ou dans toutes autres applications (fonction exportation/importation).

### Exemples d'utilisation

L'utilisation d'un bloc fonction DFB dans une application permet de :

- simplifier la conception et la saisie du programme,
- accroître la lisibilité du programme,
- faciliter sa mise au point (toutes les variables manipulées par le bloc fonction DFB sont identifiées sur son interface),
- diminuer le volume de code généré (le code correspondant au DFB n'étant chargé qu'une fois, quel que soit le nombre d'appels au DFB dans le programme).

### Comparaison avec les sous-programmes

Par rapport au sous programme, ils permettent :

- de paramétrer plus facilement le traitement,
- d'utiliser des variables internes propres au DFB donc indépendantes de l'application,
- d'être testés indépendamment de l'application.

Ils offrent en langage à contacts une visualisation graphique du bloc facilitant la programmation et la mise au point.

De plus les blocs fonction DFB exploitent des données rémanentes.

### Domaine d'utilisation

Le tableau ci-après décrit le domaine d'application des DFB.

Fonction	Domaine
Automates pour lesquels les DFB sont utilisables.	Premium
Logiciel de création des DFB.	PL7 Pro
Logiciels avec lesquels les DFB sont utilisables.	PL7 Pro ou un PL7 Junior
Langage de programmation pour la création du code des DFB.	langage littéral structuré et langage à contacts
Langage de programmation avec lesquels les DFB sont utilisables.	langage à contacts, littéral structuré et en liste d'instructions

## Comment mettre en oeuvre un bloc fonction DFB

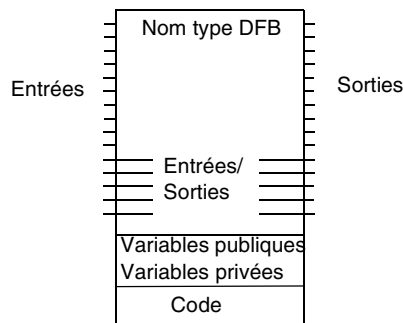
**Marche à suivre** La mise en oeuvre d'un bloc fonction DFB s'effectue en 3 étapes principales :

Etape	Action
1	Conception du DFB modèle (appelé : Type DFB).
2	Création d'une image de ce bloc appelée instance pour chaque utilisation dans l'application.
3	Utilisation de l'instance dans le programme PL7.

### Conception du type DFB

Consiste à définir et coder tous les éléments composant le DFB modèle, à l'aide de l'éditeur de DFB.

L'illustration suivante montre la composition d'un DFB modèle.



Un bloc fonction Type DFB se compose :

- d'un nom,
- de paramètres :
  - entrées,
  - sorties,
  - entrées/sorties,
- de variables :
  - variables publiques,
  - variables privées,
- du code en langage littéral structuré ou en langage à contacts,
- d'un commentaire,
- d'une fiche descriptive.

### Création d'une instance DFB

Une fois le Type DFB conçu, l'utilisateur définit une instance du DFB à l'aide de l'éditeur de variables ou lors de l'appel de la fonction dans l'éditeur de programme.

**Utilisation des  
DFB**

Cette instance du bloc s'utilise ensuite comme un bloc fonction standard en langage à contacts, ou comme une fonction élémentaire en langage littéral structuré ou liste d'instructions.

Elle peut se programmer dans les différentes tâches (excepté dans les tâches événementielles) et sections de l'application.

---



## Definition des objets des blocs fonction type DFB

### Caractéristiques générales des objets DFB

Ces objets sont des données internes au DFB, ils sont purement symboliques (pas d'adressage sous forme de repère).

Les DFB utilisent 2 types d'objet :

- les paramètres
- les variables

### Syntaxe

Pour chaque paramètre ou variable utilisé, le concepteur du bloc fonction Type DFB définit :

- un nom de 8 caractères maximum (sont autorisés les lettres non accentuées, les chiffres, le caractère "\_"; le premier caractère doit être une lettre; les mots clefs et les symboles sont interdits),
- un type d'objet (voir tableau ci-après),
- un commentaire optionnel de 80 caractères maximum,
- une valeur initiale (excepté pour les paramètres Entrées/Sorties).

