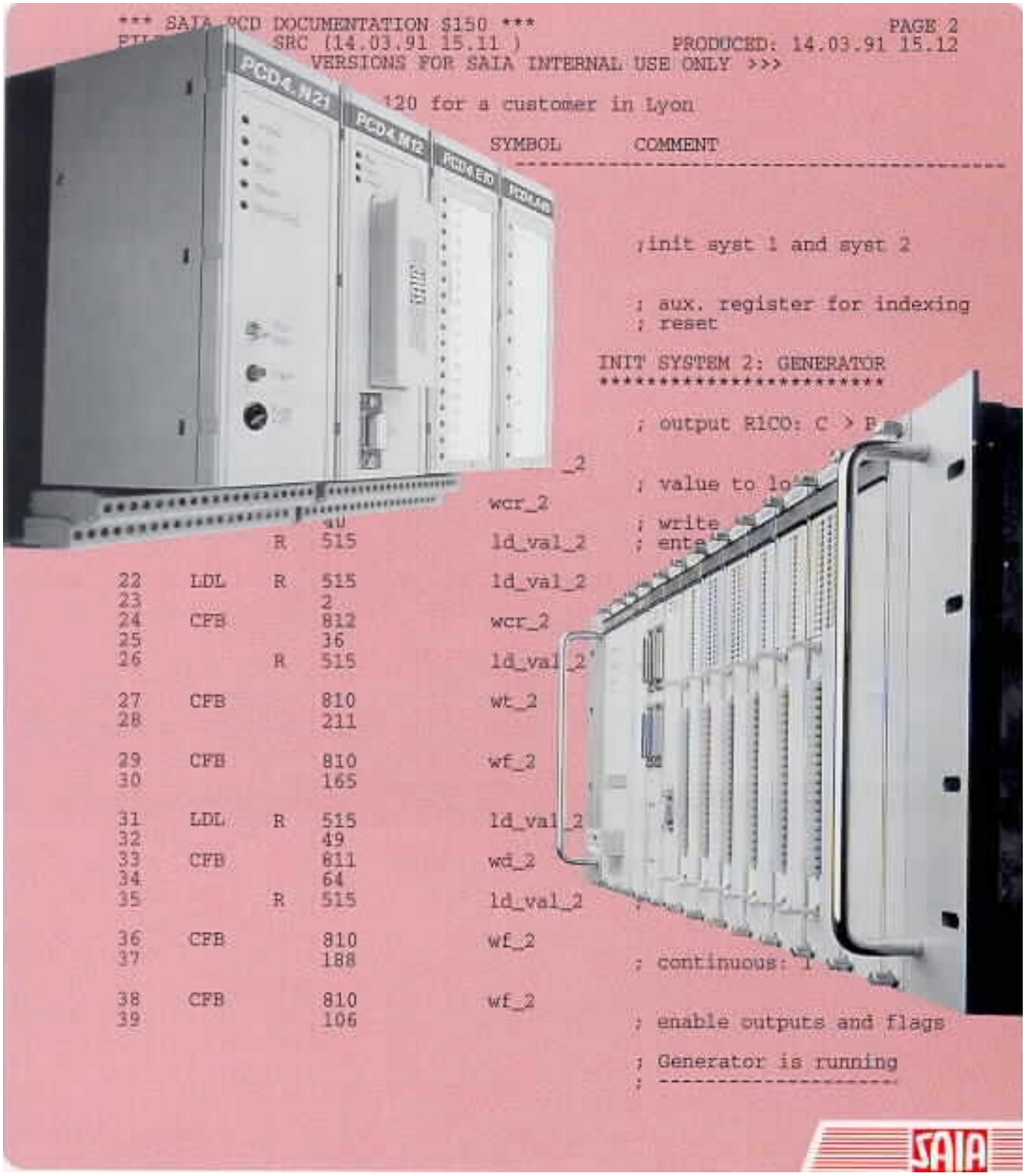
SAIA-Burgess Electronics

SWITCHES - MOTORS - CONTROLLERS

SAIA® PCD

Programmable Control Devices

Guide des instructions SAIA® PCD



Edition française 26/733 F6

Saia-Burgess Controls SA
Rue de la Gare 18
CH-3280 Morat (Suisse)
http;//www.saia-burgess.com

