Manuel de Référence

PL7 Micro/Junior/Pro

Description détaillée des Instructions et Fonctions

Juillet 2006 fre

350110057 01

a brand of Schneider Electric

Telemecanique

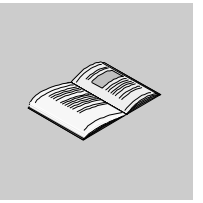
Structure de la documentation

Présentation

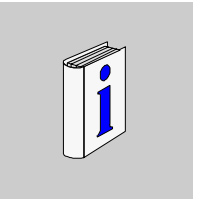
Ce manuel se compose de trois tomes:

- 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



A propos de ce manuel



Présentation

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

Instructions de base

1

Présentation

Contenu de ce chapitre

Ce chapitre décrit les instructions de base du langage PL7.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

1.1 Présentation des instructions PL7

Instructions de PL7

Généralités

Les langages PL7 exploitent tous le même jeu d'instructions.

Les instructions booléennes et les blocs fonctions ont des représentations différentes suivant le langage.

Exemple : instruction de chargement.

Instruction	Langage à contacts	Liste d'instructions	Littéral
Chargement		LD	:=

Les instructions numériques (arithmétiques, logiques, métier...) ont des représentations similaires.

Ce document décrit de façon détaillée l'ensemble des instructions, par souci de simplicité celles-ci ont été classées en 2 jeux :
- les instructions de base (Voir *Instructions de base, p. 13*),
- les instructions avancées (Voir *Instructions avancées, p. 81*).

Instructions de base

Elles comprennent les instructions booléennes de base, les blocs fonction prédéfinis, et les instructions arithmétiques et logiques sur entier.

Instructions avancées

Elles comprennent des instructions répondant à des besoins de programmation avancée. Ces instructions sont de 2 types :
- langage PL7, elles augmentent les possibilités de traitements du langage par des fonctions spécifiques (manipulation de chaînes de caractères, gestion du temps...),
- métiers, elles offrent des fonctions spécifiques au métier à traiter, exemple de fonctions pour le métier communication :
  - `PRINT` pour envoyer un message type chaîne de caractères à un terminal ou une imprimante,
  - `SEND` pour envoyer un message à une application,
  - `PID` fonction PID de régulation.

1.2 Instructions booléennes

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions booléennes du langage PL7.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

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

Définition des principaux objets booléen

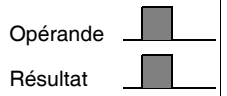
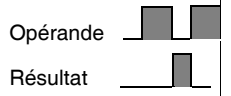
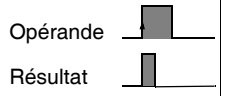
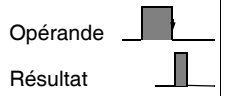
Description Le tableau suivant décrit les principaux objets booléens.

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. **Note :** 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
Etapes et macro-étapes Grafcet	Les bits Grafcet d'état des étapes, des macro-étapes et des étapes de macro-étape permettent de connaître l'état de l'étape i, de la macro-étape j ou de l'étape i de la macro-étape j du Grafcet.	%X21 %X5.9	Oui Oui

Instructions de chargement

Rôle

Le tableau suivant décrit le rôle de chacune des instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Description	Chronogramme
	LD	:=	Contacts à fermeture: contact passant (résultat à 1) quand l'objet bit qui le pilote est à l'état 1.	
	LDN	:=NOT	Contacts à ouverture: contact passant (résultat à 1) quand l'objet bit qui le pilote est à l'état 0.	
	LDR	:=RE	Contacts à front montant: détection du passage de 0 à 1 de l'objet bit qui le pilote. La mise à 1 du résultat s'effectue pendant 1 cycle.	
	LDF	:=FE	Contacts à front descendant: détection du passage de 1 à 0 de l'objet bit qui le pilote. La mise à 1 du résultat s'effectue pendant 1 cycle.	

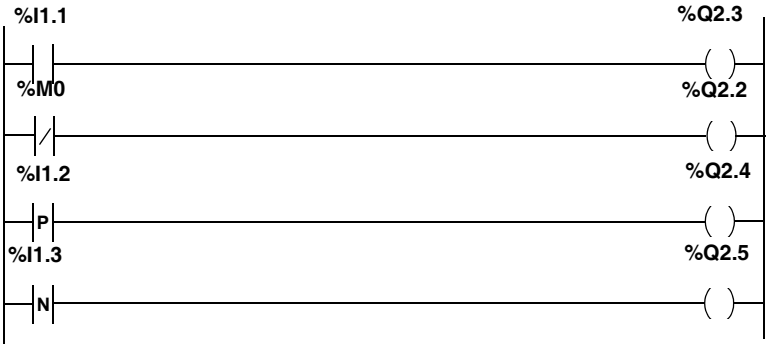
Opérandes autorisés

Le tableau suivant donne la liste des opérandes utilisés pour ces instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Opérandes
	LD	:=	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi, (True et False en liste d'instructions ou littéral structuré).
	LDN	:=NOT	%I,%Q,%M,%S,%BLK,%•:Xk, %X (True et False en liste d'instructions ou littéral structuré).
	LDR	:=RE	%I,%Q,%M
	LDF	:=FE	%I,%Q,%M

Exemple en langage à contacts

L'exemple suivant montre la programmation des instructions de chargement en langage à contacts.



Exemple en liste d'instructions

L'exemple suivant montre la programmation des instructions de chargement en langage liste d'instructions.

```
LD    %I1.1
ST    %Q2.3
LDN   %M0
ST    %Q2.2
LDR   %I1.2
ST    %Q2.4
LDF   %I1.3
ST     %Q2.5
```

Exemple en littéral structuré

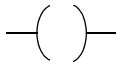
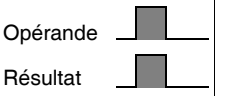
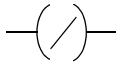
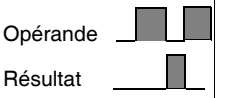
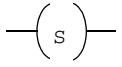
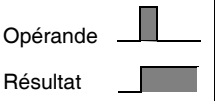
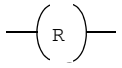
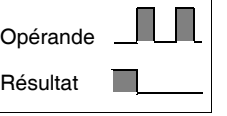
L'exemple suivant montre la programmation des instructions de chargement en langage littéral structuré.

```
%Q2.3:=%I1.1;
%Q2.2:=NOT %M0;
%Q2.4:=RE %I1.2;
%Q2.5:=FE %I1.3;
```

Instructions d'affectation

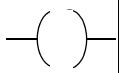
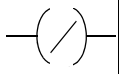
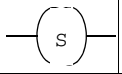
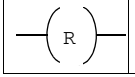
Rôle

Le tableau suivant décrit le rôle de chacune des instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Description	Chronogramme
	ST	:=	aux bobines directes: l'objet bit associé prend la valeur du résultat de l'équation.	
	STN	:=NOT	aux bobines inverses: l'objet bit associé prend la valeur de l'inverse du résultat de l'équation.	
	S	SET	aux bobines à enclenchement: l'objet bit associé est mis à 1 lorsque le résultat de l'équation est à 1.	
	R	RESET	aux bobines à déclenchement: l'objet bit associé est mis à 0 lorsque le résultat de l'équation est à 1.	

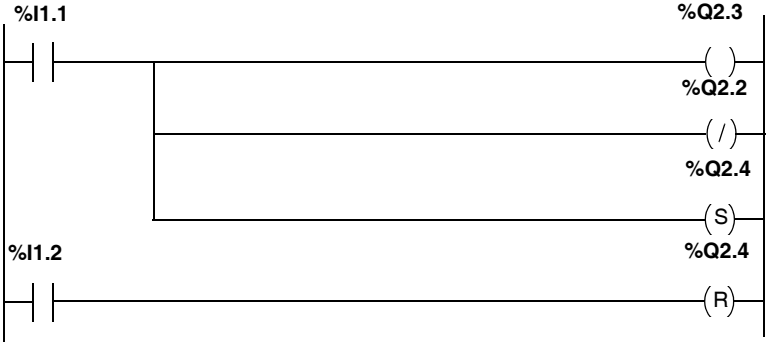
Opérandes autorisés

Le tableau suivant donne la liste des opérandes utilisés pour ces instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Opérandes
	ST	:=	%I,%Q,%M,%S,%•:Xk
	STN	:=NOT	%I,%Q,%M,%S,%•:Xk
	S	SET	%I,%Q,%M,%S,%•:Xk,%Xi Uniquement dans le traitement préliminaire.
	R	RESET	%I,%Q,%M,%S,%•:Xk,%Xi Uniquement dans le traitement préliminaire.

Exemple en langage à contacts

L'exemple suivant montre la programmation des instructions d'affectation en langage à contacts.



Exemple en liste d'instructions

L'exemple suivant montre la programmation des instructions d'affectation en langage liste d'instructions.

```
LD   %I1.1
ST    %Q2.3

STN   %Q2.2

S     %Q2.4

LD   %I1.2
R     %Q2.4
```

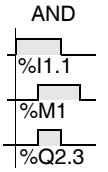
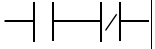
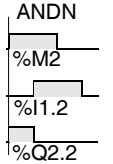
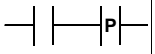
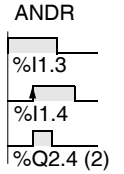
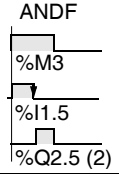
Exemple en littéral structuré

L'exemple suivant montre la programmation des instructions d'affectation en langage littéral structuré.

```
%Q2.3 := %I1.1;
%Q2.2 := NOT %I1.1;
IF %I1.1 THEN
    SET %Q2.4;
END_IF;
IF %I1.2 THEN
    RESET %Q2.4;
END_IF;
```

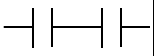
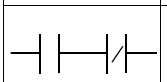
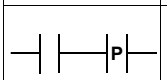
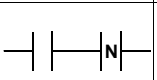
Instruction ET Logique

Rôle Le tableau suivant décrit le rôle de chacune des instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Description	Chronogramme
	AND	AND	ET logique entre l'opérande et le résultat booléen de l'instruction précédente.	
	ANDN	AND (NOT...)	ET logique entre l'inverse de l'opérande et le résultat booléen de l'instruction précédente.	
	ANDR	AND (RE...)	ET logique entre le front montant de l'opérande et le résultat booléen de l'instruction précédente. (2) Mise à 1 pendant 1 cycle.	
	ANDF	AND (FE...)	ET logique entre le front descendant de l'opérande et le résultat booléen de l'instruction précédente. (2) Mise à 1 pendant 1 cycle.	

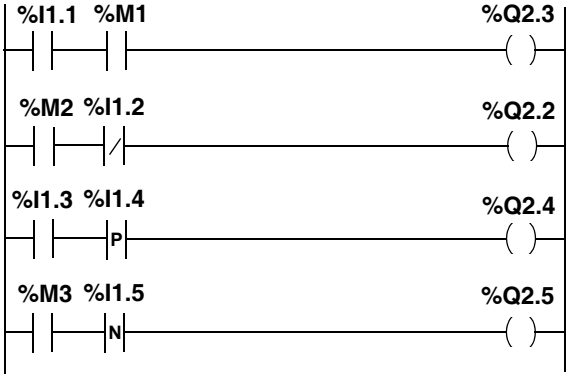
Opérandes autorisés

Le tableau suivant donne la liste des opérandes utilisés pour ces instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Opérandes
	AND	AND	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0)en langage liste d'instructions ou littéral structuré.
	ANDN	AND (NOT...)	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0)en langage liste d'instructions ou littéral structuré.
	ANDR	AND (RE...)	%I, %Q, %M
	ANDF	AND (FE...)	%I, %Q, %M

Exemple en langage à contact

L'exemple suivant montre la programmation des instructions ET Logique en langage à contacts.



Exemple en liste d'instructions

L'exemple suivant montre la programmation des instructions ET Logique en liste d'instructions.

```
LD   %I1.1
AND  %M1
ST   %Q2.3
LD   %M2
ANDN %I1.2
ST   %Q2.2
LD   %I1.3
ANDR %I1.4
ST   %Q2.4
LD   %M3
ANDF %I1.5
ST   %Q2.5
```

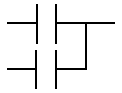
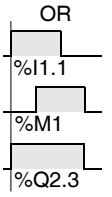
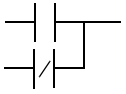
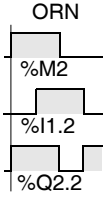
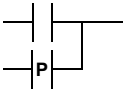
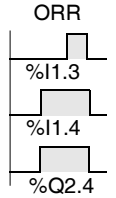
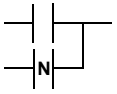
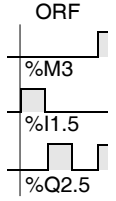
Exemple en langage littéral structuré

L'exemple suivant montre la programmation des instructions ET Logique en langage littéral structuré.

```
%Q2.3:=%I1.1 AND %M1;
%Q2.2:=%M2 AND (NOT%I1.2);
%Q2.4:=%I1.3 AND (RE%I1.4);
%Q2.5:=%M3 AND (FE%I1.5);
```

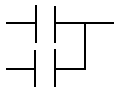
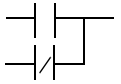
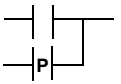
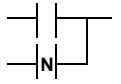
Instruction OU Logique

Rôle Le tableau suivant décrit le rôle de chacune des instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Description	Chronogramme
	OR	OR	OU logique entre l'opérande et le résultat booléen de l'instruction précédente.	
	ORN	OR (NOT...)	OU logique entre l'inverse de l'opérande et le résultat booléen de l'instruction précédente.	
	ORR	OR (RE...)	OU logique entre le front montant de l'opérande et le résultat booléen de l'instruction précédente.	
	ORF	OR (FE...)	OU logique entre le front descendant de l'opérande et le résultat booléen de l'instruction précédente.	

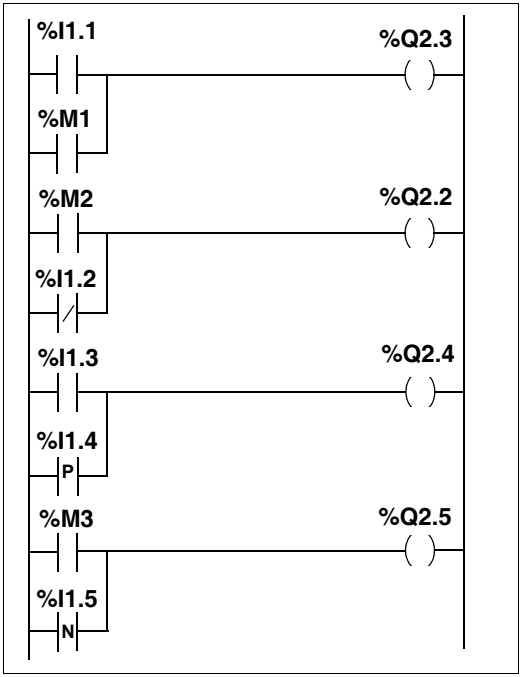
Opérandes autorisés

Le tableau suivant donne la liste des opérandes utilisés pour ces instructions.

Langage à contacts	Liste d'instructions	Littéral structuré	Opérandes
	OR	OR	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0)en langage liste d'instructions ou littéral structuré.
	ORN	OR (NOT...)	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0)en langage liste d'instructions ou littéral structuré.
	ORR	OR (RE...)	%I, %Q, %M
	ORF	OR (FE...)	%I, %Q, %M

Exemple en langage à contact

L'exemple suivant montre la programmation des instructions OU Logique en langage à contacts.



Exemple en liste d'instructions

L'exemple suivant montre la programmation des instructions OU Logique en liste d'instructions.

```
LD %I1.1
OR %M1
ST %Q2.3

LD %M2
ORN %I1.2
ST %Q2.2

LD %I1.3
ORR %I1.4
ST %Q2.4

LD %M3
ORF %I1.5
ST %Q2.5
```

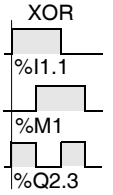
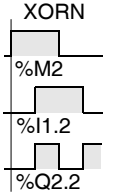
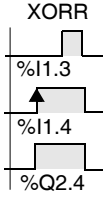
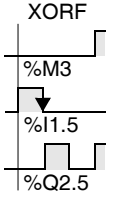
Exemple en langage littéral structuré

L'exemple suivant montre la programmation des instructions OU Logique en langage littéral structuré.

```
%Q2.3:=%I1.1 OR %M1;
%Q2.2:=%M2 OR (NOT%I1.2);
%Q2.4:=%I1.3 OR (RE%I1.4);
%Q2.5:=%M3 OR (FE%I1.5);
```

Instruction OU Exclusif

Rôle Le tableau suivant décrit le rôle de chacune des instructions.

Liste d'instructions	Littéral structuré	Description	Chronogramme
XOR	XOR	OU Exclusif entre l'opérande et le résultat booléen de l'instruction précédente.	
XORN	XOR (NOT...)	OU Exclusif entre l'inverse de l'opérande et le résultat booléen de l'instruction précédente.	
XORR	XOR (RE...)	OU Exclusif entre le front montant de l'opérande et le résultat booléen de l'instruction précédente.	
XORF	XOR (FE...)	OU Exclusif entre le front descendant de l'opérande et le résultat booléen de l'instruction précédente.	

Note : Il n'y a pas d'éléments graphiques spécifiques pour le OU exclusif en langage à contacts. Cependant le OU exclusif peut être programmé en utilisant une combinaison de contacts à ouverture et à fermeture (voir exemple ci-dessous).

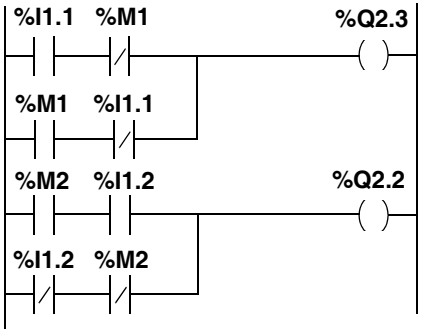
Opérandes autorisés

Le tableau suivant donne la liste des opérandes utilisés pour ces instructions.

Liste d'instructions	Littéral structuré	Opérandes
XOR	XOR	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi
XORN	XOR (NOT...)	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi
XORR	XOR (RE...)	%I, %Q, %M
XORF	XOR (FE...)	%I, %Q, %M

Exemple en langage à contact

L'exemple suivant montre la programmation des instructions OU Exclusif en langage à contacts.



Exemple en liste d'instructions

L'exemple suivant montre la programmation des instructions OU Exclusif en liste d'instructions:

```
LD %I1.1
XOR %M1
ST %Q2.3

LD %M2
XORN %I1.2
ST %Q2.2

LD %I1.3
XORR %I1.4
ST%Q2.4

LD %M3
XORF %I1.5
ST %Q2.5
```

Exemple en langage littéral structuré

L'exemple suivant montre la programmation des instructions OU Exclusif en langage littéral structuré:

```
%Q2.3:=%I1.1 XOR%M1;
%Q2.2:=%M2 XOR (NOT%I1.2);
%Q2.4:=%I1.3 XOR (RE%I1.4)
%Q2.5:=%M3 XOR (FE%I1.5);
```

Note : Les parenthèses sont facultatives mais facilitent la lisibilité du programme.

1.3 Blocs fonctions prédéfinis

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les Blocs fonctions prédéfinis du langage PL7.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Présentation du bloc fonction temporisateur %TMi

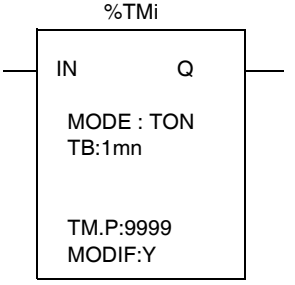
Généralités

Le temporisateur a 3 modes de fonctionnement :
- **TON** : permet de gérer des retards à l'enclenchement,
- **TOF** : permet de gérer des retards au déclenchement,
- **TP** : permet d'élaborer une impulsion de durée précise.

Les retards ou durées d'impulsion sont programmables et peuvent être modifiables ou non par terminal.

Illustration

La représentation graphique du bloc fonction temporisateur est la suivante:



Caractéristiques

Le temporisateur possède les caractéristiques suivante:

Caractéristique	Repére	Valeur
Numéro temporisateur	%TMi	0 à 63 pour un TSX 37, 0 à 254 pour un TSX 57
Mode	TON	• retard à l'enclenchement (par défaut)
	TOF	• retard au déclenchement
	TP	• monostable
Base de temps	TB	1mn (par défaut), 1s, 100ms, 10ms (16 temporisateurs maxi à 10ms). Plus la base de temps est faible, plus la précision du temporisateur sera grande
Valeur courante	%TMi.V	Mot qui croît de 0 à %TMi.P sur écoulement du temporisateur. Peut être lu, testé, mais non écrit par programme (%TMi.V peut être modifiée par terminal)
Valeur de présélection	%TMi.P	0-%TMi.P-9999. Mot qui peut être lu, testé, et écrit par programme. Est mis à la valeur 9999 par défaut. La durée ou retard élaboré est égal à %TMi.P x TB

Caractéristique	Repére	Valeur
Réglage par terminal (MODIF)	Y/N	Y : possibilité de modification de la valeur de présélection %TMi.P en réglage. N : pas d'accès en réglage.
Entrée (instruction) "Armement"	IN	Sur front montant (mode TON ou TP) ou front"Armement"descendant (mode TOF), démarre le temporisateur.
Sortie "Temporisateur"	Q	Bit associé %TMi.Q, sa mise à 1 dépend de la fonction réalisée TON, TOF ou TP.

Mode de fonctionnement du bloc temporisateur %TMi

Description

Le tableau suivant décrit les modes de fonctionnement spécifiques du bloc temporisateur.

Incidence...	Description
d'une reprise à froid	(%S0=1), provoque la mise à 0 de la valeur courante, la mise à 0 de la sortie %TMi.Q et la valeur de présélection est ré-initialisée à la valeur définie en configuration.
d'une reprise à chaud	(%S1=1) n'a pas d'incidence sur la valeur courante du temporisateur, ni sur la valeur de présélection. La valeur courante n'évolue pas pendant le temps de la coupure secteur.
d'un passage en stop, désactivation d'une tâche ou exécution d'un point d'arrêt	ne fige pas la valeur courante.
d'un saut de programme	Le fait de ne pas scruter les instructions où est programmé le bloc temporisateur ne fige pas la valeur courante %TMi.V qui continue à croître vers %TMi.P. De même le bit %TMi.Q associé à la sortie Q du bloc temporisateur conserve son fonctionnement normal et peut être ainsi testé par une autre instruction. Par contre la sortie, directement câblée à la sortie du bloc, n'est pas activée puisque non scrutée par l'automate.
de la modification de la présélection	La modification de la valeur de présélection par instruction ou en réglage n'est prise en compte qu'à la prochaine activation du temporisateur. La modification de la valeur de présélection dans l'éditeur de variables n'est prise en compte qu'après une reprise à froid (%S0=1).

Note : il est conseillé de tester le bit %TMi.Q qu'une seule fois dans le programme.

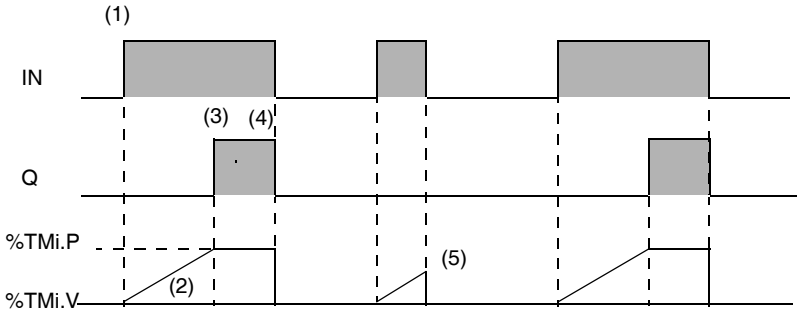
Fonctionnement du bloc fonction temporisateur %TMi en mode TON

Généralités

Le fonctionnement en mode TON du temporisateur permet de gérer des retards à l'enclenchement.

Illustration

Le chronogramme illustre le fonctionnement du temporisateur en mode TON.



Fonctionnement

Le tableau suivant décrit le fonctionnement du temporisateur en mode TON.

Phase	Description
1	Lors d'un front montant sur l'entrée IN, le temporisateur est lancé
2	La valeur courante %TMi.V du temporisateur croît de 0 vers %TMi.P d'une unité à chaque impulsion de la base de temps TB
3	Le bit de sortie %TMi.Q passe à 1 dès que la valeur courante a atteint %TMi.P
4	Le bit de sortie %TMi.Q reste à 1 tant que l'entrée IN est à 1.
5	Quand l'entrée IN est à 0, le temporisateur est arrêté même s'il était en cours d'évolution : %TMi.V prend la valeur 0.

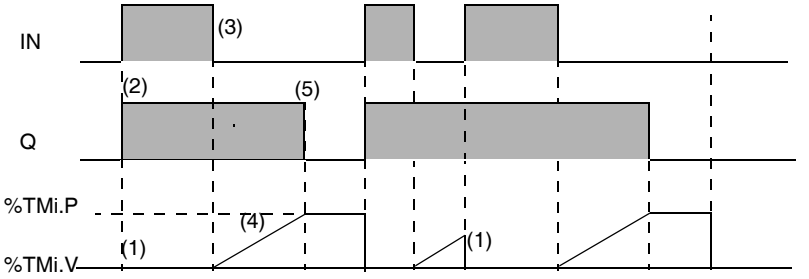
Fonctionnement du bloc fonction temporisateur %TMi en mode TOF

Généralités

Le fonctionnement en mode TOF du temporisateur permet de gérer des retards au déclenchement.

Illustration

Le chronogramme illustre le fonctionnement du temporisateur en mode TOF.



Fonctionnement

Le tableau suivant décrit le fonctionnement du temporisateur en mode TOF.

Phase	Description
1	La valeur courante %TMi.V prend la valeur 0, sur un front montant de l'entrée IN (même si le temporisateur est en cours d'évolution)
2	Le bit de sortie %TMi.Q passe à 1.
3	Lors du front descendant sur l'entrée IN, le temporisateur est lancé.
4	La valeur courante croît vers %TMi.P d'une unité à chaque impulsion de la base de temps TB.
5	Le bit de sortie %TMi.Q retombe à 0 quand la valeur courante a atteint %TMi.P

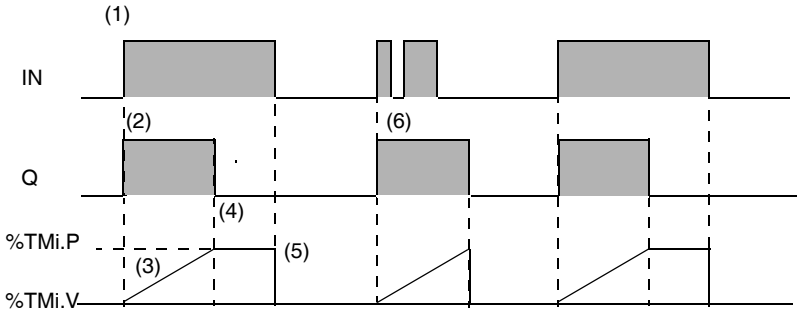
Fonctionnement du bloc fonction temporisateur %TMi en mode TP

Généralités

Le fonctionnement en mode TP du temporisateur permet de gérer d'élaborer une impulsion de durée précise (fonction monostable).

Illustration

Le chronogramme illustre le fonctionnement du temporisateur en mode TP.



Fonctionnement

Le tableau suivant décrit le fonctionnement du temporisateur en mode TP.

Phase	Description
1	Lors d'un front montant sur l'entrée IN, le temporisateur est lancé.
2	Le bit de sortie %TMi.Q passe à 1.
3	La valeur courante %TMi.V du temporisateur croît de 0 vers %TMi.P d'une unité à chaque impulsion de la base de temps TB.
4	Le bit de sortie %TMi.Q retombe à 0 quand la valeur courante a atteint %TMi.P.
5	Quand l'entrée IN et la sortie %TMi.Q sont à 0, %TMi.V prend la valeur 0.
6	Ce monostable n'est pas réarmable.

Programmation et configuration des blocs fonction temporisateur

Généralités

La programmation des blocs fonction temporisateur est identique quel que soit le mode d'utilisation sélectionné.

Le choix du fonctionnement TON, TOF ou TP s'effectue dans l'éditeur de variables.

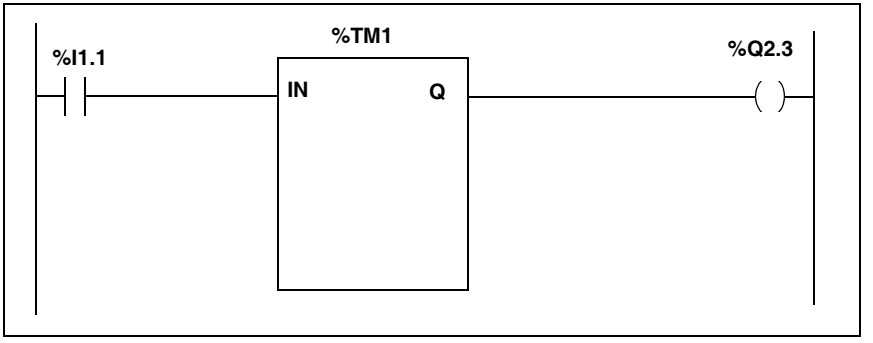
Configuration

Elle consiste à déterminer les paramètres suivants :

Paramètre	Valeurs
Mode	TON, TOF ou TP.
TB	1min, 1s, 100ms ou 10ms.
%TMi.P	0 à 9999.
MODIF	Y ou N.

Programmation en langage à contacts

Le programme ci-après illustre l'utilisation d'un bloc fonction temporisateur en langage à contacts.



Programmation en liste d'instructions

Le programme suivant illustre l'utilisation d'un bloc fonction temporisateur en langage liste d'instructions.

```
LD      %I1.1
IN      %TM1
LD      %TM1.Q
ST      %Q2.3
```

Programmation en littéral structuré

Le programme ci-après illustre l'utilisation d'un bloc fonction temporisateur en langage littéral structuré.

```
IF RE %I1.1 THEN
    START %TM1;
ELSIF FE %I1.1 THEN
    DOWN %TM1;
END_IF;
%Q2.3 := %TM1.Q;
```

L'instruction `START %TMi`, génère un front montant sur l'entrée IN du bloc temporisateur.

L'instruction `DOWN %TMi`, génère un front descendant sur l'entrée IN du bloc temporisateur.

Cas spécifiques de fonctionnement du temporisateur série 7

Cas spécifiques

- **Incidence d'une "reprise à froid"** : (%S0 = 1) provoque le chargement de la valeur de présélection (définie par l'éditeur de variables) dans la valeur courante et la mise à 0 de la sortie %Ti.D, la valeur de présélection éventuellement modifiée par le terminal étant perdue.
- **Incidence d'une "reprise à chaud"** : (%S1=1) n'a pas d'incidence sur la valeur courante du temporisateur.
- **Incidence d'un passage en stop** : le passage en stop de l'automate ne fige pas la valeur courante. Il en va de même lorsque la tâche en cours est désactivée ou lors de l'exécution d'un point d'arrêt.
- **Incidence d'un saut de programme** : le fait de ne pas scruter le réseau où est programmé le bloc temporisateur ne fige pas la valeur courante %Ti.V qui continue à décroître vers 0. De même les bits %Ti.D et %Ti.R associés aux sorties D et R du bloc temporisateur conservent leur fonctionnement normal et peuvent être ainsi testés dans un autre réseau. Par contre les bobines directement "raccordées" aux sorties du bloc ne seront pas activées puisque non scrutées par l'automate.
- **Test des bits %Ti.D et %Ti.R** : ces bits peuvent changer d'état en cours de cycle.

Présentation du bloc fonction compteur-décompteur

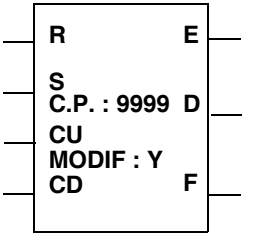
Généralités

Ce bloc fonction permet :
- Le comptage d'événement,
- Le décomptage d'événements.

Ces opérations pouvant être simultanées.

Illustration

Représentation graphique du bloc fonction compteur-décompteur.



Caractéristiques

Le compteur-décompteur a les caractéristiques suivantes :

Caractéristiques	Repère	Valeur
Numéro Compteur	%Ci	0 à 31 pour un TSX 37, 0 à 254 pour un TSX 57.
Valeur courante	%Ci.V	Mot incrémenté ou décrémenté en fonction des entrées CU et CD. Peut être lu, testé, mais non écrit par programme. Peut être modifié par terminal.
Valeur de présélection	%Ci.P	$0 \leq \%Ci.P \leq 9999$. Mot pouvant être lu, testé, écrit (valeur 9999 par défaut).
Réglage par terminal (MODIF)	Y/N	• Y : possibilité de modification de la valeur de présélection en réglage, • N : pas d'accès en réglage.
Entrée (instruction) Remise à zéro	R	Sur état 1 : %Ci.V = 0
Entrée (instruction) Présélection	S	Sur état 1 : %Ci.V = %Ci.P
Entrée (instruction) Comptage	CU	Incrémente %Ci.V sur front montant.
Entrée (instruction) Décomptage	CD	Décrémente %Ci.V sur front montant.

Caractéristiques	Repère	Valeur
Sortie Débordement	E (Empty)	Le bit associé %Ci.E=1, lorsque %Ci.V passe de 0 à 9999 (mis à 1 quand %Ci.V devient égal à 9999) est remis à 0 si le compteur continue à décompter. Quand il y a débordement, le bit %S18 passe à 1.
Sortie Présélection atteinte	D (Done)	Le bit associé %Ci.D=1, lorsque %Ci.V=%Ci.P.
Sortie Débordement	F (Full)	Le bit associé %Ci.F, lorsque %Ci.V passe de 9999 à 0 (mis à 1 quand %Ci.V devient égal à 0) est remis à 0 si le compteur continu à compter. Quand il y a débordement, le bit %S18 passe à 1.

Fonctionnement du bloc fonction Compteur/Décompteur

Fonctionnement

Fonction Comptage

Action	Résultat
Un front montant apparaît sur l'entrée comptage CU	La valeur courante %Ci.V est incrémentée d'une unité.
La valeur courante %Ci.V est égale à la valeur de présélection %Ci.P	le bit de sortie %Ci.D "présélection atteinte" associé à la sortie D passe à l'état 1.
La valeur courante %Ci.V passe de 9999 à 0	Le bit de sortie %Ci.F (débordement comptage) passe à l'état 1.
Le compteur continue à compter	Le bit de sortie %Ci.F (débordement comptage) est remis à 0.

Fonction Décomptage

Action	Résultat
Un front montant apparaît sur l'entrée décomptage CD	La valeur courante %Ci.V est décrémentée d'une unité.
La valeur courante %Ci.V passe de 0 à 9999	Le bit de sortie %Ci.E (débordement décomptage) passe à l'état 1.
Le compteur continue à décompter	Le bit de sortie %Ci.E (débordement décomptage) est remis à 0.

Fonction Comptage/Décomptage

Action	Résultat
Un front montant apparaît sur l'entrée comptage CU	La valeur courante %Ci.V est incrémentée d'une unité.
Un front montant apparaît sur l'entrée décomptage CD	La valeur courante %Ci.V est décrémentée d'une unité.
Les deux entrées sont à 1 simultanément	La valeur courante reste inchangée.

Remise à zéro

Lorsque	Résultat
L'entrée R est mise à 1 (cette entrée est prioritaire sur les autres entrées)	La valeur courante %Ci.V est forcée à 0. Les sorties %Ci.V, %CI.D et %CI.F sont à 0.

Présélection

Action	Résultat
L'entrée S "Présélection" est à l'état 1 et l'entrée R "Remise à zéro"	La valeur courante %Ci.V prend la valeur %Ci.P et la sortie %Ci.D passe à 1.

Remarque

Sur remise à 0 (entrée R ou instruction R) :

- En langage à contacts, les historiques des entrées CU et CD sont mis à jour avec les valeurs câblées.
- En langage liste d'instructions et en langage littéral structuré, les historiques des entrées CU et CD ne sont pas mis à jour ; chaque entrée garde la valeur qu'elle avait avant l'appel.

Cas spécifiques

Différents cas spécifiques

Action	Résultat
• Reprise à froid (%S0=1)	• La valeur courante %Ci.V est mise à zéro, • Les bits des sorties %Ci.E, %Ci.D et %Ci.F sont mis à zéro, • La valeur de présélection est initialisée à la valeur définie en configuration.
• Reprise à chaud (%S1=1) • Passage en stop • Désactivation d'une tâche • Exécution d'un point d'arrêt	• Aucune incidence sur la valeur courante du compteur (%Ci.V).
• Modification de la présélection %Ci.P	• La modification de la valeur de présélection par instruction ou en réglage est prise en compte lors de la gestion du bloc par l'application (activation de l'une des entrées).

Configuration et programmation

Exemple

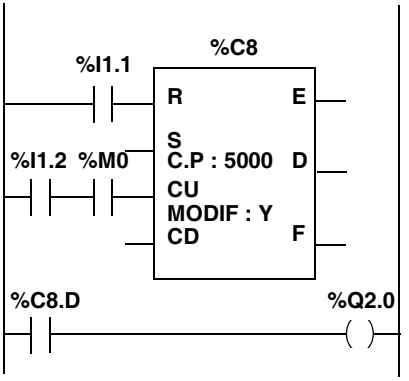
Comptage d'un nombre de pièces = 5000. Chaque impulsion sur l'entrée %I1.2 (lorsque le bit interne %M0 est à 1) provoque l'incrémentation du compteur %C8 et ce jusqu'à la valeur de présélection finale du compteur %C8 (bit %C8.D=1). La remise à zéro du compteur est provoquée par l'entrée %I1.1.

Configuration

Les paramètres à saisir par l'éditeur de variables sont les suivants :
- %Ci.P, fixé à 5000 dans cet exemple,
- MODIF : Y.

Programmation

Langage à contacts



Langage liste d'instructions

```
LD %I1.1
R   %C8
LD  %I1.2
AND %M0
CU  %C8
LD  %C8.D
ST  %Q2.0
```

Langage littéral structuré

```
IF %I1.1 THEN
    RESET %C8
END_IF;
%M1:=%I1.2 THEN
    UP %C8;
END_IF;
%Q2.0 :=%C8.D;
```

En langage littéral structuré, 4 instructions permettent de programmer les blocs fonctions compteur/décompteur :

- **RESET** %Ci : Remise à zéro de la valeur courante
- **PRESET** %Ci : Chargement de la valeur de présélection dans la valeur courante
- **UP** %Ci : Incrémente la valeur courante
- **DOWN** %Ci : Décrémente la valeur courante

Dans le cas du langage littéral structuré, l'historique des entrées CU et CD est remis à zéro lors de l'utilisation des instructions UP et DOWN. C'est donc l'utilisateur qui doit gérer les fronts montants pour ces deux instructions.

1.4 Traitements numériques sur entiers

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les traitements numériques sur entiers du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Présentation des traitements numériques sur entiers

Généralités

Les instructions numériques décrites dans ce chapitre s'appliquent aux objets de type :

- tableaux de bits
- mots
- doubles mots

Les instructions sur les autres types d'objet sont décrites dans le chapitre "Instructions avancées (Voir *Instructions avancées, p. 81*)."

Programmation en langage à contacts

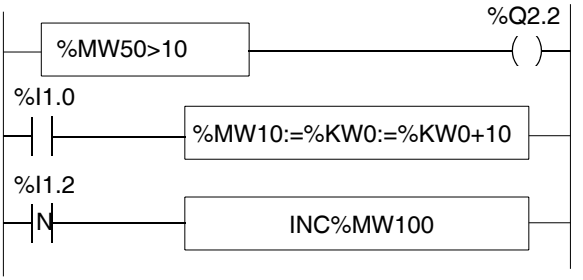
Les instructions numériques sont saisies dans des blocs :

- placées en zone test pour les blocs comparaison,
- placées en zone action pour les blocs opérations.

Ces blocs peuvent contenir :

- une expression de forme simple; ex : OP3:=OP1+OP2
- une expression de forme complexe; ex : OP5:=(OP1+OP2)*OP3-OP4.

Exemple de programmation :



Programmation en langage liste d'instructions

Les instructions sont placées entre crochets.
Elles sont exécutées si le résultat booléen de l'instruction de test précédant l'instruction numérique est à 1.

Exemple de programmation :

```
LD      [%MW50>10]
ST      %Q2.2
LD      %I1.0
[%MW10:=%KW0+10]
LDF     %I1.2
[INC%MW100]
```

Programmation en langage littéral structuré

Les instructions numériques sont saisies directement.
L'instruction conditionnelle IF, permet de conditionner ces instructions numériques par une expression booléenne.

Exemple de programmation :

```
%Q2.2:=%MW50 > 10;
IF %I1.0 THEN
   %MW10:=%KW0 + 10;
END_IF;
IF FE %I1.2 THEN
   INC %MW100;
END_IF;
```

Liste des opérandes

Liste des tableaux de bits

Abréviations	Adressage complet	Type de mot	Accès
%M:L	%Mi:L	tableau de bits internes	R/W
%I:L	%Ixy.i:L	tableau de bits d'entrées	R/W
%Q:L	%Qxy.i:L	tableau de bits de sorties	R/W
	%Xi:L ou %Xj.i:L	tableau de bits d'étapes	R

Liste des mots simples formats

Abréviations	Adressage complet	Type de mot	Accès	Forme indexée
Val. imm.	-	valeurs immédiates	R	-
%MW	%MWi	mot interne	R/W	%MWi[index]
%KW	%KWi	constante interne	R	%KWi[index]
%SW	%SWi	mot système	R/W (1)	-
%IW	%IWxy.i(.r)	mot d'entrée	R	-
%QW	%QWxy.i(.r)	mot de sortie	R/W	-
%NW	%NW{j}k	mot commun	R/W	-
%BLK	ex : %TMi.P	mot extrait de bloc fonction standard ou de bloc fonction	R/W (2)	-
%Xi.T	%Xi.T ou %Xj.i.T	temps d'activité d'étape	R	%Xi.T[index]

(1) écriture selon i.
(2) écriture suivant le type de mot, par exemple : les valeurs de présélection (%Ci.P peuvent être écrites, tandis que les valeurs courantes %Ci.V ne peuvent être que lues).

Liste des doubles mots

Abréviations	Adressage complet	Type de mot	Accès	Forme indexée
Val. imm.	-	valeurs immédiates	R	-
%MD	%MDi	mot double interne	R/W	%MDi[index]
%KD	%KDi	constante double interne	R	%KDi[index]
%SD	%SDi	double mot système	R/W (1)	-
%ID	%IDxy.i(.r)	double mot d'entrée	R	-
%QD	%QDxy.i(.r)	double mot de sortie	R/W	-

(1) uniquement double mot %SD18

Note : Il existe d'autres types de mots et doubles mots, tels que %MWxy.i %KWxy.i et %MDxy.i %KDxy.i associés aux métiers, ces doubles mots se comportent respectivement comme les mots et doubles mots %MWi %KWi et %MDi %KDi.

Note : Conversions implicites mots <--> doubles mots
Le logiciel PL7 autorise le mixage d'opérations utilisant des mots et des doubles mots. Les conversions dans l'un ou l'autre des formats s'effectuent de façon implicite, une opération faisant intervenir un double mot ou plusieurs valeurs immédiates s'exécute de façon interne automatiquement en double format.

Instructions de comparaison

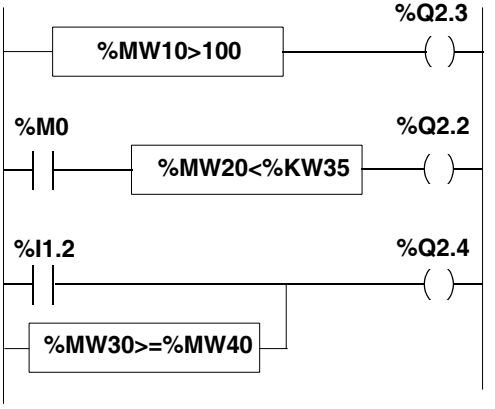
Généralités

Les instructions de comparaison permettent de comparer deux opérandes.

- **>** : teste si l'opérande 1 est supérieur à l'opérande 2,
- **>=** : teste si l'opérande 1 est supérieur ou égal à l'opérande 2,
- **<** : teste si l'opérande 1 est inférieur à l'opérande 2,
- **<=** : teste si l'opérande 1 est inférieur ou égal à l'opérande 2,
- **=** : teste si l'opérande 1 est différent de l'opérande 2.

Structure

Langage à contacts



Note : Les blocs comparaison se programment en zone de test.

Langage liste d'instructions

```
LD    [%MW10>100]
ST    %Q2.3
LD    %M0
AND   [%MW20<%KW35]
ST    %Q2.2
LD    %I1.2
OR    [%MW30>=%MW40]
ST    %Q2.4
```

Note : La comparaison est réalisée à l'intérieur de crochets figurant derrière des instructions LD, AND et OR.

Langage littéral structuré

```
%Q2.3:=%MW10>100;
%Q2.2:=%M0 AND (%MW20<%KW35);
%Q2.4:=%I1.2 OR (%MW30>=%MW40);
```

Note : Les parenthèses sont facultatives mais facilitent la lisibilité du programme.

Syntaxe

Opérateurs d'instructions de comparaison

Opérateurs	Syntaxe
>,>=,<,<=,=,<>	Op1 Opérateur Op2.

Opérandes

Type	Opérandes 1 et 2 (Op1 et Op2)
Mots indexables	%MW,%KW,%Xi.T
Mots non indexables	Val.imm.,%IW,%QW,%SW,%NW,%BLK,Expr. numérique.
Mots doubles indexables	%MD,%KD
Mots doubles non indexables	Val.imm.,%ID,%QD,%SD,Expr.numérique.

Note :
- en langage à contacts, l'opération de comparaison peut aussi s'effectuer avec le Bloc comparaison vertical (Voir *Présentation du bloc opération comparateur vertical, p. 110*).
- en langage liste d'instructions, les instructions de comparaison peuvent être utilisées au sein de parenthèses.

Instructions d'affectation

Généralités

Elles effectuent le chargement d'un opérande Op2 dans un opérande Op1
Les opérations d'affectation peuvent être réalisées :

- sur tableaux de bits,
- sur mots ou doubles mots.

Plusieurs instructions d'affectation peuvent être enchaînées dans un même bloc : Op1:=Op2:=Op3:=Op4:=...