### Type d'objets

Le tableau ci-après décrit la liste des différents types d'objets possibles lors de la déclaration des paramètres et des variables du type DFB.

Action sur...	Type	Nom	Exemples
Bits	BOOL	Booléen	Le type BOOL ne gère pas les fronts. Si la gestion de front n'est pas utile dans le traitement, il est préférable d'utiliser le type BOOL Exemple d'objet de type BOOL du langage PL7 : %MWi.Xj qui ne gère pas les fronts mais qui consomme moins de taille mémoire que le type EBOOL.
	EBOOL	Booléen étendu	Le type EBOOL gère les fronts, il est donc possible d'exécuter sur ce type de paramètre ou de variable des instructions sur front de type RE et FE. si vous souhaitez associer un type EBOOL à un paramètre d'entrées/sorties lors de l'utilisation, il doit être de type EBOOL dans le DFB. Exemple d'objet de type EBOOL du langage PL7 : %Mi,%Ixy.i,%Qxy.i.

Action sur...	Type	Nom	Exemples
Mots	WORD	Entier 16 bits	Exemple d'objet de type WORD du langage PL7 : %MWi, %KW <sub>i</sub> ,
	DWORD	Entier 32 bits	Exemple d'objet de type DWORD du langage PL7 : %MD <sub>i</sub> , %KD <sub>i</sub> ,
	REAL	Réel	Exemple d'objet de type REAL du langage PL7 : %MFi, %KF <sub>i</sub>
Tableaux	AR_X	Tableau de bits	Exemple d'objet de type AR_X du langage PL7 : %Mi:L, %Ix.i:L
	AR_W	Tableau d'entier 16 bits	Exemple d'objet de type AR_W du langage PL7 : %MWi:L, %KW <sub>i</sub> :L
	AR_D	Tableau d'entier 32 bits	Exemple d'objet de type AR_D du langage PL7 : %MD <sub>i</sub> :L, %KD <sub>i</sub> :L
	AR_R	Tableau de réels	Exemple d'objet de type AR_R du langage PL7 : %MFi:L, %KF <sub>i</sub> :L
	STRING	Chaîne de caractères	Exemple d'objet de type STRING du langage PL7: %MB <sub>i</sub> , %KB <sub>i</sub>

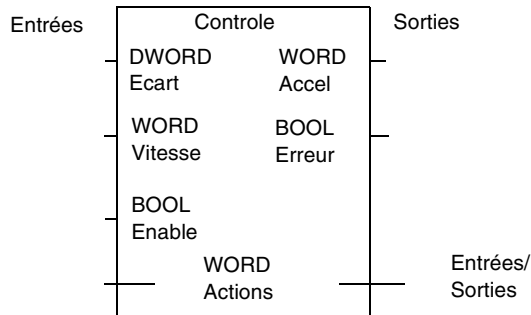
**Note :**

- Cas des tableaux : la longueur du tableau doit être obligatoirement mentionnée pour les paramètres sorties et les variables publiques et privées, par contre il n'est pas nécessaire de les définir pour les paramètres entrées et les paramètres entrées/sorties.
- Les valeurs initiales peuvent être définies pour les entrées (si elles ne sont pas de type tableau), pour les sorties et pour les variables publiques et privées.

## Définition des paramètres DFB

### Illustration

L'illustration suivante présente des exemples de paramètres



### Description des paramètres

Le tableau ci-après décrit le rôle de chaque type de paramètres.

Paramètre	Nombre maximum	Rôle
Entrées	15 (1)	Ce sont les données à fournir au DFB par le programme application. Ces paramètres en lecture seule ne peuvent pas être modifiés dans le code du DFB.
Sorties	15 (2)	Ce sont les données élaborées par le DFB à destination du programme application.
Entrées/ Sorties	15	Ce sont des paramètres d'entrées modifiables dans le code du DFB.