Téléphone 026 / 672 72 72
Télécopieur 026 / 672 74 99

Sociétés SAIA-Burgess

Suisse
Saia-Burgess Controls SA
Rue de la Gare 18
CH-3280 Morat
☎ 026 672 72 72, Fax 026 672 74 99

Allemagne
SAIA-Burgess Electronics GmbH & Co. KG
Otto-Hahn-Strasse 31 - 33
D-63303 Dreieich
☎ 06103 89 060, Fax 06103 89 06 66

Autriche
SAIA-Burgess Electronics Ges.m.b.H.
Schallmooser Hauptstrasse 38
A-5020 Salzburg
☎ 0662 88 49 10, Fax 0662 88 49 10 11

Italie
SAIA-Burgess Electronics S.r.l.
Via Cadamosto 3
I-20094 Corsico MI
☎ 02 48 69 21, Fax 02 48 60 06 92

France
SAIA-Burgess Electronics Sàrl.
10, Bld. Louise Michel
F-92230 Gennevilliers
☎ 01 46 88 07 70, Fax 01 46 88 07 99

Pays-Bas
SAIA-Burgess Electronics B.V.
Hanzeweg 12c
NL-2803 MC Gouda
☎ 0182 54 31 54, Fax 0182 54 31 51

Belgique
SAIA-Burgess Electronics Belgium
Avenue Roi Albert 1er, 50
B-1780 Wemmel
☎ 02 456 06 20, Fax 02 460 50 44

Hongrie
SAIA-Burgess Electronics Automation Kft.
Liget utca 1.
H-2040 Budaörs
☎ 23 501 170, Fax 23 501 180

Représentations

Grande-Bretagne
Canham Controls Ltd.
25 Fenlake Business Centre, Fengate
Peterborough PE1 5BQ UK
☎ 01733 89 44 89, Fax 01733 89 44 88

Danemark
Malthe Winje Automation AS
Håndværkerbyen 57 B
DK-2670 Greve
☎ 70 20 52 01, Fax 70 20 52 02

Norvège
Malthe Winje Automasjon AS
Haukelivn 48
N-1415 Oppegård
☎ 66 99 61 00, Fax 66 99 61 01

Suède
Malthe Winje Automation AB
Truckvägen 14A
S-194 52 Upplands Våsby
☎ 08 795 59 10, Fax 08 795 59 20

Suomi/ Finlande
ENERGEL OY
Atomitie 1
FIN-00370 Helsinki
☎ 09 586 2066, Fax 09 586 2046

Portugal
INFOCONTROL Electronica e Automatismo LDA.
Praceta Cesário Verde, No 10 s/cv, Massamá
P-2745 Queluz
☎ 21 430 08 24, Fax 21 430 08 04

Espagne
Tecnosistemas Medioambientales, S.L.
Poligono Industrial El Cabril, 9
E-28864 Ajalvir, Madrid
☎ 91 884 47 93, Fax 91 884 40 72

Tchéquie
ICS Industrie Control Service, s.r.o.
Modranská 43
CZ-14700 Praha 4
☎ 2 44 06 22 79, Fax 2 44 46 08 57

Pologne
SABUR Ltd.
ul. Druzynowa 3A
PL-02-590 Warszawa
☎ 22 844 63 70, Fax 22 844 75 20

Argentine
MURTEN S.r.l.
Av. del Libertador 184, 4° “A”
RA-1001 Buenos Aires
☎ 054 11 4312 0172, Fax 054 11 4312 0172

Service après-vente

USA
SAIA-Burgess Electronics Inc.
1335 Barclay Boulevard
Buffalo Grove, IL 60089, USA
☎ 847 215 96 00, Fax 847 215 96 06