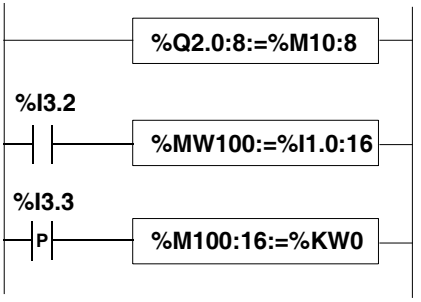
Affectation de tableaux de bits

Les opérations sur tableau de bits (("Objets de type Tableau" - Manuel de référence Tome 1)) suivantes peuvent être réalisées :

- tableau de bits -> tableau de bits (Ex: 1),
- tableau de bits -> mot ou double mot (indexé) (Ex: 2),
- mot ou double mot (indexé) -> tableau de bits (Ex: 3).

Structure

Langage à contact



Langage liste d'instructions
Exemple 1 :

```
LD  TRUE
[%Q2.0:8]
```

Exemple 2 :

```
LD  %I3.2
[%MW100:=%I1.0:16]
```

Exemple 3 :

```
LDR  %I3.3
[%MW100:16=%KW0]
```

Langage littéral structuré
Exemples 1 et 2 :

```
%Q2.0:8:=%M10:8;
IF %I3.2 THEN
  %MW100:=%I1.0:16;
END_IF;
```

Exemple 3 :

```
IF RE %I3.3 THEN
  %M100:16:=%KW0;
END_IF;
```

Syntaxe

Opérateur et syntaxe

Opérateur	Syntaxe
:=	Op1:=Op2

Opérandes

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Tableau de bits	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L,%Xi:L
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexables	%QW,%SW,%NW,%BLK	Val.imm.,%IW,%QW,%SW,%NW,%BLK,Expr.num.
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD,%SD	Val.imm.,%ID,%QD,%SD,Expr. numérique

Règles d'utilisation

- les tableaux de bits origine et destination ne sont pas forcément de longueur identique. Dans le cas où le tableau origine est plus long que le tableau destination, seuls les bits de poids faible seront transférés. Dans le cas contraire, le tableau destination est complété avec des 0,
- cas d'une affectation tableau de bits -> mot (ou double mot): les bits du tableau sont transférés dans le mot (de poids faible pour un double mot) en commençant par la droite (premier bit du tableau dans le bit 0 du mot), les bits du mot non concernés par le transfert (longueur<16 ou 32) sont positionnés à 0,
- cas d'une affectation mot -> tableau de bits : les bits du mot sont transférés à partir de la droite (le bit 0 du mot dans le premier bit du tableau).

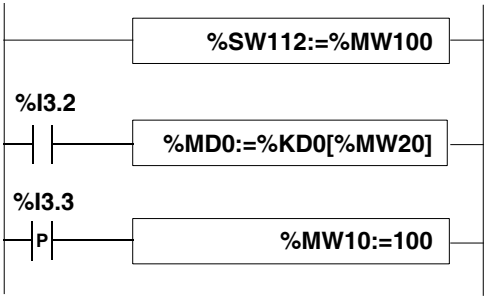
Affectation de mots

Généralités

Les opérations d'affectation sur mots suivantes peuvent être réalisées :
- mot (indexé) -> mot (indexé) ou double mot (indexé) (Ex : 1),
- double mot (indexé) -> double mot (indexé) ou mot (indexé) (Ex : 2),
- valeur immédiate -> mot (indexé) ou double mot (indexé) (Ex : 3).

Structure

Langage à contact



Langage liste d'instructions
Exemple 1 :

```
LD  TRUE
[%SW112:=%MW100]
```

Exemple 2 :

```
LD  %I3.2
[%MD10:=%KD0[%MW20]]
```

Langage littéral structuré
Exemple 3 :

```
IF %I3.3 THEN
  %MW10:=100;
END_IF;
```

Syntaxe

Opérateur et syntaxe

Opérateur	Syntaxe
:=	Op1:=Op2

Opérandes

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexables	%QW,%SW,%NW,%BLK	Val.imm.,%IW,%QW,%SW,%NW, %BLK,Expr.num.
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD,%SD	Val.imm.,%ID,%QD,%SD, Expr. numérique

Note : Les conversions mot <--> double mot sont effectuées de façon implicites, lors d'affectation double mot --> mot, si la valeur du double mot ne peut pas être contenue dans le mot, le bit %S18 est positionné à 1.
Il est possible de réaliser des affectations multiples. Exemple :
`%MW0:=%MW2:=%MW4`
Attention, dans l'exemple `%MD14:=%MW10:=%MD12`, on n'aura pas forcément `%MD14:=%MD12`, car lors de l'affectation à %MW10, il y a perte des poids forts du double mot due à la conversion double mot-simple mot.

Instructions arithmétiques sur entiers

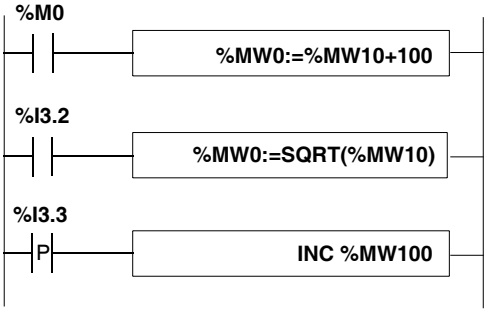
Généralités

Ces instructions permettent de réaliser une opération arithmétique entre deux opérandes ou sur un opérande.
Liste des instructions :

+	addition de deux opérandes	**SQRT**	racine carré d'un opérande
-	soustraction de deux opérandes	**INC**	incrémentation d'un opérande
*****	multiplication de deux opérandes	**DEC**	décrémentation d'un opérande
/	division de deux opérandes	**ABS**	valeur absolue d'un opérande
REM	reste de la division de 2 opérandes		

Structure

Langage à contact



Langage liste d'instructions

```
LD  %M0
[%MW0:=%MW10+100]

LD  %I3.2
[%MW0:=SQRT(%MW10)]

LD  %I3.3
[INC %MW100]
```

Langage littéral structuré

```
IF %M0 THEN
  %MW0:=%MW10+100;
END_IF;
IF %I3.2 THEN
  %MW0:=SQRT(%MW10);
END_IF;
IF RE %I1.3 THEN
  INC %MW100;
END_IF
```

Syntaxe

Opérateur et syntaxe

Opérateur	Syntaxe
+,**-**,*****,**/**,**REM**	Op1:=Op2 Opérateur Op3
SQRT,**ABS**	Op1:=Opérateur(Op2)
INC,**DEC**	Opérateur Op1

Opérandes

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexables	%QW,%SW,%NW,%BLK	Val.imm.,%IW,%QW,%SW,%NW, %BLK,Expr.num.
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD,%SD	Val.imm.,%ID,%QD,%SD, Expr. numérique

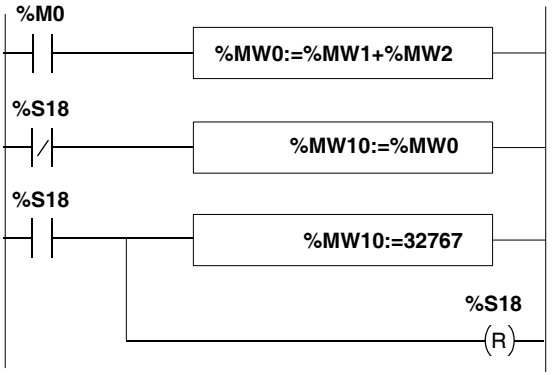
Note : Les opérations INC et DEC ne peuvent pas être utilisées dans des expressions numériques.

Règles d'utilisation

- **Addition : Dépassement de capacité pendant l'opération**
  Dans le cas où le résultat dépasse les limites :
  - -32768 ou +32767 pour un opérande simple longueur,
  - -2 147 483 648 ou +2 147 483 647 pour un opérande double longueur.

  Le bit %S18 (débordement) est mis à l'état 1. Le résultat est donc non significatif. La gestion du bit %S18 s'effectue par programme utilisateur :

Exemple en langage à contacts :



Exemple en langage liste d'instructions :

```
LD    %M0
[%MW0:=%MW1+%MW2]
LDN   %S18
[%MW10:=%MW0]
LD    %S18
[%MW10:=32767]
R     %S18]
```

Exemple en langage littéral structuré :

```
IF %M0 THEN
   %M0:=%MW1+%MW2;
END_IF;
IF %S18 THEN
   %MW10:=32767;RESET %S18;
ELSE
   %MW10:=%MW0;
END_IF;
```

Dans le cas où %MW1 =23241 et %MW2=21853, le résultat réel (45094) ne peut pas être exprimé dans un mot de 16 bits, le bit %S18 est mis à l'état 1 et le résultat obtenu (-20442) est erroné. Dans cet exemple lorsque le résultat est supérieur à 32767, sa valeur est fixée égale à 32767.

- **Multiplication :**
  Débordement de capacité pendant l'opération.
  Dans le cas ou le résultat dépasse la capacité du mot de rangement, le bit %S18 (débordement) est mis à l'état 1 et le résultat est non significatif.
- **Division/reste de la division :**
  Division par 0.
  Dans le cas ou le diviseur est égal à 0, la division est impossible et le bit système %S18 est mis à l'état 1, le résultat sera donc erroné.
  Débordement de capacité pendant l'opération.
- **Extraction de la racine carrée :**
  L'extraction de racine carrée ne s'effectue que sur des valeurs positives. Le résultat est donc toujours positif. Dans le cas où l'opérande de la racine carrée est négatif, le bit système %S18 est mis à l'état 1 et le résultat est erroné.

Note :
- Lorsque le résultat d'une opération n'est pas un entier (cas d'une division ou d'une racine carrée), le résultat est tronqué (arrondi à l'entier inférieur le plus proche).
- Le signe du reste de la division (REM) est celui du numérateur.
- La gestion du bit système %S18 est à la charge du programme utilisateur. Il est mis à 1 par l'automate, il doit être remis à zéro par le programme pour pouvoir être réutilisé (voir exemple ci-dessus).

Instructions logiques

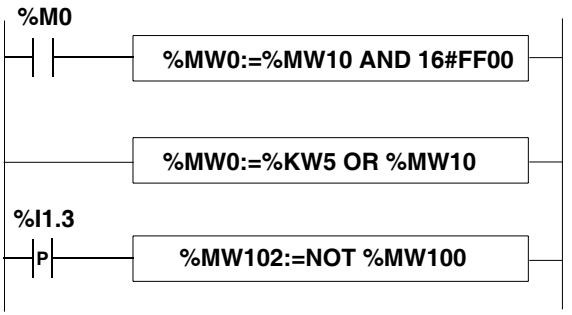
Généralités

Les instructions associées permettent de réaliser une opération logique entre deux opérandes ou sur un opérande.
Liste des instructions :

AND	ET (bit à bit) entre deux opérandes
OR	OU logique (bit à bit) entre deux opérandes
XOR	OU exclusif (bit à bit) entre deux opérandes
NOT	complément logique (bit à bit) d'un opérande

Structure

Langage à contact:



Langage liste d'instructions:

```
LD  %M0
[%MW0:=%MW10 AND 16#FF00]

LD  TRUE
[%MW0:=%KW5 OR %MW10]

LD   %I1.3
[%MW102:=NOT%MW100]
```

Langage littéral structuré:

```
IF %M0 THEN
  %MW0:=%MW10 AND 16#FF00;
END_IF;
%MW0:=%KW5 OR %MW10;
IF %I1.3 THEN
  %MW102:=NOT %MW100;
END_IF;
```

Syntaxe

Opérateur et syntaxe

Opérateur	Syntaxe
AND,**OR**,**XOR**	Op1:=Op2 Opérateur Op3
NOT	Op1:=NOT Op2

Opérandes

Type	Opérande 1 (Op1)	Opérande 2 et 3 (Op2, Op3)
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexables	%QW,%SW,%NW	Val.imm.,%IW,%QW,%SW,%NW, %BLK,Expr.num.
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD,%SD	Val.imm.,%ID,%QD,%SD, Expr. numérique

Expression numériques

Généralités

L'expression numérique est composée de plusieurs opérandes numériques et d'opérateurs arithmétiques et logiques décrits précédemment.
Le nombre d'opérateurs et d'opérandes d'une expression arithmétique n'est pas limité.
Exemple :

```
%MW25*3-SQRT(%MW10)+%KW8*(%MW15 + %MW18)AND16#FF
```

Règles d'application

- Les opérandes d'une même expression numérique peuvent être indifférement en simple ou double longueur :
  Exemple :
  ```
  %MW6*%MW15+SQRT(%DW6)/(%MW149[%MW8])+%KD29)AND16#FF
  ```
- Un opérande ou une opération à un seul opérande peut être précédé du signe + ou - (par défaut, signe +).
  Exemple :
  ```
  SQRT(%MW5)*-%MW9
  ```
- Tous les objets mots peuvent être utilisés à l'intérieur d'une expression arithmétique. L'indexation de certains mots est possible.

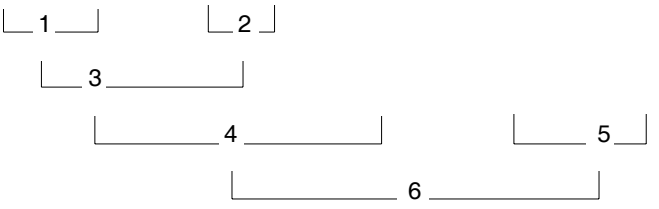
Priorité d'exécution des instructions

Dans l'expression numérique, la priorité des différentes instructions est respectée. L'exécution s'effectue dans l'ordre décrit ci-après :
Ordre d'exécution :

Rang	Instruction
1	Instruction à un opérande
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

Exemple :
L'exécution des instructions ci-dessous est réalisée suivant l'ordre de la numérotation :

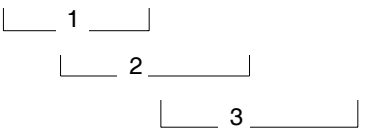
```
SQRT (%MW3) + %MW5 * 7 AND %MW8 OR %MW5 XOR %MW10
```



Parenthèses

Les parenthèses permettent de modifier l'ordre d'évolution des priorités. Leur usage est conseillé pour structurer les expressions numériques.
L'exemple ci-dessous montre l'ordre d'exécution des parenthèses.

```
((%MW5 AND %MW6) + %MW7) * %MW8
```



1.5 Instructions sur programme

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions sur programme du langage PL7.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

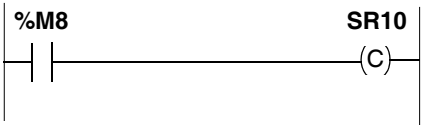
Appel à un sous-programme

Généralités

L'instruction d'appel à un sous-programme permet d'appeler un module sous-programme situé dans la même tâche.

Structure

Langage à contact :



Langage liste d'instruction :

```
LD  %M8
SR10
```

Langage littéral structuré :

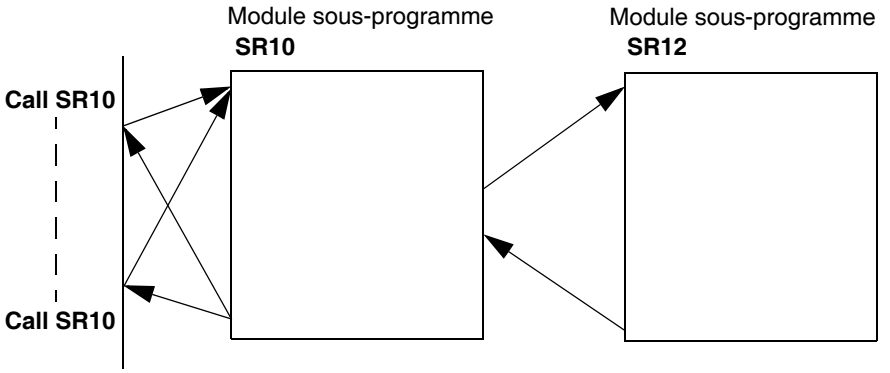
```
IF %M8 THEN
   SR10;
END_IF;
```

Règles

- L'appel à sous-programme ne peut être réalisé que si le module sous-programme a été au préalablement créé.
- Le retour d'un sous-programme se fait sur l'action suivant immédiatement l'instruction d'appel à sous-programme.
- Un sous-programme peut appeler un autre sous-programme; le nombre d'appels en cascade est limité à 8.
- Les sous-programmes sont affectés à une tâche, ils ne peuvent être appelés qu'à partir de la même tâche.

Principe

Principe d'exécution des sous-programmes :



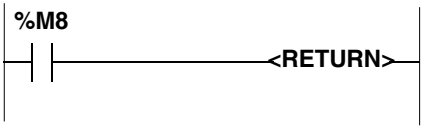
Retour de sous-programme

Généralités

L'instruction de retour de sous-programme est réservée aux modules sous-programmes et permet le retour vers le module appelant, si le résultat booléen de l'instruction de test précédente est à 1.

Structure

Langage à contact



Langage liste d'instruction

```
LD  %M8
RETC
```

Langage littéral structuré

```
IF %M8 THEN
   RETURN;
END_IF;
```

Le langage liste d'instructions comporte les instructions supplémentaires suivantes :

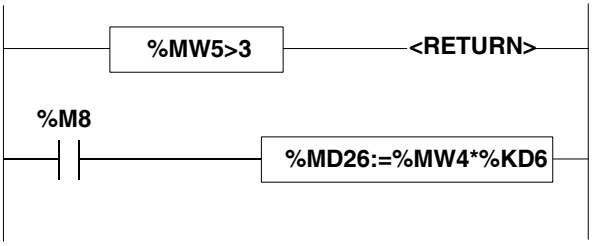
- **RETCN** : retour de sous-programme si résultat booléen de l'instruction de test précédente est à 0.
- **RET** : retour de sous-programme inconditionnel.

Règles d'utilisation

L'instruction de retour de sous-programme est implicite à la fin de chaque sous-programme, mais peut être utilisée pour un retour vers le module appelant avant la fin du sous-programme.

Exemples

Langage à contacts



Langage liste d'instructions

```
LD   [%MW5>3]
RETC
LD   %M8
[%MD26:=%MW4*%KD6]
```

Langage littéral structuré

```
IF (%M5>3) THEN
   RETURN;
END_IF;
IF %M8 THEN
   %MD26:=%MW4*%KD6;
END_IF;
```

Saut dans le programme

Généralités

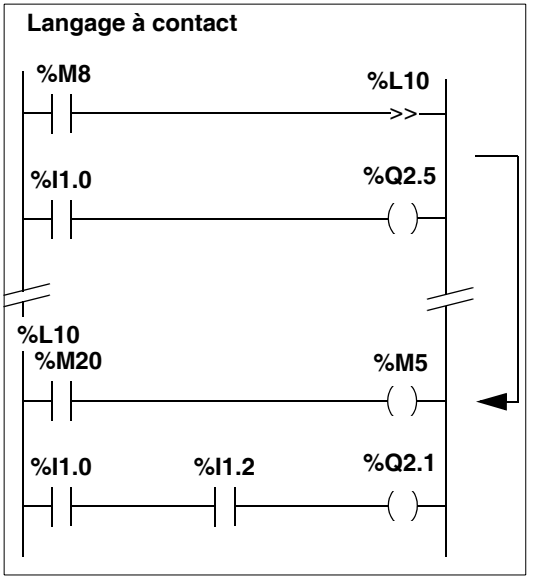
Les instructions de saut permettent un branchement à une ligne de programmation repérée par une étiquette %Li :

- **JMP** : saut de programme inconditionnel,
- **JMPC** : saut de programme si résultat booléen de l'instruction de test précédente est à 1,
- **JMPCN** : saut de programme si résultat booléen de l'instruction de test précédente est à 0. %Li représente l'étiquette de la ligne sur laquelle est effectuée le branchement (i repère de 1 à 999 avec 256 étiquettes maximum).

Structure

JUMPC

Langage à contact



Langage liste d'instructions

```
LD      %M8
JMPC    %L10
LD      %I1.0
ST      %Q2.5

----------
%L10:
LD     %M20
ST     %M5
LD     %I1.0
AND    %I1.2
ST     %Q2.1
```

Saut à l'étiquette %L10, si %M8 est à 1

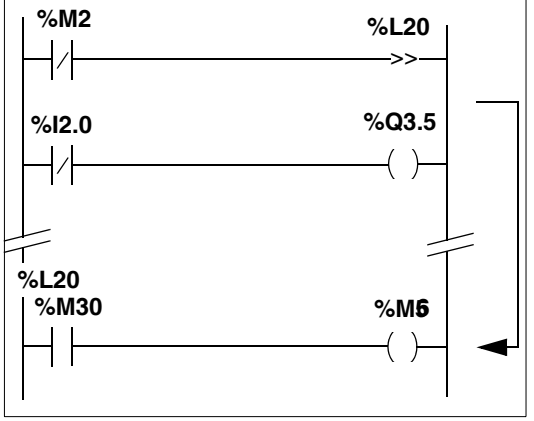
Langage littéral structuré

```
IF %M8 THEN
   JUMP %L10;
END_IF;
%Q2.5:=%I1.0;

------
%L10:
  %M5:=%M20;
  %Q2.1:=%I1.0 AND %I1.2;
```

Saut à l'etiquette %L10 si %M8 est à 1

JUMPCN

Langage à contact



Langage liste d'instructions

```
LD      %M2
JMPCN   %L20
LDN     %I2.0
ST      %Q3.5

----------
%L20:
LD     %M30
ST     %M6
```

Saut à l'étiquette %L20, si %M2 est à 0

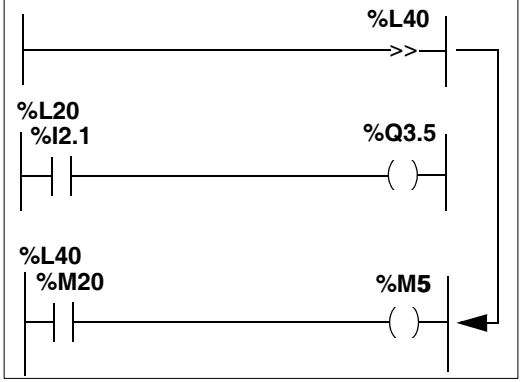
Langage littéral structuré

```
IF NOT %M2 THEN
   JUMP %L20;
END_IF;
%Q3.5:=NOT%I2.0;

------
%L20:
  %M6:=%M30;
```

Saut à l'etiquette %L10 si %M8 est à 1

JMP

Langage à contact



Langage liste d'instructions

```
JMP     %L40

%L20:
LD      %I2.1
ST      %Q3.5
---------
%L40:
LD     %M20
ST     %M5
```

Saut inconditionnel à l'étiquette %L40

Langage littéral structuré

```
JUMP %L40;

%L20
  %Q3.5:=%I2.1;

------
%L40:
  %M5:=%M20;
```

Saut inconditionnel à l'étiquette %L40

Règles

- Un saut de programme se fait à l'intérieur d'une même entité de programmation (module principal d'une tâche maître MAIN, sous-programme %SRi,...),
- Un saut de programme se fait vers une ligne de programmation située en aval ou en amont.

Dans le cas de saut amont, il faut faire attention au temps d'exécution du programme : le temps d'exécution du programme est alors allongé et peut entraîné un dépassement de la période de la tâche incluant le saut amont.

Instructions de fin de programme

Généralités

Les instructions END, ENDC et ENDCN permettent de définir la fin d'exécution de cycle du programme :

- **END** : fin de programme inconditionnelle,
- **ENDC** : fin de programme si résultat booléen de l'instruction de test précédente est à 1,
- **ENDCN** : fin de programme si résultat booléen de l'instruction de test précédente est à 0.

Note : Les instructions **END**, **ENDC** et **ENDCN** ne doivent pas être utilisées dans les sections de programme des automates **Premium** et **Micro**. Il est impératif de les remplacer respectivement avec les instructions JMP, JMPC et JMPCN avec un saut vers une étiquette à la fin du programme. Aucun contrôle de conformité n'est effectué par le logiciel de programmation PL7.

Règles

Par défaut (mode normal), lorsque la fin de programme est activée, il y a mise à jour des sorties et passage au cycle suivant.
Si la scrutation est périodique, il y a mise à jour des sorties, attente de fin de période et passage au cycle suivant.

Note : Ces instructions ne sont utilisables qu'en langage liste d'instructions dans la tâche maître.

Exemple

Langage liste d'instructions

Exemple 1 :

```
LD   %M1
ST   %Q2.1
LD   %M2
ST   %Q2.2
-------
END
```

Exemple 2 :

```
LD   %M1
ST   %Q2.1
LD   %M2
ST   %Q2.2
------
LD   %I2.2
ENDC
LD   %M2
ST   %Q2.2
------
END
```

- Si %I1.2 = 1, il y a fin de scrutation du programme
- Si %I1.2 = 0, La scrutation continue jusqu'à la prochaine instruction END

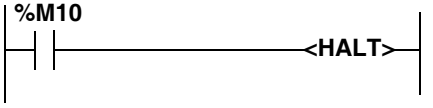
Arrêt du programme

Généralités

L'instruction HALT dans un programme application permet l'arrêt de son exécution (arrêt de toutes les tâches), ce qui a pour effet de figer les objets variables de ce programme.
Pour être à nouveau exécuté, un programme ainsi arrêté, devra être initialisé (par la commande INIT de PL7). Les instructions, qui suivent l'instruction HALT, ne seront donc pas exécutées.

Structure

Langage à contacts



Langage liste d'instructions

```
LD   %M10
HALT
```

Langage littéral structuré

```
IF %M10 THEN
    HALT;
END_IF;
```

Instructions de masquage/démasquage d'événement

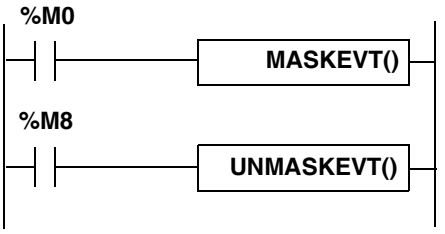
Généralités

Les instructions de masquage/démasquage permettent le masquage ou le démasquage de l'ensemble des événements qui assurent l'activation des tâches événementielles.

- **MASKEVT** : masquage global des événements. Les événements sont mémorisés par l'automate, par contre les tâches événementielles associées restent inactives tant que l'opération de masquage est valide (jusqu'à la prochaine instruction UNMASKEVT).
- **UNMASKEVT** : démasquage global des événements. Les événements qui ont été mémorisés pendant la période de masquage sont traités. Le mécanisme de traitement événementiel est opérationnel jusqu'à la prochaine instruction MASKEVT.

Structure

Langage à contact



Langage liste d'instructions

```
LD   %M0
[MASKEVT()]

LD   %M8
[UNMASKEVT()]
```

Langage littéral structuré

```
IF %M0 THEN
   MASKEVT();
END_IF;
IF %M8 THEN
   UNMASKEVT();
END_IF;
```

Instructions NOP

Généralités

L' instruction **NOP** n'effectue aucune action. Elle permet de "réserver" des lignes dans un programme et ainsi de pouvoir écrire par la suite des instructions sans modification des numéros de lignes.

Instructions avancées

2

Présentation

Contenu de ce chapitre

Ce chapitre décrit les instructions avancées du langage PL7.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

2.1 Présentation des instructions avancées

Présentation des instructions avancées

Généralités

Les instructions décrites dans ce chapitre répondent à des besoins de programmation avancée.
Elles ont les mêmes effets quel que soit le langage utilisé. Seule la syntaxe diffère.

Ce sont:
- soit des instructions de base du logiciel.
- soit des Fonctions considérées comme extensions du logiciel.

Les instructions de types Fonctions étendues permettent d' enrichir le logiciel de base par des instructions spécifiques de programmation.
- Opérations sur chaînes de caractères, tableaux de mots, etc...
- Fonctions métiers: Communication, Régulation, Dialogue opérateur, etc...

Familles d'instructions

Elles comprennent les familles suivantes:
- Chaînes de caractères,
- Tableaux d'entiers,
- Gestion des Dates, heures, durées,
- Conversions,
- Tableaux de bits,
- Fonctions "Orphée".

Les familles suivantes sont décrites dans les métiers concernés:
- Communication,
- Régulation,
- Dialogue opérateur,
- Commande de mouvement.

Note : Les instructions de type Fonction impliquent une occupation mémoire application supplémentaire (seulement lorsqu'elles sont réellement utilisées dans le programme). Cette occupation mémoire est à prendre en compte par le programmeur pour chaque fonction quel que soit leur nombre d'utilisation et ceci en accord avec la taille mémoire maxi de l'automate retenu.

2.2 Blocs fonctions prédéfinis avancés

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les Blocs fonctions prédéfinis avancés du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

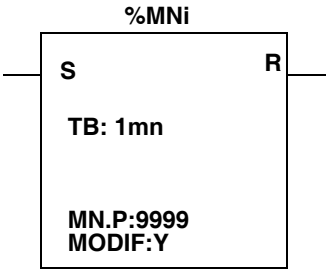
Présentation du bloc fonction Monostable

Généralités

Le bloc fonction monostable permet d'élaborer un impulsion de durée précise. Cette durée est programmable et peut être modifiable ou non par terminal

Illustration

Représentation graphique du bloc fonction monostable



Caractéristiques

Caractéristiques du bloc fonction monostable

Caractéristique	Repère	Valeur
Numéro	%MNi	0 à 7 pour un TSX 37, 0 à 254 pur un TSX 57.
Base de temps	TB	1mn, 1s, 100ms, 10ms (1mn par défaut).
Valeur courante	%MNi.V	Mot qui décroît de %MNi.P vers 0 sur écoulement du temporisateur. Peut être lu, testé, mais non écrit.
Valeur de présélection	%MNi.P	$0 \leq \%MNi.P \leq 9999$. Mot pouvant être lu, testé, écrit. La durée de l'impulsion (PRESET) est égale à : %MNi.P x TB
Modification MODIF	Y/N	• Y : possibilité de modification de la valeur de présélection en réglage, • N : pas d'accès en réglage.
Entrée "Départ" (ou instruction)	S (Start)	Sur front montant %MNi.V = %MNi.P puis %MNi.V décroît vers 0.
Sortie "Monostable"	R (Running)	Le bit associé %MNi.R est à 1 si %MNi.V > 0 ("écoulement en cours" monostable) %MNi.R = 0 si %MNi.V = 0.

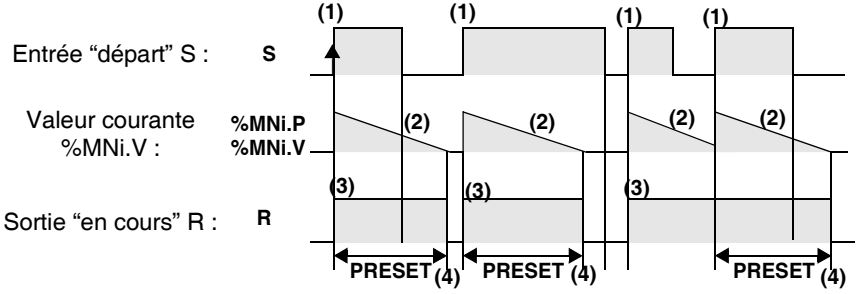
Fonctionnement du bloc fonction monostable

Généralités

Le bloc fonction monostable permet d'élaborer une impulsion de durée précise.

Illustration

Chronogramme illustrant le fonctionnement du monostable.



Fonctionnement

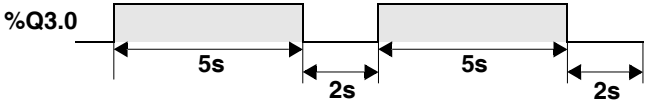
Description du fonctionnement du monostable.

Phase	Description
1	Dès l'apparition d'un front montant sur l'entrée S du monostable, la valeur courante %MNi.V prend la valeur de présélection %MNi.P
2	La valeur courante %MNi.V décroit vers 0 d'une unité à chaque impulsion de la base de temps TB.
3	Le bit de sortie %MNi.R (Running) associé à la sortie R passe à l'état 1 dès que la valeur courante %MNi.V est différente de 0.
4	Lorsque la valeur courante %MNi.V = 0, le bit de sortie %MNi.R repasse à l'état 0.

Configuration et programmation des blocs fonctions monostable

Exemple

Clignotant à périodes cycliques variable : la valeur de présélection de chaque monostable définit la durée de chaque impulsion.



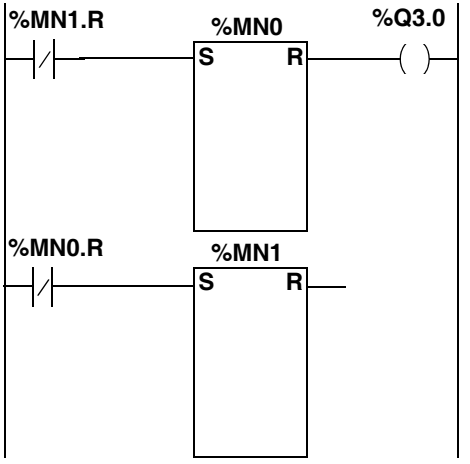
Configuration

Les paramètres suivants sont à saisir dans l'éditeur de variables :

- TB : 1mn, 1s, 100ms, 10ms ou 1ms (100ms dans cet exemple),
- %MNi.P : 0 à 9999 (%MN0.P = 50 et %MN1.P = 20 dans cet exemple),
- MODIF : Y ou N

Programmation

Langage à contact



Langage liste d'instructions

```
LDN %MN1.R
S %MN0
LD %MN0.R
ST %Q3.0
LDN %MN0.R
S %MN1
```

Langage littéral structuré

```
%M0:=NOT %MN1.R;
IF RE %M0 THEN
 START %MN0;
END_IF;
%Q3.0:=%MN0.R;
%M1:=NOT %MN0.R;
IF RE %M1 THEN
 START %MN1;
END_IF;
```

Dans l'exemple ci-dessus, la sortie %Q3.0 est mise à l'état 1 pendant 5s (%MN0.P) et remise à l'état 0 pendant 2s (%MN1.P).

Remarques

- En langage littéral structuré, l'instruction `START%MNi` permet de lancer l'exécution du bloc fonction monostable. Cette instruction force un front montant sur l'entrée S du bloc, ce qui a pour effet de réinitialiser le bloc fonction. L'utilisation de cette instruction doit donc être impulsionnelle.
- La fonction monostable peut aussi être réalisée par le bloc fonction %TMi en mode TP (Voir *Fonctionnement du bloc fonction temporisateur %TMi en mode TP, p. 37*).

Cas spécifiques

- **Incidence d'une "reprise à froid"** : (%S0 = 1) provoque le chargement de la valeur de présélection %MNi.P dans la valeur courante %MNi.V, la valeur de présélection éventuellement modifiée par le terminal étant perdue, la sortie %MNi.R est remise à 0.
- **Incidence d'une "reprise à chaud"** : (%S1) n'a pas d'incidence sur la valeur courante du monostable (%MNi.V).
- **Incidence d'un passage en stop, désactivation de la tâche et point d'arrêt** : le passage en stop de l'automate ne fige pas la valeur courante. Il en va de même lorsque la tâche en cours est désactivée ou lors de l'exécution d'un point d'arrêt.
- **Incidence d'un saut de programme** : le fait de ne plus scruter le réseau où est programmé le bloc monostable ne fige pas la valeur courante %MNi.V qui continue à décroître vers 0. De même le bit %MNi.R associé à la sortie du bloc monostable conserve son fonctionnement normal et peut être ainsi testé dans un autre réseau. Par contre les bobines directement "raccordées" à la sortie du bloc (ex %Q3.0) ne seront pas activées puisque non scrutées par l'automate.
- **Test du bit %MNi.R** : ce bit peut changer d'état en cours de cycle.

Présentation du bloc fonction Registre

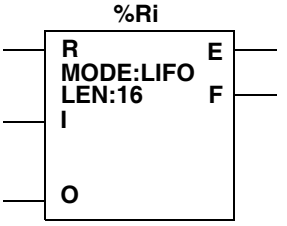
Généralités

Un registre est un bloc mémoire permettant de stocker jusqu'à 255 mots de 16 bits de deux manières différentes :

- file d'attente (premier entré, premier sorti) appelée pile FIFO (First In, First Out),
- pile (dernier entré, premier sorti) appelée pile LIFO (Last In, First Out).

Illustration

La représentation graphique du bloc fonction registre est la suivante :



Caractéristiques Liste des caractéristiques du bloc fonction Registre:

Caractéristique	Repère	Valeur
Numéro	%Ri	0 à 3 pour un TSX 37, 0 à 254 pour un TSX 57.
Mode	FIFO LIFO	File d'attente Pile (choix par défaut).
Longueur	LEN	Nombre de mots de 16 bits (1< LEN< 255) composant le bloc mémoire registre.
Mot d'entrée	%Ri.I	Mot d'accès au registre. Peut être lu, testé, écrit.
Mot de sortie	%Ri.O	Sur front montant provoque le rangement d'un mot d'information dans le mot %Ri.O
Entrée (ou instruction) "Stockage"	I (In)	Sur front montant provoque le stockage du contenu du mot %Ri.I
Entrée (ou instruction) "Destockage"	O (Out)	Sur front montant provoque le rangement d'un mot d'information dans le mot %Ri.O
Entrée (ou instruction) "Remise à zéro"	R (Reset)	Sur état 1 initialise le registre.
Sortie "Vide"	E (Empty)	E bit %Ri.E associé indique que le registre est vide. Peut être testé.
Sortie "Plein"	F (Full)	Le bit %Ri.F associé indique que le registre est plein. Peut être testé.

Note : Lorsque les deux entrées I et O sont activées simultanément, le stockage est réalisé avant le déstockage.

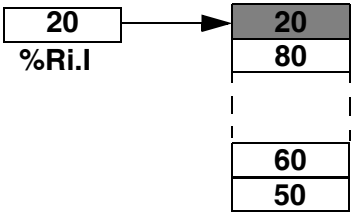
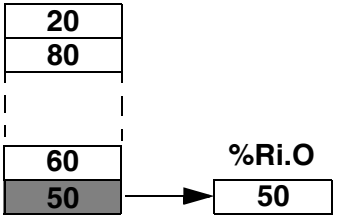
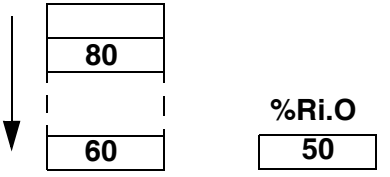
Fonctionnement du bloc fonction Registre en mode FIFO

Généralités

Dans le mode FIFO (First In - First Out), la première information entrée dans la pile du registre est la première à être sortie.

Fonctionnement

Cette table décrit le fonctionnement du mode FIFO.

Etape	Description
1	Sur un front montant sur l'entrée I ou activation de l'instruction I, le contenu du mot d'entrée %Ri.I préalablement chargé est stocké au plus haut de la pile. Lorsque la pile est pleine, le chargement est impossible et le bit système %S18 passe à 1. 
2	Sur front montant de l'entrée O ou activation de l'instruction O, le mot d'information le plus bas dans la file est rangé dans le mot de sortie %Ri.O. 
3	Dés que le mot est transféré dans Ri.O, le contenu du registre est décalé d'un pas vers le bas. Lorsque le registre est vide (sortie E=1) le déstockage est impossible, le mot de sortie %Ri.O n'évolue plus et conserve sa valeur La pile peut être réinitialisée à tout moment (état 1 sur l'entrée R ou activation de l'instruction R). 

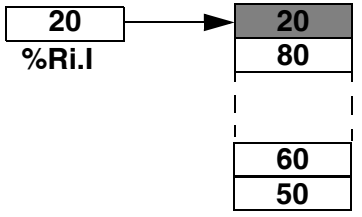
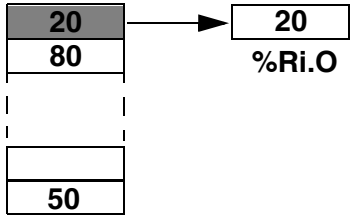
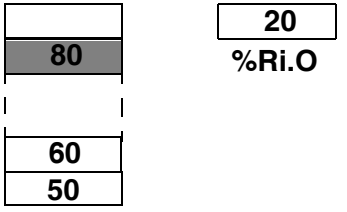
Fonctionnement du bloc fonction Registre en mode LIFO

Généralités

Dans le mode LIFO (Last In - First Out), la dernière information entrée dans la pile du registre est la première à être sortie.

Fonctionnement

Cette table décrit le fonctionnement du mode LIFO.

Etape	Description	
1	Sur un front montant sur l'entrée I ou activation de l'instruction I, le contenu du mot d'entrée %Ri.I préalablement chargé est stocké au plus haut de la pile. Lorsque la pile est pleine, le chargement est impossible et le bit système %S18 passe à 1.	
2	Sur front montant de l'entrée O ou activation de l'instruction O, le mot d'information le plus haut dans la pile (dernière information entrée) est rangé dans le mot de sortie %Ri.O.	
3	Dés que le mot est tranféré dans Ri.O, le mot suivant du registre est disponible. Lorsque le registre est vide (sortie E=1) le déstockage est impossible, le mot de sortie %Ri.O n'évolue plus et conserve sa valeur La pile peut être réinitialisée à tous moment (état 1 sur l'entrée R ou activation de l'instruction R).	

Programmation et configuration du bloc fonction Registre

Exemple

L'exemple suivant montre le chargement de %R2.I par le mot %MW34 sur demande de l'entrée %I1.2, si le registre R2 n'est pas plein (%R2.F=0). La demande d'entrée dans le registre est assurée par %M1. La demande de sortie est faite par l'entrée %I1.3 et le rangement de %R2.O dans %MW20 s'effectue si le registre n'est pas vide (%R2.E=0).

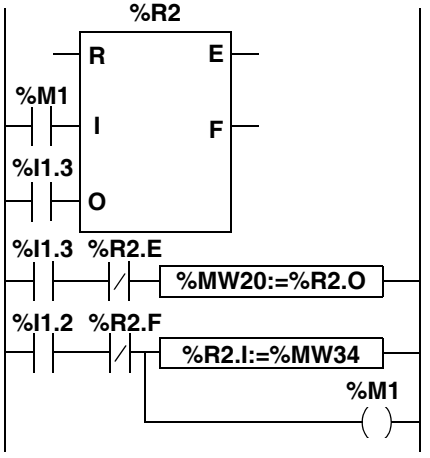
Configuration

Les paramètres suivants sont à saisir dans l'éditeur de configuration :
- Nombre : 1 à 4 pour un TSX 37, 1 à 255 pour un TSX 57,
- Longueur : 1 à 255.

Le mode de fonctionnement (FIFO ou LIFO) est à saisir dans l'éditeur de variables.

Programmation

Langage à contacts



Langage liste d'instructions

```
LD %M1
I %R2
LD %I1.3
O %R2
LD %I.3
ANDN %R2.E
[%MW20:=%R2.O]
LD %I.2
ANDN %R2.F
[%R2.I:=%MW34]
ST %M1
```

Langage littéral structuré

```
IF RE %M1 THEN
 PUT %R2;
END_IF;
IF RE %I1.3 THEN
 GET %R2;
END_IF;
IF (%I1.3 AND NOT %R2.E) THEN
 %MW20:=%R2.O;
END_IF;
%M1:=%I1.2 AND NOT %R2.F;
IF %M1 THEN
 %R2.I:=%MW34;
END_IF;
```

Remarque

En langage littéral structuré, 3 instructions permettent de programmer les blocs fonctions registre :

- **RESET** %Ri: Initialisation du registre,
- **PUT** %Ri: Provoque le stockage du contenu du mot %R.I dans le registre,
- **GET** %Ri: Provoque le rangement d'un mot d'information dans le mot %Ri.O.

Les instructions PUT et GET réalisent un front montant, respectivement sur les entrées I et O du bloc fonction. L'utilisation de ces instructions doit donc être impulsionnelle.

Cas spécifiques

- **Incidence d'une reprise "à froid"**: (%S0=1) provoque l'initialisation du contenu du registre. Le bit de sortie %Ri.E associé à la sortie E est mis à 1.
- **Incidence d'une reprise "à chaud"**: (%S1=1) n'a pas d'incidence sur le contenu du registre ainsi que sur l'état des bits de sorties.
- **Sur remise à 0** (entrée R ou instruction R)
  - En langage à contacts, les historiques des entrées I et O sont mis à jour avec les valeur câblées,
  - En langage liste d'instructions, les historiques des entrées I et O ne sont pas mis à jour : chacune garde la valeur qu'elle avait avant l'appel,
  - En langage littéral structuré, les historiques des entrées I et O sont mis à jour avec 0.

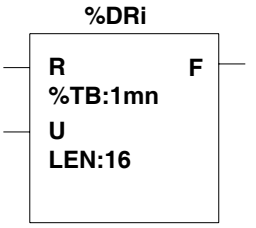
Présentation du bloc fonction Programmateur cyclique (Drum)

Généralités

D'un principe de fonctionnement similaire au programmateur à cames, le programmateur cyclique change de pas en fonction d'événements extérieurs. A chaque pas, le point haut d'une came donne un ordre exploité par l'automatisme. Dans le cas du programmateur cyclique, ces points hauts seront symbolisés par un état 1 au niveau de chaque pas et sont affectés à des bits de sortie %Qi.j ou interne %Mi appelés bits d'ordres.

Illustration

Représentation graphique du bloc fonction Programmateur cyclique (Drum)



Caractéristiques

Liste des caractéristiques du bloc fonction programmateur cyclique

Caractéristique	Repère	Valeur
Numéro	%DRi	0 à 7 pour un TSX 37, 0 à 254 pour un TSX 57.
Nombre de pas	LEN	1 à 16 (16 par défaut).
Base de temps	TB	1mn, 1s, 100ms, 10ms (1mn par défaut).
Temps enveloppe ou durée du pas en cours	%DRi.V	$0 \leq \%DRi.V \leq 9999$. Mot pouvant être remis à zéro à chaque changement de pas. Peut être lu, testé mais non écrit. La durée est égale à %DRi.V x TB
Numéro du pas en cours	%DRi.S	$0 \leq \%Di.S \leq 15$. Mot pouvant être lu et testé. Ne peut être écrit qu'à partir d'une valeur immédiate.
Entrée "retour au pas 0"	R (RESET)	Sur état 1 initialise le programmateur au pas 0.
Entrée "avance	U (UP)	Sur front montant provoque l'avance d'un pas du programmateur et la mise à jour des bits d'ordres.
Sortie	F (FULL)	Indique que le dernier pas défini est en cours. Le bit %DRi.F associé peut être testé (%DRi.F=1 si %DRi.S=nombre de pas configurés -1).
Etat d'un pas	%DRi.Wj	Mot de 16 bits définissant les états du pas j du programmateur i. Peut être lu, testé mais non écrit.

Caractéristique	Repère	Valeur
Bits d'ordres	%DRi.Wj	Sorties ou bits internes associés au pas (16 bits d'ordre).

Note : Le bit %S18 passe à 1, si un pas non configuré est écrit.