#### Légende :

- (1) Nombre d'entrées + Nombre d'entrées/sorties inférieur ou égal à 15
- (2) Nombre de sorties + Nombre d'entrées/sorties inférieur ou égal à 15

#### Note :

- Tout bloc DFB doit avoir au moins une entrée booléenne.
- La modification de l'interface d'un DFB (variables publiques ou paramètres) est possible uniquement s'il n'est pas instancié et utilisé dans l'application.

## Définition des variables DFB

---

### Description des variables

Le tableau ci-après décrit le rôle de chaque type de variables.

Variable	Nombre maximum	Rôle
Publique	100	Variables internes utilisées dans le traitement et accessibles par l'utilisateur en réglage ou par le programme application en dehors du code DFB (en tant que variable publique d'instance DFB, voir ci-dessous : Accès aux variables publiques).
Privée	100	Variables internes au code du bloc fonction, ces variables sont calculées et exploitées à l'intérieur même du DFB mais n'ont aucun lien avec l'extérieur du DFB. Ces variables sont utiles pour la programmation du bloc mais n'ont pas d'intérêt pour l'utilisateur du bloc (par exemple : variable intermédiaire de renvoi d'une expression combinatoire à l'autre, résultat d'un calcul intermédiaire...).

**Note** : la modification de l'interface d'un DFB (variables publiques ou paramètres) est possible uniquement s'il n'est pas instancié et utilisé dans l'application.

### Accès aux variables publiques

Seuls les paramètres de sorties et les variables publiques sont accessibles en tant qu'objets dans le programme application en dehors du corps du bloc fonction. Leur syntaxe est la suivante :

#### **Nom\_DFB.Nom\_paramètre**

Où **Nom\_DFB** est le nom donné à l'instance du DFB utilisé (32 caractères maximum)

et **Nom\_paramètre** est le nom donné au paramètre de sorties ou à la variable publique (8 caractères maximum).

**Exemple** : `Controle.Ecart` pour la sortie `Ecart` de l'instance DFB nommée `Controle`.

---

**Sauvegarde et restitution des variables publiques**

Les variables publiques, modifiées par programme ou par réglage, peuvent être sauvegardées en lieu et place des valeurs d'initialisation (définies dans les instances DFB) par mise à 1 du bit système %S94.

Le remplacement n'a lieu que si l'autorisation en a été donnée au niveau de chaque variable du type DFB.

Ces valeurs sauvegardées sont ré-appliquées par une mise à 1 du bit système %S95 ou sur une ré-initialisation de l'automate.

L'inhibition de la fonction "**Save/Restore**" globale pour tous les blocs fonction DFB est possible (boîte de dialogue **Propriétés du type DFB**).

---

## Règle de codage des Types DFB

---

### Généralités

Le code définit le traitement que doit effectuer le bloc DFB en fonction des paramètres déclarés.

Le code du bloc fonction DFB se programme en langage littéral ou en langage à contacts.

Dans le cas du langage littéral, le DFB est constitué d'une seule phrase littérale de longueur non limitée.

---

### Règles de programmation

Toutes les instructions et fonctions avancées du langage sont permises exceptés :

- l'appel aux blocs fonction standards,
- l'appel aux autres blocs fonction DFB,
- branchement à une étiquette `JUMP`,
- l'appel à sous-programme,
- l'instruction `HALT`,
- les instructions utilisant des variables de modules d'entrées/sorties (ex : `READ_STS`, `SMOVE...`).

Le code exploite les paramètres et les variables du DFB définies par l'utilisateur.

Le code du bloc fonction DFB ne peut utiliser ni les objets d'entrées/sorties (`%I,%Q...`), ni les objets globaux de l'application (`%MW,%KW...`) excepté les bits et mots système `%S` et `%SW`.