Issue : 12.01.2001

SAIA® Programmable Control Devices

Manuel

Guide des Instructions

SAIA® PCD

Saia-Burgess Controls SA

Edition 26/733 F6 - 06.97 Mise à jour – 10.2002

Sous réserve de modifications

Mise à jour

Manuel : Guide des instructions SAIA® PCD - édition F6

Date	Chapitre	Page	Description
12.04.2000	6	6-5 à 6-8	XOB : div. corrections et ajout de XOB 6
12.04.2000	6	6-18 et 6-19	SCOB : ancien et nouveau
12.04.2000	8	8-18 et 8-19	SASI : le texte SASI accepte le signe $
12.04.2000	8	8-48	SOCL : basculement en RS 485 et RS 422
12.04.2000	12	12-14 à 12-17	SYSRD : div. corrections et lecture horloge
12.04.2000	12	12-18 à 12-22	SYSWR : div. corrections et écriture horloge
06.10.2000	12	12-21	SYSWR : Code 6000 (écriture dans l'EEPROM)
08.03.2001	3	3-8	DEC : corrections de DEC Registre
16.10.2001	13	13-3	SBUS-PGU ERROR | Oui | Tous
29.10.2002	8	8-3	MC5

Table des matières

Page

3. Instructions WORD

Page

4. Instructions INTEGER (arithmétique)

5. Instructions FLOATING POINT (arithmétique)

Page

6. Instructions BLOCTEC

7. Instructions GRAFTEC

Page

8. Instructions de COMMUNICATION

Page

Page

9. Instructions pour le LAN 2

10. Instructions de CONTROLE

11. Instructions de DEFINITION

Page

12. Instructions SPECIALES

13. Liste Historique "History List"

Liste alphabétique des instructions

Légende :

Colonne "Ind" :

'X' indique que l'instruction peut être indexée.

Colonne "Pm" :

'=' indique que cette instruction accepte des FB paramètres comme opérande.

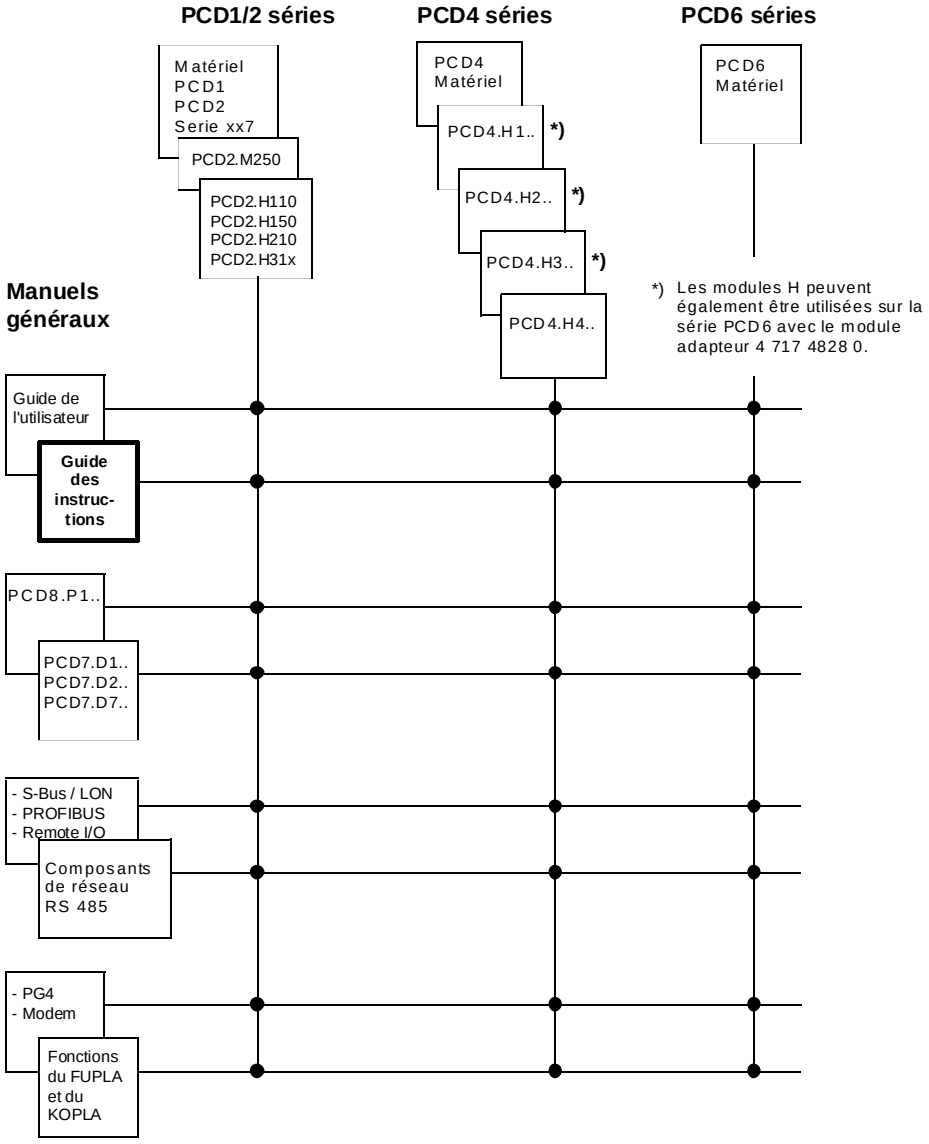
Avis aux lecteurs :