Fonctionnement du bloc fonction Programmateur cyclique (Drum)

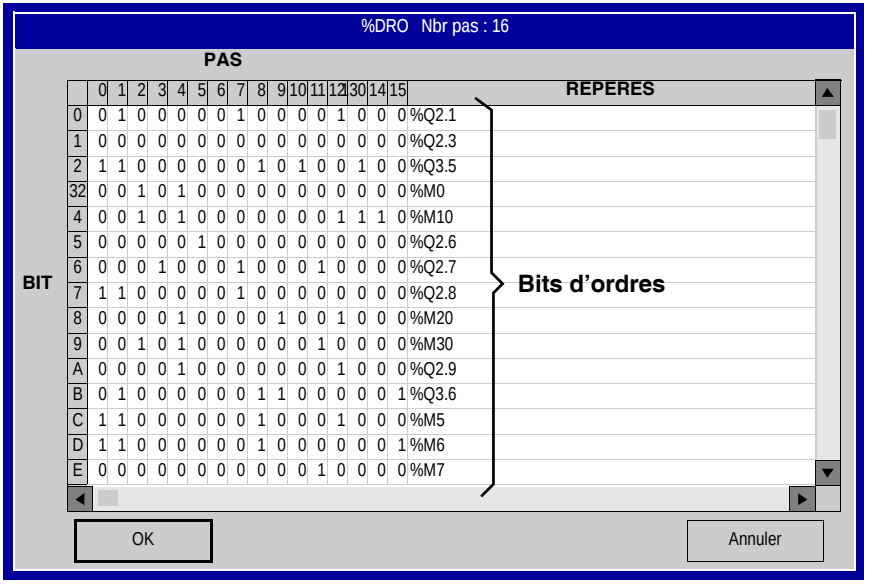
Généralités

Le programmateur cyclique se compose :.

- d'une matrice de données constantes (les cames) organisée en colonne : en pas de 0 à N-1 (N étant le nombre de pas configuré), chaque colonne donne les états du pas sous forme de 16 informations binaires repérées de 0 à F,
- d'une liste de bits d'ordres (1 par ligne) correspondants à des sorties %Qxy.i ou à des bits internes %Mi. Lors du pas en cours les bits d'ordres prennent les états binaires définis pour ce pas.

Illustration

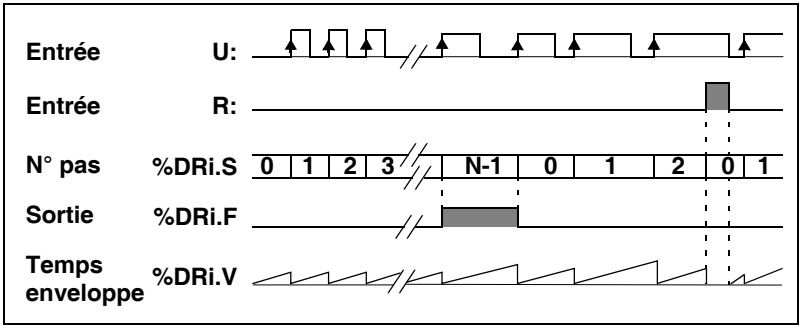
Le tableau ci-dessous résume les caractéristiques principales du programmateur cyclique (programmateur configuré avec 16 pas)



BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	30	14	15	REPERES
0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	%Q2.1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	%Q2.3
2	1	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	%Q3.5
32	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	%M0
4	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	%M10
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	%Q2.6
6	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	%Q2.7
7	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	%Q2.8
8	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	%M20
9	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	%M30
A	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	%Q2.9
B	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	%Q3.6
C	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	%M5
D	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	%M6
E	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	%M7

Dans l'exemple ci-dessus, pour le pas 1, les bits d'ordres %Q2.1;%Q3.5 ;%Q2.8;%Q3.6;%M5 et %M6 sont mis à l'état 1, les autres bits d'ordres sont mis à 0.

Diagramme de fonctionnement



Le numéro du pas en cours est incrémenté à chaque front montant sur l'entrée U (ou activation de l'instruction U). Ce numéro peut être modifié par programme.

Programmation et configuration du bloc fonction programmateur cyclique (Drum)

Exemple

Dans cet exemple, les 5 premières sorties %Q2.0 à %Q2.4 sont activées les unes à la suite des autres, chaque lois que l'entrée %I1.1 est mise à 1. L'entrée I1.0 réinitialise les sorties au pas 0.

Configuration

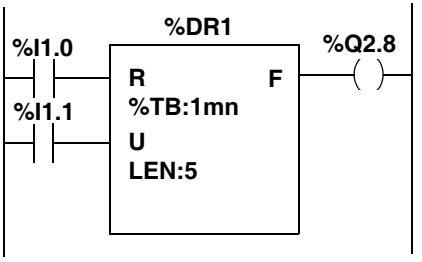
Les informations suivantes sont définies dans l'éditeur de variables:

- nombre de pas : (LEN:5),
- base de temps (TB : 1mn),
- états des sorties (bits d'ordres) pour chaque pas du programmateur.

		Pas :					
		0	1	2	3	4	
	0 :	1	0	0	0	0	%Q2.0
	1 :	0	1	0	0	0	%Q2.1
Bits :	2 :	0	0	1	0	0	%Q2.2
	3 :	0	0	0	1	0	%Q2.3
	4 :	0	0	0	0	1	%Q2.4

Programmation

Langage à contact



Langage liste d'instructions

```
LD %I1.0
R %DR1
LD %I1.1
U %DR1
LD %DR1.F
ST %Q2.8
```

Langage littéral structuré

```
IF %I1.0 THEN
  RESET %DR1;
END_IF;
IF RE %I1.1 THEN
  UP %DR1;
END_IF;
%Q2.8:=%DR1.F;
```

Remarques

En langage littéral structuré, 2 instructions permettent de programmer les blocs fonctions programmateur cyclique :

- RESET %DRi : Initialise le programmateur au pas 0,
- UP %DRi : Provoque l'avance d'un pas du programmateur et la mise à jour des bits d'ordre. Cette instruction réalise un front montant sur l'entrée U du bloc fonction, son utilisation doit donc être impulsionnelle.

Note : Sur remise à 0 (entrée R, instruction R ou instruction RESET) :

- En langage à contacts, l'historique de l'entrée U est mise à jour avec les valeurs câblées.
- En langage listes d'instructions, l'historique de l'entrée U n'est pas mis à jour ; elle garde la valeur qu'elle avait avant l'appel.
- En langage littéral structuré, l'historique de U est mis à jour avec 0.

Cas spécifiques

- **Incidence d'une "reprise à froid" :** (%S0=1) provoque la réinitialisation du programmateur au pas 0 (avec mise à jour des bits d'ordres).
- **Incidence d'une "reprise à chaud : "**(%S1=1) provoque la mise à jour des bits d'ordre, suivant le pas en cours.
- **Incidence d'un saut de programme, désactivation de la tâche et point d'arrêt :** le fait de ne pas scruter le programmateur cyclique ne provoque pas de remise à 0 des bits d'ordres.
- **Mise à jour des bits d'ordre :**ne s'effectue que lors d'un changement de pas ou lors d'une reprise à chaud ou à froid.

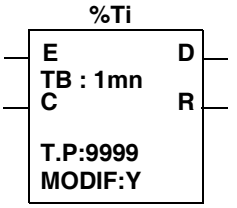
Présentation du bloc fonction temporisateur (Timer) série 7

Généralités

Ce bloc fonction temporisateur compatible avec les blocs série 7 PL7-2/3 permet de commander de façon temporisé des actions spécifiques. La valeur de ce retard est programmable et peut être modifiable ou non par terminal.

Illustration

Représentation graphique du bloc fonction temporisateur série 7



Caractéristiques

Caractéristiques du bloc fonction temporisateur série 7

Caractéristique	Repère	Valeur
Numéro	%Ti	0 à 63 pour un TSX 37, 0 à 254 pour un TSX 57.
Base de temps	TB	1mn, 1s, 100ms, 10ms (1mn par défaut).
Valeur courante	%Ti.V	Mot qui décroît de %Ti.P vers 0 sur écoulement du temporisateur. Peut être lu, testé, mais non écrit.
Valeur de présélection	%Ti.P	$0 \leq \%\mathrm{Ti.P} \leq 9999$. Mot qui peut être lu, testé, écrit. Est mis à la valeur 9999 par défaut. La durée est égale à %Ti.P x TB.
Modification MODIF	Y/N	• Y : possibilité de modification de la valeur de présélection en réglage. • N : pas d'accès en réglage.
Entrée "Armement"	E(Enable)	Sur état 0 réinitialise le temporisateur %Ti.V = %Ti.P.
Entrée "Contrôle"	C(Control)	Sur état 0 gèle la valeur courante %Ti.V.
Sortie "Temporisateur écoulé"	D(Done)	Le bit associé %Ti.D = 1, si temporisateur écoulé %Ti.V = 0.

Caractéristique	Repère	Valeur
Sortie "Temporisateur en cours"	R(Running)	Le bit associé %Ti.R = 1, si temporisateur %Ti.P > %Ti.V > 0 et si entrée C est à l'état 1.

Note : Les blocs fonctions %Ti ne sont pas programmables en liste d'instructions, les objets de blocs %Ti (%Ti.V, %Ti.P, %Ti.D et %Ti.R) sont par contre accessibles. Le nombre total de %Tmi + %Ti doit être inférieur à 64 sur le TSX 37 et 255 sur le TSX 57.

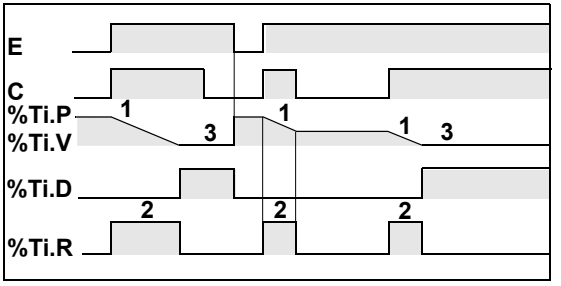
Fonctionnement du bloc fonction temporisateur (Timer) série 7

Généralités

Le temporisateur évolue lorsque ses 2 entrées (E et C) sont à l'état 1. Il se comporte comme un décompteur.

Illustration

Diagramme de fonctionnement du temporisateur série 7



E	0	0	1	1
C	0	1	0	1
%Ti.V %Ti.P	%Ti.V = %Ti.P	%Ti.V = %Ti.P	%Ti.V gelée	%Ti.V décroît de %Ti.P -> 0
%Ti.D	0	1	0	1 si Tempo écoulée
%Ti.R	0	1	0	1 si Tempo en cours

Fonctionnement

Description du fonctionnement

Phase	Description
1	La valeur courante %Ti.V décroît de la présélection %Ti.P vers 0, d'une unité à chaque impulsion de la base de temps TB,
2	le bit de sortie %Ti.R (Temporisateur en cours) associé à la sortie R est alors à l'état 1, le bit de sortie %Ti.D (Temporisateur écoulé) associé à la sortie D est à l'état 0,
3	lorsque la valeur courante %Ti.V=0, %Ti.D passe à l'état 1 et %Ti.R repasse à l'état 0.

Instructions

En langage littéral structuré, 3 instructions permettent de programmer les blocs fonctions temporisateur %Ti

- PRESET %Ti : Réinitialise le temporisateur
- START %Ti : Provoque l'écoulement du temporisateur
- STOP %Ti : Gèle la valeur courante du temporisateur

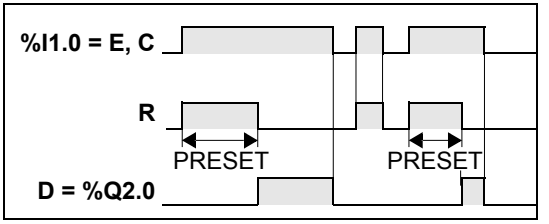
Programmation du temporisateur série 7 en mode "Retard à l'enclenchement"

Généralités

Selon sa programmation, le bloc fonction "Temporisateur" peut réaliser différentes fonctions. Ici est décrit la fonction de "retard à l'enclenchement".

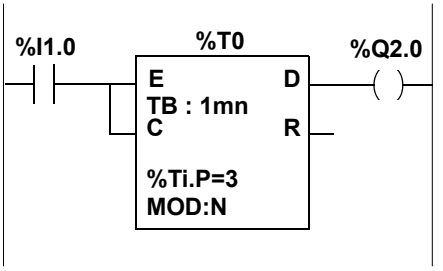
Illustration

Diagramme de fonctionnement de la fonction retard à l'enclenchement



Programmation

Programmation en langage à contacts



Programmation en langage littéral structuré

```
IF %I1.0 THEN
  START %T0;
ELSE
  PRESET %T0;
END_IF;
%Q2.0:=%T0.D;
```

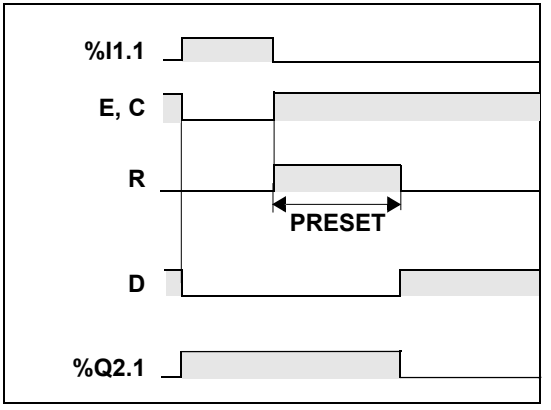
Programmation du temporisateur série 7 en mode "Retard au déclenchement

Généralités

Selon sa programmation, le bloc fonction "Temporisateur" peut réaliser différentes fonctions. Ici est décrit la fonction de "retard au déclenchement".

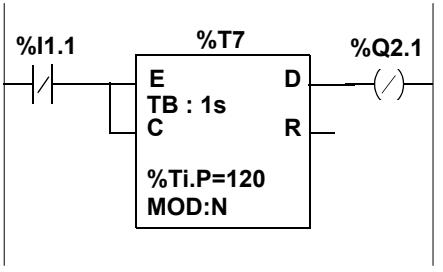
Illustration

Diagramme de fonctionnement de la fonction retard au déclenchement



Programmation

Programmation en langage à contacts



Programmation en langage littéral structuré

```
IF %I1.1 THEN
  PRESET %T7;
ELSE
  START %T7;
END_IF;
%Q2.1:=NOT%T7.D;
```

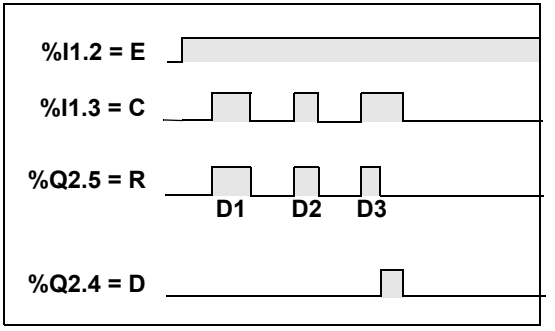
Programmation du temporisateur série 7 en mode "Retard cumulé à l'enclenchement"

Généralités

Selon sa programmation, le bloc fonction "Temporisateur" peut réaliser différentes fonctions. Ici est décrit la fonction de "retard cumulé à l'enclenchement".

Illustration

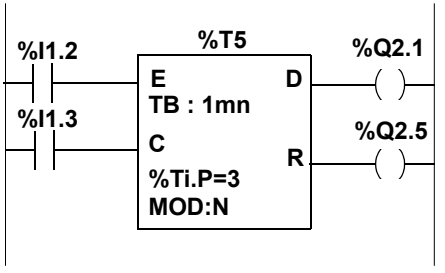
Diagramme de fonctionnement de la fonction retard cumulé à l'enclenchement



PRESET = D1+D2+D3

Programmation

Programmation en langage à contacts



Programmation en langage littéral structuré

```
IF %I1.2 THEN
  IF %I1.3 THEN
   START %T5;
  ELSE
   STOP %T5;
  END_IF;
ELSE
   PRESET %T5;
END_IF;
%Q2.4:=%T5.D;
%Q2.5:=%T5.R;
```

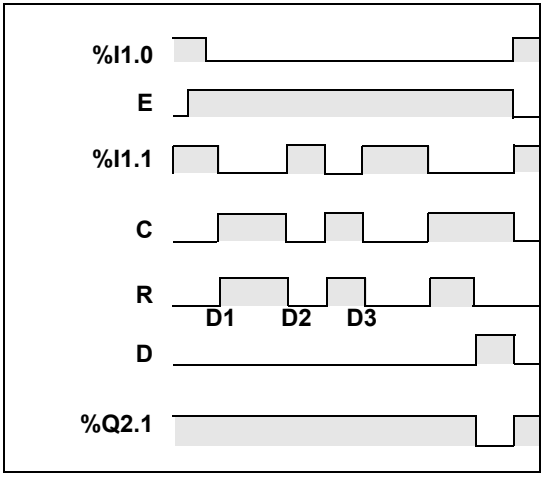
Programmation du temporisateur série 7 en mode "Retard cumulé au déclenchement"

Généralités

Selon sa programmation, le bloc fonction "Temporisateur" peut réaliser différentes fonctions. Ici est décrit la fonction de "retard cumulé au déclenchement".

Illustration

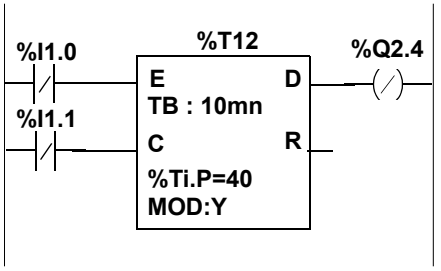
Diagramme de fonctionnement de la fonction retard cumulé au déclenchement



PRESET = D1+D2+D3

Programmation

Programmation en langage à contacts



Programmation en langage littéral structuré

```
IF %I1.0 THEN
   PRESET %T12;
ELSE
 IF %I1.1 THEN
   STOP %T12;
 ELSE
  START %T12;
 END_IF;
END_IF;
%Q2.4:=NOT %T12.D;
```

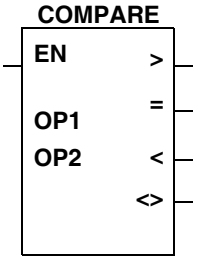
Présentation du bloc opération comparateur vertical

Généralités

Le bloc comparateur vertical permet d'effectuer la comparaison entre 2 opérandes (OP). Ces 2 opérandes sont de type mot de 16 bits éventuellement indexés ou valeur immédiate.
Le nombre de blocs comparateur vertical n'est pas limité et n'est pas numéroté.

Illustration

Représentation graphique du bloc opération comparateur vertical



Caractéristiques

Caractéristiques du bloc opération comparateur vertical

Caractéristique	Repère	Valeur
Entrée de commande	EN	Sur état 1 provoque la comparaison des deux opérandes.
Sortie "Supérieur"	>	Est à l'état 1 si le contenu de OP1 est supérieur à celui de OP2.
Sortie "Egal"	=	Est à l'état 1 si le contenu de OP1 est égal à celui de OP2.
Sortie "Inférieur"	<	Est à l'état 1 si le contenu de OP1 est inférieur à celui de OP2.
Sortie "Différent"	<>	Est à l'état 1 si le contenu de OP1 est différent de celui de OP2.
Opérande numéro 1	OP1	Cet opérande est un objet mot simple longueur (il peut être indexé).
Opérande numéro 2	OP2	Cet opérande est un objet mot simple longueur (il peut être indexé).

Fonctionnement du bloc opération comparateur vertical

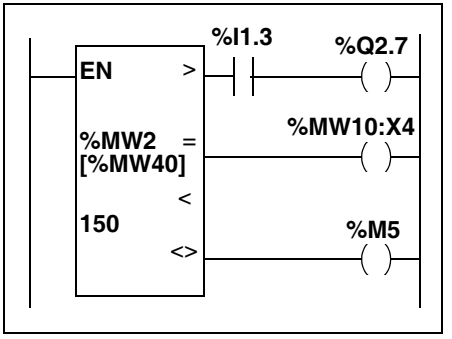
Fonctionnement

La mise à 1 de l'entrée de commande provoque la comparaison des deux opérandes, les quatre sorties sont activées en fonction du résultat de la comparaison. La mise à 0 de l'entrée de commande provoque la mise à zéro des sorties activées.

Exemple

Le programme ci-dessous montre la comparaison du mot %MW2 indexé par le mot %MW40 et de la valeur immédiate 150. Dans le cas où le contenu de %MW2[%MW40] est supérieur à 150 et si %I1.3 = 1 la bobine %Q2.7 est commandée. Si ce contenu est égal à 150 la bobine %MW10:X4 est commandée. La bobine %M5 n'est pilotée que si le contenu est différent de 150 (< ou >).

Langage à contact



Note : Ce bloc fonction n'existe pas en langage liste d'instructions et en langage littéral structuré. Utiliser les opérations de comparaison >, <, =, <>.

Cas spécifiques

- **Incidence d'une reprise "à froid" :** (%S0) provoquant une remise à zéro de l'opérande OP1 et éventuellement OP2 (si OP2 est un mot interne), les sorties sont activées en fonction de la comparaison avec les nouvelles valeurs.
- **Incidence d'une reprise "à chaud" :** (%S1) n'a pas d'incidence sur le bloc comparaison.

2.3 Instructions de décalage

Instructions de décalage

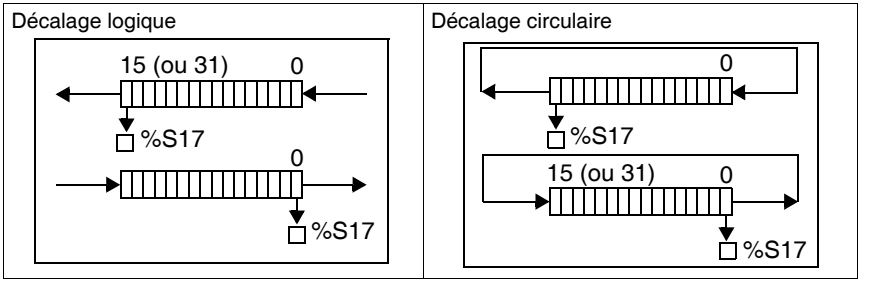
Généralités

Les instructions de décalage consistent à déplacer les bits d'un opérande mot ou double mot d'un certain nombre de positions vers la droite ou vers la gauche. Il existe deux types de décalages :

- **le décalage logique :**
  - SHL(op2,i) décalage logique à gauche de i positions.
  - SHR(op2,i) décalage logique à droite de i positions.
- **Le décalage circulaire :**
  - ROL(op2.i) décalage circulaire à gauche de i positions.
  - ROR(op2.i) décalage circulaire à droite de i positions.

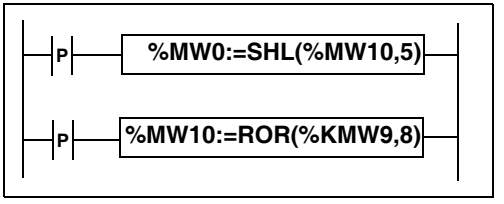
Si l'opérande à décaler est un opérande simple longueur, la variable i sera comprise entre 1 et 16. Si l'opérande à décaler est un opérande double longueur, la variable i sera comprise entre 1 et 32. L'état du dernier bit sorti est mémorisé dans le bit %S17.

Illustration des deux types de décalages



Structure

Langage à contact :



Langage liste d'instructions :

```
LDR %I1.1
[%MW0:=SHL(%MW10,5)]
```

Langage littéral structuré :

```
IF RE%I1.2 THEN
 %MW10:=ROR(%KW9,8);
END_IF;
```

Syntaxes

Opérateurs : SHL, SHR, ROL, ROR
Opérandes :

Type	Opérande 1 (Op1)	Opérande 2 (Op2
Mots indexables	%MW	%MW, %KW, %Xi.T
Mots non indexables	%QW, %SW, %NW, %BLK	Val.imm., %IW, %QW, %SW, %NW, %BLK, Expr. num.
Mots doubles indexables	%MD	%MD, %KD
Mots doubles non indexables	%QD, %SD	Val.imm., %ID, %QD, %SD, Expr. num.

Syntaxe : Op1:=Opérateur(Op2,i)

2.4 Instructions sur flottant

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions sur flottant du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Instructions sur flottant

Généralités

Le logiciel PL7 permet d'effectuer des opérations sur objets flottants.

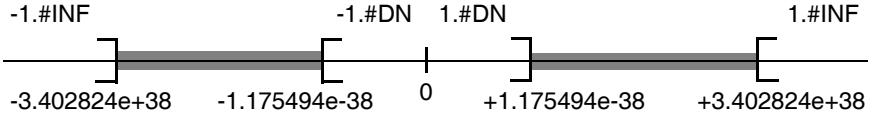
Le format flottant utilisé est celui de la norme IEEE STD 734-1985 (équivalence IEC 559). La longueur des mots est de 32 bits, ce qui correspond à des nombres flottants simple précision.

Les valeurs flottantes peuvent être représentées avec ou sans exposant, elles doivent toujours comporter un point (virgule flottante).

Exemples de valeurs flottantes :
sans exposant : 1285.28
avec exposant : 1.28528e3

Les valeurs flottantes sont comprises entre -3.402824e+38 et -1.175494e-38 ; 1.175494e-38 et 3.402824e+38 (valeurs grisées sur le schéma). Elles comportent aussi la valeur 0 notée 0.0

.



Lorsqu'un résultat de calcul est compris entre -1.175494e-38 et 1.175494e-38, il est arrondi à 0. Une valeur comprise entre ces bornes ne peut être saisie en flottant si elle est saisie dans un autre format, en flottant le symbole 1.#DN ou - 1.#DN sera affiché.

Lorsqu'un résultat de calcul est :
- inférieur à -3.402824e+38, le symbole -1.#INF (pour -infini) est affiché,
- supérieur à +3.402824e+38, le symbole 1.#INF (pour + infini) est affiché.

Lorsque le résultat d'une opération est indéfini (exemple racine carré d'un nombre négatif) le symbole 1.#NAN ou -1.#NAN est affiché.

Le bit système %S18 est positionné à 1 lorsque le résultat n'est pas dans les bornes valides.

Les bits du mot d'état %SW17 indiquent la cause d'un défaut sur une opération flottante.

Différents bits du mot SW17:

%SW17:X0	opération invalide, le résultat n'est pas un nombre (1.#NAN ou -1.#NAN)
%SW17:X1	opérande non normalisé (compris entre -1.175494e-38 et 1.175494e-38), le résultat est arrondi à 0.
%SW17:X2	division par 0, le résultat est infini (-1.#INF ou 1.#INF).
%SW17:X3	résultat supérieur en valeur absolu à +3.402824e+38, le résultat est infini (-1.#INF ou 1.#INF)
%SW17:X4	résultat inférieur à 1.175494e-38, le résultat est 0.
%SW17:X5	imprécision sur le résultat.

Ce mot est remis à 0 par le système sur démarrage à froid et par le programme pour réutilisation.

La précision de représentation est de 2-24. Pour la visualisation de nombre flottant, il est inutile d'afficher plus de 6 chiffres après la virgule.

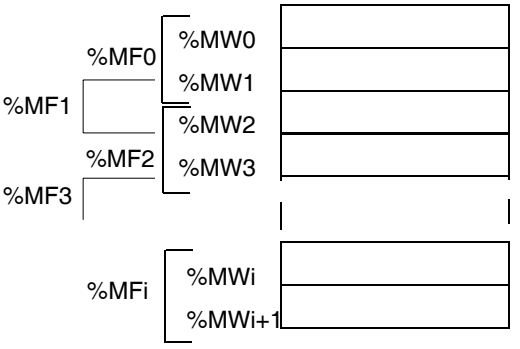
Note :
- la valeur "1285" est interprétée en tant que valeur entière, pour être prise en compte comme valeur flottante elle doit être écrite : "1285.0".
- les instructions de conversion Entier <--> Flottant permettent de passer d'un format à l'autre.

Adressage des objets flottants

Abréviations	Adressage complet	Type de flottant	Accès	Forme indexée
Val.imm.	-	Valeurs immédiates	R	-
%MF	%MFi	flottant interne	R/W	%MFi[index]
%KF	%KFi	constante flottante	R	%KFi[index]

Possibilité de recouvrement entre objets :
Les mots simple, double longueur et flottant sont rangés à l'intérieur de l'espace données dans une même zone mémoire. Ainsi, le mot flottant %MFi correspond aux mots simple longueur %MWi et %MWi+1 (le mot %MWi renfermant les poids faibles et le mot %MWi+1 les poids forts du mot %MFi).

Illustration :



Exemple :
%MF0 correspond à %MW0 et %MW. %KF543 correspond à %KW543 et %KW544.

Instructions de comparaison sur flottant

Généralités

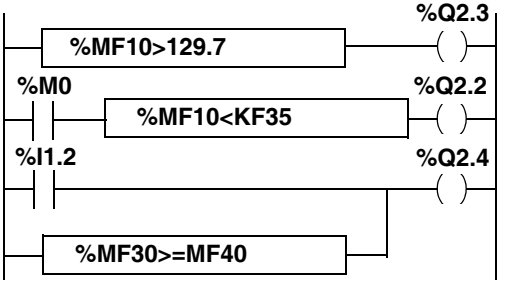
Les instructions de comparaison permettent de comparer deux opérandes.

>	teste si l'opérande 1 est supérieur à l'opérande 2,
>=	teste si l'opérande 1 est supérieur ou égal à l'opérande 2,
<	teste si l'opérande 1 est inférieur à l'opérande 2,
<=	teste si l'opérande 1 est inférieur ou égal à l'opérande 2,
=	teste si l'opérande 1 est égal à l'opérande 2,
<>	teste si l'opérande 1 est différent de l'opérande 2.

Le résultat est à 1 lorsque la comparaison demandée est vraie.

Structure

Langage à contacts



Les blocs de comparaison se programment en zone de test.

Langage liste d'instructions

```
LD [%MF10>129.7]
ST %Q2.3
LD %M0
AND [%MF10<KF35]
ST %Q2.2
LD %I1.2
OR [%MF30>=MF40]
ST %Q2.4
```

La comparaison est réalisée à l'intérieur de crochets figurant derrière des instructions LD, AND et OR.

Langage littéral structuré

```
%Q2.3:=%MF10>129.7;
%Q2.2:=(%MF20<%KF35) AND %M0;
%Q2.4:=(%MF30>=%MF40) OR %I1.2;
```

Syntaxe

Opérateurs : >, >=, <, <=, =, <>
Opérandes :

Type	Opérandes 1 et 2 (Op1 et Op2)
Flottants indexable	%MF,%KF
Flottants non indexable	Valeur immédiate flottante, Expression numérique flottante.

Note : En langage liste d'instructions, les instructions de comparaison peuvent être utilisées au sein de parenthèses.

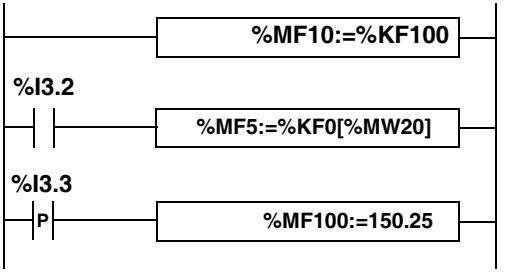
Instructions d'affectation sur flottant

Généralités

Les opérations d'affectation sur flottants suivantes peuvent être réalisées :
- flottant (indexé) -> flottant (indexé). Voir ex. 1
- valeur flottante immédiate -> flottant (indexé). Voir ex. 2

Structure

Langage à contacts :



Langage liste d'instructions :
Ex. 1

```
LD TRUE
[%MF10:=%KF10]

LD %I3.2
[%MF5:=%KF0[%MW20]]
```

Ex. 2

```
LDR %I3.3
[%MF100:=150.25]
```

Langage littéral structuré :
Ex. 1

```
%MF10:=%KF10;
IF %I3.2 THEN
  %MF5:=%KF0[%MW20];
END_IF;
```

Ex. 2

```
IF RE %I1.3 THEN
  %MF100:=150.25;
END_IF;
```

Syntaxe

Opérateurs : :=
Opérandes :

Type	Opérandes 1 (Op1)	Opérandes 2 (Op2
Flottants indexable	%MF	%MF, %KF
Flottants non indexable		Valeur immédiate flottante, Expression numérique flottante.

Syntaxe : Op1:=Op2

Note : Il est possible de réaliser des affectations multiples.
Exemple : `%MF0:=%MF2:=%MF4`

Instructions arithmétiques sur flottant

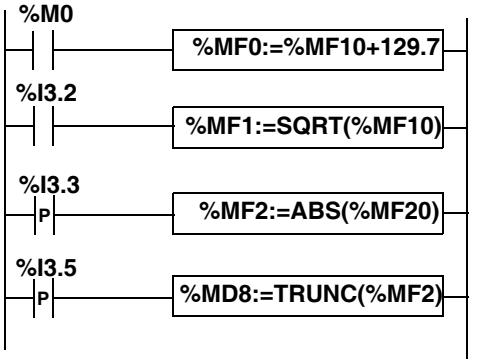
Généralités

Ces instructions permettent de réaliser une opération arithmétique entre deux opérandes ou sur un opérande.

+	addition de deux opérandes	SQRT	racine carré d'un opérande
-	soustraction de deux opérandes	ABS	valeur absolue d'un opérande
*	multiplication de deux opérandes	TRUNC	partie entière d'une valeur flottante
/	division de deux opérandes		

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MF0:=%MF10+129.7]

LD %I3.2
[%MF1:=SQRT(%MF10)]

LDR %I3.3
[%MF2:=ABS(%MF20)]

LDR %I3.5
[%MD8:=TRUNC(%MF2)]
```

Langage littéral structuré

```
IF %M0 THEN
 %MF0:=%MF10+129.7;
END_IF;
IF %I3.2 THEN
 %MF1:=SQRT(%MF10);
END_IF;
IF %I3.3 THEN
 %MF2:=ABS(%MF20);
END_IF;
IF %I3.5 THEN
 %MD8:=TRUNC(%MF2);
END_IF
```

Syntaxe

Opérateurs et syntaxe des instructions arithmétiques sur flottant

Opérateurs	Syntaxe
+, - *, /	Op1:=Op2 Opérateur Op3
SQRT, ABS, TRUNC	Op1:=Opérateur(Op2)

Note : Lorsqu'on effectue une addition ou une soustraction entre 2 nombres flottants, les 2 opérandes doivent respecter la condition $\mathrm{Op1} > \mathrm{Op2} \times 2^{-24}$, avec Op1>Op2. Si cette condition n'est pas respectée le résultat est égal à l'opérande 1 (Op1). Ce comportement est sans grande conséquence lorsqu'ils s'agit d'une opération isolée, puisque l'erreur résultante est trés faible (2^{-24}), mais a des conséquences innatendues s'il sagit d'un calcul itératif.
Ex : soit l'instruction **%MF2:= %MF2 + %MF0** répétée indéfiniment. Si les conditions initiales sont %MF.0 = 1.0 et %MW2= 0, on observe un blocage de la valeur de %MF2 à 16777216.
Il est donc déconseillé de programmer sans précaution des calculs itératifs. Si on souhaite néanmoins programmer ce type de calcul, il appartient à l'applicatif client de gérer les erreurs de troncature.

Opérandes des instructions arithmétiques sur flottant:

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MF (1)	%MF, %KF
Mots non indexable	-	Valeur immédiate flottante, Expr. numérique flottante.

(1) %MD dans le cas de l'instruction TRUNC

Règles d'utilisation

- les opérations sur flottants et sur entiers ne peuvent pas être mixées directement. Les opérations de conversion (Voir *Instructions de conversions numériques, p. 134*) assurent la conversion dans l'un ou l'autre de ces formats.
- le bit système %S18 est géré de façon identique aux opérations sur entier (Voir *Instructions arithmétiques sur entiers, p. 58*), le mot %SW17 (Voir *Instructions sur flottant, p. 115*) indique la cause du défaut.

Instructions logarithmes et exponentielles

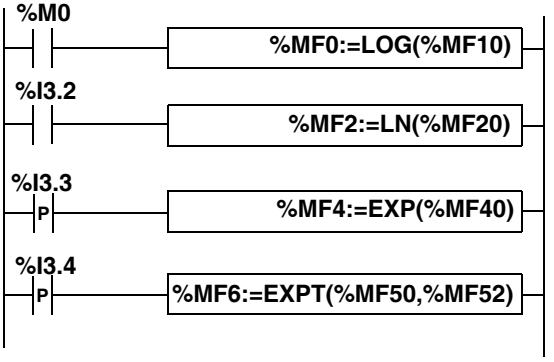
Généralités

Ces instructions permettent de réaliser des opérations logarithmiques et exponentielles.

LOG	logarithme base 10
LN	logarithme népérien
EXP	exponentielle naturelle
EXPT	exponentiation d'un réel par un réel

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MF0:=LOG(%MF10]

LD %I3.2
[%MF2:=LN(%MF20)]

LDR %I3.3
[%MF4:=EXP(%MF40)]

LDR %I3.4
[%MF6:=EXPT(%MF50,%MF52)]
```

Langage littéral structuré

```
IF %M0 THEN
 %MF0:=LOG(%MF10);
END_IF;
IF %I3.2 THEN
 %MF2:=LN(%MF20);
END_IF;
IF %I3.3 THEN
 %MF4:=EXP(%MF40);
END_IF;
IF %I3.4 THEN
 %MF6:=EXPT(%MF50,%MF52);
END_IF;
```

Syntaxe

Opérateurs et syntaxe des instructions logarithmiques et exponentielles

Opérateurs	Syntaxe
LOG, EXP, LN	Op1:=Opérateur(Op2)
EXPT	Op1:=Opérateur (Op2,Op3)

Opérandes des instructions logarithmiques et exponentielles

Type	Opérande 1 (Op1)	Opérande 2 (Op2)	Opérande 3 (Op3)
Mots indexables	%MF	%MF, %KF	%MF
Mots non indexable	-	Valeur imm. flottante Expr. num. flottante	Valeur imm. flottante

Règles d'utilisation

- lorsque l'opérande de la fonction est une valeur invalide (exemple : logarithme d'un nombre négatif), elle produit un résultat indéterminé ou infini et fait passer le bit %S18 à 1,le mot %SW17 indique la cause du défaut (Généralités (Voir *Instructions sur flottant, p. 115*)).
- dans le cas des fonctions logarithmes, pour les valeurs proches de 1.0 (comprises entre 0,99 et 1,0 ou 1,0 et 1,01), le résultat sera égal à 0, les bits %S18 et %SW17:X5 sont positionnés à 1.

Instructions Trigonométrique

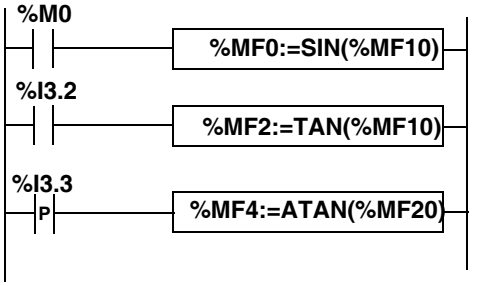
Généralités

Ces instructions permettent de réaliser des opérations trigonométriques.

SIN	sinus d'un angle exprimé en radian,	**ASIN**	arc sinus (résultat entre $-\frac{\pi}{2}$ et $\frac{\pi}{2}$)
COS	cosinus d'un angle exprimé en radian,	**ACOS**	arc cosinus (résultat entre 0 et π)
TAN	tangente d'un angle exprimé en radian.	**ATAN**	arc tangente (résultat entre $-\frac{\pi}{2}$ et $\frac{\pi}{2}$)

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MF0:=SIN(%MF10)]

LD %I3.2
[%MF2:=TAN(%MF10)]

LDR %I3.3
[%MF4:=ATAN(%MF20)]
```

Langage littéral structuré

```
IF %M0 THEN
 %MF0:=SIN(%MF10);
END_IF;
IF %I3.2 THEN
 %MF2:=TAN(%MF10);
END_IF;
IF %I3.3 THEN
 %MF4:=ATAN(%MF20);
END_IF;
```

Syntaxe

Opérateurs et syntaxe des instructions opérations trigonométriques:

Opérateurs	Syntaxe
SIN, COS, TAN, ASIN, ACOS, ATAN	Op1:=Opérateur(Op2)

Opérandes des instructions opérations trigonométriques:

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MF	%MF, %KF
Mots non indexable	-	Valeur imm. flottante Expr. num. flottante

Règles d'utilisation

- lorsque l'opérande de la fonction est une valeur invalide (exemple : arc cosinus d'un nombre supérieur à 1), elle produit un résultat indéterminé ou infini et fait passer le bit %S18 à 1, le mot %SW17 (Voir *Instructions sur flottant, p. 115*) indique la cause du défaut.
- les fonctions SIN/COS/TAN admettent en paramètre un angle entre -4096π et 4096π mais leur précision décroît progressivement pour les angles en dehors de l'intervalle -2π et $+2\pi$ en raison de l'imprécision apportée par le modulo 2π effectué sur le paramètre avant toute opération.
- Pour les valeurs 0<Op2<0.01 et 0.999<Op2<1.0 de ASIN, le bit %S18 et le bit %SW17:X5 passent à 1, signifiant une imprécision de mesure.

Instructions de conversion

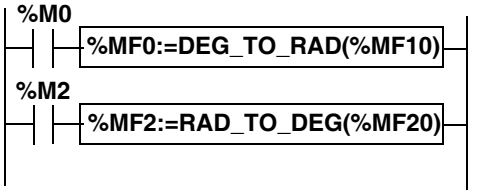
Généralités

Ces instructions permettent de réaliser des opérations de conversion.

DEG_TO_RAD	conversion de degré en radian, le résultat est la valeur de l'angle compris entre 0 et 2π
RAD_TO_DEG	cosinus d'un angle exprimé en radian, le résultat est la valeur de l'angle compris entre 0 et 360 degrés.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MF0:=DEG_TO_RAD(%MF10)]

LD %M2
[%MF2:=RAD_TO_DEG(%MF20)]
```

Langage littéral structuré

```
IF %M0 THEN
 %MF0:=DEG_TO_RAD(%MF10);
END_IF;
IF %M2 THEN
 %MF2:=RAD_TO_DEG(%MF20);
END_IF;
```

Syntaxe

Opérateurs et syntaxe des instructions de conversion:

Opérateurs	Syntaxe
DEG_TO_RAD **RAD_TO_DEG**	Op1:=Opérateur(Op2)

Opérandes des instructions de conversion:

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MF	%MF, %KF
Mots non indexable	-	Valeur imm. flottante Expr. num. flottante

Règles d'utilisation

L'angle à convertir doit être compris entre -737280.0 et +737280.0 (pour les conversions DEG_TO_RAD) ou entre -4096π et 4096π (pour les conversions RAD_TO_DEG).
Pour des valeurs non comprises entre ces bornes le résultat affiché sera + 1.#NAN, les bits %S18 et %SW17:X0 étant positionnés à 1.

Arrondi d'une valeur flottante sous format ASCII

Généralités

La fonction ROUND fournit la valeur approchée d'un nombre flottant représenté par une chaîne de caractères.

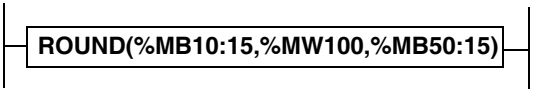
Cette fonction utilise 3 paramètres :

ROUND(chaîne 1, Long, Chaîne 2)

- **Chaîne 1 :** Tableaux d'octets constituant la chaîne de caractères origine.
- **Long :** Mot donnant la position, dans la chaîne de caractères, à partir de laquelle s'effectue l'arrondi (la position se calcule en comptant le nombre de caractères à partir de la virgule, virgule comprise).
- **Chaîne 2 :** Tableaux d'octets constituant la chaîne de caractères résultat.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[ROUND(%MB10:15,%MW100,%MB50:15)]
```

Langage littéral structuré

```
ROUND(%MB10:15,%MW100,%MB50:15);
```

Exemples

Exemples d'arrondis de valeurs flottantes ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB10:15	"-"	"1"	"."	"2"	"3"	"4"	"5"	"6"	"7"	"0"	"e"	"+"	"2"	"6"	$00

%MW100 = 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB50:15	"-"	"1"	"."	"2"	"3"	"4"	"5"	"0"	"0"	"0"	"e"	"+"	"2"	"6"	$00

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB10:15	"-"	"1"	"."	"1"	"3"	"5"	"4"	"9"	"4"	"2"	"e"	"-"	"3"	"0"	$00

%MW100 = 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB50:15	"+"	"1"	"."	"1"	"0"	"0"	"0"	"0"	"0"	"0"	"e"	"-"	"3"	"0"	$00

Syntaxe

Opérateurs et syntaxe des instructions de conversion:

Opérateurs	**Syntaxe**
ROUND	Op(chaîne 1,Long, chaîne 2)

Opérandes des instructions de conversion:

Type	**Opérande 1 (Op1)**	**Opérande 2 (Op2)**
Tableaux d'octets	%MB:15	-
Mots non indexable	-	%MW

Règles d'utilisation

- La longueur des chaînes de caractères origine et résultat doit être comprise entre 15 et 255 octets. Dans le cas contraire, le bit %S15 est positionné à 1.
- Le paramètres de longueur Long doit être compris entre 0 et 8. Dans le cas contraire, le bit %S20 (débordement d'index) est positionné à 1. Cas particulier: pour L=0 ou L=8 l'arrondi n'est pas effectué (chaîne origine = chaîne résultat).
- Lorsque le dernier caractère différent de 0 est > à 5, le caractère le précédant est incrémenté.

2.5 Instructions de conversions numériques

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions sur flottant du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Instructions de conversion BCD <-> Binaire

Généralités

Six instructions de conversion sont proposées.

Liste des instructions :

BCD_TO_INT	conversion d'un nombre BCD 16 bits en entier 16 bits
INT_TO_BCD	conversion d'un entier 16 bits en nombre BCD 16 bits
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

Rappel sur le code BCD

Le code BCD (Binary Coded Decimal) qui signifie Décimal codé binaire permet de représenter un chiffre décimal 0 à 9 par un ensemble de 4 bits. Un objet mot de 16 bits peut ainsi contenir un nombre exprimé sur 4 chiffres (0 < N < 9999).

Equivalence entre décimale et BCD :

Décimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

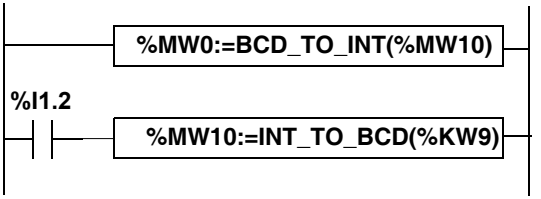
Exemples de codage BCD :

- Mot %MW5 exprimant la valeur BCD "2450" correspondant à la valeur binaire : 0010 0100 0101 0000
- Mot %MW12 exprimant la valeur décimale "2450" correspondant à la valeur binaire : 0000 1001 1001 0010

Le passage du mot %MW5 au mot %MW12 s'effectue par l'instruction BCD_TO_INT.
Le passage du mot %MW12 au mot %MW5 s'effectue par l'instruction INT_TO_BCD.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MW0:=BCD_TO_INT(%MW10)]

LD I1.2
[%MW10:=INT_TO_BCD(%KW9)]
```

Langage littéral structuré

```
%MW0:=BCD_TO_INT(%MW10);
IF %I1.2 THEN
 %MW10:=INT_TO_BCD(%KW9);
END_IF;
```

Syntaxe

Opérateurs et syntaxe (conversion d'un nombre de 16 bits) :

Opérateurs	Syntaxe
BCD_TO_INT	Op1=opérateur(Op2)
INT_TO_BCD	
INT_TO_DBCD	

Opérandes (conversion d'un nombre de 16 bits) :

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexable	%QW,%SW,%NW, %BLK	Val. imm.,%IW,%SW%NW,%BLK,Expr. num
Mots doubles indexables	%MD	-
Mots doubles non indexables	%QD,%SD	-

Opérateurs et syntaxe (conversion d'un nombre de 32 bits):

Opérateurs	Syntaxe
DBCD_TO_DINT	Op1=opérateur(Op2)
DINT_TO_DBCD	
DBCD_TO_INT	

Opérandes (conversion d'un nombre de 32 bits):

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexable	%QW,%SW,%NW, %BLK	-
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD,%SD	Val. imm.,%ID,%QD%SD,Expr. num

Exemple d'applications

L'instruction BCD_TO_INT s'utilise pour traiter une valeur de consigne présente en entrée automate sur des roues codeuses encodées BCD.
L'instruction INT_TO_BCD s'utilise pour afficher des valeurs numériques (ex: résultat de calcul, valeur courante de bloc fonction) sur afficheurs codés BCD.