**Note :** pas d'utilisation possible d'étiquette.

---

### Fonctions spécifiques

Le tableau ci-après décrit les fonctions spécifiquement adaptées pour être utilisées dans le code.

Fonctions	Rôle
FTON, FTOF, FTP, FPULSOR	Ces fonctions de temporisation sont destinées à être utilisées à la place des blocs fonction temporisation standard.
LW, HW, COCATW	Ces instructions permettent de manipuler des mots et de doubles mots.
LENGTH_ARW, LENGTH_ARD, LENGTH_ARR	Ces instructions permettent de calculer les longueurs de tableau.

---

**Exemple de code** Le programme suivant donne l'exemple code en littéral.

```
CHR_200:=CHR_100;
CHR_114:=CHR_104;
CHR_116:=CHR_106;
RESET DEMARRE;
(*On incremente 80 fois CHR_100*)
FOR CHR_102:=1 TO 80 DO
    INC CHR_100;
    WHILE((CHR_104-CHR_114)<100)DO
        IF(CHR_104>400) THEN
EXIT;
            END_IF;
            INC CHR_104;
            REPEAT
                IF(CHR_106>300) THEN
EXIT;
                    END_IF;
                    INC CHR_106;
                    UNTIL ((CHR_100-CHR_116)>100)
                    END_REPEAT;
                    END_WHILE;
                    (* On boucle tant que CHR_106)
                    IF (CHR_106=CHR_116)
                    THEN EXIT;
                    ELSE
                        CHR_114:=CHR_104;
                        CHR_116:=CHR_106;
                        END_IF;
                        INC CHR_200;
END_FOR;
```

## Création des instances de DFB

---

### Généralités

Une instance DFB est une copie du Type DFB :

- elle exploite le code du Type DFB, (il n'y a pas duplication du code),
- elle crée une zone de données spécifique à cette instance, qui est la recopie des paramètres et des variables du Type DFB. Cette zone est située dans l'espace donnée de l'application.

Chaque instance DFB est repérée par un nom de 32 caractères maximum défini par l'utilisateur.

Les caractères permis sont identiques à ceux autorisés pour les symboles, c'est à dire sont autorisés :

- les lettres non accentuées,
- les chiffres,
- le caractère "\_".

Le premier caractère doit être une lettre; les mots clefs et les symboles sont interdits.

---

### Règles

Il est possible de créer autant d'instances souhaitées à partir d'un même type de DFB. Néanmoins le nombre d'instances de DFB ne doit pas utiliser plus de 32 K mots de la mémoire interne de l'automate.

Les valeurs initiales des variables publiques définies pour les blocs fonction Type DFB peuvent être modifiées pour chaque instance.

---



## Règle d'utilisation des DFB dans un programme

### Généralités

Les instances de DFB sont utilisables dans tous les langages (langages à contacts, littéral et liste d'instructions) et dans toutes parties de l'application : sections, sous programme, module Grafcet, (excepté dans les tâches événementielles).

### Règles générales d'utilisation

Les règles suivantes doivent être respectées quel que soit le langage utilisé :

- tous les paramètres d'entrées de type tableau ainsi que les paramètres d'entrées/sorties doivent être renseignés
- les paramètres d'entrées non câblés gardent la valeur du précédent appel ou la valeur d'initialisation si le bloc n'a jamais été appelé avec cette entrée renseignée ou câblée.
- tous les objets affectés aux paramètres d'entrées, de sorties et d'entrées/sorties doivent être obligatoirement de même types que ceux définis lors de la création du Type DFB (par exemple : si le type WORD est défini pour le paramètre d'entrée "vitesse", il n'est pas autorisé d'y affecter des doubles mots %MDi, %KD*i*).

Seule exception les types BOOL et EBOOL pour les paramètres d'entrées ou de sorties (pas pour les paramètres entrées/sorties) peuvent être mixés.