De nombreux manuels techniques précis et détaillés ont été élaborés par SAIA-Burgess Electronics SA afin de faciliter l'installation et l'exploitation de ses automates programmables ; ils s'adressent à un personnel qualifié ayant suivi au préalable nos stages de formation.

Pour optimiser les performances des appareils de commande de processus SAIA® PCD, nous vous conseillons de respecter scrupuleusement les consignes de montage, de câblage, de programmation et de mise en service figurant dans ces manuels. Cette démarche rigoureuse vous donnera l'assurance d'une satisfaction totale.

Toutefois, si vous souhaitez formuler des propositions ou des commentaires visant à améliorer la qualité et le contenu de nos documentations, nous vous serions reconnaissants de compléter le formulaire situé en dernière page de cette notice.

Vue d'ensemble de la gamme et de la documentation PCD



Fiabilité et sécurité des automates programmables

Soucieux d'offrir à sa clientèle des automates programmables fiables et sûrs, SAIA-Burgess Electronics SA apporte le plus grand soin à la conception, au développement et à la fabrication de ses produits.

Parmi ces mesures, citons :

- Technologie de pointe,
- Conformité aux normes,
- Certification ISO 9001,
- Agrément de nombreux organismes internationaux (Germanischer Lloyd, UL, Det Norske Veritas, marquage CE...),
- Choix de composants de haute qualité,
- Contrôles qualité aux différents stades de fabrication,
- Essais en conditions réelles de fonctionnement,
- Déverminage à 85°C pendant 48 heures.

Malgré l'excellence et le grand soin apporté à sa production, SAIA-Burgess Electronics SA ne saurait être tenu responsable des défaillances naturelles d'un composant. A cet égard, les « Conditions générales de vente » exposent clairement les limites de garantie offertes par SAIA-Burgess Electronics SA.

Le responsable de production doit également s'assurer de la fiabilité de son installation ; il lui incombe en effet de se conformer aux spécifications techniques de l'automate sans jamais le soumettre à des conditions extrêmes d'utilisation (respect de la plage de températures, protection contre les surtensions, immunité aux parasites et tenue aux chocs).

Il lui faut en outre veiller à l'application de toutes les règles de sécurité en vigueur afin de garantir qu'aucun produit défectueux ne risque de porter atteinte à la sécurité des biens et des personnes. Tout défaut générateur de danger doit donner lieu à des mesures complémentaires visant à l'identifier et à en prévenir les conséquences. Ainsi les sorties directement liées à la sécurité de fonctionnement du matériel doivent être raccordées aux entrées et surveillées par logiciel. Il convient enfin de faire systématiquement appel aux fonctions de diagnostic du PCD (chien de garde, blocs d'organisation des exceptions « XOB », instructions de test ou de recherche d'erreurs).