Règles d'utilisation

- Conversion BCD->Binaire
  Les instructions de conversion BCD-->Binaire vérifient que l'opérateur de conversion s'effectue bien sur une valeur codée en BCD. Dans le cas où la valeur n'est pas une valeur BCD, le bit système %S18 est mis à 1 et le résultat retourne la valeur du premier quartet qui fait défaut.
  Ex : `BCD_TO_INT(%MW2)` avec %MW2=4660 donne comme résultat 1234. Par contre %MW2=242 (16#00F2) provoque la mise à 1 de %S18 et le résultat est 15.
  Pour l'instruction DBCD_TO_INT, si le nombre BCD est supérieur à 32767, le bit système %S18 est mis à 1 et la valeur -1 est rangée dans le résultat.
- Conversion Binaire->BCD
  Lorsque le dernier caractère différent de 0 est > à 5, le caractère le précédant est incrémenté.
  L'instruction INT_TO_BCD (ou DINT_TO_BCD) vérifie que l'opérateur de conversion s'effectue bien sur une valeur comprise entre 0 et 9999 (ou 0 et 9999 9999). Dans le cas contraire, le bit système %S18 est mis à 1 et le résultat retourne la valeur du paramètre d'entrée.
  Ex : `INT_TO_BCD (%MW2)` avec %MW2=2478 donne comme résultat 9336. Par contre %MW2=10004 provoque la mise à 1 de %S18 et le résultat est 10004.
  Pour l'instruction INT_TO_DBCD, si le paramètre d'entrée est négatif, le bit système %S18 est mis à 1 et le résultat retourne la valeur du paramètre d'entrée.

Instructions de conversion Entier <-> Flottant

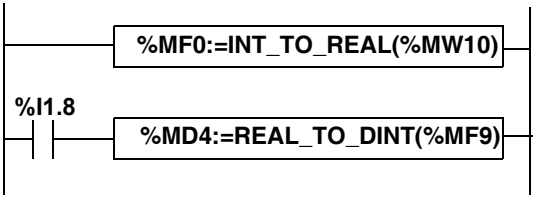
Généralités

Quatre instructions de conversion sont proposées.
Liste des instructions de conversion entier<-> flottant :

INT_TO_REAL	conversion d'un mot entier --> flottant
DINT_TO_REAL	conversion double mot entier --> flottant
REAL_TO_INT	conversion flottant --> mot entier (le résultat est la valeur algébrique la plus proche)
REAL_TO_DINT	conversion flottant --> double mot entier (le résultat est la valeur algébrique la plus proche)

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MF0:=INT_TO_REAL(%MW10)]

LD I1.8
[%MD4:=REAL_TO_DINT(%MF9)]
```

Langage littéral structuré

```
%MF0:=INT_TO_REAL(%MW10);
IF %I1.8 THEN
 %MD4:=REAL_TO_DINT(%MF9);
END_IF;
```

Syntaxe

Opérateurs et syntaxe (conversion d'un mot entier --> flottant) :

Opérateurs	Syntaxe
INT_TO_REAL	Op1=INT_TO_REAL(Op2)

Opérandes (conversion d'un mot entier --> flottant) :

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	-	%MW,%KW,%Xi.T
Mots non indexable	-	Val. imm.,%IW,%QW,%SW%NW,%BLK,Expr. num
Mots flottants indexables	%MF	-

Exemple : conversion mot entier --> flottant : 147 --> 1.47e+02

Opérateurs et syntaxe (conversion double mot entier --> flottant) :

Opérateurs	Syntaxe
DINT_TO_REAL	Op1=DINT_TO_REAL(Op2)

Opérandes (conversion double mot entier --> flottant) :

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	-	%MD,%KD
Mots non indexable	-	Val. imm.,%ID,%QD%SD,Expr. num
Mots flottants indexables	%MF	-

Exemple : conversion mot double entier --> flottant : 68905000 --> 6.8905e+07

Opérateurs et syntaxe (conversion flottant --> mot entier ou mot double entier) :

Opérateurs	Syntaxe
REAL_TO_INT	Op1=Opérateur(Op2)
REAL_TO_DINT	

Opérandes (conversion flottant --> mot entier ou mot double entier) :

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MW	-
Mots non indexable	%QW,%NW,%BLK	-

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots doubles indexables	%MD	-
Mots doubles non indexables	%QD	-
Mots flottants indexables	-	%MF,%KF
Mots flottants non indexables	-	Val. imm. flottante

Exemple :
conversion flottant --> mot entier : 5978.6 --> 5979
conversion flottant --> mot double entier : -1235978.6 --> -1235979

Note : Si lors d'une conversion réel vers entier (ou réel vers entier double mot) la valeur flottante est en dehors des bornes du mot (ou du double mot), le bit %S18 est positionné à 1.

Précision d'arrondi

La norme IEEE 754 définit 4 modes d'arrondi pour les opérations sur flottant.
Le mode utilisé par les instructions ci-dessus est le mode "arrondi au plus près":
"si les valeurs représentables les plus proches sont à égale distance du résultat théorique, la valeur founie sera celle dont le bit de poids faible est égal à 0".
Dans certains cas, le résultat de l'arrondi peut donc prendre une valeur par défaut ou un valeur par excès.

Par exemple:
Arrondi de la valeur 10.5 -> 10
Arrondi de la valeur 11.5 -> 12

Instructions de conversion Gray <-> Entier

Généralités

L' instruction GRAY_TO_INT convertit un mot de code Gray en entier (code binaire pur).

Rappel sur le code Gray

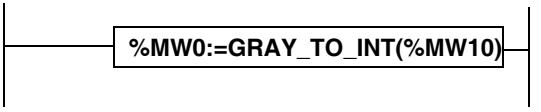
Le code Gray ou "binaire réfléchi" permet de coder une valeur numérique en cours d'évolution en une suite de configurations binaires se différenciant l'une de l'autre par le changement d'état d'un et d'un seul bit.
Ce code permet par exemple d'éviter l'aléa suivant : en binaire pur, le passage de la valeur 0111 à 1000 peut engendrer des valeurs aléatoires comprises entre 0 et 1000, les bits ne changeant pas de valeur de façon parfaitement simultanée.

Equivalence entre décimale, BCD et Gray :

Décimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Gray	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MW0:=GRAY_TO_INT(%MW10)]
```

Langage littéral structuré

```
%MW0:=GRAY_TO_INT(%MW10);
```

Syntaxe

Opérateurs et syntaxe :

Opérateurs	Syntaxe
GRAY_TO_INT	Op1=GRAY_TO_INT(Op2)

Opérandes:

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Mots indexables	%MW	%MW,%KW,%Xi.T
Mots non indexable	%QW,%SW,%NW, %BLK	Val.imm.,%IW,%QW,%SW,%NW,%BLK ,Expr. num.

Instructions de conversion mot <--> double mot

Généralités

Les instructions décrites ci-après sont utiles dans le cas d'objets purement symboliques (cas des blocs fonction DFB).
Dans le cas des objets adressables les mécanismes de recouvrement (exemple : double mot %MD0 est constitué des mots %MW0 et %MW1) rend inutile l'usage de ces instructions.

Liste des instructions de conversion mot <--> double mot :

LW	Instructions d'extraction du mot de poids faible d'un double mot
HW	Instructions d'extraction du mot de poids fort d'un double mot
CONCATW	Instructions de concaténation de 2 mots

Syntaxe

Opérateurs et syntaxe des instructions d'extraction du mot de poids faible d'un double mot :

Opérateurs	Syntaxe
LW	Op1=LW(Op2)

Opérandes des instructions d'extraction du mot de poids faible d'un double mot :

Op1	Mot simple longueur (type Word)
Op2	Mot double longueur (type DWord)

Exemple :

```
Pression_cuve:=LW(Parametre_1)
```

si `Parametre_1=16#FFFF1234`, `Pression_cuve=16#1234`

Opérateurs et syntaxe des instructions d'extraction du mot de poids fort d'un double mot :

Opérateurs	Syntaxe
HW	Op1=HW(Op2)

Opérandes des instructions d'extraction du mot de poids fort d'un double mot :

Op1	Mot simple longueur (type Word)
Op2	Mot double longueur (type DWord)

Exemple :

```
Pression_cuve:=HW(Parametre_1)
si Parametre_1=16#FFFF1234, Pression_cuve=16#FFFF
```

Opérateurs et syntaxe des instructions de concaténation de 2 mots simples et le transfert dans un double mot :

Opérateurs	Syntaxe
CONCATW	Op1=CONCATW(Op2, Op3)

Opérandes des instructions de concaténation de 2 mots simples et le transfert dans un double mot :

Op1	Mot double longueur (type DWord)
Op2	Mot simple longueur (type Word)
Op3	Mot simple longueur (type Word)

Exemple :

```
Pression_cuve:=CONCATW(Parametre_1,Parametre_2)
```

si `Parametre_1=16#1234`, `Parametre_1=16#FFFF`,
`Pression_cuve=16#FFFF1234`

2.6 Instructions sur tableaux de mots

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions sur tableaux de mots du langage PL7

Contenu de ce sous-chapitre

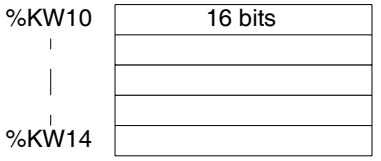
Ce sous-chapitre contient les sujets suivants :

Instructions sur tableaux de mots

Généralités

Le logiciel PL7 permet d'effectuer des opérations sur tableaux :
- de mots,
- de doubles mots,
- de mots flottants.

Les tableaux de mots sont des suites de mots adjacents de même type et de longueur définie : L
Exemple de tableau de mots : %KW10:5



Caractéristiques des tableaux de mots

Type	Format	Adresse maximum	Taille	Accès écriture
Mots internes	Simple longueur	%MWi:L	i+L<=Nmax (1)	Oui
	Double longueur	%MWDi:L	i+L<=Nmax-1 (1)	Oui
	Flottant	%MFi:L	i+L<=Nmax-1 (1)	Oui
Mots constants	Simple longueur	%KWi:L	i+L<=Nmax (1)	Non
	Double longueur	%KWDi:L	i+L<=Nmax-1 (1)	Non
	Flottant	%KFi:L	i+L<=Nmax-1 (1)	Non
Mots système	Simple longueur	%SW50:4 (2)	-	Oui

(1) Nmax = nombre maximum de mots défini en configuration logicielle
(2) Seules les mots %SW50 à %SW53 peuvent être adressés sous forme de tableaux.

Règle générales sur les opérations de tableaux

- les opérations sur tableaux ne s'effectuent que sur des tableaux contenant des objets de même types,
- les opérations sur tableaux ne s'effectuent que sur 2 tableaux maximum
- si dans une opération les tableaux ont des tailles différentes, le tableau résultat correspondra au minimum des 2 tableaux opérandes,
- l'utilisateur doit éviter d'effectuer des opérations sur tableaux avec recouvrement (par exemple : `%MW100[20]:=%MW90[20]+%KW100[20]`),
- l'opération sur 2 tableaux s'effectuent sur chaque élément de même rang des 2 tableaux et le résultat est transféré dans l'élément de même rang du tableau résultat,
- si lors d'une opération entre 2 éléments le bit système %S18 est positionné à 1, alors le résultat pour cette opération est erroné, mais l'opération pour les éléments suivants est effectuée correctement,
- lorsqu'un des opérandes est une expression numérique, celle-ci doit être placée entre parenthèse,
- le rang d'un mot dans un tableau correspond à la position du mot dans le tableau, la première position correspond au rang 0.

Instructions arithmétiques sur tableaux

Généralités

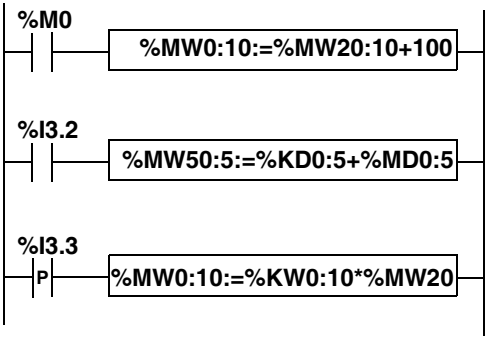
Ces instructions permettent de réaliser une opération arithmétique entre deux opérandes de type tableaux de mots (ou mot et tableau de mots).

Liste des instructions

+ : addition	***** : multiplication
- : soustraction	**/** : division
REM : reste de la division	-

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MW0:10:=%MW20:10+100]

LD %I3.2
[%MD50:5:=%KD0:5+%MD0:5]
```

Langage littéral structuré

```
IF RE %I3.3 THEN
 %MW0:10:=%KW0:10*%MW20;
END_IF;
```

Syntaxe

Opérateurs et syntaxe des instructions arithmétiques sur tableaux:

Opérateurs	Syntaxe
+,-,*,/,REM	Op1:=Op2 Opérateur Op3

Opérandes des instructions arithmétiques sur tableaux de mots:

Type	Opérande 1 (Op1)	Opérande 2 et 3 (Op2 et 3)
Tableaux de mots indexables	%MW:L	%MW:L,%KW:L,%Xi.T:L
Mots indexables	-	%MW,%KW,%Xi.T
Mots non indexable	-	Val.imm.,%IW,%QW,%SW,%NW, %BLK,Expr. num.

Opérandes des instructions arithmétiques sur tableaux de doubles mots:

Type	Opérande 1 (Op1)	Opérande 2 et 3 (Op2 et 3)
Tableaux de mots indexables	%MD:L	%MD:L,%KD:L
Mots doubles indexables	-	%MD,%KD
Mots doubles non indexable	-	Val.imm.,%ID,%QD,Expr. numérique

Instructions logiques sur tableaux

Généralités

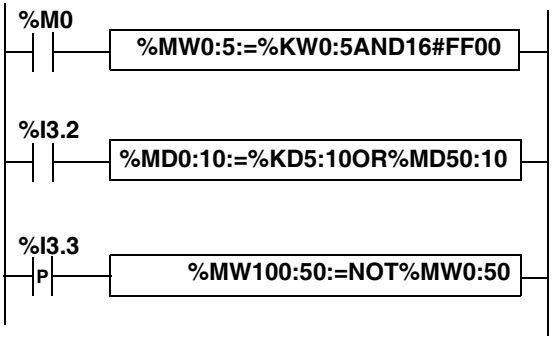
Ces instructions permettent de réaliser une opération arithmétique entre deux opérandes de type tableaux de mots (ou mot et tableau de mots).

Liste des instructions

AND : ET (bit à bit)	**XOR** : OU exclusif (bit à bit)
OR : OU logique (bit à bit)	**NOT** : complément logique (bit à bit) d'un tableau (1 seul opérande)

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MW0:5:=%KW0:5 AND 16#FF00]
```

Langage littéral structuré

```
IF %I3.2 THEN
 %MD0:10:=%KD5:10 OR %MD50:10;
END_IF;
IF RE %I3.3 THEN
 %MW100:50 := NOT %MW0:50;
END_IF;
```

Syntaxe

Opérateurs et syntaxe des instructions arithmétiques sur tableaux:

Opérateurs	Syntaxe
AND,OR,XOR	Op1:=Op2 Opérateur Op3
NOT	Op1:=NOT Op2

Opérandes des instructions logiques sur tableaux de mots:

Type	Opérande 1 (Op1)	Opérande 2 et 3 (Op2 et 3)
Tableaux de mots indexables	%MW:L	%MW:L,%KW:L,%Xi.T:L
Mots indexables	-	%MW,%KW,%Xi.T
Mots non indexable	-	Val.imm.,%IW,%QW,%SW,%NW,%BLK, Expr. num.

Opérandes des instructions logiques sur tableaux de doubles mots:

Type	Opérande 1 (Op1)	Opérande 2 et 3 (Op2 et 3)
Tableaux de mots indexables	%MD:L	%MD:L,%KD:L
Mots doubles indexables	-	%MD,%KD,%SD
Mots doubles non indexable	-	Val.imm.,%ID,%QD,Expr. numérique

Fonctions de sommation sur tableaux

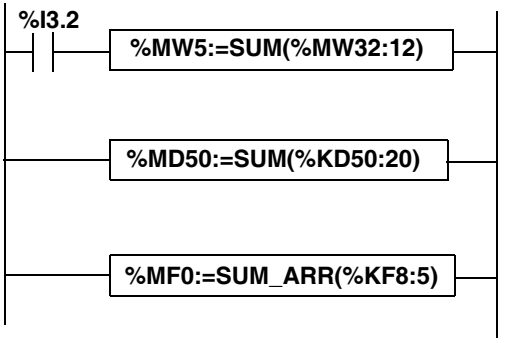
Généralités

Les fonctions SUM et SUM_ARR effectuent la somme de tous les éléments d'un tableau de mots :

- si le tableau est constitué de mots simple format, le résultat est donné sous la forme d'un mot simple format (fonction SUM),
- si le tableau est constitué de doubles mots, le résultat est donné sous la forme d'un double mot (fonction SUM),
- si le tableau est constitué de mots flottants, le résultat est donné sous la forme d'un mot flottant (fonction SUM_ARR).

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[%MW5:=SUM(%MW32:12]
```

Langage littéral structuré

```
%MD50:=SUM(%KD50:20)

%MF0:=SUM_ARR(%KF8:5)
```

Syntaxe

Syntaxe des instructions de sommation sur tableaux:

Res:=SUM(Tab)
Res:=SUM_ARR(Tab)

Paramètres des instructions de sommation sur tableaux

Type	**Résultat (res)**	**Tableau (Tab)**
Tableaux de mots indexables	-	%MW:L,%KW:L,%Xi.T:L
Mots indexables	%MW	-
Mots non indexable	%QW,%SW,%NW	-
Tableaux de doubles mots indexables	-	%MD:L,%KD:L
Mots doubles indexables	%MD	-
Mots doubles non indexables	%QD,%SD	-
Tableaux de flottants indexables	-	%MF:L,%KF:L
Mots flottants indexables	%MF	-

Note : le bit %S18 est mis à 1 lorsque le résultat n'est pas dans les bornes du format mot ou double mot suivant l'opérande tableau.

Exemple

```
%MW5:=SUM(%MW30:4)
```

avec %MW30=10, %MW31=20, %MW32=30, %MW33=40
%MW5=10+20+30+40=100

Fonctions de comparaison de tableaux

Généralités

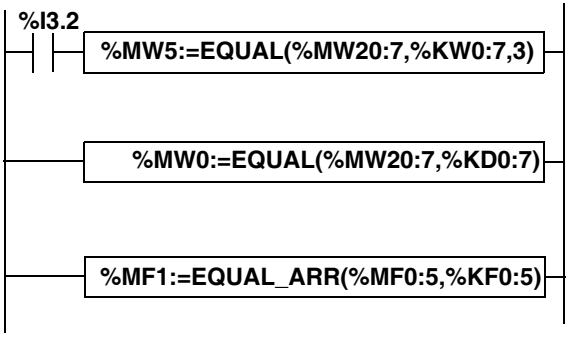
Les fonctions EQUAL (sur entier) et EQUAL _ARR (sur flottant) effectuent la comparaison de 2 tableaux élément par élément.

Si une différence apparaît, le rang des premiers éléments dissemblables est retourné sous forme d'un mot, sinon la valeur retournée est égale à -1.

Le troisième paramètre fournit le rang à partir duquel débute la comparaison (exemple: 0 pour commencer au début). Ce troisième paramètre est optionnel (il n'est pas autorisé avec la fonction EQUAL_ARR), lorsqu'il est omis, la comparaison est effectuée sur la totalité du tableau.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[%MW5:=EQUAL(%MD20:7,KD0:7,3)]
```

Langage littéral structuré

```
%MW0:=EQUAL(%MD20:7,%KD0:7)

%MW1:=EQUAL_ARR(%MF0:5,%KF0:5)
```

Syntaxe

Syntaxe des instructions de comparaison de tableaux:

Res:=EQUAL(Tab1,Tab2,rang)
Res:=EQUAL_ARR(Tab1,Tab2)

Paramètres des instructions de comparaison de tableaux.

Type	Résultat (Res)	Tableau (Tab)	Rang
Tableaux de mots	-	%MW:L,%KW:L,%Xi.T:L	-
Mots indexables	%MW	-	%MW,%KW,%Xi.T
Mots non indexable	%QW,%SW,%NW	-	Val.imm.,%QW,%IW, %SW,%NW, Expr. num.
Tableaux de doubles mots	-	%MD:L,%KD:L	-
Mots doubles indexables	%MD	-	%MD,%KD
Mots doubles non indexables	%QD,%SD	-	Val.imm.,%QD,%ID, %SD,Expr. num.
Tableaux de flottants	-	%MF:L,%KF:L	-
Mots flottants	%MF	-	-

Note :
- les tableaux doivent être obligatoirement de même longueur,
- si le paramètre rang est supérieur à la taille des tableaux alors le résultat est égal à ce rang.

Exemple

```
%MW5:=EQUAL(%MW30:4,%KW0:4,1)
```

Comparaison des 2 tableaux :

Rang	Tableau de Mots	Tableaux de Constantes	Différence
0	%MW30=10	%KW0=20	Ignoré (rang<1)
1	%MW31=20	%KW1=20	=
2	%MW32=30	%KW2=30	=
3	%MW33=40	%KW3=60	Différent

Le mot %MW5 vaut 3 (premier rang différent).

Fonctions de recherche sur tableaux

Généralités

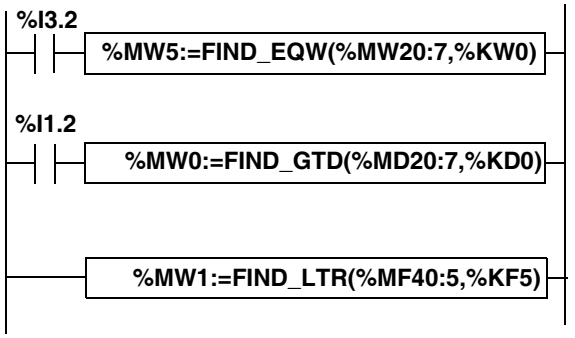
11 fonctions de recherche sont proposées :

- **FIND_EQW** : recherche de la position dans un tableau de mots du premier élément égal à une valeur donnée,
- **FIND_GTW** : recherche de la position dans un tableau de mots du premier élément supérieur à une valeur donnée,
- **FIND_LTW** : recherche de la position dans un tableau de mots du premier élément inférieur à une valeur donnée,
- **FIND_EQD** : recherche de la position dans un tableau de doubles mots du premier élément égal à une valeur donnée,
- **FIND_GTD** : recherche de la position dans un tableau de doubles mots du premier élément supérieur à une valeur donnée,
- **FIND_LTD** : recherche de la position dans un tableau de doubles mots du premier élément inférieur à une valeur donnée,
- **FIND_EQR** : recherche de la position dans un tableau de flottants du premier élément égal à une valeur donnée,
- **FIND_GTR** : recherche de la position dans un tableau de flottants du premier élément supérieur à une valeur donnée,
- **FIND_LTR** : recherche de la position dans un tableau de flottants du premier élément inférieur à une valeur donnée,
- **FIND_EQWP** : recherche de la position dans un tableau de mots du premier élément égal à une valeur donnée depuis un rang,
- **FIND_EQDP** : recherche de la position dans un tableau de doubles mots du premier élément égal à une valeur donnée depuis un rang.

Le résultat de ces instructions est égal au rang du premier élément trouvé ou à -1 si la recherche est infructueuse.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[%MW5:=FIND_EQW(%MW20:7,Kw0)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 %MW0:=FIND_GTD(%MD20:7,%KD0);
END_IF;

%MW1:=FIND_LTR(%MF40:5,%KF5);

%MW9:=FIND_EQWP(%MW30:8,%KF5,%MW4);
```

Syntaxe

Syntaxe des instructions de recherche sur tableaux:

Fonction	Syntaxe
FIND_EQW	Res:=Fonction(Tab,Val)
FIND_GTW	
FIND_LTW	
FIND_EQD	
FIND_GTD	
FIND_LTD	
FIND_EQR	
FIND_GTR	
FIND_LTR	
FIND_EQWP	Res:=Fonction(Tab,Val,rang)
FIND_EQDP	

Paramètres des instructions recherche sur tableaux de mots (FIND_EQW,FIND_GTW,FIND_LTW,FIND_EQWP)

Type	Résultat (Res)	Tableau (Tab)	Valeur (val), rang
Tableaux de mots indexables	-	%MW:L,%KW:L,%Xi.T:L	-
Mots indexables	%MW	-	%MW,%KW,%Xi.T
Mots non indexable	%QW,%SW,%NW	-	Val.imm.,%QW,%IW,%SW,%NW,Expr. num.

Paramètres des instructions recherche sur tableaux de doubles mots (FIND_EQD,FIND_GTD,FIND_LTD,FIND_EQDP)

Type	Résultat (Res)	Tableau (Tab)	Valeur (val)
Tableaux de mots indexables	-	%MD:L,%KD:L,%Xi.T:L	-
Mots doubles indexables	%MW	-	%MD,%KD
Mots doubles non indexable	%QW,%SW,%NW	-	Val.imm.,%QD,%ID,%SD,Expr. num.

Note : Pour le rang voir le tableau de mots (idem FIND_EQWP).

Paramètres des instructions recherche sur tableaux de flottants (FIND_EQR,FIND_GTR,FIND_LTR)

Type	Résultat (Res)	Tableau (Tab)	Valeur (val)
Tableaux de flottants	-	%MF:L,%KF:L	-
Mots flottants indexables	%MW	-	%MF,%KF
Mots flottants non indexable	%QW,%SW,%NW	-	Val.imm.,Expr. num.

Exemple

```
%MW5:=FIND_EQW(%MW30:4,%KW0)
```

Recherche de la position du premier mot =%KW0=30 dans le tableau :

Rang	Tableau de Mots	Résultat
0	%MW30=10	-
1	%MW31=20	-
2	%MW32=30	%MW5=2 (valeur du rang)
3	%MW33=40	-

Fonctions de recherche de valeurs maxi et mini sur tableaux

Généralités

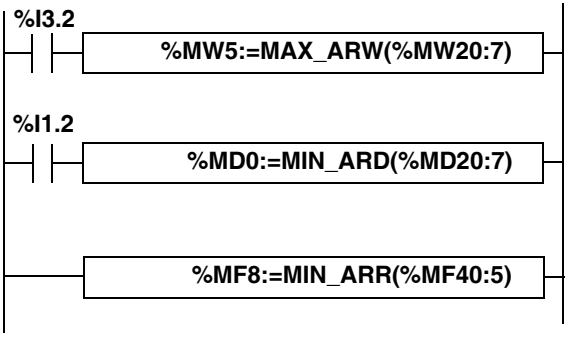
6 fonctions de recherche sont proposées :

- **MAX_ARW** : recherche de la valeur maximum dans un tableau de mots,
- **MIN_ARW** : recherche de la valeur minimum dans un tableau de mots,
- **MAX_ARD** : recherche de la valeur maximum dans un tableau de doubles mots,
- **MIN_ARD** : recherche de la valeur minimum dans un tableau de doubles mots,
- **MAX_ARR** : recherche de la valeur maximum dans un tableau de flottants,
- **MIN_ARR** : recherche de la valeur minimum dans un tableau de flottants.

Le résultat de ces instructions est égal à la valeur maximum (ou minimum) trouvée dans le tableau.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[%MW5:=MAX_ARW(%MW20:7)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 %MD0:=MIN_ARD(%MD20:7);
END_IF;
%MF8:=MIN_ARR(%MF40:5);
```

Syntaxe

Syntaxe des instructions de recherche de valeurs maxi et mini sur tableaux:

Fonction	Syntaxe
MAX_ARW	Res:=Fonction(Tab)
MIN_ARW	
MAX_ARD	
MIN_ARD	
MAX_ARR	
MIN_ARR	

Paramètres des instructions de recherche de valeurs maxi et mini sur tableaux:

Type	Résultat (Res)	Tableau (Tab)
Tableaux de mots indexables	-	%MW:L,%KW:L,%Xi.T:L
Mots indexables	%MW	-
Mots non indexable	%QW,%SW,%NW	-
Tableaux de doubles mots indexables	-	%MD:L,%KD:L
Mots doubles indexables	%MD	-
Mots doubles non indexable	%QD,%SD	-
Tableaux de flottants	-	%MF:L,%KF:L
Mots flottants indexables	%MF	-

Nombre d'occurrences d'une valeur dans un tableau

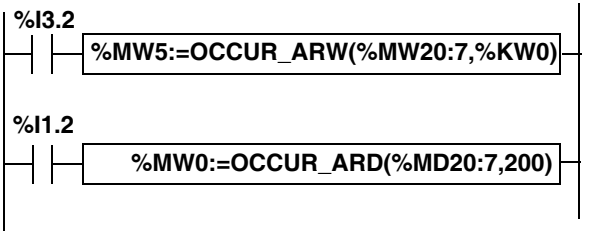
Généralités

3 fonctions de recherche sont proposées :

- **OCCUR_ARW** : effectue la recherche dans un tableau de mots du nombre d'éléments égaux à une valeur donnée,
- **OCCUR_ARD** : effectue la recherche dans un tableau de doubles mots du nombre d'éléments égaux à une valeur donnée,
- **OCCUR_ARR** : effectue la recherche dans un tableau de flottants du nombre d'éléments égaux à une valeur donnée.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[%MW5:=OCCUR_ARW(%MW20:7,%KW0)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 %MW0:=OCCUR_ARD(%MD20:7,200);
END_IF;
```

Syntaxe

Syntaxe des instructions de recherche de valeurs maxi et mini sur tableaux:

Fonction	Syntaxe
OCCUR_ARW	Res:=Fonction(Tab,Val)
OCCUR_ARD	
OCCUR_ARR	

Paramètres des instructions de recherche de valeurs maxi et mini sur tableaux:

Type	Résultat (Res)	Tableau (Tab)	Valeur (Val)
Tableaux de mots indexables	-	%MW:L,%KW:L,%Xi .T:L	-
Mots indexables	%MW	-	%MW,%KW,%Xi.T
Mots non indexable	%QW,%SW,%NW	-	Val.imm.,%QW,%IW, %SW,%NW,Expr. num.
Tableaux de doubles mots indexables	-	%MD:L,%KD:L	-
Mots doubles indexables	%MW	-	%MD,%KD
Mots doubles non indexable	%QW,%SW,%NW	-	Val.imm.,%QD,%ID, %SD,Expr. num.
Tableaux de flottants	-	%MF:L,%KF:L	-
Mots flottants indexables	%MF	-	%MF,%KF
Mots flottants non indexables	%QW,%SW,%NW	-	Val. imm., Expr. num.

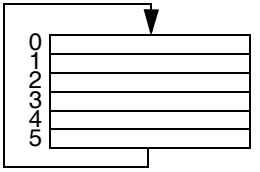
Fonction décalage circulaire sur un tableau

Généralités

6 fonctions de décalage sont proposées :

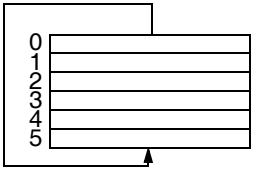
- **ROL_ARW** : réalise le décalage circulaire de n positions de haut en bas des éléments du tableau de mots,
- **ROL_ARD** : réalise le décalage circulaire de n positions de haut en bas des éléments du tableau de doubles mots,
- **ROL_ARR** : réalise le décalage circulaire de n positions de haut en bas des éléments du tableau de flottants.

Illustration des fonctions ROL_



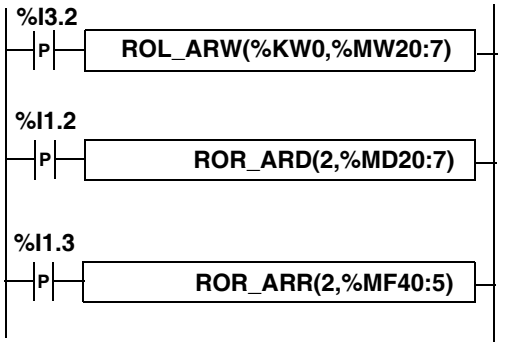
- **ROL_ARW** : réalise le décalage circulaire de n positions de bas en haut des éléments du tableau de mot,
- **ROR_ARD** : réalise le décalage circulaire de n positions de bas en haut des éléments du tableau de doubles mots,
- **ROR_ARR** : réalise le décalage circulaire de n positions de bas en haut des éléments du tableau de flottants.

Illustration des fonction ROR_



Structure

Langage à contacts



Langage liste d'instructions

```
LDR %I3.2
[ROL_ARW(%KW0,%MW0)]
```

Langage littéral structuré

```
IF RE %I1.2 THEN
 ROR_ARD(2,%MD20:7);
END_IF;
IF RE %I1.3 THEN
 ROR_ARR(2,%MF40:5);
END_IF;
```

Syntaxe

Syntaxe des instructions de décalage circulaire sur tableaux de mots **ROL_ARW** et **ROR_ARW**

Fonction	Syntaxe
ROL_ARW	Fonction(n,Tab)
ROR_ARW	

Paramètres des instructions de décalage circulaire sur tableaux de mots **ROL_ARW** et **ROR_ARW :**

Type	Nombre de positions (n)	Tableau (Tab)
Tableaux de mots indexables	-	%MW:L
Mots indexables	%MW,%KW,%Xi.T	-
Mots non indexable	Val.imm.,%QW,%IW,%SW, %NW,Expr.num.	-

Syntaxe des instructions de décalage circulaire sur tableaux de doubles mots **ROL_ARD** et **ROR_ARD :**

Fonction	Syntaxe
ROL_ARD	Fonction(n,Tab)
ROR_ARD	

Paramètres des instructions de décalage circulaire sur tableaux de doubles mots **ROL_ARD** et **ROR_ARD :**

Type	Nombre de positions (n)	Tableau (Tab)
Tableaux de mots indexables	-	%MD:L
Mots indexables	%MW,%KW,%Xi.T	-
Mots non indexable	Val.imm.,%QW,%IW,%SW, %NW,Expr.num.	-

Syntaxe des instructions de décalage circulaire sur tableaux de flottants **ROL_ARR** et **ROR_ARR**

Fonction	Syntaxe
ROL_ARR	Fonction(n,Tab)
ROR_ARR	

Paramètres des instructions de décalage circulaire sur tableaux de flottants: **ROL_ARR** et **ROR_ARR :**

Type	**Nombre de positions (n)**	**Tableau (Tab)**
Tableaux de mots indexables	-	%MF:L
Mots indexables	%MW,%KW,%Xi.T	-
Mots non indexable	Val.imm.,%QW,%IW,%SW, %NW,Expr.num.	-

Note : si la valeur de n est négative ou nulle, aucun décalage n'est effectué.

Fonction de tri sur tableau

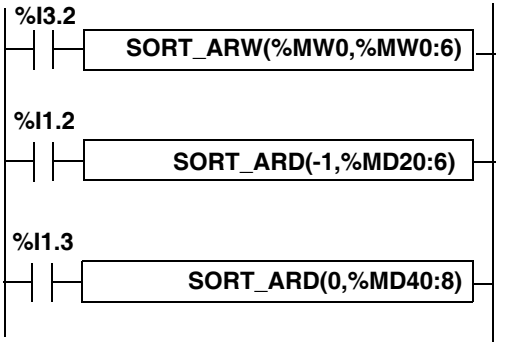
Généralités

3 fonctions de tri sont proposées :

- **SORT_ARW** : réalise les tris par ordre croissant ou décroissant des éléments du tableau de mots et range ce qui en résulte dans ce même tableau,
- **SORT_ARD** : réalise les tris par ordre croissant ou décroissant des éléments du tableau de doubles mots et range ce qui en résulte dans ce même tableau,
- **SORT_ARR** : réalise les tris par ordre croissant ou décroissant des éléments du tableau de flottants et range ce qui en résulte dans ce même tableau.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[SORT_ARW(%MW20,%MW0:6)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 SORT_ARD(-1,%MD20:6);
END_IF;
IF %I1.3 THEN
 SORT_ARR(0,%MF40:8);
END_IF;
```

Syntaxe

Syntaxe des fonctions de tri sur tableaux :

Fonction	Syntaxe
SORT_ARW	Fonction(sens,Tab)
SORT_ARD	
SORT_ARR	

- le paramètre "sens" donne l'ordre du tri: sens > 0 le tri se fait par ordre croissant, sens < 0 le tri s'effectue par ordre décroissant,
- le résultat (tableau trié) est retourné dans le paramètre Tab (tableau à trier).

Paramètres des fonctions de tri sur tableaux :

Type	Sens du tri	Tableau (Tab)
Tableaux de mots (SORT_ARW)	-	%MW:L
Tableaux de mots doubles (SORT_ARD)	-	%MD:L
Tableaux de flottants (SORT_ARR)	-	%MF:L
Mots indexables	%MW,%KW	-
Mots non indexable	Val.imm.,%QW,%IW,%SW, %NW,Expr.num.	-

Fonction de calcul de longueur de tableaux

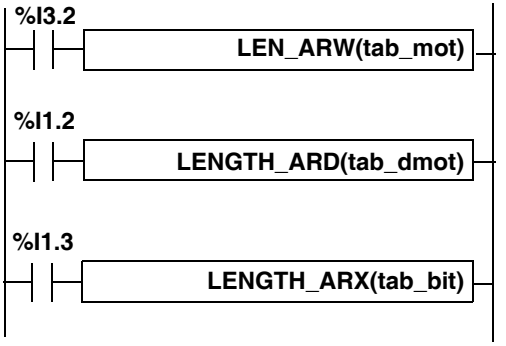
Généralités

4 fonctions de calcul de longueur de tableaux sont proposées, ces fonctions sont utiles notamment pour la programmation des blocs fonction DFB, lorsque les longueurs de tableaux n'ont pas été explicitement définies:

- **LENGTH_ARW** : calcule la longueur d'un tableau de mots en nombre d'éléments
- **LENGTH_ARD** : calcule la longueur d'un tableau de doubles mots en nombre d'éléments
- **LENGTH_ARR** : calcule la longueur d'un tableau de flottants nombre d'éléments
- **LENGTH_ARX** : calcule la longueur d'un tableau de bits en nombre d'éléments

Structure

Langage à contacts



Langage liste d'instructions

```
LD %I3.2
[LENGTH_ARW(tab_mot)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 LENGTH_ARD(tab_dmot);
END_IF;
IF %I1.3 THEN
 LENGTH_ARX(tab_bit);
END_IF;
```

Syntaxe

Syntaxe des fonctions de calcul de longueur de tableaux :

Fonction	Syntaxe
LENGTH_ARW	Result=Fonction(Tab)
LENGTH_ARD	
LENGTH_ARR	
LENGTH_ARX	

Paramètres des fonctions de calcul de longueur de tableaux :

Type	Tableau (Tab)	Résultat (Result)
Tableaux (LENGTH_ARW)	mot	-
Tableaux (LENGTH_ARD)	double mot	-
Tableaux (LENGTH_ARR)	flottant	-
Tableaux (LENGTH_ARX)	bit	-
Mots indexables	-	%MW
Mots non indexable	-	%QW,%SW,NW

Note : les paramètres de tableau seront des objets purement symboliques.

2.7 Instructions sur chaînes de caractères

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions sur chaînes de caractères du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Format d'une chaîne de caractères ou tableau de caractères

Généralités

- Un tableau de caractères est constitué d'une suite d'octets dans laquelle on peut ranger une chaîne de caractères. La taille du tableau permet de spécifier la longueur maximale que peut avoir la chaîne de caractères (255 maximum). Exemple : **%MB4:6 représente un tableau de 6 octets contenant une chaîne de 6 caractères maximum.**
- Le premier octet de début d'un tableau doit être pair (il n'est pas possible de saisir un tableau d'octets commençant par un octet impair, ex : %MB5:6).
- Les tableaux d'octets utilisent la même zone mémoire que les mots %MW, %MD, il y a donc risque de recouvrement ("Règle de recouvrement" - Manuel de référence Tome 1) .
- Le terme chaîne de caractères représente l'ensemble des caractères compris entre le début du tableau et le premier terminateur de chaîne rencontré.
- Le caractère NUL (code hexa 00) est appelée Terminateur de chaîne. Elle est symbolisée par **Ø** dans la suite du chapitre.
- La longueur d'une chaîne de caractères est donc donnée soit par le nombre de caractères avant le terminateur de chaîne, soit par la taille du tableau si aucun teminateur n'est détecté.

Exemples :

Le tableau suivant (de 12 éléments) contient la chaîne de caractères 'ABCDE' (de longueur 5) :

'A'	**'B'**	**'C'**	**'D'**	**'E'**	**Ø**	**'J'**	**'K'**	**'L'**	**'M'**	**'N'**	**'O'**

Le tableau suivant (de 10 éléments) contient la chaîne de caractères 'ABCDEJKLMN' (de longueur 10) :

'A'	**'B'**	**'C'**	**'D'**	**'E'**	**'J'**	**'K'**	**'L'**	**'M'**	**'N'**

Note : Le bit système %S15 est positionné à 1 dans les cas suivants :
- Lors de l'écriture d'une chaîne dans un tableau et que celle-ci est plus longue que la taille de ce dernier (impossibilité d'écrire le terminateur de chaîne Ø),
- Lorsqu'on tente d'accéder à un caractère ne se trouvant pas dans la chaîne considérée,
- Incohérence des paramètres : Longueur à effacer nulle (fonction DELETE), longueur à extraire nulle (fonction MID), longueur à remplacer nulle (fonction REPLACE), recherche d'une sous-chaîne plus longue que la chaîne (fonction FIND).

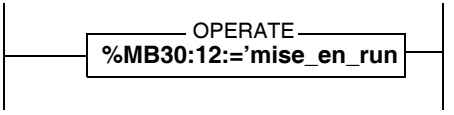
Affectation sur chaîne de caractères

Généralités

Permet de transférer une chaîne de caractères dans un tableau d'octets de longueur L.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB30:12:='mise_en_run']
```

Langage littéral structuré

```
%MB30:12:='mise_en_run';
```

Exemple

Transfert de la chaîne de caractères 'mise_en_run' dans le tableau d'octets de longueur 12

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'r'**	**'u'**	**'n'**	**Ø**

Syntaxe

Opérateurs d'affectation sur chaîne de caractères

Op1:=Op2

Opérandes d'affectation sur chaîne de caractères

Type	Opérande 1 (Op1)	Opérande 2 (Op2)
Tableaux d'octets	%MB:L	%MB:L,KB:L,Valeur immédiate

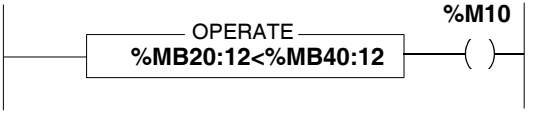
Comparaisons alphanumériques

Généralités

Ces opérateurs permettent de comparer deux chaînes de caractères contenues dans les tableaux d'octets passés en paramètres. La comparaison s'effectue caractère par caractère.
Le résultat est un bit qui vaut 1 si les deux chaînes satisfont la condition fournie par l'opérateur, caractère par caractère, dans le cas contraire le bit vaut 0.
L'ordre des caractères est donné par la table des codes ASCII (ISO 646). Par exemple la chaîne 'Z' est plus grande que la chaîne 'AZ' qui est plus grande que la chaîne 'ABC'.

Structure

Langage à contacts



Note : Les blocs comparaison se programment en zone de test.

Langage liste d'instructions

```
LD [%MB20:12<%MB40:12]
ST %M10
```

Note : La comparaison est réalisée à l'intérieur de crochets figurant derrière des instructions LD, AND et OR.

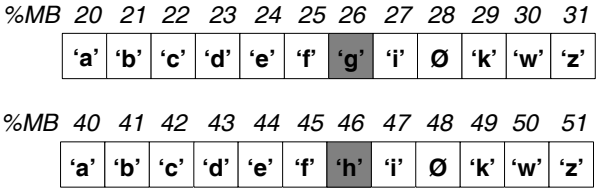
Langage littéral structuré

```
%MB10<%MB40:12;
```

Exemple

Exemple: `%MB20:12<%MB40:12` ==> OUI Le résultat vaut 1

illustration



Les éléments après le terminateur ne sont pas pris en compte.

Syntaxe

Opérateurs de comparaisons alphanumériques

Opérateurs	**Syntaxe**
<, >, <=, >=, =, <>	Op1 Opérateur Op2

Opérandes de comparaisons alphanumériques

Type	**Opérande 1 (Op1) et Opérande 2 (Op2)**
Tableaux d'octets	%MB:L, %KB:L, valeur immédiate

Fonctions de conversion Numérique <---> ASCII

Généralités

Ces fonctions permettent de convertir une valeur numérique (ou flottante) en chaîne de caractères codée en ASCII ou inversement.
Le résultat de la conversion doit être transféré dans un objet PL7 par une opération d'affectation: tableau d'octets, mot simple ou double longueur, flottant.

Liste des fonctions de conversions Numérique <---> ASCII possibles

Opérateurs	Description
INT_TO_STRING	Conversion Binaire -->ASCII (mots)
DINT_TO_STRING	Conversion Binaire -->ASCII (mots doubles)
STRING_TO_INT	Conversion ASCII-->Binaire (mots simples)
STRING_TO_DINT	Conversion ASCII-->Binaire (mots doubles)
REAL_TO_STRING	Conversion Flottant-->ASCII
STRING_TO_REAL	Conversion ASCII-->Flottant

Rappel sur le format flottant (Voir *Instructions sur flottant, p. 115*)

Rappel sur le code ASCII :
L'ensemble des 256 caractères alphanumériques et de contrôle peut être codé sur 8 bits. Ce code appelé ASCII (American Standard Code for Information Interchange) est compatible avec la notion d'octets. Tout tableau de n octets peut donc être formé par n codes ASCII définissant n caractères.

Conversion binaire-->ASCII

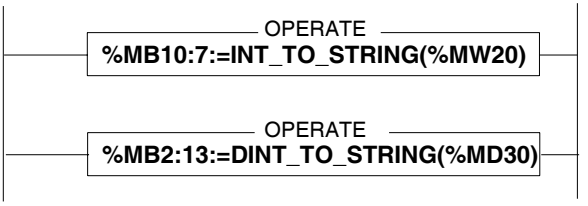
Généralités

Ces fonctions permettent de convertir une valeur numérique (mot simple ou double longueur) en chaîne de caractères codée en ASCII.
Chaque chiffre ainsi que le signe de la valeur passée en paramètre est codé en ASCII dans un élément du tableau d'octets résultat.

- Fonction **INT_TO_STRING** : Le contenu d'un mot simple longueur pouvant être compris entre -32768 et +32767, soit 5 chiffres plus le signe, le résultat sera un tableau de 6 caractères plus le terminateur. Le signe '+' ou '-' est rangé dans le premier caractère et les unités dans le sixième caractère, les dizaines dans le cinquième, ainsi de suite.
- Fonction **DINT_TO_STRING** : Le contenu d'un mot double longueur pouvant être compris entre -2147483648 et +2147483647, soit 10 chiffres plus le signe, le résultat sera un tableau de 12 caractères plus terminateur. Le signe '+' ou '-' est rangé dans le premier caractère, l'unité dans le douzième caractère, les dizaines dans le onzième, ainsi de suite. Le deuxième caractère est toujours '0'.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB10:7:=INT_TO_STRING(%MW20)]
```

Langage littéral structuré

```
%MB2:13:=DINT_TO_STRING(%MD30);
```

Exemples

Conversion Binaire ---> ASCII
`%MB10:7:=INT_TO_STRING(%MW20)` avec %MW20 = - 3782 en décimal
==> Le résultat est rangé dans le tableau de 7 octets suivant %MB10:

Illustration

%MB	*10*	*11*	*12*	*13*	*14*	*15*	*16*
	'_'	**'0'**	**'3'**	**'7'**	**'8'**	**'2'**	**Ø**

Exemple: `%MB2:13:=DINT_TO_STRING(%MD30)`
avec %MD30 = - 234701084

Illustration

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
	'-'	**'0'**	**'0'**	**'2'**	**'3'**	**'4'**	**'7'**	**'0'**	**'1'**	**'0'**	**'8'**	**'4'**	**Ø**

Syntaxe

Opérateurs de conversion Binaire-->ASCII

Syntaxe
Result:=**INT_TO_STRING** (valeur)

Opérandes de conversion Binaire-->ASCII

Type	**Result (résultat)**	**valeur**
Tableaux de 6 octets + terminateur	%MB:7	-
Mots indexables	-	%MW,%KW,%Xi.T
Mots non indexables	-	%IW,%QW,%SW,%NW,Val imm.,Expr. num.