**Exemple** : le paramètre d'entrée "Validation" peut être défini en tant que BOOL et peut être associé à un bit interne %Mi qui est de type EBOOL, par contre dans le code interne du type DFB le paramètre d'entrée aura bien la propriété d'un type BOOL, il ne sait pas gérer les fronts.

### Affectation des paramètres

Le tableau suivant résume les différentes possibilités d'affectation des paramètres dans les différents langages de programmation :

Paramètre	Type	Affectation paramètre	Affectation
Entrées	Booléen	Câblé (1)	optionnelle (2)
	Numérique	Objet ou expression	optionnelle
	Tableau	Objet	obligatoire
Entrées/ Sorties	Booléen	Objet	obligatoire
	Numérique	Objet	obligatoire
	Tableau	Objet	obligatoire
Sorties	Booléen	Câblé (1)	optionnelle
	Numérique	Objet	optionnelle
	Tableau	Objet	optionnelle

(1) câblé en langage à contacts, ou objet en langage booléen ou littéral

(2) en langage à contacts tout bloc DFB doit avoir au moins une entrée booléenne (binaire) câblée.

## Utilisation d'un DFB dans un programme en langage à contacts

### Principe

Il y a deux possibilités pour appeler un bloc fonction DFB :

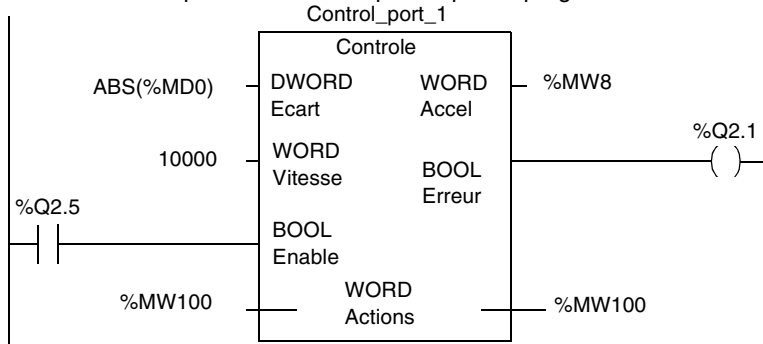
- un appel textuel dans un bloc opération, la syntaxe et les contraintes sur les paramètres sont identiques à celles du langage littéral.
- un appel graphique, voir exemple ci-après.

Les blocs fonctionnels DFB graphiques ont des entrées /sorties qui sont affectées directement par des objets ou des expressions, ces objets ou expressions occupent une cellule du réseau graphique.

2 blocs fonctionnels DFB connectés en série doivent être séparés d'au moins 2 colonnes

### Exemple

L'illustration suivante présente un exemple simple de programmation d'un DFB.



Le tableau ci-après repère les différents éléments du DFB.

Repère	Rôle
1	Nom du DFB
2	Nom du Type DFB
3	Paramètre effectif de la première entrée
4	Paramètres d'entrées (nom et type)
5	Paramètres de sorties (nom et type)
6	Paramètres d'entrées/sorties (nom et type)

#### Note :

- Un bloc fonction DFB doit avoir au moins une entrée booléenne câblée.
- Les entrées, sorties ou entrées/sorties numériques du bloc ne sont pas câblées. A ces broches sont associés des objets mentionnés sur la cellule face à la broche.

## Utilisation d'un DFB dans un programme en langage liste d'instructions ou littéral

**Généralités** L'appel du bloc fonction DFB constitue une action, qui peut être placée dans une phrase comme pour tout autre action du langage.

**Syntaxe générale** La syntaxe de programmation des DFB est la suivante :

Nom\_DFB (E1, ..., En, ES1, ..., ESn, S1, ..., Sn)

Le tableau suivant décrit le rôle des paramètres de l'instructions.

Paramètres	Rôle
E1, ..., En	Expressions (excepté pour les objets de type BOOL/EBOOL), objets ou valeurs immédiates servant de paramètres effectifs aux paramètres d'entrées.
ES1, ..., ESn	Paramètres effectifs correspondant aux entrées/sorties; ce sont toujours des objets langages en lecture/écriture.
S1, ..., Sn	Paramètres effectifs correspondant aux sorties; ce sont toujours des objets langages en lecture/écriture.

**Syntaxe en littéral** L'instruction en langage littéral a la syntaxe suivante :  
Nom\_DFB (E1, ..., En, ES1, ..., ESn, S1, ..., Sn);

**Exemple :** Cpt\_boulons(%I2.0,%MD10,%I2.1,%Q1.0);

**Syntaxe en liste d'instructions** L'instruction en langage liste d'instructions a la syntaxe suivante :  
[Nom\_DFB (E1, ..., En, ES1, ..., ESn, S1, ..., Sn)]

**Exemple :**  
LD TRUE  
[Cpt\_boulons(%I2.0,%MD10,%I2.1,%Q1.0)]

## Exécution d'une instance DFB

---

**Fonctionnement** L'exécution d'une instance DFB s'effectue dans l'ordre suivant :

Etape	Action
1	Chargement des paramètres d'entrées et d'entrées/sorties à l'aide des paramètres effectifs. Tout entrée laissée libre prend à l'initialisation ou sur reprise à froid la valeur d'initialisation définie dans le type DFB, elle prend ensuite la valeur courante du paramètre.
2	Passage par valeur des paramètres d'entrées (excepté pour le type tableau).
3	Passage par adresse des paramètres entrées/sorties.
4	Exécution du code littéral.
5	Ecriture des paramètres de sorties.

**Outils de mise au point**

Le logiciel PL7 offre plusieurs outils de mise au point du programme PL7 et des DFB :

- table d'animation : tous les paramètres et variables publiques sont affichés et animés en temps réel, il est possible de modifier et forcer les objets désirés,
  - point d'arrêt, pas à pas et diagnostic programme,
  - écrans d'exploitation : pour la mise au point unitaire.
-

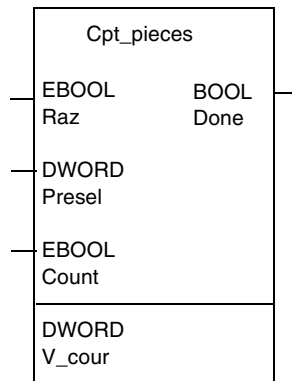
## Exemple de programmation de bloc fonction DFB

### Généralités

Cet exemple est donné à titre didactique, le DFB à programmer est un compteur.

### Caractéristiques du type DFB

Le compteur est réalisé à partir du Type DFB suivant :



Le tableau ci-après décrit les caractéristiques du Type DFB à programmer.

Caractéristiques	Valeurs
Nom	<b>Cpt_pieces</b>
Entrées	<ul style="list-style-type: none"> <li>● <b>Raz</b> : remise à zéro du compteur</li> <li>● <b>Presel</b> : valeur de présélection du compteur</li> <li>● <b>Count</b> : entrée comptage</li> </ul>
Sorties	<b>Done</b> : sortie valeur de présélection atteinte
Variable publique	<b>V_cour</b> : Valeur courante incrémentée par l'entrée <b>Count</b>

### Fonctionnement du compteur

Le tableau suivant décrit le fonctionnement que doit avoir le compteur.

Phase	Description
1	Ce bloc compte les fronts montants sur l'entrée <b>Count</b> .
2	Le résultat est placé dans la variable <b>V_cour</b> , cette valeur est remise à zéro par un front montant sur l'entrée <b>Raz</b> .
3	Le comptage s'effectue jusqu'à la valeur de présélection, lorsque cette valeur est atteinte la sortie <b>Done</b> est mise à 1, elle est remise à 0 sur front montant sur l'entrée <b>Raz</b> .