Opérateurs de conversion Binaire-->ASCII (mots doubles)

Syntaxe
Result:=**DINT_TO_STRING** (valeur)

Opérandes de conversion Binaire-->ASCII (mots doubles)

Type	**Result (résultat)**	**valeur**
Tableaux de 12 octets + terminateur	%MB:13	-
Mots indexables	-	%MD,%KD
Mots non indexables	-	%ID,%QD,%SD,Val imm.,Expr. num.

Conversion ASCII-->binaire

Généralités

Ces fonctions permettent de convertir en binaire une chaîne de caractères représentant une valeur numérique (résultat transféré dans un mot simple ou double longueur). Chaque élément du tableau passé en paramètre représente le code ASCII d'un caractère. Les caractères autorisés sont les chiffres et les caractères '+' et '-'.

- Fonction **STRING_TO_INT** : converti une chaîne de 6 caractères représentant une valeur numérique comprise entre -32768 et +32767 . Le premier caractère doit représenter le signe et les caractères suivant la valeur : le deuxième, les dizaines de mille ; ... ; le sixième caractère, les unités. La valeur doit être cadrée à droite dans la chaîne.
- Fonction **STRING_TO_DINT** : converti une chaîne de 12 caractères représentant une valeur numérique comprise entre -2147483648 et +2147483647. Le premier caractère doit représenter le signe et les caractères suivant la valeur: le deuxième est le caractère '0' ; le troisième, les milliards ;... ; le douzième, les unités. La valeur doit être cadrée à droite dans la chaîne.

Structure

Langage à contacts

Langage liste d'instructions

```
LD TRUE
[%MW13:=STRING_TO_INT(%MB20:7)]
```

Langage littéral structuré

```
%MD2:=STRING_TO_DINT(%MB30:13);
```

Exemples

Exemple : `%MW13:=STRING_TO_INT(%MB20:7)`, avec

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*
	'-'	**'0'**	**'2'**	**'3'**	**'4'**	**'7'**	**Ø**

Le résultat dans %MW13 = -2347 en décimal.

Syntaxe

Opérateurs de conversion ASCII-->Binaire

Syntaxe
Result:=**STRING_TO_INT** (chaîne)

Opérandes de conversion ASCII-->Binaire.

Type	**Result (résultat)**	**valeur**
Mots indexables	%MW	-
Mots non indexables	%QW,%SW,%NW	-
Tableaux de 6 octets + terminateur	-	%MB:7,%KB:7,Val. imm.

Note : Le bit %S18 est positionné à 1 si la valeur décrite par la chaîne n'est pas comprise entre -32768 et +32767 ou si l'un des 6 caractères est erroné.

Opérateurs de conversion ASCII-->Binaire (mots doubles).

Syntaxe
Result:=**DINT_TO_STRING** (chaîne)

Opérandes de conversion ASCII-->Binaire (mots doubles)

Type	**Result (résultat)**	**valeur**
Mots indexables	%MD	-
Mots non indexables	%QD,%SD	-
Tableaux de 12 octets + terminateur	-	%MB:13,%KB:13, Val. imm.

Note : Le bit %S18 est positionné à 1 si la valeur décrite par la chaîne n'est pas comprise entre -2147483648 et +2147483647 ou si l'un des 12 caractères est erroné.

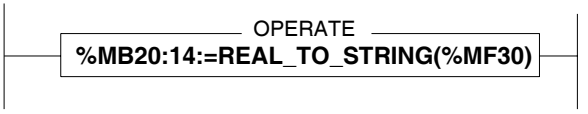
Conversion Flottant-->ASCII

Généralités

Cette fonction permet de convertir une valeur numérique réelle contenue dans un mot de type flottant en chaîne de caractères codée en ASCII. Le résultat est transféré dans un tableau de 13 octets + le terminateur.
Chaque chiffre de la valeur ainsi que les caractères '+', '-', '.', 'e' et 'E' sont codés en ASCII dans un élément du tableau résultat.
Le signe de la valeur se trouve dans le premier caractère, la virgule (.) dans le troisième, l'exposant 'e' dans le dixième, le signe de l'exposant dans le onzième.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB20:14:=REAL_TO_STRING(%MF30)]
```

Langage littéral structuré

```
%MB20:14:=REAL_TO_STRING(%MF30);
```

Exemples

Exemple : `%MB20:14:=REAL_TO_STRING(%MF30)` avec %MF30=-3.234718e+26
===> résultat :

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
	'-'	**'3'**	**'.'**	**'2'**	**'3'**	**'4'**	**'7'**	**'1'**	**'8'**	**'e'**	**'+'**	**'2'**	**'6'**	**Ø**

Syntaxe

Opérateurs de conversion Flottant-->ASCII

Syntaxe
Result:=**REAL_TO_STRING** (valeur)

Opérandes de conversion Flottant-->ASCII

Type	**Result (résultat)**	**valeur**
Tableaux de 13 octets + terminateur	%MB14	-
Mots indexables	-	%MF,%KF
Mots non indexables	-	Val. imm.,Expr. num.

Note : Le bit %S18 est positionné à 1 si la valeur flottante passée en paramètre n'est pas comprise entre -3.402824e+38 et -1.175494e-38 ou +1.175494e-38 et +3.402824e+38. Dans ce cas la valeur du résultat est erronée.

Conversion ASCII-->Flottant

Généralités

Cette fonction permet de convertir en flottant une chaîne de caractères représentant une valeur numérique réelle (résultat transféré dans un mot de type flottant).
Chaque élément du tableau passé en paramètre représente le code ASCII d'un caractère. Les caractères autorisés sont les chiffres et les caractères '+', '-', '.', 'e' et 'E'. On n'utilise pas le terminateur de chaîne pour déterminer la fin de la chaîne ce qui signifie que les 13 caractères du tableau doivent tous être corrects.
Le signe de la valeur doit se trouver dans le premier caractère, la virgule (.) dans le troisième, le 'e' dans le dixième, le signe de l'exposant dans le onzième. Par exemple la valeur 3.12 doit être donnée sous la forme '+3.120000e+00'.

Structure

Langage à contacts

Langage liste d'instructions

```
LD TRUE
[%MF18:=STRING_TO_REAL(%MF20:13)]
```

Langage littéral structuré

```
%MB18:=STRING_TO_REAL(%MB20:13);
```

Exemples

Exemple : `%MF18:=STRING_TO_REAL(%MB20:13)` avec

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
	'-'	**'3'**	**'.'**	**'2'**	**'3'**	**'4'**	**'7'**	**'1'**	**'8'**	**'e'**	**'+'**	**'2'**	**'6'**

===> résultat : %MF18 = -3.234718e+26

Syntaxe

Opérateurs de conversion ASCII-->Flottant

Syntaxe
Result:=**STRING_TO_REAL** (chaîne)

Opérandes de conversion ASCII-->Flottant

Type	**Result (résultat)**	**valeur**
Mots indexables	%MF	-
Tableaux de 13 octets	-	%MB:13,%KB:13,Valeur immédiate

Note : Le bit %S18 est positionné à 1 :

- si la valeur décrite par la chaîne n'est pas comprise entre -3.402824e+38 et -1.175494e-38
- si la valeur décrite par la chaîne n'est pas comprise entre +1.175494e-38 et +3.402824e+38
- si l'un des 13 caractères est erroné.

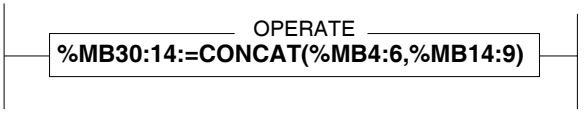
Concaténation de deux chaînes

Généralités

Ces instructions réalisent la concaténation de deux chaînes de caractères définies en paramètres. Le résultat est un tableau d'octets contenant une chaîne de caractères.

Structure

Langage à contacts



Langage liste d'instructions

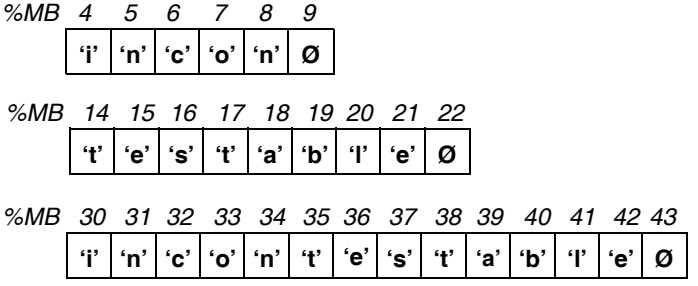
```
LD TRUE
[%MB30:14:=CONCAT(%MB4:6,%MB14:9)]
```

Langage littéral structuré

```
%MB30:14:=CONCAT(%MB4:6,%MB14:9);
```

Exemples

Exemple : `%MB30:14:=CONCAT(%MB4:6,%MB19:9)`



Syntaxe

Opérateurs de concaténation de chaînes

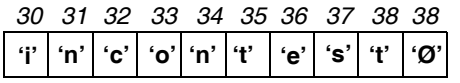
Syntaxe
Result:=**CONCAT**(chaîne1, chaîne 2)

Opérandes de concaténation de chaînes

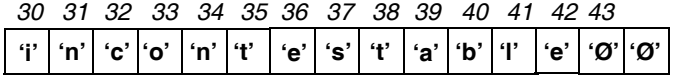
Type	**Result (résultat)**	**Chaîne 1 et 2**
Tableaux d'octets	%MB:L	%MB:L,%KB;L,Valeur immédiate

Note :

- Si le tableau résultat est trop court, alors il y a troncature et le bit système %S15 est positionné à 1. `%MB30:10:=CONCAT(%MB4:6, %MB14:9)`



- Si le tableau résultat est trop long, alors on complète la chaîne avec des caractères terminateurs 'Ø'. `%MB30:15:=CONCAT(%MB4:6,%MB14:9)`



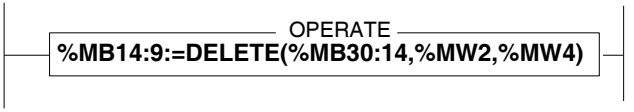
Suppression d'une sous-chaîne de caractères

Généralités

Réalise l'effacement d'un certain nombre de caractères (zone de longueur L), à partir d'un rang donné (position du premier caractère à effacer) dans la chaîne définie en paramètre. Le résultat est un tableau d'octets contenant une chaîne de caractères.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB14:9:=DELETE(%MB30:14,%MW2,%MW4)]
```

Langage littéral structuré

```
%MB14:9:=DELETE(%MB30:14,%MW2,%MW4);
```

Exemples

Exemple : `%MB14:9:=DELETE(%MB30:14,%MW2,%MW4)`
avec %MW2 = 5 (5 caractères à effacer) %MW4 = 3 (position = 3)

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*
	'i'	'n'	'c'	'o'	'n'	't'	'e'	's'	't'	'a'	'b'	'l'	'e'	Ø

%MB	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
	'i'	'n'	's'	't'	'a'	'b'	'l'	'e'	Ø

Syntaxe

Opérateur de suppression d'une sous-chaîne de caractères

Syntaxe
Result:=**DELETE**(chaîne1, long, pos)

Opérandes de suppression d'une sous-chaîne de caractères

Type	**Result (résultat)**	**Chaîne**	**Long (longueur), Pos (position)**
Tableaux d'octets	%MB:L	%MB:L,%KB;L,Val. immédiate	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mot non indexable	-	-	%IW,%QW,%SW,%NW, Val.imm.,Expr.num.

Note : Possibilité de recouvrement entre les paramètres suivant les indices des objets PL7 :

- Tableau contenant la chaîne origine.
- Tableau contenant la chaîne résultat.
- Mot contenant la longueur à effacer.
- Mot contenant la position du premier caractère à effacer.

Une longueur ou une position négative est interprétée comme étant égale à 0. Le paramètre position démarre à la valeur 1 correspondant à la première position dans la chaîne de caractères.

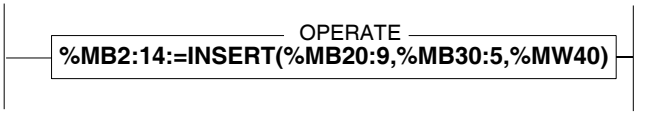
Insertion d'une sous-chaîne de caractères

Généralités

Insertion de la sous-chaîne de caractères définie par le deuxième paramètre (chaîne2) dans la chaîne de caractères définie par le premier paramètre (chaîne1). L'insertion est effectuée dans la première chaîne, après le caractère situé à la position donnée par le paramètre position (Pos).
Le résultat de l'insertion est une nouvelle chaîne de caractères transférée dans un tableau d'octets.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB2:14:=INSERT(%MB20:9,%MB30:6,%MW40)]
```

Langage littéral structuré

```
%MB2:14:=INSERT(%MB20:9,%MB30:6,%MW40);
```

Exemples

Exemple : `%MB2:14:=INSERT(%MB20:9,%MB30:6,%MW40)`
avec %MW40=position 2

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*
	'i'	**'n'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

%MB	*30*	*31*	*32*	*33*	*34*	*35*
	'c'	**'o'**	**'n'**	**'t'**	**'e'**	**Ø**

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
	'i'	**'n'**	**'c'**	**'o'**	**'n'**	**'t'**	**'e'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

Syntaxe

Opérateurs d'insertion d'une sous-chaîne de caractères

Syntaxe
Result:=INSERT (chaîne1, chaîne2, pos)

Opérandes d'insertion d'une sous-chaîne de caractères

Type	Result (résultat)	Chaîne 1 et 2	Pos (position)
Tableaux d'octets	%MB:L	%MB:L,%KB;L	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mot non indexable	-	-	%IW,%QW,%SW,%NW, Val.imm.,Expr.num.

Note :
- Le paramètre position démarre à la valeur 1 correspondant à la première position dans la chaîne de caractères.
- Il est impossible d'effectuer une insertion en début de chaîne. Pour ce cas utiliser la fonction CONCAT.
- Si le tableau est trop long, alors on complète avec des caractères de type terminateur.
- Mot contenant la position du premier caractère à effacer.
- Le bit système %S15 est positionné à 1 dans les cas suivants:
  - La valeur du paramètre position est négative ou égale à 0. Dans ce cas elle est interprétée comme étant égale à 0 et le tableau résultat contient une chaîne vide (composée de terminateurs),
  - Le tableau resultat est trop court, il y a alors troncature.

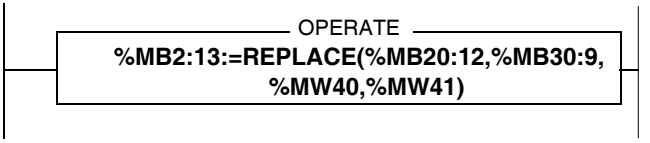
Remplacement d'une sous-chaîne de caractères

Généralités

Remplace un tronçon d'une chaîne de caractères définie dans le tableau d'origine (chaîne1) par une sous-chaîne de caractères définie dans le tableau de remplacement (chaîne2). Le remplacement à effectuer est défini par les paramètres de position (pos.) et de longueur (long.). Cette longueur correspond à la longueur de la chaîne qui disparait et non à la longueur de la sous-chaîne qui la remplace.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB2:13:=REPLACE(%MB20:12,%MB30:9,%MW40,%MW41)]
```

Langage littéral structuré

```
%MB2:13:=REPLACE(%MB20:12,%MB30:12,%MW40,%MW41);
```

Exemples

Exemple : `%MB2:13:=REPLACE(%MB20:12,%MB30:12,%MW40,%MW41)` avec `%MW40=3` (longueur=3) et `%MW41=9` (position 9)

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
Chaîne 1	**'m'**	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'r'**	**'u'**	**'n'**	**Ø**

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*
Chaîne 2	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**	**'r'**	**'u'**	**'n'**	**Ø**

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**

Syntaxe

Opérateurs de remplacement d'une sous-chaîne de caractères

Syntaxe
Result:=REPLACE (chaîne1, chaîne2, long., pos.)

Opérandes de remplacement d'une sous-chaîne de caractères

Type	**Result (résultat)**	**Chaîne 1 et 2**	**Long (longueur) Pos (position)**
Tableaux d'octets	%MB:L	%MB:L,%KB;L	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mot non indexable	-	-	%IW,%QW,%SW,%NW, Val.imm.,Expr.num.

Note :
- Le paramètre position démarre à la valeur 1 correspondant à la première position dans la chaîne de caractères.
- Si le tableau de sortie est trop long, alors on complète la chaîne avec des caractères de type terminateur.caractères.

Le bit système %S15 est positionné à 1 dans les cas suivants :
- Si la valeur du paramètre position est négative ou égale à 0. Dans ce cas elle est interprétée comme étant égale à 0 et le tableau résultat contient une chaîne vide (composée de terminateurs).
- Si la position passée en paramètre est supérieure ou égale à la longueur de la chaîne origine, le tableau résultat contient alors une chaîne vide (composée de terminateurs).
- Si le tableau resultat est trop court, il y a alors troncature.
- Mot contenant la position du premier caractère à effacer.
- Si la position du premier terminateur de chaîne est inférieur ou égal à la position du premier caractère à remplacer, le tableau de sortie est une recopie du tableau origine jusqu'au terminateur, complétés par des caractères terminateurs.

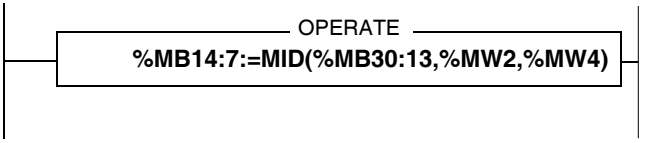
Extraction d'une sous-chaîne de caractères

Généralités

Extraction d'un certain nombre de caractères dans une chaîne origine passée en paramètre (chaîne).
Le rang du premier caractère à extraire est donné par le paramètre position (pos), et le nombre de caractères à extraire est donné par le paramètre longueur (long.).
La chaîne extraite est rangée dans un tableau d'octets (résult.).

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB14:7:=MID(%MB30:13,%MW2,%MW4)]
```

Langage littéral structuré

```
%MB14:7:=MID(%MB30:13,%MW2,%MW4);
```

Exemples

Exemple : `%MB14:7:=MID(%MB30:13,%MW2,%MW4)`
avec `%MW2=4` (longueur) et `%MW4=9` (position)

%MB	*30*	*31*	*32*	*33*	*34*	35	36	37	38	39	40	41	42
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**

Résultat :

%MB	*14*	*15*	*16*	*17*	*18*	*19*	*20*
	's'	**'t'**	**'o'**	**'p'**	**Ø**	**Ø**	**Ø**

Syntaxe

Opérateurs d'extraction d'une sous-chaîne de caractères

Syntaxe
Result:=MID (chaîne, long., pos.)

Opérandes d'extraction d'une sous-chaîne de caractères

Type	**Result (résultat)**	**Chaîne**	**Long (longueur) Pos (position)**
Tableaux d'octets	%MB:L,Val. imm.	%MB:L,%KB;L	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mot non indexable	-	-	%IW,%QW,%SW,%NW, Val.imm.,Expr.num.

Note :
- Le paramètre position démarre à la valeur 1 correspondant à la première position dans la chaîne de caractères.
- Si le tableau de sortie est trop long, alors on complète la chaîne avec des caractères de type terminateur.caractères.
- Si la longueur passée en paramètre est supérieure à la taille de la chaîne origine, le tableau résultat contient alors la chaîne origine.
- Si on atteint le dernier élément du tableau ou le terminateur de chaîne avant d'avoir extrait le nombre de caractères défini par le paramètre longueur, l'extraction s'arrête là.

Le bit système %S15 est positionné à 1 dans les cas suivants:
- Si la valeur du paramètre longueur à extraire est négative ou nulle. Dans ce cas elle est interprétée comme étant égale à 0 et le tableau résultat contient une chaîne vide (composée de terminateurs),
- Si la valeur du paramètre position de début d'extraction est nulle ou supérieure ou égale à la longueur du tableau ou supérieure ou égale à la position du premier terminateur. Dans ce cas le tableau résultat contient une chaîne vide (composée de terminateurs),
- Si le tableau résultat est trop court, il y a alors troncature.

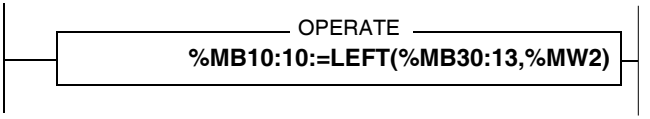
Extraction des caractères

Généralités

Extraction d'un certain nombre de caractères les plus à gauche (LEFT) ou les plus à droite (RIGHT) dans une chaîne origine passée en paramètre (chaîne).
Le nombre de caractères à extraire est défini par le paramètre longueur (long.). La chaîne extraite est rangée dans un tableau d'octets (résult.).

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB10:10:=LEFT(%MB30:13,%MW2)]
```

Langage littéral structuré

```
%MB10:10:=LEFT(%MB30:13,%MW2);
```

Exemples

Exemple : `%MB10:10:=LEFT(%MB30:13,%MW2)`
avec `%MW2=8` (longueur)

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**

Résultat :

%MB	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**Ø**	**Ø**

Syntaxe

Opérateurs d'extraction de caractères

Syntaxe
Result:=LEFT (chaîne, long)
Result:=RIGHT (chaîne, long)

Opérandes Opérateurs d'extraction de caractères

Type	**Result (résultat)**	**Chaîne**	**Long (longueur) Pos (position)**
Tableaux d'octets	%MB:L	%MB:L,%KB;L,Val. imm.	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mot non indexable	-	-	%IW,%QW,%SW,%NW, Val.imm.,Expr.num.

Note :
- Si le tableau de sortie est trop long, alors on complète la chaîne résultat avec des caractères de type terminateur.
- Si la longueur passée en paramètre est supérieure à la taille de la chaîne origine, le tableau résultat contient alors la chaîne origine.

Le bit système %S15 est positionné à 1 dans les cas suivants :
- Si la valeur du paramètre longueur à extraire est négative ou nulle. Dans ce cas le tableau résultat contient une chaîne vide (composée de terminateurs),
- Si la valeur du paramètre position de début d'extraction est nulle ou supérieure ou égale à la longueur du tableau ou supérieure ou égale à la position du premier terminateur. Dans ce cas le tableau résultat contient une chaîne vide (composée de terminateurs),
- Si le tableau résultat est trop court, il y a alors troncature.

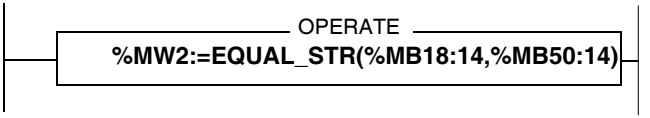
Comparaison de deux chaînes de caractères

Généralités

Comparaison de deux chaînes de caractères. Le résultat est un mot contenant la position du premier caractère différent.
En cas d'égalité parfaite entre les deux chaînes de caractères, le résultat vaut -1.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MW2:=EQUAL_STR(%MB18:14,%MB50:14)]
```

Langage littéral structuré

```
%MW2:=EQUAL_STR(%MB18:14,%MB50:14);
```

Exemples

Exemple : `%MW2:=EQUAL_STR(%MB18:14,%MB50:14)`
avec

%MB	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
	'a'	**'b'**	**'c'**	**'d'**	**'e'**	**'f'**	**'g'**	**'h'**	**'i'**	**'p'**	**'w'**	**'x'**	**'y'**	**'z'**

Résultat :

%MB	*50*	*51*	*52*	*53*	*54*	*55*	*56*	*57*	*58*	*59*	*60*	*61*	*62*	*63*
	'a'	**'b'**	**'c'**	**'d'**	**'?'**	**'f'**	**'g'**	**'h'**	**'Ø'**	**'v'**	**'w'**	**'x'**	**'y'**	**'z'**

==> MW2:= 5

Syntaxe

Opérateurs de comparaison de deux chaînes de caractères

Syntaxe
Result:=EQUAL_STR (chaîne1, chaîne2)

Opérandes de comparaison de deux chaînes de caractères

Type	**Result (résultat)**	**Chaîne 1 et 2**
Tableaux d'octets	-	%MB:L,%KB;L,Val. imm.
Mots indexables	%MW	-
Mot non indexable	%QW,%SW,%NW	-

Note :
- Une longueur ou une position négative est interprétée comme étant égale à 0.
- Les lettres majuscules sont différentes des lettres minuscules.

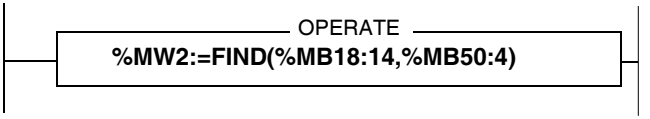
Recherche d'une sous-chaîne de caractères

Généralités

Recherche de la sous-chaîne de caractères définie par le deuxième paramètre dans la chaîne de caractères définie par le premier paramètre.
Le résultat est un mot contenant la position, dans la première chaîne, du début de la sous-chaîne recherchée.
En cas d'échec dans la recherche, le résultat vaut -1.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MW2:=FIND(%MB18:14,%MB50:4)]
```

Langage littéral structuré

```
%MW2:=FIND(%MB18:14,%MB50:4);
```

Exemples

Exemple : `%MW2:=FIND(%MB18:14,%MB50:4)` avec :

%MB	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
	'a'	**'b'**	**'c'**	**'d'**	**'e'**	**'f'**	**'g'**	**'h'**	**'i'**	**Ø**	**'w'**	**'x'**	**'y'**	**'z'**

%MB	*50*	*51*	*52*	*53*
	'f'	**'g'**	**'h'**	**Ø**

==> MW2:= 6 Indique Indique que le début de la chaîne recherchée se situe à partir du sixième caractère.

Syntaxe

Opérateurs de recherche de sous-chaînes de caractères

Syntaxe
Result:=FIND (chaîne1, chaîne2)

Opérandes de recherche de sous-chaînes de caractères

Type	**Result (résultat)**	**Chaîne 1 et 2**
Mots indexables	%MW	-
Mot non indexable	%QW,%SW,%NW	-
Tableaux d'octets	-	%MB:L,%KB;L,Val. imm.

Note : Une longueur ou une position négative est interprétée comme étant égale à 0.

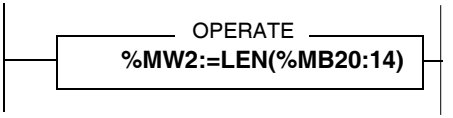
Longueur d’une chaîne de caractères

Généralités

Cette fonction retourne la longueur de la chaîne de caractères passée en paramètres, c'est à dire le nombre de caractères qui se trouvent avant le terminateur.

Structure

Langage à contacts



Langage liste d’instructions

```
LD TRUE
[%MW2:=LEN(%MB20:14)]
```

Langage littéral structuré

```
%MW2:=LEN(%MB20:14);
```

Exemples

Exemple : %MW2:=LEN(%MB20:14 avec :))

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘Ø’	‘n’	o’	‘p’	‘r’

==> MW2:= 7

Syntaxe

Opérateur d'une longueur de chaîne de caractères

Syntaxe
Result:=LEN (chaîne)

Opérandes d'une longueur de chaîne de caractères

Type	**Result (résultat)**	**Chaîne 1 et 2**
Mots indexables	%MW	-
Mot non indexable	%QW,%SW,%NW	-
Tableaux d'octets	-	%MB:L,%KB;L,Val. imm.

Note : Si aucun terminateur n'est trouvé alors cette fonction retourne la taille du tableau comme indiqué dans: "Formats d'une chaîne de caratères ou tableau de caractères" (Voir *Format d'une chaîne de caractères ou tableau de caractères, p. 175*).

2.8 Instructions de gestion du temps: Dates, Heures, Durées

Présentation

Objet de ce sous-chapitre

Ce sous chapitre décrit les instructions de gestion du temps: Dates, Heures, Durées, du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Format des paramètres des instructions de gestion du temps

Généralités

Les paramètres Date, Heure, Durée utilisés par ces instructions correspondent aux formats types définis par la norme IEC1131-3.

Format Durée (Type TIME)

Ce format permet de coder des durées exprimées en dixièmes de seconde et correspond au format TIME de la norme.
L' affichage de telles valeurs est fait sous la forme : **sssssssss.d**

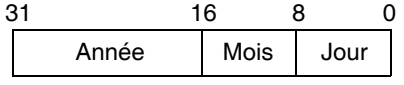
Ce qui donne par exemple : 3674.3 , pour 1 heure, 1 minute, 14 secondes et 3 dixièmes
La valeur est codée sur 32 bits (un double mot) dont les bornes sont fixées à [0, 4294967295] dixièmes de secondes, ce qui représente approximativement 13 ans et 7 mois.

Note : Seules les valeurs comprises dans l'intervalle [00:00:00, 23:59:59] sont autorisées.

Format Date (Type DATE)

Ce format permet de coder l'année, le mois, et le jour. Il correspond au format DATE de la norme.
L' affichage de la valeur est fait sous la forme: **yyyy-mm-dd**

Ce qui donne par exemple: 1984-06-25
La valeur est codée en BCD sur 32 bits (un double mot) avec 3 champs :



Année : 4 digits
Mois : 2 digits
Jour : 2 digits

Exemple, exprimé en hexadécimal :

19H	84H	06H	25H

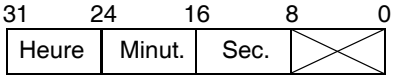
= 1984-06-25

Note : Seules les valeurs comprises dans l'intervalle [1990-01-01, 2099-12-31] sont autorisées.

Format Heure du jour (type TOD)

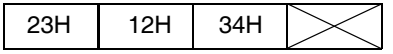
Ce format permet de coder l'heure, les minutes et les secondes. Il correspond au format TIME_OF_DAY de la norme.
L' affichage de la valeur est fait sous la forme: **hh:mm:ss**

Ce qui donne par exemple: 23:12:34
La valeur est codée en BCD sur 32 bits (un double mot) avec 3 champs :) :



Heures : 2 digits (mots poids fort)
Minutes : 2 digits (mots poids fort)
Secondes : 2 digits (mots poids faible)

Exemple, exprimé en hexadecimal:



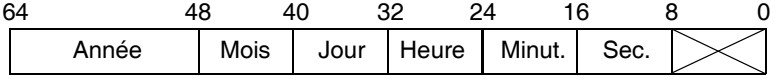
= 23:12:34

Note : Seules les valeurs comprises dans l'intervalle [00:00:00, 23:59:59] sont autorisées.

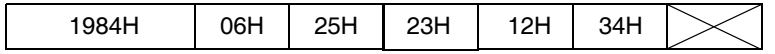
Format Date et heure (Type DT)

Ce format permet de coder l'année, le mois, le jour, l'heure, les minutes et les secondes. Il correspond au format DATE_AND_TIME de la norme.
L' affichage de la valeur est fait sous la forme: **yyyy-mm-dd-hh:mm:ss**

Ce qui donne par exemple: 1984-06-25-23:12:34
La valeur est codée en BCD sur 64 bits (un tableau de mots de longueur 4) :



Exemple, exprimé en hexadecimal :

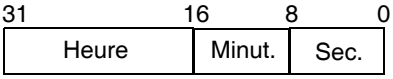


Note : Seules les valeurs comprises dans l'intervalle [1990-01-01-00:00:00, 2099-12-31-23:59:59] sont autorisées.

Format Heure, Minute, Seconde (Type HMS)

Ce format utilisé exclusivement par la fonction TRANS_TIME, permet de coder les heure, les minutes et les secondes.
L' affichage de la valeur est fait sous la forme: **hh:mm:ss**

Ce qui donne par exemple : 23:12:34
La valeur codée en BCD sur 32 bits (un double mot) avec 3 champs :



Heures : 4 digits (mots poids fort)
Minutes : 2 digits (mots poids faible)
Secondes : 2 digits (mots poids faible)

Exemple, exprimé en hexadecimal :

= 23:12:34

Utilisation des bits et mots système

Bit système %S17

Le bit système **%S17** est positionné dans les cas suivants :

- Résultat d'une opération hors de l'intervalle de valeurs autorisé,
- Un paramètre d' entrée n'est pas interprétable et cohérent au format souhaité (DATE, DT ou TOD),
- Opération sur un format Heure du jour (TOD) entraînant un changement de jour,,
- Conflit d'accès à l'horodateur.

Bit système %S15

Le bit système **%S15** est positionné à 1 lors de l'écriture d'une chaîne dans un tableau lorsque celle-ci est plus longue que la taille de ce dernier.

Mots système

Les mots système :

- **%SD18** : compteur de temps absolu permet aussi de faire des calculs de durée (incrémenté tous les 1/10 de secondes par le système),
- **%SW49** à **%SW53** (Voir *Description des mots système %SW48 à %SW59, p. 318*) peuvent aussi être utilisés pour faire des affichages de dates.

Fonction horodateur

Généralités

Cette fonction permet de commander des actions à des horaires et des dates prédéfinis ou calculés.
Elle positionne à 1 le paramètre de sortie OUT si la date fournie par l' horloge automate au moment de l'appel de la fonction appartient à la période programmée dans les paramètres d'entrées.

Syntaxe

Opérateur de la fonction horodateur.

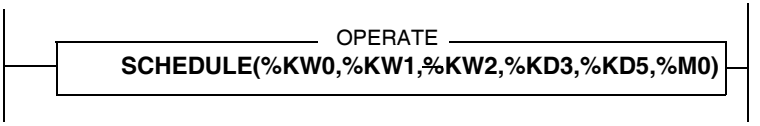
SCHEDULE (DBEG, DEND, WEEK, HBEG, HEND, OUT)

Caractéristiques des paramètres

Sortie	**OUT**	Bit contenant le résultat des comparaisons effectuées par la fonction horodateur : à 1 pendant les périodes définies par les paramètres.
Date de début	**DBEG**	Mot codant la date de début de la période (mois-jour) en BCD (valeurs limites : 01-01 à 12-31).
Date de fin	**DEND**	Mot codant la date de fin de la période (mois-jour) en BCD (valeurs limites : 01-01 à 12-31).
Jour de la semaine	**WEEK**	Mot codant le ou les jours de la semaine pris en compte dans la période définie par les paramètres DBEG et DEND. Les 7 bits de poids faible représentent les 7 jours de la semaine : bit 6=lundi, bit 5 = mardi,..., bit 0 = dimanche.
Heure de début	**HBEG**	Double mot codant l'heure de début de la période dans la journée (heures-minutes-secondes) en BCD format heure du jour (type : TOD). Valeurs limites : 00:00:00, 23:59:59
Heure de fin	**HEND**	Double mot codant l'heure de fin de la période dans la journée (heures-minutes-secondes) en BCD format heure du jour (type : TOD). Valeurs limites : 00:00:00, 23:59:59

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[SCHEDULE(%KW0,%KW1,%KW2,%KD3,%KD5,%M0)]
```

Langage littéral structuré

```
SCHEDULE(%KW0,%KW1,%KW2,%KD3,%KD5,%M0);
```

Exemples

Exemple : Programmation de 2 plages horaires non continues

SCHEDULE	(16#0501, 16#1031, 2#0000000001111100, 16"08300000, 16#12000000, %M0);	(*date de début : 1er mai*) (*date de fin : 31 octobre*) (*lundi à vendredi*) (*heure de début : 8h30*) (*heure de fin : 18h*) (*résultat dans %M0*)
SCHEDULE	(16#0501, 16#1031, 2#0000000001111100, 16"14000000, 16#18000000, %M1);	(*date de début : 1er mai*) (*date de fin : 31 octobre*) (*lundi à vendredi*) (*heure de début : 14h*) (*heure de fin : 18h*) (*résultat dans %M1*)
	%Q0.0:=%M0 OR %M1;	

Opérandes

Opérandes de la fonction horodateur

Type	DBEG,DEND,WEEK	HBEG,HEND	OUT
Mots indexables	%MW,%KW,%Xi.T	-	-
Mot non indexable	%IW,%QW,%SW,%NW,Val.imm.,expr.num.	-	-
Doubles mots indexables	-	%MD,%KD	-
Doubles mots non indexables	-	%ID,%QD,Val.imm., Expr.num.	-
Bits	-	-	%I,%Q,%M,%S, %BLK,%*:Xk,%X

Note :

- Les 2 paramètres DBEG et DEND définissent une plage de jours dans l'année, cette plage peut être à cheval sur 2 années civiles. Exemple : du 10 Octobre au 7 Avril. Le 29 Février peut être utilisé dans cette période, il sera ignoré les années non bissextiles.
- Les 2 paramètres HBEG et HEND définissent une plage horaire dans le jour, cette plage peut être à cheval sur 2 jours. Exemple : de 22h à 6h 10min 20s.
- Si une des dates DBEG et DEND ou l'une des heures HBEG et HEND est erronée, c'est à dire ne correspond pas à une date ou une heure réelle, la sortie OUT sera à 0 et le bit %S17 sera mis à 1.
- Si l'automate cible ne possède pas d'horloge interne (cas du TSX37-10), la sortie sera à 0 et le bit système %S17 mis à 1.
- Il est possible d'alléger la charge du processeur automate lorsque la précision n'est pas importante en cadençant l'appel de la fonction SCHEDULE par le bit système %S6 ou %S7.
- Pour une plage horaire programmée à cheval sur deux jours, par exemple de 15h (jour 1) à 8h (jour 2), les conditions seront de nouveaux valides de 15h à 24h le deuxième jour. Si on veut exécuter cette plage horaire qu'une seule fois dans la semaine, il vaut mieux utiliser deux fois la fonction SCHEDULE avec une plage horaire de 15h à 24h (jour 1) et 0h à 8h (jour 2).

Fonction Horodateur réseau

Généralités

La fonction R_NTPC permet de récupérer la date et l'heure sur un serveur NTP selon deux formats :

- un format pour l'affichage,
- un format pour effectuer des calculs.

Note : cette fonction nécessite la connexion sur un réseau Ethernet permettant l'accès à un serveur NTP.
Le service de synchronisation horaire est disponible sur TSX ETY 5103 avec une version processeur = V5.8 et une version V4.5 de PL7.

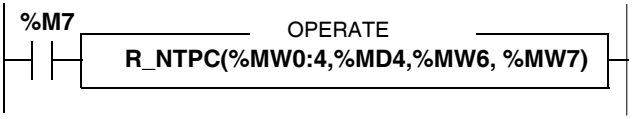
Syntaxe

Opérateur de la fonction horodateur réseau

Syntaxe
R_NTPC(N_DT,SEC,MSEC,Status)

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[R_NTPC(%MW0:4,%MD4,%MW6,%MW7)]
```

Langage littéral structuré

```
IF %M7 THEN
  R_NTPC(%MW0:4,%MD4,%MW6,%MW7);
END_IF;
```

Opérandes

Opérandes de la fonction horodateur réseau

Paramètres	Type	Description
N_DT	Tableau de 4 Mots (%MW)	Mot 1 : ● octet 0 : réservé ● octet 1 : secondes Mot 2 : ● octet 0 : minutes ● octet 1 : heure Mot 3 : ● octet 0 : jour ● octet 1 : mois Mot 4 : ● octet 0 et 1 : année
SEC	DWORD (%MD)	C'est la date et l'heure depuis le 1er Janvier 1980 qui sont converties en secondes.
MSEC	WORD (%MW)	Valeurs des Millli secondes de l'heure.

STATUS	WORD (%MW)	La variable STATUS indique la validité du résultat de la fonction R_NTPC.
		L'octet de poids faible est contrôlé par l'automate.
		● Si sa valeur est égale à 0 :
		  ● valeur d'horloge non disponible,
		  ● date/heure non mis à jour au cours des deux dernières minutes.
		● Si sa valeur est égale 1 :
		  ● date/heure mis à jour au cours des deux dernières minutes,
		  ● date et heure acceptable.
		L'octet de poids fort est contrôlé par le module ETY.
		● Si sa valeur est égale à 0 :
		  ● valeur d'horloge transmise à l'UC non acceptable.
		● Si sa valeur est égale à 1, la date et l'heure mises à jour reçues du serveur sont transmises au module :
		  ● dans un délai de deux minutes,
		  ● acceptable (décalage de 10ms maximum).

	Remarque : pour que l'heure soit valide dans l'UC, l'octet de poids faible et de poids fort du mot STATUS doivent être égaux à 1.	

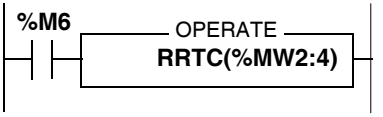
Lecture date système

Généralités

Lecture de la date système (Real Time Clock) et transfert dans l'objet donné en paramètre dans le format Date et heure (DT).

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M6
[RRTC(%MW2:4)]
```

Langage littéral structuré

```
IF %M6 THEN
 RRTC(%MW2:4);
END_IF;
```

Exemples

Exemple : `RRTC(%MW2:4)`
Le résultat est transféré dans le tableau de mots internes de longueur 4 : %MW2 à %MW5.

Syntaxe

Opérateur de lecture date système

Syntaxe
RRTC(date)

Opérandes de lecture date système

Type	Date
Tableau de 4 Mots au format date et heure	%MW:4

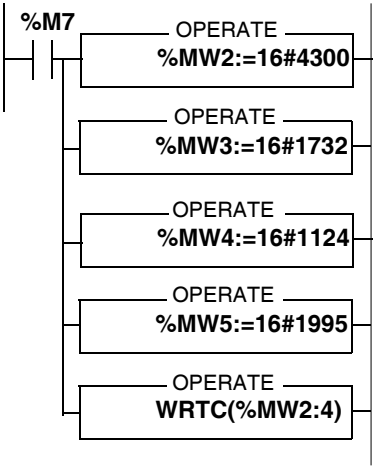
Mise à jour date système

Généralités

Mise à jour de la date système (Real Time Clock) et transfert dans l' objet donné en paramètre dans le format Date et heure (DT).

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[%MW2:=16#4300]
[%MW3:=16#1732]
[%MW4:=16#1124]
[%MW5:=16#1995]
[WRTC(%MW2:4)]
```

Langage littéral structuré

```
IF %M7 THEN
 %MW2:=16#4300;
 %MW3:=16#1732;
 %MW4:=16#1124;
 %MW5:=16#1995;
 WRTC(%MW2:4);
END_IF;
```

Exemples

Exemple : La nouvelle date est chargée dans un tableau de mots internes de longueur 4 %MW2:4 puis envoyée au système par la fonctions WRTC.

Syntaxe

Opérateur de mise à jour date système

Syntaxe
WRTC(date)

Opérandes de mise à jour date système

Type	Date
Tableau de 4 Mots	%MW:4,%KW:4 au format date et heure

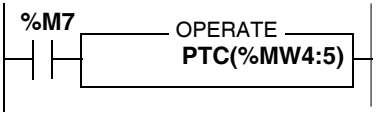
Lecture date et code arrêt

Généralités

Lecture de la date du dernier arrêt automate et du code spécifiant la cause de cet arrêt (dans le 5ème mot, équivalent à %SW58 (Voir *Description des mots système %SW48 à %SW59, p. 318*)).

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[PTC(%MW4:5)]
```

Langage littéral structuré

```
IF %M7 THEN
  PTC(%MW4:5);
END_IF;
```

Exemples

Exemple : `PTC(%MW4:5)`
Le résultat est transféré dans le tableau de mots internes de longueur 5 : %MW4 à %MW8.

Syntaxe

Opérateur de lecture date et code arrêt

Syntaxe
PTC (date)

Opérandes de lecture date et code arrêt

Type	Date
Tableau de 5 Mots au format date et heure	%MW:5

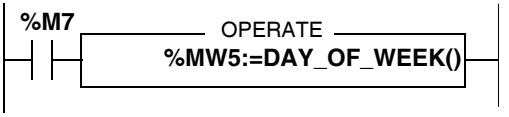
Lecture du jour de la semaine

Généralités

Cette fonction fournit en résultat le jour courant de la semaine sous la forme d'un chiffre de 1 à 7 transféré dans un mot (1 = Lundi, 2 = Mardi,3 = Mercredi, 4 = Jeudi, 5 = Vendredi, 6 = Samedi, 7 = Dimanche).

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[%MW5:=DAY_OF_WEEK()]
```

Langage littéral structuré

```
IF %M7 THEN
  %MW5:=DAY_OF_WEEK();
END_IF;
```

Exemples

Exemple : `%MW5:=DAY_OF_WEEK()`
%MW5:=4 correspond à jeudi

Syntaxe

Opérateur de lecture du jour de la semaine

Syntaxe
Result:=**DAY_OF_WEEK()**

Opérandes de lecture du jour de la semaine

Type	Result (résultat)
Mots indexables	%MW
Mots non indexables	%QW,%SW,%NW

Note : Si la fonction n' a pas pu mettre à jour le résultat suite à une erreur d' accès à l'horodateur le résultat retourné est 0 et le bit système %S17 est positionné à 1.

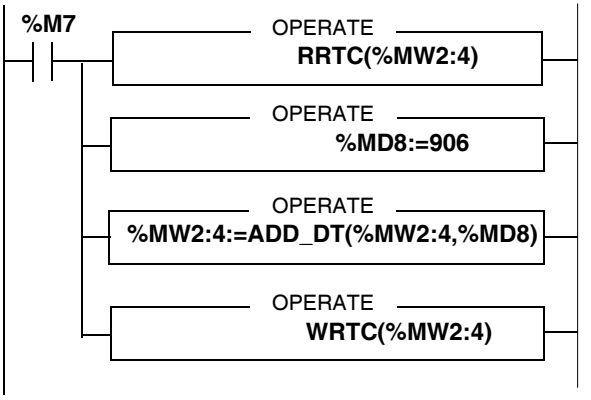
Ajout / Retrait d'une durée à une date

Généralités

Ajout ou retrait d'une durée (en dixièmes de secondes) (In2) à une date origine (In1). Le résultat est une nouvelle date, transférée dans un tableau de 4 mots.
ADD_DT () = Ajout d'une durée
SUB_DT () = Retrait d'une durée

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[RRTC(%MW2:4)]
[%MD8:=906]
[%MW2:4:=ADD_DT(%MW2:4,%MD8)]
[WRTC(%MW2:4)]
```

Langage littéral structuré

```
IF %M7 THEN
 RRTC(%MW2:4);
 %MD8:=906;
 %MW2:4:=ADD_DT(%MW2:4,%MD8);
 WRTC(%MW2:4);
END_IF;
```

Exemples

Exemple : `%MW2:4:=ADD_DT(%MW2:4,%MD8)`
%MW2:4:= Date origine
%MD8:=906 (906 dixièmes de seconde arrondi à 1 min. 31s)
%MW2:4:= Nouvelle date

Syntaxe

Opérateurs d'ajout/retrait d'une durée à une date

Syntaxe
Result:=**ADD_DT**(In1, In2)
Result:=**SUB_DT**(In1, In2)

Opérandes d'ajout/retrait d'une durée à une date

Type	**Result (résultat)**	**In1 (date origine)**	**In2 (durée)**
Tableaux de 4 mots au format date et heure	%MW4	%MW4:4,%KW:4	-
Mots doubles indexables	-	-	%MD,%KD
Mots doubles non indexables	-	-	%ID,%QD,Val.imm., Expr.num.

Note :

- Le principe de l'arrondi sera appliqué sur le paramètre "durée" (exprimée en 10ème de seconde) pour permettre l'ajout ou retrait à la date (précision à la seconde).
  - sssssssss.0 à sssssssss.4 arrondi à sssssssss.0
  - sssssssss.5 à sssssssss.9 arrondi à sssssssss.0 +1.0
- La gestion des années bissextiles est à prévoir dans l'application.
- Si le résultat de l'opération est hors de l'intervalle des valeurs autorisées, alors le bit système %S17 est positionné à 1 et la valeur du résultat est égale à la borne minimum (pour SUB_DT) ou reste bloqué au maximum (pour ADD_DT).
- Si le paramètre d'entrée "date origine" n'est pas interprétable et cohérent en format DT (DATE_AND_TIME) alors le bit système %S17 est positionné à 1 et la valeur du résultat est égal à 0001-01-01-00:00:00.

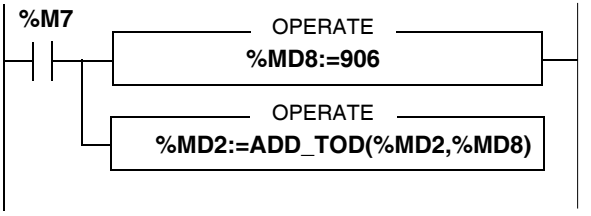
Ajout / Retrait d'une durée à une heure du jour

Généralités

Ajout ou retrait d'une durée à une heure du jour. Le résultat est une nouvelle heure du jour qui est transférée dans un double mot.
ADD_TOD () = Ajout d'une durée
SUB_TOD () = Retrait d'une durée

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[%MD8:=906]
[%MD2:=ADD_TOD(%MD2,%MD8)]
```

Langage littéral structuré

```
IF %M7 THEN
 %MD8:=906;
 %MD2:=ADD_TOD(%MD2,%MD8);
END_IF;
```

Exemples

Exemple : `%MD2:=ADD_TOD(%MD2,%MD8)`
%MD2:= Heure origine (ex : 12:30:00)
%MD8:= 906 (906 dixièmes de seconde arrondie à 1 min. 31s
%MD2:= Nouvelle heure (ex : 13:31:31)

Syntaxe

Opérateurs d'ajout/retrait d'une durée à une heure du jour

Syntaxe
Result:=**ADD_TOD**(In1, In2)
Result:=**SUB_TOD**(In1, In2)

Opérandes d'ajout/retrait d'une durée à une heure du jour

Type	Result (résultat)	In1 (heure origine) et In2 (durée)
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD	%ID,%QD,Val.imm.,Expr.num.

result et **In1** sont au format TOD, **In2** est au format durée.

Note :
- Le principe de l'arrondi sera appliqué sur le paramètre "durée" (exprimée en 10ème de seconde) pour permettre l'ajout ou retrait à la date (précision à la seconde).
  - ssssssssss.0 à ssssssssss.4 arrondi à ssssssssss.0
  - ssssssssss.5 à ssssssssss.9 arrondi à ssssssssss.0 +1.0
- Il y a changement de jour si le résultat de l'opération est hors de l'intervalle des valeurs autorisées. Dans ce cas le bit système %S17 est positionné à 1 et la valeur du résultat est interprétable avec un modulo 24:00:00.
- Si le paramètre d'entrée "heure du jour" n'est pas interprétable en format TOD alors le bit système %S17 est positionné à 1 et le résultat est égal à 00:00:00.

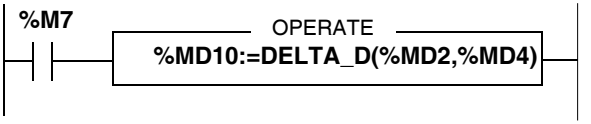
Ecart entre deux dates (sans heure)

Généralités

Calcule l'écart de temps entre deux dates. Le résultat, donné en valeur absolue, est transféré dans un double mot.

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M7
[%MD10:=DELTA_D(%MD2,%MD4)]
```

Langage littéral structuré

```
IF %M7 THEN
 %MD10:=DELTA_D(%MD2,%MD4);
END_IF;
```

Exemples

```
%MD10:=DELTA_D(%MD2,%MD4)
```

%MD2:= Date numéro1 (ex : 1994-05-01)
%MD4:= Date numéro2 (ex: 1994-04-05)
==> %MD10:= 22464000 (==> écart = 26 jours)

Syntaxe

Opérateur d'écart entre deux dates (sans heure)

Syntaxe
Result:=**DELTA_D**(Date1, Date2)

Opérandes d'écart entre deux dates (sans heure)

Type	**Result (résultat)**	**Date 1 et 2**
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD	%ID,%QD,Val.imm.,Expr.num.

result est au format TIME, **Date 1** et **2** sont au format DATE.
Le format TIME est défini avec une précision au dixième de seconde. Le format DATE est défini avec une précision à la journée. L'écart de temps calculé sera alors un multiple de 864000 (= 1jour = 24 h x 60 mn x 60 s x 10 dixième).

Note :
- Il y a débordement si le résultat dépasse la valeur maximale admise pour une durée (TIME). Dans ce cas le résultat est égal à 0 et le bit système %S18 est positionné à 1.
- Si un des paramètres d'entrée n'est pas interprétable et cohérent au format DATE, alors le bit système %S17 est positionné à 1 et le résultat est égal à 0.

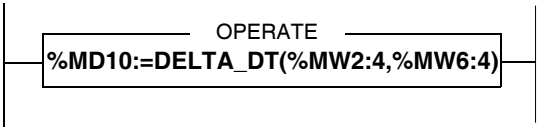
Ecart entre deux dates (avec heure)

Généralités

Calcule l'écart de temps entre deux dates. Le résultat, donné en valeur absolue, est transféré dans un double mot.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MD10:=DELTA_DT(%MW2:4,%MW6:4)]
```

Langage littéral structuré

```
%MD10:=DELTA_DT(%MW2:4,%MW6:4);
```

Exemples

```
%MD10:=DELTA_DT(%MW2:4,%MW6:4)
```

%MW2:4:= Date numéro1 (ex : 1994-05-01-12:00:00)
%MW6:4:= Date numéro2 (ex: 1994-05-01-12-01-30)
==> %MD10:= 900 (==> écart = 1 minute et 30 secondes)

Syntaxe

Opérateur d'écart entre deux dates (avec heure)

Syntaxe
Result:=**DELTA_DT**(Date1, Date2)

Opérandes d'écart entre deux dates (avec heure)

Type	**Result (résultat)**	**Date 1 et 2**
Mots doubles indexables	%MD	-
Mots doubles non indexables	%QD	-
Tableau de 4 mots au format DT	-	%MW:4,%KW:4

result est au format TIME, **Date 1** et **2** sont au format DT.
Le format TIME est défini avec une précision au dixième de seconde. Le format DT est lui défini avec une précision à la seconde. L'écart de temps calculé sera alors un multiple de 10.

Note :
- Il y a débordement si le résultat dépasse la valeur maximale admise pour une durée (TIME). Dans ce cas le résultat est égal à 0 et le bit système %S18 est positionné à 1.
- Si un des paramètres d'entrée n'est pas interprétable et cohérent au format DT, alors le bit système %S17 est positionné à 1et le résultat est égal à 0.

Ecart entre deux heures

Généralités

Calcule l'écart de temps entre deux heures du jour. Le résultat est transféré dans un double mot en valeur absolue donnant une durée.

Structure

Langage à contacts

Langage liste d'instructions

```
LD TRUE
[%MD10:=DELTA_TOD(%MD2,%MD4)]
```

Langage littéral structuré

```
%MD10:=DELTA_TOD(%MD2,%MD4);
```

Exemples

```
%MD10:=DELTA_TOD(%MD2,%MD4)
```

%MD2:= Heure1 (ex : 02:30:00)
%MD4:= Heure2 (ex : 02 41 00)
==> %MD10:= 6600 (==> écart = 11 minutes)

Syntaxe

Opérateur d'écart entre deux heures

Syntaxe
Result:=**DELTA_TOD**(Date1, Date2)

Opérandes d'écart entre deux heures

Type	**Result (résultat)**	**Heure 1 et 2**
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD	%ID,%QD,Valeur immédiate, Expr. numérique

result est au format TIME, **Heure 1** et **2** sont au format TOD.

Le format TIME est défini avec une précision au dixième de seconde. Le format TOD est lui défini avec une précision à la seconde. L'écart de temps calculé sera alors un multiple de 10.

Note : Si un des paramètres d'entrée n'est pas interprétable et cohérent au format TOD, alors le bit système %S17 est positionné à 1 et le résultat est égal à 0.

Conversion d'une date en chaîne de caractères

Généralités

Cette instruction convertit une date en chaîne de caractères (sans heure) au format: YYYY-MM-DD (10 caractères). Cette chaîne se termine par la caractère terminateur Ø. Chaque caractère Y,M,D symbolise un chiffre.

Structure

Langage à contacts

Langage liste d'instructions

```
LD TRUE
[%MB2:11=DATE_TO_STRING(%MD40)]
```

Langage littéral structuré

```
%MB2:11=DATE_TO_STRING(%MD40);
```

Exemples

```
%MB2:11=DATE_TO_STRING(%MD40)
```

%MD40:= Date (ex : 1998-12-27)

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
	'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**	**'2'**	**'7'**	**Ø**

Syntaxe

Opérateur de conversion d'une date en chaîne

Syntaxe
Result:=**DATE_TO_STRING**(Date)

Opérandes de conversion d'une date en chaîne

Type	**Result (résultat)**	**Date**
Tableaux de 11 octets	%MB:11	-
Mots doubles indexables	-	%MD,%KD
Mots doubles non indexables	-	%ID,%QD,Valeur immédiate, Expr. numérique

Note :

- Si le paramètre d'entrée (date) n'est pas interprétable et cohérent en format DATE, alors le bit système %S17 est positionné à 1 et la fonction retourne la chaîne `**** - ** - **` .
- Si la chaîne de sortie est trop courte, alors il y a troncature et le bit système %S15 est positionné à 1.

```
%MB2:8 := DATE_TO_STRING(%MD40)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**

==> %S15 = 1

- Si la chaîne de sortie est trop longue, alors on complète la chaîne de sortie avec des caractères de type terminateur Ø.

```
%MB2:12:=DATE_TO_STRING(%MD40)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**	**'2'**	**'7'**	**Ø**	**Ø**

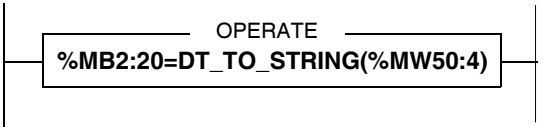
Conversion d'une date complète en chaîne de caractères

Généralités

Cette instruction convertit une date complète (avec heure) en chaîne de caractères au format: YYYY-MM-DD-HH:MM:SS (19 caractères). Cette chaîne se termine par la caractère terminateur Ø. Chaque caractère Y,M,D,H,M,S symbolise un chiffre.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB2:20=DT_TO_STRING(%MW50:4)]
```

Langage littéral structuré

```
%MB2:20=DT_TO_STRING(%MW50:4);
```

Exemples

```
%MB2:20=DT_TO_STRING(%MW50:4)
```

%M50:4:= Date et heure (type DT) (ex : 1998-12-27-23:14:37)

==>																					
%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'	'2'	'7'	'-'	'2'	'3'	':'	'1'	'4'	':'	'3'	'7'	Ø	Ø

Syntaxe

Opérateur de conversion d'une date complète en chaîne

Syntaxe
Result:=DT_TO_STRING(Date)

Opérandes de conversion d'une date complète en chaîne

Type	**Result (résultat)**	**Date**
Tableaux de 20 octets	%MB:20	-
Tableau de 4 mots au format DT	-	%MW:4,%KW:4

Note :

- Si le paramètre d'entrée (date) n'est pas interprétable et cohérent en format DT (DATE_AND_TIME), alors le bit système %S17 est positionné à 1 et la fonction retourne la chaîne `****-**-**-**:**:**` .
- Si la chaîne de sortie est trop courte, alors il y a troncature et le bit système %S15 est positionné à 1.

```
%MB2:8:=DT_TO_STRING(%MW50:4)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**

==> %S15 = 1

- Si la chaîne de sortie est trop longue, alors on complète la chaîne de sortie avec des caractères de type terminateur Ø.

```
%MB2:21:=DT_TO_STRING(%MD50:4)
```

==>
%MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**	**'2'**	**'7'**	**'-'**	**'2'**	**'3'**	**':'**	**'1'**	**'4'**	**':'**	**'3'**	**'7'**	**Ø**	**Ø**

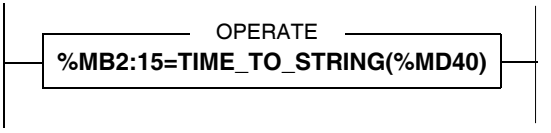
Conversion d'une durée en chaîne de caractères

Généralités

Cette instruction convertit une durée (au format TIME) en chaîne de caractères. Le format du résultat se décompose en heures, minutes, secondes et dixièmes sur 15 caractères : HHHHHH:MM:SS.D. Cette chaîne se termine par le caractère terminateur Ø. Chaque caractère H,M,S,D symbolise un chiffre.
La durée maximale correspond à 119304 heures, 38 minutes, 49 secondes et 5 dixièmes.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB2:15=TIME_TO_STRING(%MD40)]
```

Langage littéral structuré

```
%MB2:15=TIME_TO_STRING(%MD40);
```

Exemples

```
%MB2:15=TIME_TO_STRING(%MD40)
```

%MD40:= 27556330.3 (format TIME)

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
	'0'	**'0'**	**'7'**	**'6'**	**'5'**	**'4'**	**':'**	**'3'**	**'2'**	**':'**	**'1'**	**'0'**	**'.'**	**'3'**	**Ø**

Syntaxe

Opérateur de conversion d'une durée en chaîne

Syntaxe
Result:=**TIME_TO_STRING**(Durée)

Opérandes de conversion d'une durée en chaîne

Type	**Result (résultat)**	**Durée**
Tableaux de 15 octets	%MB:15	-
Mots doubles indexables	-	%MD,%KD
Mots doubles non indexables	-	%ID,%QD,Valeur immédiate, Expr. numérique

Durée est au format TIME

Note :

- Si la chaîne de sortie est trop courte, alors elle est tronquée et le bit système %S15 est positionné à 1.

```
%MB2:8:=TIME_TO_STRING(%MD40)
```

==> %MB *2 3 4 5 6 7 8 9*

'0'	**'0'**	**'7'**	**'6'**	**'5'**	**'4'**	**':'**	**'3'**

==> %S15 = 1

- Si la chaîne de sortie est trop longue, alors on complète la chaîne de sortie avec des caractères de type terminateur Ø.

```
%MB2:16:=TIME_TO_STRING(%MD40)
```

==>

%MB *2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17*

'0'	**'0'**	**'7'**	**'6'**	**'5'**	**'4'**	**':'**	**'3'**	**'2'**	**':'**	**'1'**	**'0'**	**'.'**	**'3'**	**Ø**	**Ø**

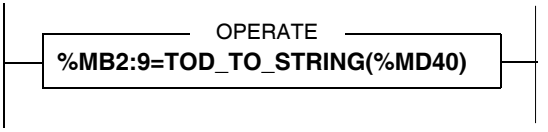
Conversion d'une heure du jour en chaîne de caractères

Généralités

Cette instruction convertit une heure du jour (au format TOD - TIME_OF_DAY) en chaîne de caractères au format HH:MM:SS sur 8 caractères plus un caractère terminateur Ø. Chaque caractère H,M,S symbolise un chiffre.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%MB2:9=TOD_TO_STRING(%MD40)]
```

Langage littéral structuré

```
%MB2:9=TOD_TO_STRING(%MD40);
```

Exemples

```
%MB2:9=TOD_TO_STRING(%MD40)
%MD40:= 23:12:27 (format TOD)
```

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
	'2'	**'3'**	**':'**	**'1'**	**'2'**	**':'**	**'2'**	**'7'**	**Ø**

Syntaxe

Opérateur de conversion d'une heure du jour en chaîne

Syntaxe
Result:=TOD_TO_STRING(Durée)

Opérandes de conversion d'une heure du jour en chaîne

Type	**Result (résultat)**	**Heure**
Tableaux de 9 octets	%MB:9	-
Mots doubles indexables	-	%MD,%KD
Mots doubles non indexables	-	%ID,%QD,Valeur immédiate, Expr. numérique

Heure est au format TOD

Note :

- Si la chaîne de sortie est trop courte, alors il y a troncature et le bit système %S15 est positionné à 1.
  %MB2:8 := TOD_TO_STRING (%MD40) (avec %MD40 := 23:12:27)

  ==> %MB *2 3 4 5 6 7 8 9*

'2'	**'3'**	**':'**	**'1'**	**'2'**	**':'**	**'2'**	**'7'**

  ==> %S15 = 1

- Si la chaîne de sortie est trop longue, alors on complète la chaîne de sortie avec des caractères de type terminateur Ø.
  %MB2:10 := TOD_TO_STRING (%MD40) (avec %MD40 := 23:12:27)

  ==>

  %MB *2 3 4 5 6 7 8 9 10 11*

'2'	**'3'**	**':'**	**'1'**	**'2'**	**':'**	**'2'**	**'7'**	**Ø**	**Ø**

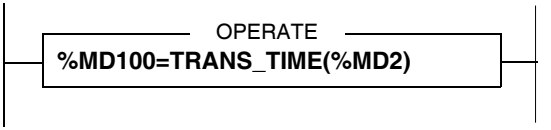
Conversion d'une durée en HHHH:MM:SS

Généralités

Cette instruction convertit une durée (au format TIME) en nombre d'heures-minutes-secondes, HHHH:MM:SS. Valeurs limites [0000:00:00 , 9999:59:59].

Structure

Langage à contacts



Langage liste d'instructions

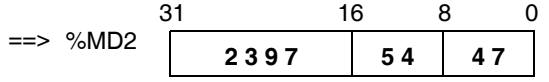
```
LD TRUE
[%MD100=TRANS_TIME(%MD2)]
```

Langage littéral structuré

```
%MD100=TRANS_TIME(%MD2);
```

Exemples

%MD100=TRANS_TIME(%MD2)
avec %MD2:= 36324873 dixièmes de secondes



valeurs exprimées en hexadécimal

Syntaxe

Opérateur de conversion d'une durée en HHHH:MM:SS

Syntaxe
Result:=**TRANS_TIME**(Durée)

Opérandes de conversion d'une durée en HHHH:MM:SS

Type	Result (résultat)	Durée
Mots doubles indexables	%MD	%MD,%KD
Mots doubles non indexables	%QD	%ID,%QD,Valeur immédiate, Expr. numérique

Result est au format HMS

Durée est au format TIME

Note :

- Le principe de l'arrondi sera appliqué sur le paramètre "durée" (exprimée en 10ème de seconde) pour permettre la conversion (précision à la seconde).
  - sssssssss.0 à sssssssss.4 arrondi à sssssssss.0
  - sssssssss.5 à sssssssss.9 arrondi à sssssssss.0 + 1.0
- La durée maximale convertie peut atteindre 10000 heures. Cela signifie que si la valeur de la durée (TIME) passée en paramètre est supérieure ou égale à 360000000, elle n'est pas convertible. Le bit système %S15 est positionné à 1 et le résultat est égal à 0000:00:00.

2.9 Instructions sur tableau de bits

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les instructions sur tableaux de bits du langage PL7.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants:

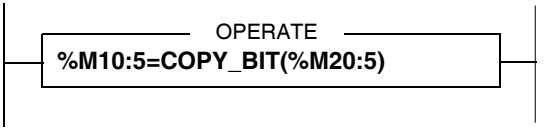
Copie d’un tableau de bits dans un tableau de bits

Généralités

Cette fonction effectue la recopie bit à bit d'un tableau de bits dans un autre tableau de bits.

Structure

Langage à contacts



Langage liste d’instructions

```
LD TRUE
[%M10:5=COPY_BIT(%M20:5)]
```

Langage littéral structuré

```
%M10:5=COPY_BIT(%M20:5);
```

Syntaxe

Opérateur de copie de tableau de bits

Syntaxe
Result:=**COPY_BIT**(Tab)

Opérandes de copie de tableau de bits

Type	Result (résultat)	Tab (tableau)
Tableau de bits	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L,%Xi:L

Note :

- Les tableaux peuvent être de tailles différentes. Dans ce cas, le tableau résultat contient le résultat de la fonction exécutée sur une longueur équivalente à la plus petite des tailles des tableaux, et le reste du tableau résultat n'est pas modifié.
- Attention aux recouvrements entre le tableau en entrée et le tableau résultat.

Instructions logiques sur tableaux de bits

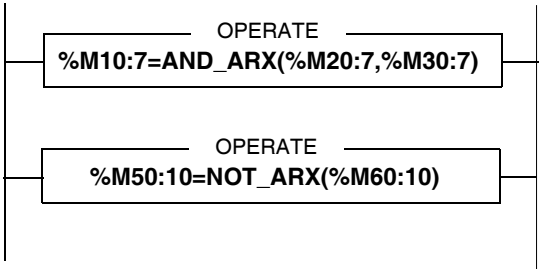
Généralités

Les fonctions associées permettent de réaliser une opération logique bit à bit entre deux tableaux de bits et range le résultat dans un autre tableau de bit.

- **AND_ARX** : ET logique (bit à bit),
- **OR_ARX** : OU logique (bit à bit),
- **XOR_ARX** : OU exclusif (bit à bit),
- **NOT_ARX** : Complément logique (bit à bit) d'un tableau.

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[%M10:7=AND_ARX(%M20:7,%M30:7)]

LD TRUE
[%M50:10=NOT_ARX(%M60:10)]
```

Langage littéral structuré

```
%M10:7=AND_ARX(%M20:7,%M30:7);
%M50:10=NOT_ARX(%M60:10);
```

Syntaxe

Opérateurs d'instructions logiques sur tableaux de bits

Syntaxe
Result:=**AND_ARX**(Tab 1, Tab 2)
Result:=**OR_ARX**(Tab 1, Tab 2)
Result:=**XOR_ARX**(Tab 1, Tab 2)
Result:=**NOT_ARX**(Tab 1)

Opérandes d'instructions logiques sur tableaux de bits

Type	**Result (résultat)**	**Tab 1 et Tab 2 (tableau)**
Tableau de bits	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L,%Xi:L

Note :
- Les tableaux peuvent être de tailles différentes. Dans ce cas, le tableau résultat contient le résultat de la fonction exécutée sur une longueur équivalente à la plus petite des tailles des tableaux, et le reste du tableau résultat n'est pas modifié.
- Possibilité de recouvrement entre le tableau en entrée et le tableau résultat.

Copie d'un tableau de bits dans un tableau de mots

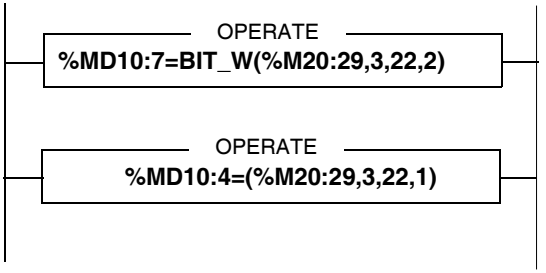
Généralités

La fonction effectue la recopie des bits d'un tableau ou d'une partie de tableau de bits dans un tableau de mots (ou doubles mots).
Dans le tableau de bits, le prélèvement est effectué à partir d'un certain rang (brow) pour un nombre de bits (nbit).
Dans le tableau de mots (ou doubles mots), la recopie est effectuée à partir du rang (wrow ou drow) en commençant par le poids le plus faible de chaque mot.

- **BIT_W** : Recopie d'un tableau de bits dans un tableau de mots.
- **BIT_D** : Recopie d'un tableau de bits dans un tableau de doubles mots.

Structure

Langage à contacts



Langage liste d'instructions

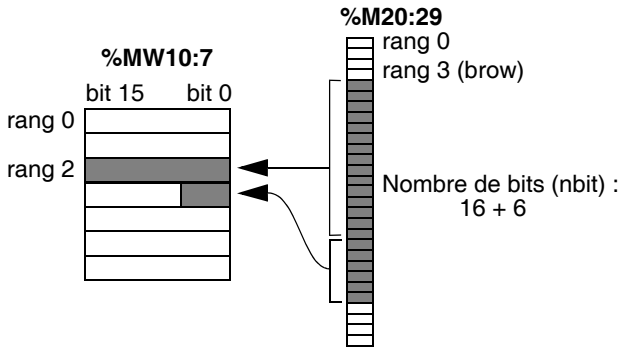
```
LD TRUE
[%MD10:7=BIT_W(%M20:29,3,22,2)]

LD TRUE
[%MD10:4=(%M20:29,3,22,1)]
```

Langage littéral structuré

```
%MD10:7=BIT_W(%M20:29,3,22,2);
%MD10:4=(%M20:29,3,22,1);
```

Exemple

```
%MD10:7=BIT_W(%M20:29,3,22,2);
```



Syntaxe

Opérateurs de copie d'un tableau de bits dans un tableau de mots

Syntaxe
Result:=**BIT_W**(Tab, brow, nbit, wrow)
Result:=**BIT_D**(Tab, brow, nbit, drow)

Opérandes de copie d'un tableau de bits dans un tableau de mots

Type	**Result (résultat)**	**Tab (tableau)**	**brow - nbit wrow ou drow**
Tableaux de mots	%MW:L	-	-
Tableaux de mots doubles	%MD:L	-	-
Tableau de bits	-	%M:L,%Q:L,%I:L,%Xi:L	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mots non indexables	-	-	%IW,%QW,%SW, %NW, Valeur imm., Expr. num.

Note :
- Si le nombre de bits à traiter est supérieur au nombre de bits restants dans le tableau à partir du rang (brow), la fonction exécute la recopie jusqu'au dernier élément du tableau.
- Si le nombre de bits à recopier est supérieur au nombre de bits constituant les mots restants dans le tableau résultat, la fonction arrête la recopie au dernier élément du tableau de mots (ou mots doubles).
- Une valeur négative dans les paramètres brow, nbit, wrow ou drow sera interprétée comme nulle.

Copie d'un tableau de mots dans un tableau de bits

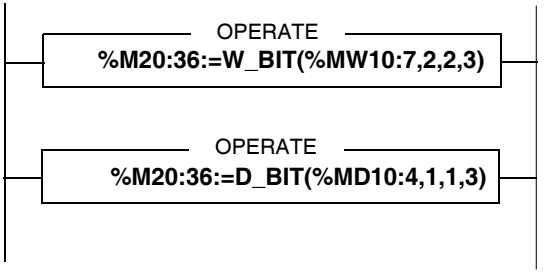
Généralités

La fonction effectue la recopie des bits constituant un tableau ou une partie de tableau de mots (ou doubles mots) dans un tableau de bits.
Dans le tableau de mots (ou doubles mots), le prélèvement est effectué à partir du mot de rang (wrow ou drow) pour un nombre de mots (nwd).
Dans le tableau de bits, la recopie est effectuée à partir du rang (brow) en commençant par le bit de poids le plus faible de chaque mot.

- **W_BIT** : Recopie d'un tableau de mots dans un tableau de bits.
- **D_BIT** : Recopie d'un tableau de doubles mots dans un tableau de bits.

Structure

Langage à contacts



Langage liste d'instructions

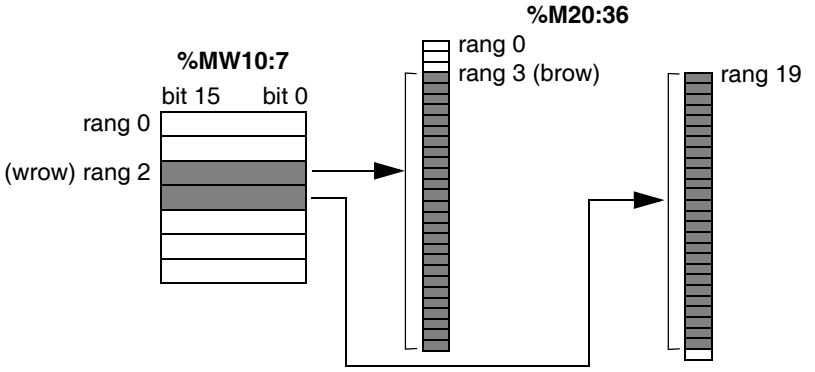
```
LD TRUE
[%M20:36:=W_BIT(%MW10:7,2,2,3)]

LD TRUE
[%M20:36:=D_BIT(%MD10:4,1,1,3)]
```

Langage littéral structuré

```
%M20:36:=W_BIT(%MW10:7,2,2,3);
%M20:36:=D_BIT(%MD10:4,1,1,3);
```

Exemple

```
%M20:36:=W_BIT(%MW10:7,2,2,3);
```



Syntaxe

Opérateurs de copie d'un tableau de mots dans un tableau de bits

Syntaxe
Result:=**W_BIT**(Tab, wrow, nwd, brow)
Result:=**D_BIT**(Tab, drow, nwd, brow)

Opérandes de copie d'un tableau de mots dans un tableau de bits

Type	**Result (résultat)**	**Tab (tableau)**	**wrow ou drow nwd - brow**
Tableaux de bits	%M:L,%Q:L,%I:L	-	-
Tableaux de mots	-	%MW:L,%KW:L	-
Tableau de doubles mots	-	%MD:L,%KD:L	-
Mots indexables	-	-	%MW,%KW,%Xi.T
Mots non indexables	-	-	%IW,%QW,%SW,%NW, Valeur imm.,Expr. num.

Note :
- Si le nombre de bits à traiter est supérieur au nombre de bits restants dans le tableau à partir du rang (wrow), la fonction exécute la recopie jusqu'au dernier élément du tableau.
- Si le nombre de bits à recopier est supérieur au nombre de bits constituant les mots restants dans le tableau résultat, la fonction arrête la recopie au dernier élément du tableau de mots (ou mots doubles).
- Si le nombre de bits à recopier est supérieur au nombre de bits restants dans le tableau résultat, la fonction arrête la recopie au dernier élément du tableau.
- Une valeur négative dans les paramètres brow, nbit, wrow ou drow sera interprétée comme nulle.

2.10 Fonctions "Orphée" : Décalages, compteur

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les fonctions "Orphée": Décalages, compteur, du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Décalages sur mots avec récupération des bits décalés

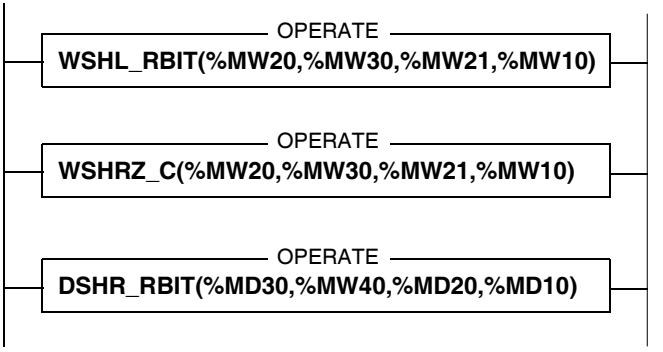
Généralités

Les fonctions effectuent des décalages arithmétiques à gauche ou à droite sur un nombre de décalages (nbit) sur un mot ou sur un double mot (a).
Après décalage, la valeur est rangée dans (résu) et les bits décalés sont rangés dans (rest).

- **WSHL_RBIT** : Décalage à gauche sur mot avec récupération des bits décalés.
- **DSHL_RBIT** : Décalage à gauche sur double mot avec récupération des bits décalés.
- **WSHRZ_C** : Décalage à droite sur mot avec remplissage par des 0 et récupération des bits décalés.
- **DSHRZ_C** : Décalage à droite sur double mot avec remplissage par des 0 et récupération des bits décalés.
- **WSHR_RBIT** : Décalage à droite sur mot avec extension de signe et récupération des bits décalés.
- **DSHR_RBIT** : Décalage à droite sur double mot avec extension de signe et récupération des bits décalés.

Structure

Langage à contacts



Langage liste d'instructions

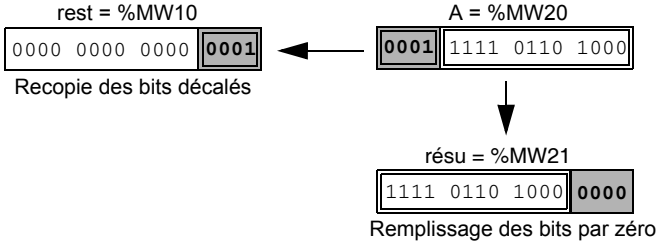
```
LD TRUE
[WSHL_RBIT(%MW20,%MW30,%MW21,%MW10)]

LD TRUE
[WSHRZ_C(%MW20,%MW30,%MW21,%MW10)]

LD TRUE
[DSHR_RBIT(%MD30,%MW40,%MD20,%MD10)]
```

Langage littéral structuré

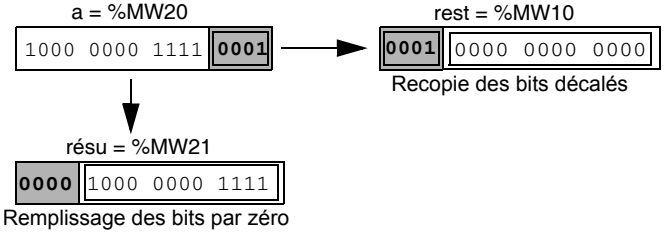
```
WSHL_RBIT(%MW20,%MW30,%MW21,%MW10);

WSHRZ_C(%MW20,%MW30,%MW21,%MW10);

DSHR_RBIT(%MD30,%MW40,%MD20,%MD10);
```

Exemples

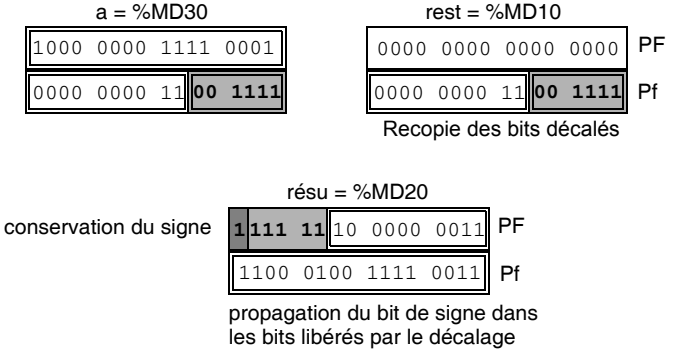
`WSHL_RBIT(%MW20,%MW30,%MW21,%MW10)` avec %MW30 = 4



`WSHRZ_C(%MW20,%MW30,%MW21,%MW10)` avec %MW30 = 4



`DSHR_RBIT(%MD30,%MW40,%MD20,%MD10)` avec %MW40 = 6



Syntaxe

Opérateurs de décalage sur mots avec récupération de bits décalés

Syntaxe
WSHL_BIT(a, nbit, résu, rest)
WSHRZ_C(a, nbit, résu, rest)
WSHR_RBIT(a, nbit, résu, rest)

Opérandes de décalage sur mots avec récupération de bits décalés

Type	a	nbit	résu, rest
Mots indexables	%MW,%KW	%MW,%KW,%Xi.T	%MW
Mots non indexables	%IW,%QW,%SW, %NW,Valeur imm., Expression num.	%IW,%QW,%SW, %NW,Valeur imm., Expression num.	%QW,%SW,%NW

Opérateurs de décalage sur doubles mots avec récupération de bits décalés

Syntaxe
DSHL_BIT(a, nbit, résu, rest)
DSHRZ_C(a, nbit, résu, rest)
DSHR_RBIT(a, nbit, résu, rest)

Opérandes de décalage sur doubles mots avec récupération de bits décalés

Type	a	nbit	résu, rest
Mots doubles indexables	%MD,%KD	-	%MD
Mots doubles non indexables	%ID,%QD,%SD, Valeur immédiate, Expression num.	-	%QD,%SD
Mots indexables	-	%MW,%KW,%Xi.T	-
Mots non indexables	-	%IW,%QW,%SW, %NW,Valeur imm., Expression num.	-

Note : Si le paramètre (nbit) n'est pas entre 1 et 16 pour les décalages sur mot, ou entre 1 et 32 pour les décalages sur double mot, les sorties (résu) et (rest) ne sont pas significatives et le bit système %S18 est positionné à 1.

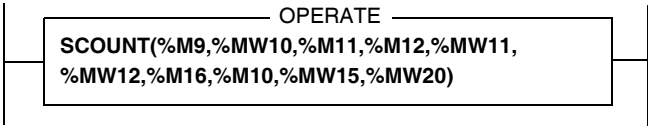
Comptage/décomptage avec signalisation de dépassement

Généralités

La fonction effectue un comptage/décomptage avec signalisation de dépassement. Cette fonction n'est exécutée que si l'entrée de validation (en) est à l'état 1.
Deux entrées séparées (cu et cd) permettent de compter et décompter des événements. La sortie (Qmin) est positionnée à 1 dès que le seuil minimum (min) est atteint, la sortie (Qmax) est positionnée à 1 dès que le seuil maximum (max) est atteint.
La valeur initiale du comptage est fixée par le paramètre (pv) et la valeur courante du comptage est donnée par le paramètre (cv).
Un mot de 16 bits (mwd) permet de mémoriser l'état des entrées cu et cd (bit 0 pour la mémorisation de cu et bit 1 pour la mémorisation de cd).

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW2
0)]
```

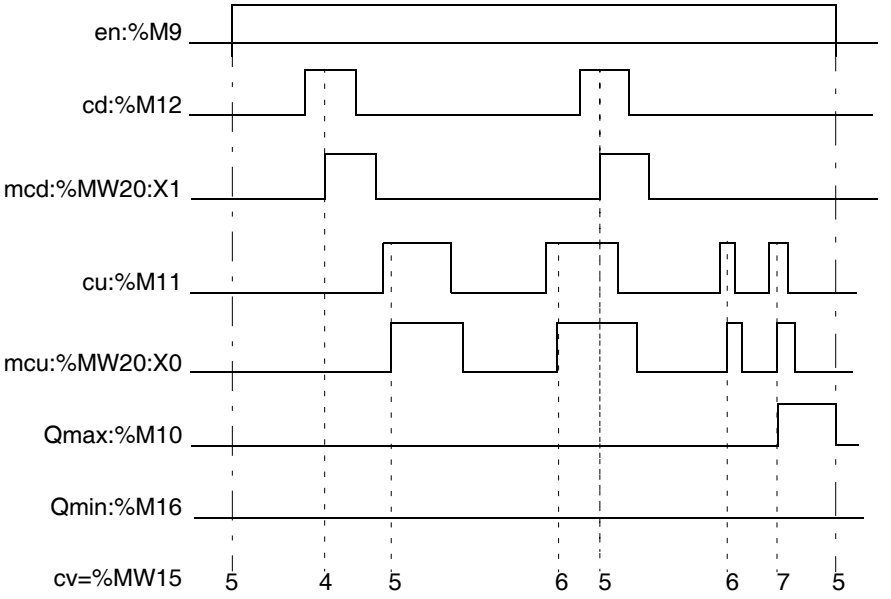
Langage littéral structuré

```
SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW20
);
```

Exemples

```
SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW20
)
```

avec %MW10 (pv) = 5, %MW11 (min) = 0, %MW12 (max) = 7



Syntaxe

Opérateurs de Comptage/décomptage avec signalisation de dépassement

Syntaxe
SCOUNT(en, pv, cu, cd, min, max, Qmin, Qmax, cv, mwd)

Opérandes de Comptage/décomptage avec signalisation de dépassement

Type	**en, cu, cd**	**Qmin, Qmax**	**pv, min, max**	**cv,mwd**
Bits	%I,%Q,%M,%S, %BLK,%.:Xk	%I,%Q,%M	-	-
Mots indexables	-	-	%MW,%KW,%Xi.T	%MW
Mots non indexables	-	-	%IW,%QW,%SW, %NW,Valeur imm., Expression num.	%QW,%SW, %NW

Note :

- Si (en) = 0 alors la fonction n'est plus validée et sur chaque appel, on a :
  Qmin = Qmax = 0
  mcu = mcd = 0 cv = pv
- Si max > min alors :
  cv >= max ---> Qmax = 1 et Qmin = 0
  min < cv < max ---> Qmax = Qmin = 0
  cv <= min ---> Qmax = 0 et Qmin = 1
- Si max < min alors :
  max <= cv <= min ---> Qmax = 1 et Qmin = 0
  cv < max ---> Qmax = 0 et Qmin = 1
  cv > min ---> Qmax = 1 et Qmin = 0
- Si max = min alors :
  cv < min et max ---> Qmax = 0 et Qmin = 1
  cv >= min et max ---> Qmax = 1 et Qmin = 0
- Une modification du paramètre (pv) avec (en) à l'état 1 n'a aucune incidence sur le fonctionnement.
- Une valeur négative pour les paramètres (pv) et (min) est interprétée comme une valeur nulle.
- Une valeur inférieure à 1 pour le paramètre (max) est interprétée comme égale à 1.

Décalages circulaire

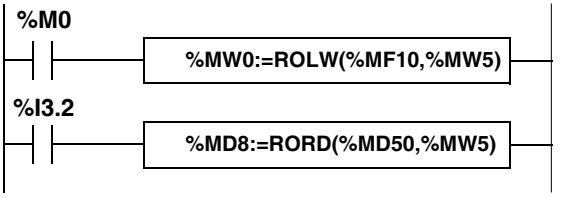
Généralités

Les fonctions effectuent des décalages circulaires à gauche ou à droite sur un mot ou sur un double mot.

- **ROLW** : décalage circulaire vers la gauche sur un mot avec nombre de décalages calculé
- **RORW** : décalage circulaire vers la droite sur un mot avec nombre de décalages calculé
- **ROLD** : décalage circulaire vers la gauche sur un double mot avec nombre de décalages calculé
- **RORD** : décalage circulaire vers la droite sur un double mot avec nombre de décalages calculé

Structure

Langage à contacts



Langage liste d'instructions

```
LD %M0
[%MW0:=ROLW(%MW10,%MW5)]

LD %I3.2
[%MD10:=RORD(%MD100,%MW5)]
```

Langage littéral structuré

```
IF %M0 THEN
 %MW0:=ROLW(%MW10,%MW5);
END_IF;
IF %I3.2 THEN
 %MD8:=RORD(%MD100,%MW5);
END_IF
```

Syntaxe

Opérateurs de décalages circulaires

Opérateurs	Syntaxe
ROLW, RORW, ROLD, RORD	Op1:=Opérateur(Op2,n)

Opérandes de décalages circulaires sur mot **ROLW**, **RORW**

Type	Opérande 1 (Op1)	Opérande 2 (Op2)	Nombre de position (n)
Mots indexables	%MW	%MW,%KW,%Xi.T	%MW,%KW,%Xi.T
Mots non indexables	-	Val.imm.,%IW,%QW,%SW,%NW,%BLK, Expr.num.	Val.imm.,%IW,%QW, %SW,%NW,%BLK, Expr.num.

Opérandes de décalages circulaires sur double mot **ROLD**, **RORD**

Type	Opérande 1 (Op1)	Opérande 2 (Op2)	Nombre de position (n)
Mots indexables	%MD	%MD,%KD	%MW,%KW,%Xi.T
Mots non indexables	%QD,%SD	Val.imm.,%ID,%QD,%SD ,Expr.num.	Val.imm.,%IW,%QW, %SW,%NW,%BLK, Expr.num.

Note : On utilisera de préférence les instructions de base ROL et ROR (lorsque le nombre de décalage est statique, car ces instructions sont plus performantes.

2.11 Fonctions de temporisation

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les fonctions de temporisation du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Fonctions de temporisation

Généralités

Ces fonctions de temporisation contrairement aux blocs fonction prédéfinis ne sont pas limitées en nombre et peuvent être utilisées dans le code des blocs fonction DFB.

4 fonctions de temporisation sont proposées.