**Code du DFB**

La programmation du code du Type DFB est donnée ci-après.

```
!(*Programmation du DFB Cpt_pieces*)
IF RE Raz THEN
    V_cour:=0;
END_IF;
IF RE Count THEN
    V_cour:=V_cour+1;
END_IF;
IF (V_cour>=Presel) THEN
    SET Done;
ELSE
    RESET Done;
END_IF;
```

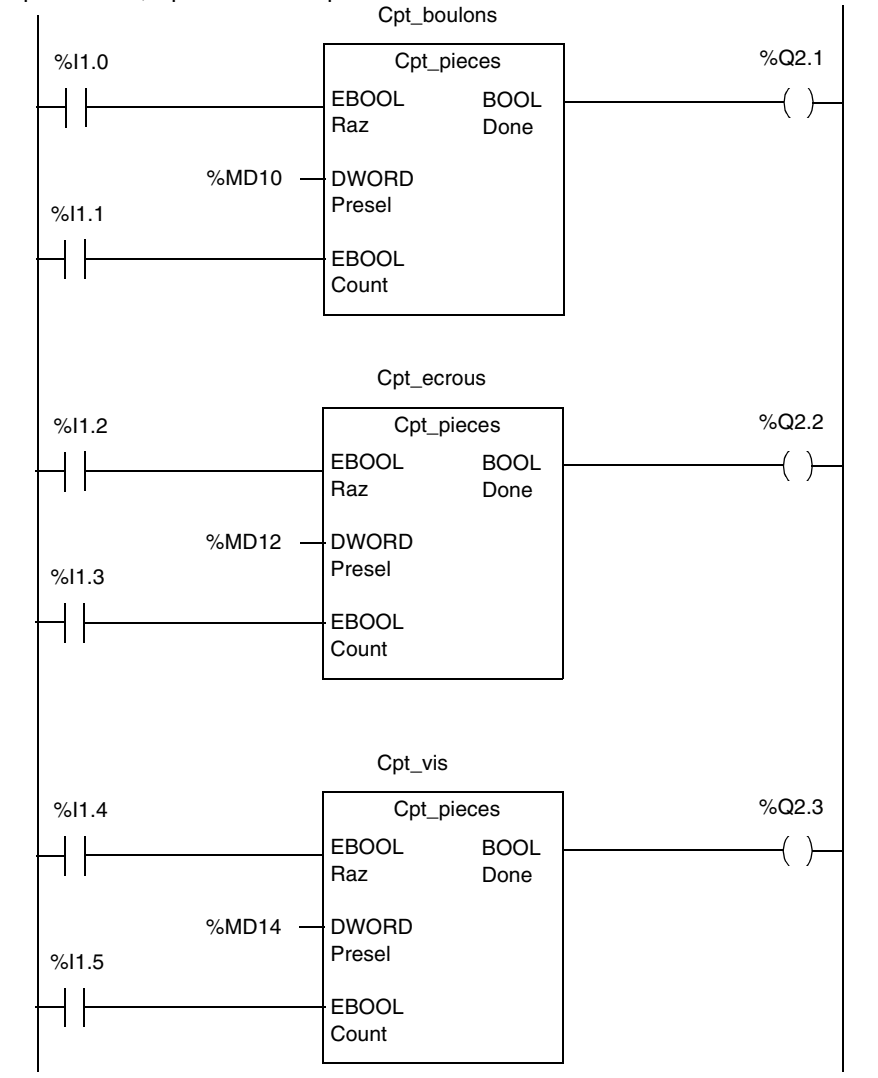
---

**Exemple  
d'utilisation**

Dans cet exemple le Type DFB créé, est utilisé 3 fois (3 instances DFB) pour le comptage de 3 types de pièces.

Lorsque le nombre de pièces programmé (dans les mots %MD10, %MD12, et %MD14) est atteint, la sortie du compteur pilote l'arrêt du système d'approvisionnement de pièces correspondant.

Le programme suivant utilise les 3 instances du type DFB Cpt\_pieces :  
 Cpt\_boulons, Cpt\_ecrous et Cpt\_vis.

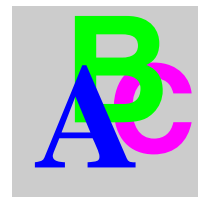






---

## Index



### A

Action, 179  
Action à l'activation, 181  
Action continue, 182  
Activation, 181  
Adressage  
    Bus AS-i, 39  
    Bus FIPIO, 36  
    E/S Micro, 31  
    Modules en rack, 34  
    Momentum, 36  
    TBX, 36  
Aiguillage, 171

### B

Bloc fonction DFB, 206

### C

Commentaire, 177  
    Grafcet, 177  
    Liste d'instructions, 125  
    littéral, 141  
    Réseau de contacts, 111  
Convergence ET, 172  
Convergence OU, 171  
Convergences en ET, 164, 165  
Coupure secteur, 72

### D

Démarrage à froid, 76  
DFB, 206  
Divergence ET, 172  
Divergence OU, 171  
Divergences en ET, 164, 165

### E

Eléments graphiques, 112  
Etape d'entrée, 189  
Etape de sortie, 189  
Etiquette  
    Liste d'instructions, 124  
    littéral, 140  
    Réseau de contacts, 110  
Exécution  
    Cyclique, 90  
    Périodique, 92  
    Réseau de contacts, 118  
Exécution d'un programme littéral, 157  
EXIT, 156

### F

Figeage du Grafcet, 198  
FOR...END\_FOR, 155

### G

Grafcet, 163

## I

- IF...THEN, 151
- Initialisation du Grafcet, 196
- Instance DFB, 216
- Instruction
  - arithmétique, 143
  - chaîne de caractères, 145
  - conversion, 148
  - Gestion du temps, 149
  - logique, 143
  - objet bits, 142
  - programme, 149
  - tableaux, 145
- Instructions
  - Liste d'instructions, 126

## L

- Langage
  - Littéral structuré, 138
- Langage à contacts, 108
- Langages
  - PL7, 17
- Liaison orientée, 176
- Liste d'instructions, 122
- Littéral structuré, 138
- Logiciel PL7, 16

## M

- Macro-étape, 188
- Mémoire
  - Bits, 59
  - Micro, 62
  - Mots, 61
  - Premium, 58, 64, 66, 68
  - TSX 37, 56, 62
  - TSX 57, 64, 66, 68
  - TSX Micro, 56
  - TSX57, 58
- Module fonctionnel, 22, 103
- Monotâche, 89
- Multitâche, 20, 97, 99

## O

- Objet
  - Bit, 29
  - Bloc fonction, 44
  - Booléen, 26
  - DFB, 209
  - Indexé, 48
  - Mot, 27, 41
- Objet langage P7, 25
- Objets Grafcet, 51, 166

## P

- Page Grafcet, 168, 170
- Paramètre
  - DFB, 211
- Phrase
  - Liste d'instructions, 123
  - littérale, 139
- PL7, 16
- Prépositionnement du Grafcet, 195
- Présymbolisation, 54
- Programmation
  - Réseau de contacts, 115

## R

- Réceptivité, 183
- Remise à zéro des macro-étapes, 199
- Remise à zéro du Grafcet, 197
- Renvoi de d'origine, 173
- Renvoi de destination, 173
- REPEAT...END\_REPEAT, 154
- Reprise à chaud, 74
- Reprise secteur, 72
- Réseau de contacts, 109

## S

- Section, 20, 82
- Section Grafcet, 193
- Sous-programme, 20, 82
- Symboles Grafcet, 164
- Symbolisation, 52

**T**

Tableau, 46

Tâche, 20

    Maître, 81

    Rapide, 85

Traitement

    Événementiel, 86, 101

    postérieur, 203

Traitement préliminaire, 194

Traitement séquentiel, 201

**V**

Variable

    DFB, 212

**W**

WHILE...END, 153