- **FTON** : Temporisation d'enclenchement.
- **FTOF** : Temporisation de déclenchement.
- **FTP** : Temporisation d'impulsion : Temporisation.
- **FPULSOR** : signal rectangulaire : Temporisation.

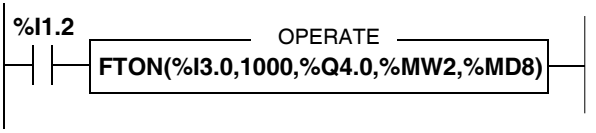
Fonction temporisation d'enclenchement

Généralités

Cette fonction permet de gérer des retards à l'enclenchement. Ce retard est programmable.

Structure

Langage à contacts



Langage liste d'instructions

```
LD I1.2
[FTON(%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 FTON(%I3.0,1000,%Q4.0,%MW2,%MD8);
END_IF;
```

Syntaxe

Opérateurs de la fonction temporisation d'enclenchement FTON

Syntaxe
FTON(EN,PT,Q,ET,PRIV)

Opérandes de la fonction temporisation d'enclenchement FTON

Type	EN	PT	Q	ET	PRIV
Mots indexables	-	%MW,%KW, %Xi.T	-	%MW	-
Mots non indexables	-	%IW,%QW, %SW,%NW, Valeur immédiate, Expression numérique	-	%IW,%QW	-
Mots doubles indexables	-	-	-	-	%MD

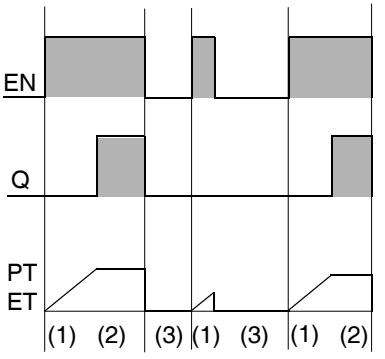
Type	EN	PT	Q	ET	PRIV
Bits	%I,%Q,%M, %S,%BLK, %*:Xk,%X		%I,%Q,%M %S,%*:Xk, %X	-	-

Caractéristiques Caractéristiques de la fonction temporisation d'enclenchement FTON

Caractéristique	Repère	Valeur
Entrée "Armement"	EN	Sur front montant démarre la temporisation
Valeur de présélection	PT	Mot d'entrée qui détermine la durée (en centième de seconde) de la temporisation. Il permet de définir une durée maximum de 5 min et 27 s avec une précision de 10 ms. (1)
Sortie "Temporisateur"	Q	Sortie mise à 1 en fin de temporisation.
Valeur courante	ET	Mot de sortie qui croît de 0 à PT sur écoulement du temporisateur.
Variable de calcul	PRIV	Double mot pour mémorisation des états internes. Associer à ce double mot une variable de l'application exclusivement réservée à cet effet.

Note : (1) une modification de ce mot est prise en compte pendant la temporisation.

Fonctionnement Description du fonctionnement de la fonction temporisation d'enclenchement FTON

Etape	Action	Description	Illustration
1	Front montant sur l'entrée EN	Le temporisateur est lancé : sa valeur courante ET croît de 0 vers PT (centième de seconde).	
2	La valeur courante a atteint PT	Le bit de sortie Q passe à 1, puis reste à 1 tant que l'entrée EN est à 1.	
3	l'entrée EN est à 0	Le temporisateur est arrêté même s'il était en cours d'évolution: ET prend la valeur 0.	

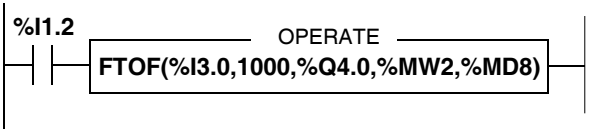
Fonction temporisation de déclenchement

Généralités

Cette fonction permet de gérer des retards au déclenchement. Ce retard est programmable.

Structure

Langage à contacts



Langage liste d'instructions

```
LD I1.2
[FTOF(%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 FTOF(%I3.0,1000,%Q4.0,%MW2,%MD8);
END_IF;
```

Syntaxe

Opérateurs de la fonction temporisation de déclenchement FTOF

Syntaxe
FTOF(EN,PT,Q,ET,PRIV)

Opérandes de la fonction temporisation de déclenchement FTOF : identiques à FTON (Voir *Fonction temporisation d'enclenchement, p. 265*)

Caractéristiques

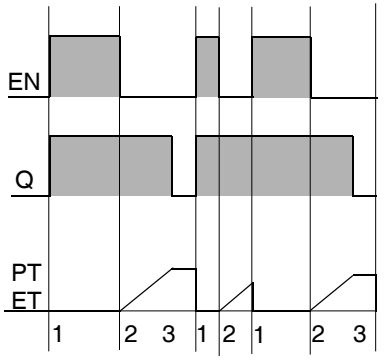
Caractéristiques de la fonction temporisation de déclenchement FTOF

Caractéristique	Repère	Valeur
Entrée "Armement"	EN	Sur front descendant démarre la temporisation.
Valeur de présélection	PT	Mot d'entrée qui détermine la durée (en centième de seconde) de la temporisation. Il permet de définir une durée maximum de 5 min et 27 s avec une précision de 10 ms. (1)
Sortie "Temporisateur"	Q	Sortie mise à 1 sur front montant de EN et mis à 0 en fin de temporisation.

Caractéristique	Repère	Valeur
Valeur courante	ET	Mot de sortie qui croît de 0 à PT sur écoulement du temporisateur.
Variable de calcul	PRIV	Double mot pour mémorisation des états internes. Associer à ce double mot une variable de l'application exclusivement réservée à cet effet.

Note : (1) une modification de ce mot est prise en compte pendant la temporisation.

Fonctionnement Description du fonctionnement de la fonction temporisation de déclenchement FTOF

Etape	Action	Description	Illustration
1	Front montant sur l'entrée EN.	La valeur courante ET prend la valeur 0 (même si le temporisateur est en cours d'évolution) et le bit de sortie Q passe à 1 (ou reste à 1).	
2	Lors du front descendant sur l'entrée EN.	le temporisateur est lancé, puis la valeur courante croît de 0 vers PT (centième de seconde).	
3	Quand la valeur courante a atteint PT.	Le bit de sortie Q retombe à 0.	

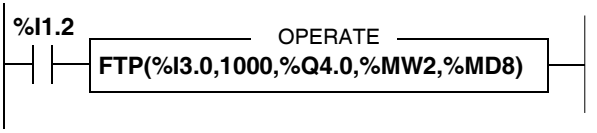
Fonction temporisation d'impulsion

Généralités

Cette fonction permet d'élaborer une impulsion de durée précise. Cette durée est programmable.

Structure

Langage à contacts



Langage liste d'instructions

```
LD I1.2
[FTP(%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 FTP(%I3.0,1000,%Q4.0,%MW2,%MD8);
END_IF;
```

Syntaxe

Opérateurs de la fonction temporisation d'impulsion FTP

Syntaxe
FTP(EN,PT,Q,ET,PRIV)

Opérandes de la fonction temporisation d'impulsion FTP : identique à FTON (Voir *Fonction temporisation d'enclenchement, p. 265*)

Caractéristiques

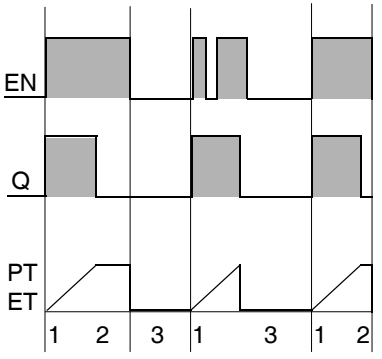
Caractéristiques de la fonction temporisation d'impulsion FTP

Caractéristique	Repère	Valeur
Entrée "Armement"	EN	Sur front descendant démarre la temporisation.
Valeur de présélection	PT	Mot d'entrée qui détermine la durée (en centième de seconde) de la temporisation. Il permet de définir une durée maximum de 5 min et 27 s avec une précision de 10 ms. (1)
Sortie "Temporisateur"	Q	Sortie mise à 1 en fin de temporisation.
Valeur courante	ET	Mot de sortie qui croît de 0 à PT sur écoulement du temporisateur.

Caractéristique	Repère	Valeur
Variable de calcul	PRIV	Double mot pour mémorisation des états internes. Associer à ce double mot une variable de l'application exclusivement réservée à cet effet.

Note : (1) une modification de ce mot est prise en compte pendant la temporisation.

Fonctionnement Description du fonctionnement de la fonction temporisation d'impulsion FTP

Etape	Action	Description	Illustration
1	Front montant sur l'entrée EN.	Le temporisateur est lancé (si le celui-ci n'est pas déjà en cours d'évolution) et la valeur courante ET croît de 0 vers PT (centième de seconde).Le bit de sortie Q passe à 1.	 Ce monostable n'est pas réarmable.
2	Quand la valeur courante a atteint PT.	Le bit de sortie Q retombe à 0.	
3	L'entrée EN et la sortie Q sont à 0	PT prend la valeur 0.	

Fonction générateur de signal rectangulaire

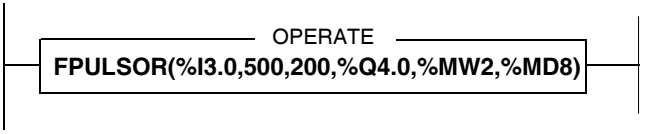
Généralités

Cette fonction permet de générer un signal rectangulaire périodique dont on peut faire varier la largeur du créneau à 1 et du créneau à 0 par programme au moyen de 2 temporisateurs :

- **TON** : temporisation à la montée (pour le créneau à 1).
- **TOFF** : temporisation à la retombée (pour le créneau à 0).

Structure

Langage à contacts



Langage liste d'instructions

```
LD TRUE
[FPULSOR(%I3.0,500,200,%Q4.0,%MW2,%MD8)]
```

Langage littéral structuré

```
IF %I1.2 THEN
 FPULSOR(%I3.0,500,200,%Q4.0,%MW2,%MD8);
END_IF;
```

Syntaxe

Opérateurs de la fonction générateur de signal rectangulaire FPULSOR

Syntaxe
FPULSOR(EN,TON,TOFF,Q,ET,PRIV)

Opérandes de la fonction générateur de signal rectangulaire FPULSOR:

Type	EN	TON,TOFF	Q	ET	PRIV
Mots indexables	-	%MW,%KW, %Xi.T	-	%MW	-
Mots non indexables	-	-	-	%IW,%QW	-

Type	EN	TON,TOFF	Q	ET	PRIV
Mots doubles indexables	-	-	-	-	%MD
Bits	%BLK,%*:Xk, %X	%I,%Q,%M, %S	%S,%*:Xk,%X	%I,%Q,%M	-

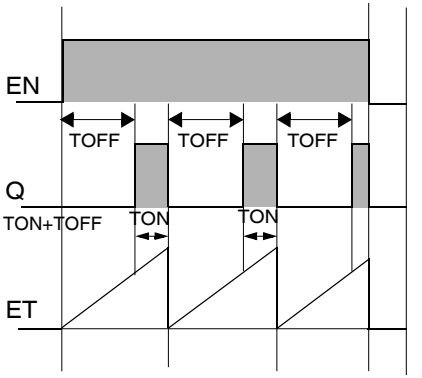
Caractéristiques

Caractéristiques de la fonction générateur de signal rectangulaire FPULSOR:

Caractéristique	Repère	Valeur
Entrée "Armement"	EN	Sur front montant démarre la génération du signal rectangulaire.
Valeur de présélection (créneau à 1)	TON	Mot d'entrée qui détermine la durée (en centième de seconde) de la durée du créneau à 1. Il permet de définir une durée maximum de 5 min et 27 s avec une précision de 10 ms. (1)
Valeur de présélection (créneau à 0)	TOFF	Mot d'entrée qui détermine la durée (en centième de seconde) de la durée du créneau à 0. Il permet de définir une durée maximum de 5 min et 27 s avec une précision de 10 ms. (1)
Sortie signal rectangulaire	Q	Sortie créneau à 0 sur la durée TOFF, à 1 sur la durée TON.
Valeur courante	ET	Mot de sortie qui croît de 0 à TON+TOFF sur écoulement du temporisateur.
Variable de calcul	PRIV	Double mot pour mémorisation des états internes. Associer à ce double mot une variable de l'application exclusivement réservée à cet effet.

Note : (1) une modification de ces mots est prise en compte pendant la temporisation. La somme TOFF+TON a une durée maximum de 5 min et 27 s.

Fonctionnement Description du fonctionnement de la fonction générateur de signal rectangulaire FPULSOR:

Etape	Action	Description	Illustration
1	Front montant sur l'entrée EN	la génération du signal rectangulaire est lancée : (si le signal n'est pas déjà en cours d'évolution) sa valeur courante ET croît de 0 vers TON+TOFF (centièmes de seconde).	
2	Tant que la temporisation TOFF n'est pas écoulée	Le bit de sortie Q reste à 0.	
3	TOFF est écoulée, TON s'enclenche	Le bit de sortie Q passe à 1 jusqu'à la fin de TON et le générateur boucle sur (2) et (3).	
4	EN passe à 0	TON et TOFF sont remis à 0, le bit de sortie Q passe à 0.	

2.12 Fonctions d’archivage de données

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les fonctions d’archivage de données du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Fonctions d'archivage de données

Présentation

Ces fonctions permettent d'archiver des données par programme dans une zone dédiée des cartes mémoire utilisateur.

Exemple d'application

- stockage automatique, de données (consignations d'état, historiques, ...) de l'application dans la carte mémoire utilisateur située dans l'emplacement mémoire du processeur de l'automate.
- sauvegarde de recettes de production dans cette même carte mémoire.

Différentes fonctions

6 fonctions permettent l'archivage et la restitution des données.

Les fonctions suivantes s'appliquent indifféremment aux cartes mémoire PCMCIA type 1 (cartes mémoire situées dans l'emplacement 0 du processeur) et Type 3 (cartes mémoire situées dans l'emplacement 1 du processeur).

- **SET_PCM_EXT** : pour initialiser à une valeur tout ou partie de la zone d'archivage de la carte mémoire,
- **WRITE_PCM_EXT** pour écrire les données dans la zone d'archivage de la carte mémoire,
- **READ_PCM_EXT** : pour lire les données dans la zone d'archivage de la carte mémoire.

Note : Ces fonctions nécessitent :
- PL7 V4.2 ou supérieure,
- une version d'OS de l'automate (SV) égale ou supérieure à 5.2.

Les fonctions suivantes s'appliquent seulement pour les cartes mémoire PCMCIA Type 1 (cartes mémoire situées dans l'emplacement 0 du processeur).

- **SET_PCMCIA** : pour initialiser à une valeur tout ou partie de la zone d'archivage de la carte mémoire,
- **WRITE_PCMCIA** pour écrire les données dans la zone d'archivage de la carte mémoire,
- **READ_PCMCIA** : pour lire les données dans la zone d'archivage de la carte mémoire.

Note : l'accès aux données stockées dans la zone archivage d'une carte mémoire n'est possible que depuis l'application résidente dans l'automate par ces 6 fonctions de base. En aucun cas une station distante ne peut y accéder directement au travers d'un réseau ou bus de communication.

Initialisation de la zone d'archivage étendue

Présentation

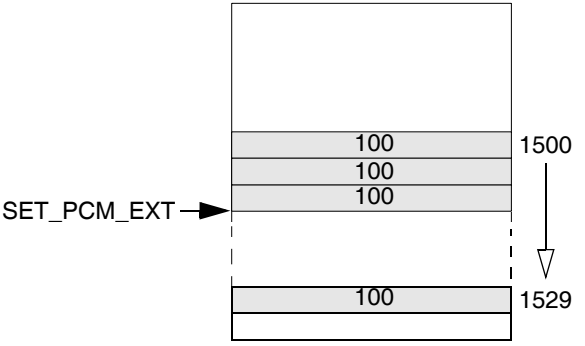
La fonction **SET_PCM_EXT** permet d'initialiser à la valeur désirée tout ou partie de la zone archivage d'une carte mémoire.

Cette fonction utilise 5 paramètres :

- **SLOT** : numéro de la voie ou est insérée la carte mémoire PCMCIA :
  - 0 pour une carte située dans l'emplacement 0 du processeur (carte PCMCIA Type 1),
  - 1 pour une carte située dans l'emplacement 1 du processeur (carte PCMCIA Type 3).
- **DEST** : adresse de la zone d'archivage à partir de laquelle s'effectue l'Initialisation.
- **NUM** : nombre de mots à initialiser.
- **VAL** : valeur d'Initialisation.
- **CR** : code donnant le résultat de l'exécution de la commande d'Initialisation.

Exemple

Représentation de la carte mémoire utilisateur :

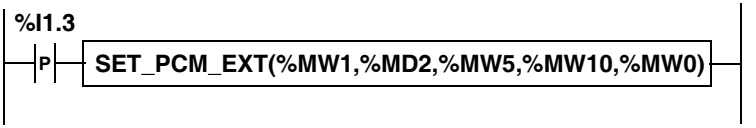


Dans cet exemple :

- **SLOT** = %MW1 , %MW1 contenant la valeur 1.
- **DEST** = %MD2 , %MD2 contenant la valeur 1500.
- **NUM** = %MW5, %MW5 contenant la valeur 30.
- **VAL** = %MW10, %MW10 contenant la valeur 100.

Structure

Langage à contact :



Langage liste d'instructions

```
LDR   %I1.3
[SET_PCM_EXT(%MW1,%MD2,%MW5,%MW10,%MW0)]
```

Langage littéral structuré :

```
IF RE %I1.3 THEN
  SET_PCM_EXT(%MW1,%MD2,%MW5,%MW10,%MW0);
END_IF;
```

Syntaxe

Syntaxe de la fonction :

SET_PCM_EXT (SLOT,DEST,NUM,VAL,CR)

Paramètres :

Type	SLOT	DEST	NUM	VAL	CR
Mots indexables	%MW, Val.imm.	-	%MW, Val.imm.	%MW, Val.imm.	%MW
Mots non indexables	-	-	-	-	%QW,%SW, %NW
Mots doubles indexables	-	%MD,Val.imm.	-	-	-
Mots doubles non indexables	-	%QD,%SD	-	-	-

Codage du paramètre **état** retourné après la commande d'initialisation :

Valeur (en hexadécimal)	Signification
0000	initialisation correctement effectuée
0201	pas de zone fichier dans la carte mémoire
0202	défaut carte mémoire
0204	carte mémoire protégée en écriture
0241	DEST < 0

Valeur (en hexadécimal)	Signification
0242	DEST + NUM - 1 -> adresse la plus haute de la carte
0401	NUM = 0 ou négatif
0402	numéro d'emplacement incorrect (différent de 0 ou 1)
0501	Service non supporté

Initialisation de la zone d'archivage

Présentation

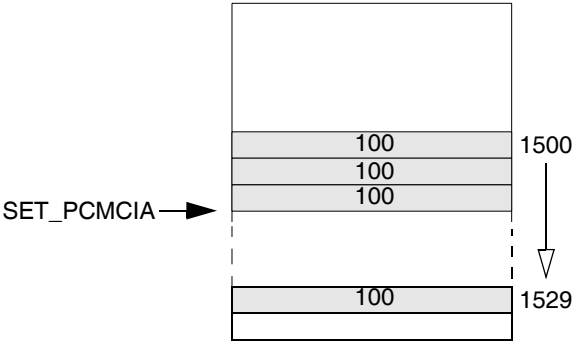
La fonction **SET_PCMCIA** permet d'initialiser à la valeur désirée tout ou partie de la zone archivage de la carte mémoire utilisateur (PCMCIA type 1).

Cette fonction utilise 4 paramètres :

- **DEST** : adresse de la zone d'archivage à partir de laquelle s'effectue l'Initialisation.
- **NUM** : nombre de mots à initialiser.
- **VAL** : valeur d'Initialisation.
- **CR** : code donnant le résultat de l'exécution de la commande d'Initialisation.

Exemple

Représentation de la carte mémoire utilisateur :

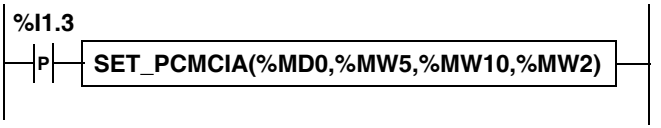


Dans cet exemple :

- **DEST** = %MD0 , %MD0 contenant la valeur 1500
- **NUM** = %MW5, %MW5 contenant la valeur 30
- **VAL** = %MW10, %MW10 contenant la valeur 100

Structure

Langage à contact :



Langage liste d'instructions

```
LDR   %I1.3
[SET_PCMCIA(%MD0,%MW5,%MW10,%MW2)]
```

Langage littéral structuré :

```
IF RE %I1.3 THEN
  SET_PCMCIA(%MD0,%MW5,%MW10,%MW2);
END_IF;
```

Syntaxe

Syntaxe de la fonction :

SET_PCMCIA (DEST,NUM,VAL,CR)

Paramètres :

Type	DEST	NUM	VAL	CR
Mots indexables	-	%MW,Val.imm.	%MW,Val.imm.	%MW
Mots non indexables	-	-	-	%QW,%SW, %NW
Mots doubles indexables	%MD,Val.imm.	-	-	-
Mots doubles non indexables	%QD,%SD	-	-	-

Codage du paramètre **CR** retourné après la commande d'initialisation :

Valeur (en hexadécimal)	Signification
0000	initialisation correctement effectuée
0201	pas de zone fichier dans la carte mémoire
0202	défaut carte mémoire
0204	carte mémoire protégée en écriture
0241	DEST négative
0242	EST + NUM - 1 -> l'adresse la plus haute de la carte mémoire
0401	NUM = 0 ou négatif

Ecriture des données dans la zone d'archivage étendue

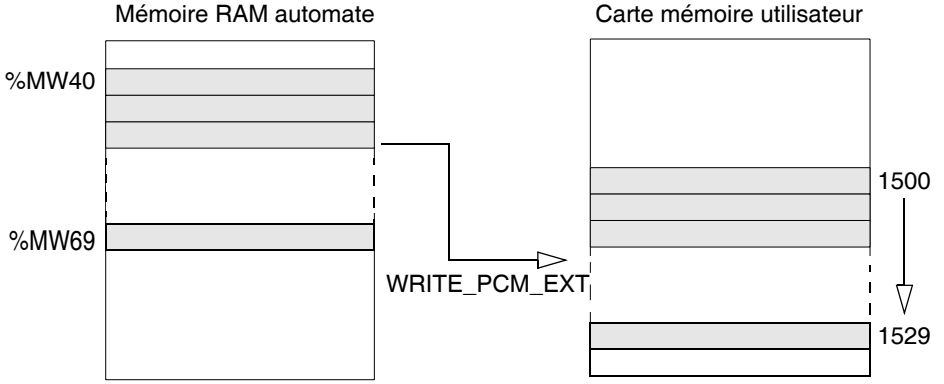
Présentation

La fonction **WRITE_PCM_EXT** permet de transférer des données de la mémoire RAM de l'automate (mots %MW) dans la zone d'archivage d'une carte mémoire. Cette fonction utilise 5 paramètres :

- **SLOT** : numéro de la voie ou est insérée la carte mémoire PCMCIA :
  - 0 pour une carte située dans l'emplacement 0 du processeur (carte PCMCIA Type 1),
  - 1 pour une carte située dans l'emplacement 1 du processeur (carte PCMCIA Type 3).
- **DEST** : adresse de la zone d'archivage à partir de laquelle seront stockés les données
- **NUM** : nombre de mots à stocker
- **EMIS** : mot contenant l'adresse de début de la zone à transférer dans la carte mémoire
- **CR** : code donnant le résultat de la commande d'écriture.

Exemple

Illustration :

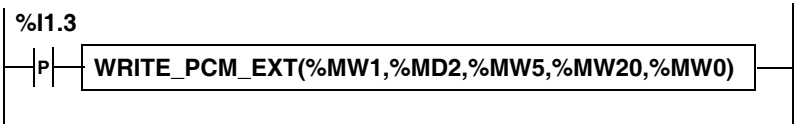


Dans cet exemple :

- **SLOT** = %MW1, %MW1 contenant la valeur 1
- **DEST** = %MD2, %MD2 contenant la valeur 1500
- **NUM** = %MW5, %MW5 contenant la valeur 30
- **EMIS** = %MW20, %MW20 contenant la valeur 40

Structure

Langage à contact :



Langage liste d'instructions :

```
LDR   %I1.3
[WRITE_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0)]
```

Langage littéral structuré :

```
IF RE %I1.3 THEN
  WRITE_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0);
END_IF;
```

Syntaxe

Syntaxe de la fonction :

WRITE_PCM_EXT (SLOT,DEST,NUM,VAL,CR)

Paramètres :

Type	SLOT	DEST	NUM	EMIS	CR
Mots indexables	%MW, Val.imm.	-	%MW, Val.imm.	%MW, Val.imm.	%MW
Mots non indexables	-	-	-	-	%QW,%SW, %NW
Mots doubles indexables	-	%MD,Val.imm.	-	-	-
Mots doubles non indexables	-	%QD,%SD	-	-	-

Codage du paramètre **état** retourné après la commande d'écriture :

Valeur (en hexadécimal)	Signification
0000	écriture correctement effectuée
0102	EMIS + NUM - 1 -> nombre maximal de %MW déclaré dans l'automate
0104	aucune application valide ou aucun %MW dans l'automate
0201	pas de zone fichier dans la carte mémoire
0202	défaut carte mémoire
0204	carte mémoire protégée en écriture

Valeur (en hexadécimal)	Signification
0241	DEST < 0
0242	DEST + NUM - 1 -> l'adresse la plus haute de la carte mémoire
0401	NUM = 0
0402	numéro d'emplacement incorrect (différent de 0 ou 1)
0501	Service non supporté

Ecriture des données dans la zone d'archivage

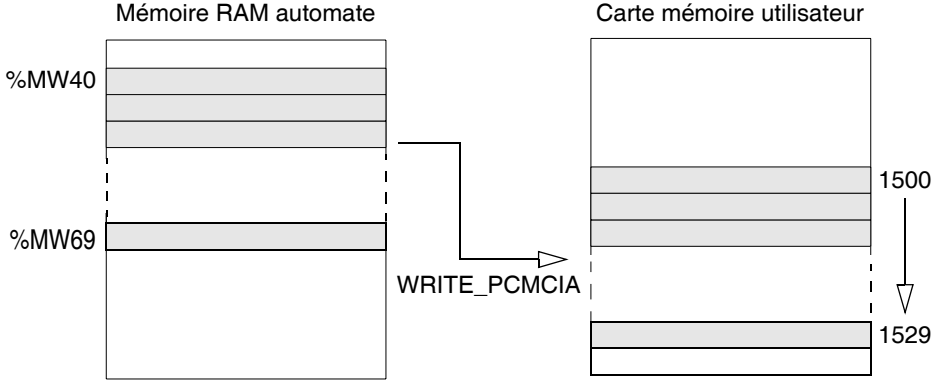
Présentation

La fonction **WRITE_PCMCIA** permet de transférer des données de la mémoire RAM de l'automate (mots %MW) dans la zone d'archivage de la carte mémoire utilisateur (PCMCIA type 1).
Cette fonction utilise 4 paramètres :

- **DEST** : adresse de la zone d'archivage à partir de laquelle seront stockés les données,
- **NUM** : nombre de mots à stocker,
- **EMIS** : mot contenant l'adresse de début de la zone à transférer dans la carte mémoire,
- **CR** : code donnant le résultat de la commande d'écriture.

Exemple

Illustration :

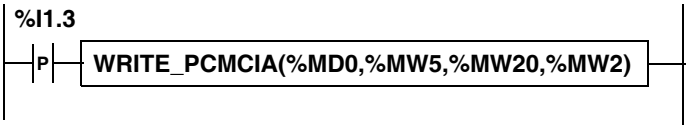


Dans cet exemple :

- **DEST** = %MD0, %MD0 contenant la valeur 1500
- **NUM** = %MW5, %MW5 contenant la valeur 30
- **EMIS** = %MW20, %MW20 contenant la valeur 40

Structure

Langage à contact :



Langage liste d'instructions :

```
LDR   %I1.3
[WRITE_PCMCIA(%MD0,%MW5,%MW20,%MW2)]
```

Langage littéral structuré :

```
IF RE %I1.3 THEN
  WRITE_PCMCIA(%MD0,%MW5,%MW20,%MW2);
END_IF;
```

Syntaxe

Syntaxe de la fonction :

WRITE_PCMCIA (DEST,NUM,EMIS,CR)

Paramètres :

Type	DEST	NUM	EMIS	CR
Mots indexables	-	%MW,Val.imm.	%MW,Val.imm.	%MW
Mots non indexables	-	-	-	%QW,%SW, %NW
Mots doubles indexables	%MD,Val.imm.	-	-	-
Mots doubles non indexables	%QD,%SD	-	-	-

Codage du paramètre **CR** retourné après la commande d'écriture :

Valeur (en hexadécimal)	Signification
0000	écriture correctement effectuée
0102	EMIS + NUM - 1 -> nombre maximal de %MW déclaré dans l'automate
0104	aucune application valide ou aucun %MW dans l'automate
0201	pas de zone fichier dans la carte mémoire
0202	défaut carte mémoire
0204	carte mémoire protégée en écriture

Valeur (en hexadécimal)	Signification
0241	DEST < 0
0242	DEST + NUM - 1 -> l'adresse la plus haute de la carte mémoire
0401	NUM = 0

Lecture des données dans la zone d'archivage étendue

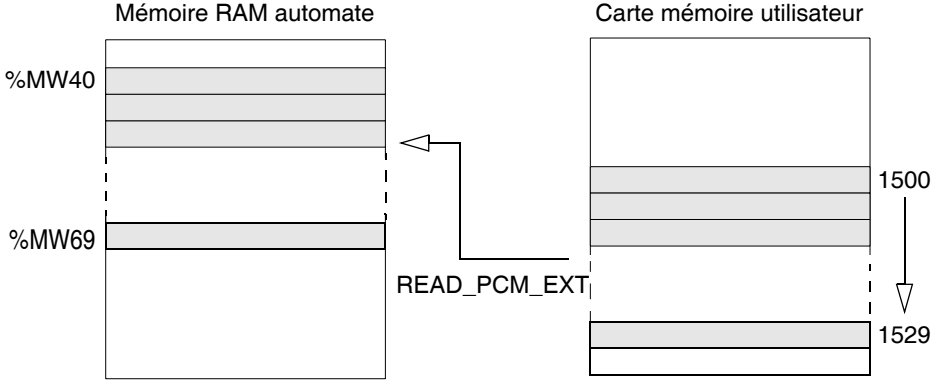
Présentation

La fonction **READ_PCM_EXT** permet de transférer des données de la zone d'archivage de la carte mémoire utilisateur dans la mémoire RAM de l'automate (mots %MW).
Cette fonction utilise 5 paramètres :

- **SLOT** : numéro de la voie ou est insérée la carte mémoire PCMCIA :
  - 0 pour une carte située dans l'emplacement 0 du processeur (carte PCMCIA Type 1),
  - 1 pour une carte située dans l'emplacement 1 du processeur (carte PCMCIA Type 3).
- **SRC** : adresse de la zone d'archivage dans laquelle sont stockés les données à lire
- **NUM** : nombre de mots à lire
- **RCPT** : mot contenant l'adresse de début de la zone transférée par la carte mémoire
- **CR** : code donnant le résultat de l'exécution de la commande de lecture

Exemple

Illustration :

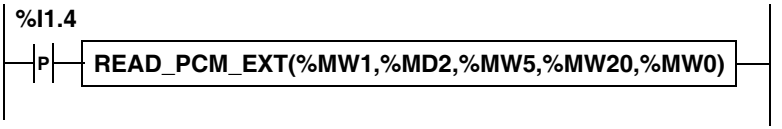


Dans cet exemple :

- **SLOT** = %MW1, %MW1 contenant la valeur 1
- **SRC** = %MD2, %MD2 contenant la valeur 1500
- **NUM** = %MW5, %MW5 contenant la valeur 30
- **RCPT** = %MW20, %MW20 contenant la valeur 40

Structure

Langage à contact :



Langage liste d'instructions:

```
LDR   %I1.4
[READ_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0)]
```

Langage littéral structuré :

```
IF RE %I1.4 THEN
  READ_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0);
END_IF;
```

Syntaxe

Syntaxe de la fonction :

READ_PCM_EXT (SLOT,SRC,NUM,RCPT,CR)

Paramètres :

Type	SLOT	SRC	NUM	RCPT	CR
Mots indexables	%MW, Val.imm.	-	%MW, Val.imm.	%MW, Val.imm.	%MW
Mots non indexables	-	-	-	-	%QW,%SW, %NW
Mots doubles indexables	-	%MD,Val.imm.	-	-	-
Mots doubles non indexables	-	%QD,%SD	-	-	-

Codage du paramètre **CR** retourné après la commande d'écriture :

Valeur (en hexadécimal)	Signification
0000	lecture correctement effectuée
0102	SRC + NUM -1 -> nombre maximal de %MW déclaré dans l'automate
0104	aucune application valide ou aucun %MW dans l'automate
0201	pas de zone fichier dans la carte mémoire
0202	défaut carte mémoire
0204	carte mémoire protégée en écriture

Valeur (en hexadécimal)	Signification
0241	SRC < 0
0242	SRC + NUM -1 -> l'adresse la plus haute de la carte mémoire
0401	NUM = 0
0402	numéro d'emplacement incorrect (différent de 0 ou 1)
0501	Service non supporté

Lecture des données dans la zone d'archivage

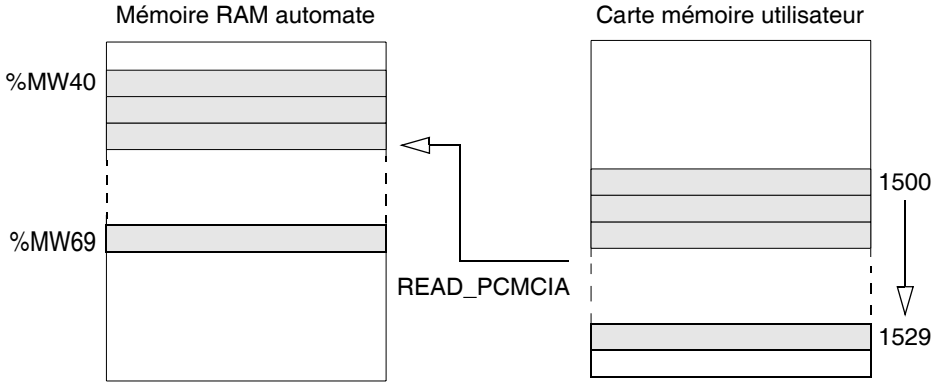
Présentation

La fonction **READ_PCMCIA** permet de transférer des données de la zone d'archivage de la carte mémoire utilisateur (PCMCIA type 1) dans la mémoire RAM de l'automate (mots %MW).
Cette fonction utilise 4 paramètres :

- **SRC** : adresse de la zone d'archivage dans laquelle sont stockés les données à lire,
- **NUM** : nombre de mots à lire,
- **RCPT** : mot contenant l'adresse de début de la zone transférée par la carte mémoire,
- **CR** : code donnant le résultat de l'exécution de la commande de lecture.

Exemple

Illustration :

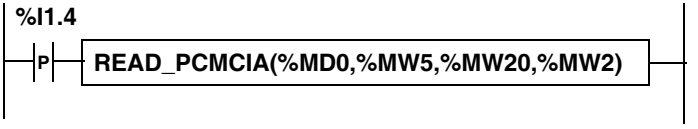


Dans cet exemple :

- **SRC** = %MD0, %MD0 contenant la valeur 1500
- **NUM** = %MW5, %MW5 contenant la valeur 30
- **RCPT** = %MW20, %MW20 contenant la valeur 40

Structure

Langage à contact :



Langage liste d'instructions:

```
LDR   %I1.4
[READ_PCMCIA(%MD0,%MW5,%MW20,%MW2)]
```

Langage littéral structuré :

```
IF RE %I1.4 THEN
  READ_PCMCIA(%MD0,%MW5,%MW20,%MW2);
END_IF;
```

Syntaxe

Syntaxe de la fonction :

READ_PCMCIA (SRC,NUM,RCPT,CR)

Paramètres :

Type	SRC	NUM	RCPT	CR
Mots indexables	-	%MW,Val.imm.	%MW,Val.imm.	%MW
Mots non indexables	-	-	-	%QW,%SW, %NW
Mots doubles indexables	%MD,Val.imm.	-	-	-
Mots doubles non indexables	%QD,%SD	-	-	-

Codage du paramètre **CR** retourné après la commande d'écriture :

Valeur (en hexadécimal)	Signification
0000	lecture correctement effectuée
0102	RCPT + NUM - 1 -> nombre maximal de %MW déclaré dans l'automate
0104	aucune application valide ou aucun %MW dans l'automate
0201	pas de zone fichier dans la carte mémoire
0202	défaut carte mémoire
0204	carte mémoire protégée en écriture
0241	SRC < 0
0242	RCPT + NUM - 1 > l'adresse la plus haute de la carte mémoire
0401	NUM= 0

2.13 Fonctions Grafcet

Fonction de remise à zéro des temps d'activités d'étapes

Généralités

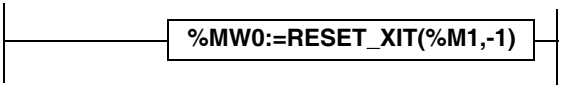
Cette fonction réinitialise tous les temps d'activités des étapes du traitement séquentiel "Chart" ou d'une macro étape.
Cette fonction est valide sur automate Premium/Atrium (version logicielle supérieure ou égale à V3.0).

Elle possède les paramètres d'entrées et de sortie suivants :

Type	Paramètres	Rôle
Entrée	En	Condition d'activation de la fonction
	Num	Numéro du module Grafcet à réinitialiser. Il vaut : ● -1 si le module est le traitement séquentiel "Chart", ● ou le numéro de la macro étape concernée.
Sortie	Result	Compte rendu de l'exécution de la fonction.

Structure

Langage à contacts



Langage liste d'instructions

```
LD True
[%MW0:=RESET_XIT(%M1,-1)]
```

Langage littéral structuré

```
%MW0:=RESET_XIT(%M1,-1);
```

Syntaxe

Opérateur :

Syntaxe
Result:=RESET_XIT(En,Num)

Opérandes :

Type	Resultat (Résult)	Condition de validation (En)	Numéro de module Grafcet (Num)
Bits	-	%M	-
Mots	%MW	-	%MW, %KW, Valeur immédiate

Résultat

Codage du paramètre résultat retourné après exécution de l'instruction :

Valeur (en hexadécimal)	Signification
0000	Opération correcte
FFFF	Paramètre d'entrée hors bornes : la macro étape n'existe pas dans l'application.
FFFA	Le type d'automate est un MICRO

Objets système

3

Présentation

Contenu de ce chapitre

Ce chapitre décrit les tous bits sytème et mots sytème du langage PL7

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

3.1 Bits système

Présentation

Objet de ce sous-chapitre

Ce chapitre décrit les bits système du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Présentation des bits système

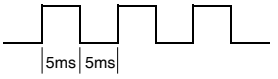
Généralités

Les automates TSX 37 et TSX 57 disposent de bits système %Si qui indiquent les états de l'automate ou permettent d'agir sur le fonctionnement de celui-ci.

Ces bits peuvent être testés dans le programme utilisateur afin de détecter tout événement de fonctionnement devant entraîner une procédure particulière de traitement. Certains d'entre eux doivent être remis dans leur état initial ou normal par programme. Cependant, les bits système qui sont remis dans leur état initial ou normal par le système ne doivent pas l'être par programme ou par le terminal.

Description des bits système %S0 à %S7

Description détaillée Description des bits système %S0 à %S7

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S0	Démarrage à froid	Normalement à l'état 0, est mis à l'état 1 par : ● reprise secteur avec perte des données (défaut batterie), ● programme utilisateur, ● terminal, ● changement de cartouche, ● appui sur le bouton de RESET. Ce bit est mis à 1 durant le premier cycle complet. Il est remis à 0 avant le cycle suivant. (Fonctionnement ("Modes de Marche" - Manuel de référence Tome 1))	0	OUI	OUI
%S1	Reprise à chaud	Normalement à l'état 0, est mis à l'état 1 par : ● reprise secteur avec sauvegarde des données, ● programme utilisateur, ● terminal. Il est remis à 0 par le système à la fin du premier cycle complet et avant la mise à jour des sorties. (Fonctionnement ("Modes de Marche" - Manuel de référence Tome 1))	0	OUI	OUI
%S4	Base de temps 10 ms	Bit dont le changement d'état est cadencé par une horloge interne. Il est asynchrone par rapport au cycle de l'automate. Diagramme :  	-	OUI	OUI
%S5	Base de temps 100 ms	Idem %S4	-	OUI	OUI
%S6	Base de temps 1 s	Idem %S4	-	OUI	OUI
%S7	Base de temps 1 mn	Idem %S4	-	OUI	OUI

Description des bits système %S8 à %S16

Description détaillée Description des bits système %S8 à %S16

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S8	Test du câblage	Normalement à l'état 1, ce bit est utilisé pour le test du câblage, câblage lorsque l'automate TSX 37 est dans l'état "non configuré" ● état 1 : les sorties sont forcées à 0, ● état 0 : les sorties peuvent être modifiées par un terminal de réglage.	1	OUI	NON
%S9	Mise à 0 des sorties	Normalement à l'état 0, peut être mis à l'état 1 par programme des sorties ou par le terminal : ● état 1 : mise à 0 de toutes les sorties TOR et analogiques, quelque soit le mode de repli configuré pour chaque module, ● état 0 : les sorties sont mises à jour normalement.,	0	OUI	NON
%S9	Passage en repli des sorties sur tous les bus	Normalement à l'état 0, peut être mis à l'état 1 par programme ou par le terminal : ● état 1 : passage en repli (0 ou 1) selon le choix fait en configuration de toutes les sorties TOR, Analogiques ..., ● état 0 : les sorties sont mises à jour normalement.,	0	NON	OUI
%S10	Défaut E/S	Normalement à l'état 1. Est mis à l'état 0 quand un défaut d'E/S d'un module en rack ou d'un module déporté (FIPIO) (configuration non conforme, défaut d'échange, défaut matériel) est détecté. Le bit %S10 est remis à 1 dès la disparition du défaut.	1	OUI	OUI
%S11	Débordement du chien de garde	Normalement à l'état 0, est mis à l'état 1 par le système dès que le temps d'exécution d'une tâche devient supérieur au temps d'exécution maximum (chien de garde) déclaré en configuration. Le débordement du chien de garde provoque le passage en STOP de l'automate et l'application s'arrête en erreur (voyant ERR clignotant).	0	OUI	OUI
%S13	Premier cycle après mise en RUN	Normalement à l'état 0, est mis à l'état 1 par le système durant le premier cycle après la mise en RUN automate.	-	OUI	OUI

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S15	Défaut chaîne de caractères	Normalement à l'état 0, mis à l'état 1 quand la zone de destination d'un transfert de chaîne de caractères n'a pas la taille suffisante pour accueillir cette chaîne de caractères, ce bit doit être remis à 0 par l'utilisateur. Chaque tâche gère son propre bit %S15.	0	OUI	OUI
%S16	Défaut d'E/S tâche	Normalement à l'état 1, est mis à l'état 0 par le système sur défaut d'un module d'E/S en rack ou déporté sur FIPIO configuré dans la tâche, ce bit doit être remis à 1 par l'utilisateur. Chaque tâche gère son propre bit %S16.	1	OUI	OUI

Description des bits système %S17 à %S20

Description détaillée Description des bits système %S17 à %S20

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S17	Bit sorti sur décalage ou report arithmétique	Normalement à l'état 0, est mis à l'état 1 par le système : • lors d'une opération de décalage contient l'état du dernier bit, • dépassement en arithmétique non signé (dates). Ce bit doit être remis à 0 par l'utilisateur.	0	OUI	OUI
%S18	Débordement ou erreur arthmétique	Normalement à l'état 0. Est mis à l'état 1 en cas de débordement de capacité lors d'une opération sur 16 bits soit : • résultat supérieur à + 32767 ou inférieur à - 32768, en simple longueur, • résultat supérieur à + 2.147.483.647 ou inférieur à - 2.147.483.648, en double longueur, • résultat supérieur à +3.402824E+38 ou inférieur à -3.402824E+38, en flottant (version logicielle > 1.0), • débordement de capacité en DCB, • division par 0, • racine d'un nombre négatif, • forçage à un pas inexistant sur un programmateur cyclique, • empilage d'un registre plein, dépilage d'un registre vide. Doit être testé, par le programme utilisateur, après chaque opération où il y a risque de débordement puis remis à 0 par l'utilisateur en cas de débordement. Chaque tâche gère son propre bit %S18.	0	OUI	OUI
%S19	Débordement période de tâche (scrutation périodique)	Normalement à l'état 0, ce bit est mis à l'état 1 par le système en cas de dépassement de la période d'exécution (temps d'exécution de tâche de la tâche supérieur à la période définie par l'utilisateur en configuration ou programmé dans le mot %SW associé à la tâche). Ce bit est remis à l'état 0 par l'utilisateur. Chaque tâche gère son propre bit %S19.	0	OUI	OUI

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S20	Débordement d'index	Normalement à l'état 0, est mis à l'état 1 lorsque l'adresse de l'objet indexé devient inférieure à 0 ou dépasser le nombre d'objets déclaré en configuration. Doit être testé, par le programme utilisateur, après chaque opération où il y a risque de débordement, puis remis à 0 en cas de débordement. Chaque tâche gère son propre bit %S20.	0	OUI	OUI

Description des bits système %S21 à %S26

Description détaillée Bits système %S21 à %S26 associés au Grafcet

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S21	Initialisation	Ce bit est géré par l'utilisateur pour initialiser le Grafcet (mise à 1 de préférence dans le traitement préliminaire). Il est repositionné à 0 par le système après initialisation du Grafcet (en fin de traitement préliminaire, lors de l'évaluation du nouvel état du Grafcet). L'initialisation du Grafcet consiste en la désactivation de toutes les étapes actives et en l'activation des étapes initiales. Sur un démarrage à froid, ce bit est positionné à 1 par le système pendant le traitement préliminaire.	0	OUI	OUI
%S22	Remise à zéro du Grafcet	Normalement à l'état 0, ce bit ne peut être mis à l'état 1 par programme que dans le traitement préliminaire. A l'état 1, il provoque la désactivation de toutes les étapes du Grafcet. Il est remis à 0 par le système après prise en compte à la fin du traitement préliminaire.	0	OUI	OUI
%S23	Figeage du Grafcet	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. Ce bit est géré par le programme utilisateur, il est positionné à 1 ou à 0 uniquement dans le traitement préliminaire.	0	OUI	OUI
%S24	Remise à zéro des macro-étapes	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. Il est remis à 0 par le système après prise en compte à la fin du traitement préliminaire.	0	NON	OUI
%S26	Débordement des tables (étapes/ transitions)	Normalement à l'état 0, ce bit est mis à l'état 1 par le système des tables lorsque les possibilités d'activation (étapes ou transitions) sont dépassées ou lors de l'exécution d'un graphe incorrect (renvoi de destination sur une étape qui n'appartient pas au graphe). Un débordement provoque le passage en STOP de l'automate. Ce bit est remis à l'état 0 sur initialisation du terminal.	0	OUI	OUI

Description des bits système %S30 à %S59

Description détaillée Description des bits système %S30 à %S59

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S30	Activation/ désactivation tâche maître	Normalement à l'état 1, la mise à l'état 0 par l'utilisateur provoque la désactivation de la tâche maître.	1	OUI	OUI
%S31	Activation tâche rapide	Normalement à l'état 1, la mise à l'état 0 par l'utilisateur provoque la désactivation de la tâche rapide.	1	OUI	OUI
%S38	Validation/ inhibition des évènement	Normalement à l'état 1, la mise à l'état 0 par l'utilisateur provoque inhibition l'inhibition des événements.	1	OUI	OUI
%S39	Saturation dans le traitement des évènements	Ce bit est mis à 1 par le système pour indiquer qu'un ou plusieurs événements ne peuvent pas être traités par suite de saturation des files d'attente. Ce bit est remis à l'état 0 par l'utilisateur.	0	OUI	OUI
%S40 à **%S47**	Défaut E/S (racks) statiques disjonctées	Les bits %S40 à %S47 sont affectés respectivement aux racks 0 à 7. Normalement à l'état 1, chacun de ces bits est mis à 0 sur défaut d'entrées/sorties du rack correspondant. Le bit est remis à l'état 1 à la disparition du défaut.	1	NON	OUI
%S49	Réarmement des sorties	Normalement à l'état 0, ce bit peut être mis à 1 par l'utilisateur pour une demande de réarmement toutes les 10s à partir de l'apparition du défaut des sorties statiques déclenchées sur sur-intensité ou court-circuit.	0	OUI	NON
%S50	Mise à jour de la date et heure par mots %SW50à53	Normalement à l'état 0, ce bit peut être mis à 1 ou à 0 par programme ou par le terminal. ● à l'état 0 : accès à la date et à l'heure par lecture des mots par mots système %SW50 à 53, ● à l'état 1 : mise à jour de la date et l'heure par écriture des mots système %SW50 à 53.	0	OUI	OUI
%S51	Perte de l'heure de l'horodateur	Ce bit géré par le système signale à l'état 1, que l'horodateur est absent ou que ses mots système ne sont pas significatifs. Dans ce cas une mise à l'heure de l'horodateur doit être effectuée.	0	OUI	OUI
%S59	Mise à jour de la date et heure par mot %SW59	Normalement à l'état 0, ce bit peut être mis à 1 ou à 0 par de la date programme ou par le terminal. ● à l'état 0 : le système ne gère pas le mot système %SW59, ● à l'état 1 : le système gère les fronts sur le mot %SW59 pour réglage de la date et l'heure courante (par incrément).	0	OUI	OUI

Description des bits système %S60 à %S69

Description détaillée Description des bits système %S60 à %S69

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S60	Commande de commutation volontaire	bit système qui commande la commutation volontaire dans le cas de mise en oeuvre d'une architecture redondante (voir utilisation dans le manuel "Warm Standby Premium"). Ce bit peut être remis à 0 par l'utilisateur ou par applicatif.	0	NON	OUI
%S66	Gestion du voyant batterie	Normalement à l'état 0, ce bit peut être mis à 1 ou à 0 par programme ou par le terminal. Il permet d'allumer ou non le voyant batterie, en cas d'absence ou de défaut de la pile de sauvegarde : • état 0 : le voyant batterie s'allume quand la pile de sauvegarde est absente ou en défaut, • état 1 : le voyant batterie est toujours éteint. Lors d'un démarrage à froid, %S66 est remis à 0 par le système.	0	OUI	NON
%S67	Etat pile cartouche	Ce bit permet de surveiller l'état de la pile principale quand la carte mémoire est dans l'emplacement PCMCIA du haut: • état 1 : pile principale de tension faible (application toujours préservée mais il faut remplacer la pile selon procédure dite de maintenance prédictive ("Changement des piles sur carte mémoire PCMCIA" - Premium et Atrium sous Unity Pro, Manuel de mise en oeuvre)), • état 0 : pile principale de tension suffisante (application toujours préservée). Le bit %S67 est géré : • sur les cartes mémoires RAM de PV06 (product version inscrite sur l'étiquette de la carte) de petite et moyenne capacité, c'est à dire offrant sous PL7 une taille mémoire ≤ 768K : TSX MRP P 128K, TSX MRP P 224K, TSX MCP C 224K, TSX MRP P 384K, TSX MRP C 448K, TSX MCP C 512K, TSX MRP C 768K,	-	OUI	OUI

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S68	Etat pile processeur	Ce bit permet de contrôler l'état de fonctionnement de la pile de sauvegarde des données et du programme en mémoire RAM ● état 0 : pile présente et en service ● état 1 : pile absente ou hors service	-	OUI	OUI
%S69	Visualisation de données utilisateur sur afficheurs automate	Normalement à l'état 0, ce bit peut être mis à 1 ou à 0 par programme ou par le terminal. ● état 0 : mode "Word" sur les afficheurs désactivé, ● état 1 : mode "Word" sur les afficheurs activé.	0	OUI	NON

Description des bits système %S70 à %S92

Description détaillée Description des bits système %S70 à%S92

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S70	Rafraîchissement des données sur bus AS-i ou liaison TSX Nano	Ce bit est mis à 1 par le système à chaque fin de cycle de la liaisonTSX Nano ou de scrutation bus AS-i. Lors d'une mise sous tension, il indique que toutes les données ont été rafraîchies au moins une fois et qu'elles sont donc significatives. Ce bit est remis à 0 par l'utilisateur.	0	OUI	OUI
%S73	Passage en mode protégé sur bus AS-i	Normalement à l'état 0, ce bit est mis à 1 par l'utilisateur pour passer en mode protégé sur bus AS-i. Au préalable, le bit %S74 devra être à l'état 1. Ce bit ne sera utilisé qu'en test de câblage, sans application dans l'automate.	0	OUI	NON
%S74	Sauvegarde configuration présente sur bus AS-i	Normalement à l'état 0, ce bit est mis à 1 par l'utilisateur pour provoquer la sauvegarde de la configuration présente sur le bus AS-i. Ce bit ne sera utilisé qu'en test de câblage, sans application dans l'automate.	0	OUI	NON
%S75	Test de la pile de la carte mémoire Data Archiving	Ce bit permet de surveiller l'état de la pile principale quand la carte mémoire est dans l'emplacement PCMCIA du bas (CPU TSX P57 4•• et TSX P57 5••) : • état 1 : pile principale de tension faible (application toujours préservée mais il faut remplacer la pile selon procédure dite de maintenance prédictive ("Changement des piles sur carte mémoire PCMCIA" - Premium et Atrium sous Unity Pro, Manuel de mise en oeuvre)), • état 0 : pile principale de tension suffisante (application toujours préservée). Le bit %S75 est géré sur les cartes mémoires RAM de PV06 (product version inscrite sur l'étiquette de la carte) de petite et moyenne capacité, c'est à dire offrant sous PL7 une taille mémoire ≤ 768K : TSX MRP P 128K, TSX MRP P 224K, TSX MCP C 224K, TSX MRP P 384K, TSX MRP C 448K, TSX MCP C 512K, TSX MRP C 768K.	1	NON	OUI
%S80	RAZ compteur de messages	Normalement à l'état 0, ce bit peut être mis à 1 par l'utilisateur pour remettre à zéro les compteurs de messages %SW80 à %SW86.	0	OUI	OUI

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S90	Rafraîchissement des mots communs	Rafraîchi toutes les secondes. Ce bit peut être mis à 0 par programme ou par terminal.	0	OUI	OUI
%S92	Passage en mode mesure de fonction de communication	Normalement à l'état 0, ce bit peut être mis à 1 par l'utilisateur pour positionner les fonctions de communication en mode mesure de performance. Le paramètre de Time-out des fonctions de communication affiche alors le temps d'échange aller-retour en dizaine de ms (si ce temps <10s, sinon non significatif).	0	NON	OUI

Description des bits système %S94 à %S99

Description détaillée Description des bits système %S94 à%S99

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S94	Sauvegarde des réglages DFB	Normalement à l'état 0, ce bit peut être mis à 1 par l'utilisateur pour sauvegarder les valeurs de réglage des blocs fonction utilisateur.	0	NON	OUI
%S95	Restitution des réglages DFB	Normalement à l'état 0, ce bit peut être mis à 1 par l'utilisateur pour restituer les valeurs de réglage des blocs fonction utilisateur.	0	NON	OUI
%S96	Validité de la sauvegarde du programme application	• A l'état 0 : sauvegarde du programme application invalide, • A l'état 1 : sauvegarde du programme application valide. Ce bit peut être lu à tout moment (par programme ou en réglage) et notamment après un démarrage à froid ou une reprise à chaud. Il est significatif vis à vis d'un Backup application réalisé à l'aide de PL7 dans la Flash EPROM interne.	-	OUI	NON
%S97	Validité de la sauvegarde de %MW	• A l'état 0 : sauvegarde des %MW invalide, • A l'état 1 : sauvegarde des %MW valide, Ce bit peut être lu à tout moment (par programme ou en réglage) et notamment après un démarrage à froid ou une reprise à chaud.	-	OUI	NON
%S98	Déport du bouton poussoir du coupleur TSX SAZ 10	Normalement à l'état 0, ce bit est géré par l'utilisateur : • A l'état 0 : bouton poussoir du coupleur TSX SAZ 10 actif, • A l'état 1 : bouton poussoir, du coupleur TSX SAZ 10 remplacé par une entrée TOR (Mot %SW98 (Voir *Description des mots système %SW98 à %SW109, p. 329*)).	0	OUI	NON
%S99	Déport du bouton poussoir du bloc de visualisation	Normalement à l'état 0, ce bit est géré par l'utilisateur : • A l'état 0 : bouton poussoir du bloc de visualisation centralisée actif, • A l'état 1 : bouton poussoir du bloc de visualisation centralisée, remplacé par une entrée TOR (Mot %SW99 (Voir *Description des mots système %SW98 à %SW109, p. 329*)).	0	OUI	NON

Description des bits système %S100 à %S119

Description détaillée Description des bits système %S100 à%S119

Bit	Fonction	Description	Etat initial	TSX37	TSX57
%S100	Protocole sur prise terminal	Positionné à 0 ou 1 par le système suivant l'état du shunt INL/DPT sur la prise console. ● si le shunt est absent (%S100=0) alors le protocole UNI-TELWAY maître est utilisé, ● si le shunt est présent (%S100=1) alors le protocole utilisé est celui indiqué par la configuration de l'application.	-	OUI	OUI
%S101	Buffer de diagnostic configuré	Ce bit est mis à 1 par le système lorsque l'option diagnostic est configurée, un buffer de diagnostic destiné au stockage des erreurs, issues des DFB de diagnostic, est alors réservé.	-	OUI	OUI
%S102	Buffer de diagnostic plein	Ce bit est mis à 1 par le système lorsque le buffer recevant les erreurs des blocs fonction de diagnostic est rempli.	-	OUI	OUI
%S118	Défaut général d'E/S FIPIO	Normalement à 1 ce bit est mis à 0 par le système lors de l'apparition d'un défaut sur un équipement connecté sur le bus FIPIO. Lorsque le défaut disparaît ce bit est remis à 1 par le système.	1	OUI	OUI
%S119	Défaut général d'E/S en rack	Normalement à 1 ce bit est mis à 0 par le système lors de l'apparition d'un défaut sur un module d'E/S implanté dans un des racks. Lorsque le défaut disparaît ce bit est remis à 1 par le système.	1	OUI	OUI

3.2 Mots système

Présentation

Objet de ce sous-chapitre

Ce sous-chapitre décrit les mots système du langage PL7

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Description des mots système %SW0 à %SW11

Description détaillée Description des mots système %SW0 à %SW11

Mots	Fonction	Description	Gestion
%SW0	Période de scrutation de la tâche maître	Permet de modifier la période de la tâche maître définie en configuration, par le programme utilisateur ou par le terminal. La période est exprimée en ms (1..255ms). %SW0=0 en fonctionnement cyclique. Sur reprise à froid : prend la valeur définie par configuration.	Utilisateur
%SW1	Période de scrutation de la tâche rapide	Permet de modifier la période de la tâche rapide définie en configuration, par le programme utilisateur ou par le terminal. La période est exprimée en ms (1..255ms). Sur reprise à froid : prend la valeur définie par configuration.	Utilisateur
%SW8	Contrôle d'acquisition des entrées des tâches	Normalement à l'état 0, ce bit peut être mis à 1 ou à 0 par programme ou par le terminal. Permet d'inhiber la phase d'acquisition des entrées de chaque tâche. ● %SW8:X0 =1 affecté à la tâche MAST : les sorties relatives à cette tâche ne sont plus pilotées. ● %SW8:X1 = 1 affecté à la tâche FAST : les sorties relatives à cette tâche ne sont plus pilotées.	Utilisateur
%SW9	Contrôle de la mise à jour des sorties des tâches	Normalement à l'état 0, ce bit peut être mis à 1 ou à 0 par programme ou par le terminal. Permet d'inhiber la phase de mise à jour des sorties de chaque tâche. ● %SW9:X0 = 1 affecté à la tâche MAST : les sorties relatives à cette tâche ne sont plus pilotées. ● %SW9:X1 = 1 affecté à la tâche FAST : les sorties relatives à cette tâche ne sont plus pilotées.	Utilisateur
%SW10	Premier cycle après démarrage à froid	La valeur 0 du bit de la tâche en cours signifie qu'elle exécute son premier cycle après démarrage à froid. ● %SW10:X0 : est affecté à la tâche Maître MAST ● %SW10:X1 : est affecté à la tâche rapide FAST	Système
%SW11	Durée du chien de garde	Permet de lire la durée du chien de garde définie en configuration. Il est exprimé en ms (10...500ms).	Système

ATTENTION

Relatif au mots systèmes %SW8 et %SW9 :

Attention : les sorties des modules connectés sur le bus X passent automatiquement en mode repli, les sorties des équipements connectés sur le bus FIPIO restent maintenues dans l'état précédent la mise à 1 du bit.

Le non-respect de cette directive peut entraîner des lésions corporelles et/ou des dommages matériels.

Description des mots système %SW12 à %SW18

Description détaillée Description des mots système %SW12 à %SW18

Mots	Fonction	Description	Gestion
%SW12	Adresse UNI-TELWAY prise terminal	Adresse UNI_TELWAY de la prise terminal (en mode esclave) définie en configuration et chargée dans ce mot lors d'un démarrage à froid.	Système
%SW13	Adresse principale de la station	Indique pour le réseau principal : ● le numéro de station (octet de poids faible) de 0 à 127 ● le numéro de réseau (octet de poids fort) de 0 à 63 (valeur des micro-interrupteur sur la carte PCMCIA)	Système
%SW17	Statut de défaut sur opération flottante	Sur détection d'un défaut dans une opération en arithmétique flottante, le bit %S18 est mis à 1 et le statuts de défaut %SW17 est mis à jour selon le codage suivant : ● %SW17:X0 = Opération invalide/le résultat n'est pas un nombre ● %SW17:X1 = Opérande non normalisé/le résultat est bon ● %SW17:X2 = Division par 0/le résultat est l'infini ● %SW17:X3 =Overflow/le résultat est l'infini ● %SW17:X4 = Underflow/le résultat est 0 ● %SW17:X5 = Imprécision sur le résultat Ce mot est remis à 0 par le système sur démarrage à froid et par le programme pour réutilisation.	Système Utilisateur
%SW18	Compteur de temps absolu	Ce double mot permet d'effectuer des calculs de durée. Il est incrémenté tous les 1/10ème de secondes par le système (même automate en STOP). Il peut être lu et écrit par programme utilisateur ou par terminal.	Système Utilisateur

Description des mots système %SW20 à %SW25

Description détaillée

Description des mots système %SW20 à %SW25 (associés au Grafcet)

Mots	Fonction	Description	Gestion
%SW20	Niveau d'activité du Grafcet	Ce mot contient le nombre d'étapes actives, à activer et à désactiver pour le cycle courant. Il est remis à jour par le système lors de chaque évolution du graphe.	Système
%SW21	Table de validité des transitions Grafcet	Ce mot contient le nombre de transitions valides, à valider et à invalider pour le cycle courant. Il est remis à jour par le système lors de chaque évolution du graphe.	Système
%SW22 à %SW25	Table de remise à 0 macro-étape	A chaque bit de cette table correspond une macro-étape avec %SW22:X0 pour XM0 ....%SW25:X16 pour XM63. Les macro-étapes dont le bit associé dans cette table est à 0, seront remis à 0 sur mise à 1 du bit %S24.	Utilisateur

Description des mots système %SW30 à %SW35

Description détaillée Description des mots système %SW30 à %SW35

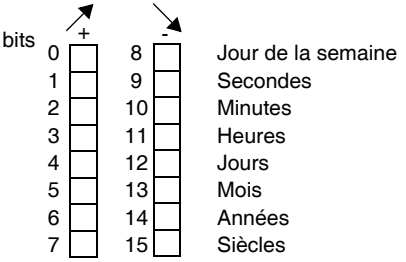
Mots	Fonction	Description	Gestion
%SW30	Temps d'exécution tâche maître	Indique le temps d'exécution du dernier cycle de la tâche maître (en ms).	Système
%SW31	Temps d'exécution maxi tâche maître	Indique le temps d'exécution le plus long de la tâche maître, depuis le dernier démarrage à froid (en ms).	Système
%SW32	Temps d'exécution mini tâche maître	Indique le temps d'exécution le plus court de la tâche maître , depuis le dernier démarrage à froid (en ms).	Système
%SW33	Temps d'exécution tâche rapide	Indique le temps d'exécution du dernier cycle d'exécution de la tâche rapide (en ms).	Système
%SW34	Temps d'exécution maxi tâche rapide	Indique le temps d'exécution le plus long de la tâche rapide, depuis le dernier démarrage à froid (en ms).	Système
%SW35	Temps d'exécution mini tâche rapide	Indique le temps d'exécution le plus court de la tâche rapide, depuis le dernier démarrage à froid (en ms).	Système

Note : Précision sur le temps d'exécution : c'est le temps écoulé entre le début (acquisition des entrées) et la fin (mise à jour des sorties) d'un cycle de scrutation. Ce temps inclut le traitement des tâches événementielles et de la tâche rapide ainsi que le traitement des requêtes console.

Description des mots système %SW48 à %SW59

Description détaillée Description des mots système %SW48 à %SW59

Mots	Fonction	Description	Gestion
%SW48	Nombre d'événements	Indique le nombre d'événements traités,depuis le dernier démarrage à froid (en ms). Ce mot peut être écrit par programme ou par terminal.	Système Utilisateur
%SW49 **%SW50** **%SW51** **%SW52** **%SW53**	Fonction Horodateur (1)	Mots système contenant la date et l'heure courante (en BCD) : ● %SW49 : jour de la semaine (1 pour Lundi à 7 pour Dimanche). ● %SW50 : Secondes (SS00) ● %SW51 : Heures et Minutes (HHMM) ● %SW52 : Mois et Jour (MMJJ) ● %SW53 : Année (AAAA) Ces mots sont gérés par le système lorsque le bit %S50 est à 0. Ces mots peuvent être écrits par programme utilisateur ou par terminal lorsque le bit %S50 est mis à 1 (Voir *Description des bits système %S30 à %S59, p. 304*).	Système Utilisateur
%SW54 **%SW55** **%SW56** **%SW57** **%SW58**	Fonction Horodateur (1)	Mots système contenant la date et l'heure du dernier défaut secteur ou arrêt automate (en BCD) : ● %SW54 : Secondes (00SS) ● %SW55 : Heures et Minutes (HHMM) ● %SW56 : Mois et Jour (MMJJ) ● %SW57 : Année (AAAA) ● %SW58 : l'octet de poids fort contient le jour de la semaine (1 pour Lundi à 7 pour Dimanche).	Système
%SW58	Code du dernier arrêt	L'octet de poids faible contient le code du dernier arrêt : ● 1 = passage de RUN à STOP par le terminal ● 2 = arrêt sur défaut logiciel (débordement de tâche automate) ● 4 = coupure secteur ● 5 = arrêt sur défaut matériel ● 6 = arrêt sur instruction HALT	Système

Mots	Fonction	Description	Gestion
%SW59	Réglage de la date courante	Contient deux séries de 8 bits pour régler la date courante. L'action est toujours réalisée sur front montant du bit. Ce mot est validé par le bit %S59. Illustration :  	Utilisateur

Note : (1) Uniquement sur automate TSX 37-21/22 et TSX 57

Description des mots système %SW60 à %SW62

Description détaillée Description des mots système %SW60 à %SW61 spécifiques au diagnostic du Warm Standby Premium

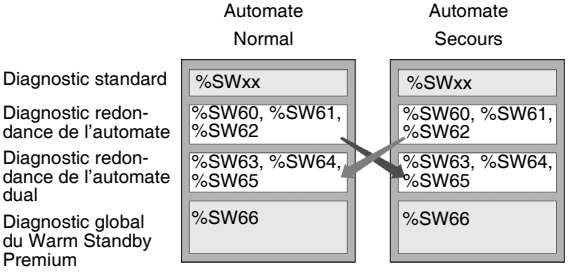
Mots	Fonction	Description	Gestion
%SW60	Diagnostic automate redondant	Diagnostic spécifique à la redondance pour un automate local (Warm Standby Premium) Signification des différents bits du mot %SW60 : ● %SW60:X0=1 indique que l'automate est en état 'Normal' ● %SW60:X1=1 indique que l'automate est en état 'Secours' ● %SW60:X3=0 indique un défaut d'entrées/sorties sur FIPIO dans l'automate Normal ; c'est l'image du bit %S118 ● %SW60:X4=0 indique un défaut d'entrées/sorties en rack;c'est l'image du bit %S119 ● %SW60:X5=1 indique un défaut détecté par autotests sur au moins l'un des TSX ETY 210 ● %SW60:X7=1 indique un défaut grave du réseau FIPIO, par exemple un court-circuit ou un bornier débranché ● %SW60:X8=1 indique un défaut sur le module TSX ETY 110 utilisé pour la liaison inter-automates ● %SW60:X9=1 indique un défaut de la communication inter-automates, il y a impossibilité de récupérer le diagnostic de l'automate dual ● %SW60:X10 est un bit réservé ● %SW60:X11=1 indique que l'automate Secours est inapte à devenir automate Normal, cette information est élaborée uniquement dans l'automate Normal, elle n'est pas significative dans l'automate Secours ● %SW60:X12=0 indique que l'automate est la station A ● %SW60:X12=1 indique que l'automate est la station B ● %SW60:X13=1 indique le mode Run de l'automate ● %SW60:X14=1 indique le mode Stop de l'automate ● %SW60:X15=1 indique le mode Halt de l'automate	Système

Mots	Fonction	Description	Gestion
%SW61	Diagnostic automate redondant	Signification des différents bits du mot %SW61 : ● %SW61:X0=1 indique un problème sur l'échange de la Base de Données par la liaison Ethway inter-automates, cette information est élaborée uniquement pour l'automate Normal en Run ● %SW60:X1=1 indique que l'automate est en état 'Secours' ● %SW61:X1=1 indique un problème sur des communications entre un module TSX ETY 210 client TCP-IP d'un équipement tiers. Cette information est élaborée uniquement pour l'automate Normal en Run. Quand ce bit passe à 1, une commutation est provoquée si l'automate Secours est apte à devenir Normal ● %SW60:X4=0 indique un défaut d'entrées/sorties en rack;c'est l'image du bit %S119 ● %SW60:X5=1 indique un défaut détecté par autotests sur au moins l'un des TSX ETY 210 ● %SW61:X2 est un bit réservé ● %SW61:X3 est un bit réservé ● %SW61:X4=1 indique un premier échange correct de la Base de Données ● %SW61:X5=1 indique que le processeur a été mis en Stop par la fonction redondance, le diagnostic est donné dans le mot %MWp.14.2 ● %SW61:X6=1 est un bit réservé ● %SW61:X7=0 indique un problème de configuration ou de fonctionnement de la fonction redondance, le diagnostic est donné dans le mot %MWp.14.2 ● %SW61:X7=1 indique que la fonction redondance est correctement configurée ● %SW61:X8 à %SW61:X15 sont des bits réservés p : designe l'emplacement du processeur	Système

Mots	Fonction	Description	Gestion
%SW62	Visualisation de la fonction arbitre de bus et producteur / consommateur du bus FIPIO.	L'octet de poids faible indique l'état de la fonction producteur / consommateur, L'octet de poids fort indique l'état de la fonction arbitre de bus (BA) Valeur de l'octet : ● 16#00 : la fonction n'existe pas (pas d'application FIPIO), ● 16#07 : la fonction est en cours de STOP BA (l'ordre de STOP BA est envoyé, la commande n'est pas terminée). ● 16#0F : la fonction est en cours de RUN BA (l'ordre de RUN BA est envoyé, la commande n'est pas terminée). ● 16#70 : la fonction est initialisée mais pas opérationnelle (en STOP BA), ● 16#F0 : la fonction est en cours d'exécution normale (en RUN BA). BA),	Système

Description des mots système %SW63 à %SW65

Description détaillée Description des mots système %SW63 à %SW65 spécifiques au diagnostic du Warm Standby Premium

Mots	Fonction	Description	Gestion
%SW63 à %SW65	Echange des mots de diagnostic entre automates	Le diagnostic redondance de l'automate dual est disponible dans les mots %SW63 à %SW65. Les mots %SW63, %SW64 et %SW65 de l'automate Normal contiennent respectivement les mots %SW60, %SW61 et %SW62 de l'automate Secours. De même, les mots %SW63, %SW64 et %SW65 de l'automate Secours contiennent respectivement les mots %SW60, %SW61 et %SW62 de l'automate Normal. Illustration  Cet échange de mots est réalisé par la liaison Ethway inter-automates (module TSX ETY 110)	Système

Description des mots système %SW66 à %SW69

Description détaillée Description du mot système %SW66 spécifique au diagnostic du Warm Standby Premium

Mots	Fonction	Description	Gestion
%SW66	Diagnostic global de l'architecture Warm Standby Premium	Dans chacun des automates, un diagnostic global de l'architecture Warm Standby Premium est élaboré à partir des diagnostics redondance des deux automates. Ce diagnostic global est stocké en %SW66 dont les bits ont la signification ci-dessous :	Système
	• %SW66:X0=0 indique un fonctionnement Dégradé du Warm Standby Premium • %SW66:X0=1 indique un fonctionnement Nominal du Warm Standby Premium • %SW66:X1=1 indique que l'automate A est l'automate Normal • %SW66:X2=1 indique que l'automate B est l'automate Normal • %SW66:X3=1 indique une défaillance de communication inter-automates **Informations pour l'automate A** • %SW66:X4=1 indique une défaillance grave du réseau FIPIO sur l'automate A • %SW66:X5=1 indique que l'automate A est en STOP • %SW66:X6=1 indique que l'automate A est en Halt • %SW66:X7=1 indique une défaillance de communication Ethernet TCP-IP de l'automate A (module TSX ETY 210 ou fonction client) • %SW66:X8=1 indique une défaillance sur au moins l'un des modules en rack de l'automate A • %SW66:X9=1 indique une défaillance sur au moins l'un des équipements FIPIO de l'automate A **Informations pour l'automate B** • %SW66:X10=1 indique une défaillance grave du réseau FIPIO sur l'automate B • %SW66:X11=1 indique que l'automate B est en STOP • %SW66:X12=1 indique que l'automate B est en Halt • %SW66:X13=1 indique une défaillance de communication Ethernet TCP-IP de l'automate B (module TSX ETY 210 ou fonction client) • %SW66:X14=1 indique une défaillance sur au moins l'un des modules en rack de l'automate B • %SW66:X15=1 indique une défaillance sur au moins l'un des équipements FIPIO de l'automate B	

Note : Les informations %SW66:X4 à %SW66:X15 ne sont pas significatives si une défaillance de communication inter-automates est présente (%SW66:X3=1)

Description du mot système %SW67 à %SW69 spécifiques au diagnostic du Warm Standby Premium

Mots	Fonction	Description	Gestion
%SW67	Adresse réseau et l'adresse station de l'automate dual	Ce mot contient l'adresse réseau et l'adresse station de l'automate dual, permettant d'établir la communication inter-automates. Ce mot doit être visualisé en hexadécimal pour être interprété de la maniére suivante : Poids forts / Poids faibles adresse réseau | adresse station	Système
%SW68 **%SW69**	Base de temps utilisée par les EF Tempo.	Ces mots contiennent une base de temps utilisée par les EF Tempo. Elle est transférée de l'automate Normal vers l'automate Secours pour mise à jour et synchronisation.	Système

Description des mots système %SW80 à %SW89

Description détaillée Description des mots système %SW80 à %SW89

Mots	Fonction	Description	Gestion
%SW80 **%SW81** **%SW82** **%SW83** **%SW84** **%SW85** **%SW86**	Gestion des messages et télégrammes	• %SW80 : Nb de messages émis par le système vers la prise terminal. • %SW81 : Nb de messages reçus par le système depuis la prise terminal. • %SW82 : Nb de messages émis par le système vers le coupleur PCMCIA. • %SW83 : Nb de messages reçus par le système depuis le coupleur PCMCIA. • %SW84 : Nb de télégrammes émis par le système. • %SW85 : Nb de télégrammes reçus par le système. • %SW86 : Nb de messages refusés par le système.	Système Utilisateur
%SW87 **%SW88** **%SW89**	Gestion des flux de communication (1)	• %SW87 : Nombre de requêtes traitées par le serveur synchrone par cycle de la tâche maître (MAST). • %SW88 : Nombre de requêtes traitées par le serveur asynchrone par cycle de la tâche maître (MAST). • %SW89 : Nombre de requêtes traitées par des fonctions serveur (immédiate) par cycle de la tâche maître (MAST).	Système

Note : (1) uniquement sur automate TSX/PCX/PMX 57

Description des mots système %SW96 et %SW97

Description détaillée

Ces mots n'existent que sur TSX 37

Description des mots système %SW96 et %SW97

Mots	Fonction	Description	Gestion

%SW96	Commande et ou diagnostic de la fonction sauvegarde/ restitution du programme application et des %MW	• bit 0 : demande de transfert vers la zone de sauvegarde. Ce bit est actif sur front montant. Il est remis à 0 par le système dès la restitution prise en compte du front montant.
		• bit 1 : quand ce bit a la valeur 1, cela signifie que la fonction de sauvegarde est terminée. Ce bit est remis à 0 dès la prise en compte du front montant sur le bit 0.
		• bit 2 : compte rendu de la sauvegarde :
		  • 0 -> sauvegarde sans erreur,
		  • 1 -> erreur pendant la sauvegarde.
		• bits 3 à 5 : réservés.
		• bit 6 : validité de la sauvegarde du programme application (idem %S96).
		• bits 8 à 15 : cet octet n'est significatif que si le bit de compte-rendu est à 1 (bit 2 = 1, erreur pendant la sauvegarde) :
		  • 1 -> nombre de %MW à sauvegarder supérieur au nombre de %MW configuré
		  • 2 -> nombre de %MW à sauvegarder supérieur à 1000 ou inférieur à 0,
		  • 3 -> nombre de %MW à restituer supérieur au nombre de %MW configuré,
		  • 4 -> taille de l'application en RAM interne supérieure à 15 Kmots (on rappelle que la sauvegarde des %MW est toujours associée à une sauvegarde du programme application dans la Flash EPROM interne),
		  • 5 -> service interdit en RUN,
		  • 6 -> présence d'une cartouche Backup dans l'automate,
		  • 7 -> défaut d'écriture dans la Flash EPROM.
			Système Utilisateur

Mots	Fonction	Description	Gestion
%SW97	Nombre de %MW à sauvegarder	Permet de paramétrer le nombre de %MW à sauvegarder. Quand ce mot est compris entre 1 et 1000, les 1 à 1000 premiers %MW sont transférés dans la Flash EPROM interne. Quand ce mot vaut 0, seul le programme application contenu dans la RAM interne est transféré dans la Flash EPROM interne. **Une éventuelle sauvegarde de %MW est alors effacée.** Lors d'un démarrage à froid, ce mot est initialisé à -1 si la Flash EPROM interne ne contient aucune sauvegarde de %MW. Dans le cas contraire, il est initialisé à la valeur du nombre de mots sauvegardés.	Utilisateur

Description des mots système %SW98 à %SW109

Description détaillée Description des mots système %SW98 à %SW109

Mots	Fonction	Description	Gestion
%SW98	Adresse géographique module/voie de l'entrée TOR (2)	Lorsque le bit %S98 = 1, ce mot indique l'adresse géographique l'entrée TOR (module / voie) de l'entrée TOR, en remplacement du bouton poussoir du coupleur TSX SAZ 10 : Poids fort / Poid faible Numéro de module / Numéro de voie	Utilisateur
%SW99	Adresse de l'entrée TOR (2)	Lorsque le bit %S99 = 1, ce mot indique l'adresse géographique (module / voie) de l'entrée TOR, en remplacement du bouton poussoir du bloc de visualisation centralisée : Poids fort / Poid faible Numéro de module / Numéro de voie	Utilisateur
%SW108	Compteur de voies forcées	Comptabilise les voies forcées à 0 ou à 1 dans l'application. Il voies est remis à jour par forçage ou déforçage des voies.	Système
%SW109	Compteur de voies analogiques forcées	Comptabilise les voies analogiques forcées à 0.	Système

Note : (2) uniquement sur TSX 37

Description du mot système %SW116

Description détaillée Description du mot système %SW116 - FIPIO

Mots	Fonction	Description	Gestion
%SW116	Défaut d'E/S FIPIO dans la tâche	Normalement à 0, chaque bit de ce mot est significatif d'un status d'échange FIPIO **dans la tâche** dans laquelle il est testé. Ce mot est remis à 0 par l'utilisateur. Détail des bits du mot %SW116 : ● x0 = 1 erreur d'échange explicite (la variable n'est pas échangée sur le bus), ● x1 = 1 time-out sur un échange explicite (non réponse au bout du time-out), ● x2 = 1 nombre maximum d'échanges explicites simultanés atteint, ● x3 = 1 une trame n'est pas correcte, ● x4 = 1 la longueur d'une trame reçue supérieure à la longueur déclarée, ● x5 = réservé à 0, ● x6 = 1 une trame est invalide, ou un agent s'initialise, ● x7 = 1 absence d'un équipement configuré, ● x8 = 1 défaut voie (au moins une voie d'un équipement signale un défaut), ● x9 à x14 = réservé à 0, ● x15 = Défaut global (OU des bits 3, 4, 6, 7, 8).	Système Utilisateur

Description des mots système %SW124 à %SW127

Description détaillée Description des mots système %SW124 à %SW127

Mots	Fonction	Description	Gestion
%SW124	Type de défaut CPU	Le système écrit dans ce mot le dernier type de défaut CPU rencontré (ces codes sont inchangés sur reprise à froid) : ● 16#30 : défaut du code système ● 16#60 à 64 : débordement de pile ● 16#90 : défaut du système d'interruption : IT non prévue ● 16#53 : défaut de time-out lors des échanges d'E/S	Système

%SW125	Type de défaut bloquant	Le système écrit dans ce mot le dernier type de défaut bloquant rencontré :
		● 16#DEB0 : débordement du chien de garde
		● 16#2258 : exécution de l'instruction HALT
		● 16#DEF8 : exécution d'une instruction JMP à une étiquette non définie
		● 16#2XXX : exécution d'une instruction CALL à un sous programme non défini
		● 16#0XXX : exécution d'une fonction inconnue
		● 16#DEFE : le programme grafcet comporte des renvois à des étapes non définies.
		● 16#DEFF : flottant non implémenté
		● division par 0 :
		  ● avec des entiers alors (16#DEF0 --> %SW125), (1-->%S18) et (0-->%SW17),
		  ● avec des flottants alors (16#DE87 --> %SW125), (1-->%S18) et (4-->%SW17).
		● 16#DEF1 : erreur de transfert de chaîne de caractères (1-->%S15)
		● débordement de capacité (overflow) :
		  ● avec des entiers alors (16#DEF2 --> %SW125), (1-->%S18) et (0-->%SW17),
		  ● avec des flottants alors (16#DE87 --> %SW125), (1-->%S18) et (8-->%SW17).
		● 16#DEF3 : débordement d'index (1-->%S20)
			Système

%SW126 **%SW127**	Adresse de l'instruction du défaut bloquant	Adresse de l'instruction ayant généré le défaut application bloquant. ● %SW126 contient l'offset de cette adresse ● %SW127 contient la base de cette adresse	Système

Description du mot système %SW128 à %SW143

Description détaillée Description des mots système %SW128 à SW143 - FIPIO

Mots	Fonction	Description	Gestion
%SW128 à %SW143	Point de connexion FIPIO en défaut	Chaque bit de ce groupe de mots est significatif de l'état d'un équipement connecté sur le bus FIPIO. Normalement à 1, la présence à 0 d'un de ces bits indique l'apparition d'un défaut sur ce point de connexion. Pour un point de connexion non configuré, le bit correspondant vaut toujours 1.	Système

Tableau de correspondance entre les bits des mots et l'adresse d'un point de connexion

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
%SW128 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
%SW129 :	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
%SW130 :	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
%SW131 :	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
%SW132 :	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
%SW133 :	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
%SW134 :	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
%SW135 :	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
%SW136 :	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
%SW137 :	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
%SW138 :	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
%SW139 :	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
%SW140 :	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
%SW141 :	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
%SW142 :	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
%SW143 :	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Description des mots système %SW144 à %SW146

Description détaillée Description des mots système %SW144 à %SW146 - FIPIO

Mots	Fonction	Description	Gestion
%SW144	Mode de marche fonction arbitre de bus FIPIO	Ce mot système permet l'arrêt et le démarrage de la fonction arbitre de bus et de la fonction producteur / consommateur. Il permet de modifier le mode de démarrage, automatique et manuel du bus en cas d'arrêt. ● x0 = 1 fonction producteur / consommateur en RUN ● x0 = 0 fonction producteur / consommateur en STOP (aucune variable est échangée sur le bus), ● x1 = 1 l'arbitre de bus est en RUN, ● x1 = 0 l'arbitre de bus est en STOP (aucune scrutation de variables et de messages est faite sur le bus), ● x2 = 1 démarrage automatique en cas d'arrêt automatique du bus ● x2 = 0 démarrage manuel si arrêt automatique du bus ● x3 = réservé à 1 ● x4 à x15 réservés à 0.	Utilisateur
%SW145	Modification des paramètres de l'arbitre de bus FIPIO	Les bits sont mis à 1 par l'utilisateur et remis à 0 par le système lorsque l'initialisation est effectuée. ● x0 = 1 modification de le priorité de l'arbitre de bus : l'octet de poids fort de ce mot système contient la valeur de la priorité de l'arbitre de bus qui sera appliquée sur le bus, ● x1 à x2 sont réservés, ● x3 à x7 réservés à 0, ● x8 à x15 : cet octet contient la valeur qui est appliquée sur le bus, suivant la valeur du bit x0. La modification de ces paramètres peut se faire lorsque l'arbitre de bus est en RUN, mais la prise en compte par l'application nécessite un arrêt puis un redémarrage de celle-ci.	Utilisateur Système

Mots	Fonction	Description	Gestion
%SW146	Visualisation de la fonction arbitre de bus FIPIO	L'octet de poids faible indique l'état de la fonction producteur / consommateur L'octet de poids fort indique l'état de la fonction arbitre de bus. Valeur de l'octet : ● 16#00 : la fonction n'existe pas (pas d'application FIPIO), ● 16#07 : la fonction est en cours de STOP (l'ordre de STOP est envoyé, la commande n'est pas terminée). ● 16#0F : la fonction est en cours de RUN (l'ordre de RUN est envoyé, la commande n'est pas terminée). ● 16#70 : la fonction est initialisée mais pas opérationnelle (en STOP), ● 16#F0 : la fonction est en cours d'exécution normale (en RUN).	Système

ATTENTION

Concerne les mots %SW144 et %SW145

la modification de ces mots système peut entraîner l'arrêt de la station automate.

Le non-respect de cette directive peut entraîner des lésions corporelles et/ou des dommages matériels.

Description des mots système %SW147 à %SW152

Description détaillée Description des mots système %SW147 à %SW152

Mots	Fonction	Description	Gestion
%SW147	Temps de cycle réseau MAST	Une valeur non nulle indique, en ms, la valeur du temps de cycle réseau (TCR-MAST) de la tâche MAST.	Système
%SW148	Valeur du temps de cycle réseau de la tâche FAST.	Une valeur non nulle indique, en ms, la valeur du temps de cycle réseau (TCR-FAST) de la tâche FAST.	Système
%SW149		Réservé à 0.	Système
%SW150	Nombre de trames émises	Ce mot indique le nombre de trames émises par le gestionnaire de la voie FIPIO	Système
%SW151	Nombre de trames reçues	Ce mot indique le nombre de trames reçues par le gestionnaire de la voie FIPIO.	Système
%SW152	Nombre de messages repris	Ce mot indique le nombre de reprises de messages effectuées par le gestionnaire de la voie FIPIO.	Système

Description du mot système %SW153

Description détaillée

Description du mot système %SW153 - FIPIO

Mots	Fonction	Description	Gestion
%SW153	liste des défauts du gestionnaire de la voie FIPIO.	Chaque bit est mis à 1 par le système et remis à 0 par l'utilisateur. Voir liste ci-dessous.	Utilisateur Système

Description des bits

- X0 = défaut d'overrun de la station : correspond à une perte de symbole MAC en réception, liée à une réaction trop lente du récepteur.
- X1 = défaut de refus message : indique un message avec acquittement refusé ou sans acquittement. MAC en réception,
- X3 = défaut d'underrun de la station : correspond à une incapacité de la station à respecter la vitesse d'émission sur le réseau.
- X4 = défaut de couche physique : correspond à une absence prolongée de transmission au niveau couche physique.
- X5 = défaut de non écho : correspond à un défaut pour lequel l'émetteur est en cours d'émission, avec un courant d'émission compris dans la plage de fonctionnement, et simultanément détection d'absence de signal sur la même voie.
- X6 = défaut de bavardage : correspond à un défaut pour lequel l'émetteurdispose du contrôle de la ligne depuis un temps supérieur à la limite maximale de fonctionnement définie. Ce défaut est par exemple provoqué par une détérioration du modulateur ou par une couche liaison de données défectueuse.
- X7 = défaut d'hypocourant : correspond à un défaut pour lequel l'émetteur produit sur la ligne, lorsqu'il est sollicité, un courant inférieur à la limite minimale de fonctionnement définie. Ce défaut est par exemple provoqué par une élévation de l'impédance de ligne (ligne ouverte...).
- X8 = défaut de trame trouée : indique la réception d'un silence dans le corps d'une trame après l'identification d'un délimiteur de début de trame et avant l'identification d'un délimiteur de fin de trame. L'apparition d'un silence dans des conditions normales de fonctionnement a lieu après l'identification d'un délimiteur de fin de trame.
- X9 = défaut de CRC trame en réception : indique une différence de valeur entre le CRC calculé sur la trame normalement reçue et le CRC contenu dans cette trame.
- X10 = défaut de codage trame en réception : indique la réception de certains symboles, appartenant exclusivement aux séquences de délimitation du début et de la fin de trame, dans le corps d'une trame.
- X11 = défaut de longueur de la trame reçue : le nombre d'octets reçus pour le corps d'une trame est supérieur à 256 octets.
- X12 = réception d'une trame de type inconnu : dans le corps d'une trame, le premier octet identifie le type de trame liaison. Un certain nombre de types de trames sont définis dans le protocole liaison de la norme WORLDFIP. L'existence de tout autre code dans une trame correspond à un défaut de type trame inconnue.
- X13 = réception d'une trame tronquée : un fragment de trame se caractérise par la reconnaissance d'une séquence de symboles du délimiteur de fin de trame alors que la station destinataire s'attend à recevoir un délimiteur de début de trame.
- X14 = inutilisé, valeur non significative
- X15 = inutilisé, valeur non significative

Description du mot système %SW154

Description détaillée

Description du mot système %SW154 - FIPIO

Mots	Fonction	Description	Gestion
%SW154	liste des défauts du gestionnaire de la voie FIPIO.	Chaque bit est mis à 1 par le système et remis à 0 par l'utilisateur. Voir liste ci-dessous.	Utilisateur Système

Description des bits

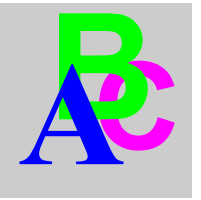
- X0 = time out séquence apériodique : indique un dépassement de la fenêtre de messages ou de variables apériodiques dans un cycle élémentaire du macro-cycle.
- X1 = refus de demande de messagerie : indique une saturation de la file d'attente des messages, l'arbitre de bus n'est momentanément plus en mesure de mémoriser puis de satisfaire une demande.
- X2 = refus de commande d'update urgente : indique une saturation de la file d'attente des demandes d'échanges de variables apériodiques urgentes, l'arbitre de bus n'est momentanément plus en mesure de mémoriser ni de satisfaire la demande.
- X3 = refus de commande d'update non urgente : indique une saturation de la file d'attente des demandes d'échanges de variables apério diques non urgentes, l'arbitre de bus n'est momentanément plus en mesure de mémoriser ni de satisfaire la demande.
- X4 = défaut de silence : l'arbitre de bus n'a détecté aucune activité sur le bus pendant une durée supérieure à un temps normalisé WorldFip.
- X5 = collision sur le réseau à l'émission d'identifieur : indique une activité sur le réseau pendant des périodes théoriques de silence. Entre une émission et l'attente d'une réponse par l'arbitre de bus, rien ne doit circuler sur le bus. Si l'arbitre de bus détecte une activité il génère un défaut de collision (Par exemple lorsque plusieurs arbitres sont actifs simultanément sur le bus).
- X6 = défaut d'overrun de l'arbitre de bus : indique un conflit d'accès à la mémoire de la station arbitre de bus.
- X7 = inutilisé, valeur non significative
- X8 à X15 = réservé à 0.

Description des mots système %SW155 à %SW162

Description détaillée Description des mots système %SW155 à %SW162

Mots	Fonction	Description	Gestion
%SW155	Nombre d'échanges explicites	Nombre d'échanges explicites en cours de traitement	Système
%SW160		Résultat de la dernière registration (fonction diagnostic).	Système
%SW161		Résultat de la dernière déregistration (fonction diagnostic).	Système
%SW162		Nombre d'erreurs en cours dans le buffer de diagnostic.	Système

Index



Symbols

A

B

C

D

E

F

G

H

I

L

M

N

X