Exploitée dans les règles de l'art, la gamme SAIA® PCD intègre des constituants d'automatismes modernes, alliant sécurité et haute fiabilité, et capables d'assurer pendant des années les fonctions de contrôle-commande, de régulation et de surveillance de votre équipement.

1. Introduction

Ce Guide de référence décrit en détail chaque instruction de la famille SAIA® PCD.
Les instructions sont groupées en fonction de leur type afin de faciliter leur apprentissage.

Ce guide est destiné à compléter le "Guide Utilisateur" qui donne une description détaillée des "PCD Utilities" et des méthodes de programmation structurée utilisées dans le SAIA® PCD.

Important

Les instructions décrites dans ce guide sont valides pour:

PCD1	version V001 (ou plus récente)
PCD2	version V004 (ou plus récente)
PCD4	version V00C (ou plus récente)
PCD6.M5	version V005 (ou plus récente)
PCD6.M1/M2	version V009 (ou plus récente)
PCD6.M3	version V001 (ou plus récente)
PCD Utilities	version V2.0

Si vous possédez un PCD avec une version plus ancienne: certaines instructions peuvent être différentes ou être non-existantes.

Comment lire ce manuel

Une ou plusieurs pages sont consacrées pour chaque instruction du SAIA® PCD.

Ligne supérieure
: La première ligne donne pour chaque instruction: son mnémonique et sa dénomination en anglais.

Description
: Décrit le fonctionnement de l'instruction et son opérande.

Usage
: Montre comment l'instruction est utilisée, avec le type et la plage de chaque opérande possible. Un "[X]" après le mnémonique signifie que l'adressage indexé est possible en ajoutant un 'X' optionnel au mnémonique (ex: STHX, INCX). Pour l'adressage indexé, l'opérande qui est indexée est indiquée par un "(i)".

Exemple
: Un exemple typique de l'instruction.

Flags
: Montre les flags qui peuvent être modifiés (ACCU, N, P, Z, E)

Voir aussi
: Une liste des instructions ou des paragraphes ayant un rapport avec l'instruction considérée.

Pratique
: Un exemple pratique montrant l'utilisation de l'instruction dans un contexte réel.

Conventions typographiques :

[ ]

Les crochets indiquent que le texte ou la donnée est optionnel. Par exemple: [;commentaires] signifie que ";commentaires" est optionnel et ne doit pas être nécessairement présent.

[X]

Un "[X]" qui suit le mnémonique indique que l'adressage indexé est possible en ajoutant un 'X' optionnel au mnémonique (ex: STHX, INCX).

(i)

Quand l'adressage indexé est possible (voir [X]); le ou les opérandes qui sont indexées sont marquées par un "(i)"

.....

Une série de "." dans un exemple montre que celui-ci peut être complété par vous même.

LABEL:

Dans les exemples, tous les labels sont représentés avec leur nom suivit d'un ':'; ceci est seulement nécessaire si vous utilisez un autre éditeur que SEDIT.
Si vous utilisez SEDIT, supprimez ce ':' après la label.

< >

Les parenthèses angulaires entourent un texte ou une expression qui ne doit pas être introduit tel quel, mais remplacé par un texte ou une expression valide.

1.1 MEDIUM code (MC)

Le Medium control code [mc] est utilisé pour sélectionner le type d'élément.

CODE	TYPE	ADRESSES
I	Input	0..8191
O	Output	0..8191
F	Flag	0..8191
R	Register	0..4095
T	Timer	0..450
C	Counter	0..1599
K	Constant	0..16383
X	teXt	0..7999
DB	Data Block	0..7999

Entrées (Inputs) et Sorties (Outputs)

Les entrées et sorties sous forme de modules peuvent être placés dans le PCD, l'adresse d'un module est fournie par la position de ce module sur les PCD1, PCD2 et PCD4 ou par des interrupteurs sur le PCD6.
Les entrées peuvent seulement être interrogées.
Les sorties peuvent être enclenchées (set) ou déclenchées (reset) mais aussi interrogées quant à leur état.

Indicateurs (Flags)

Les indicateurs sont des mémoires de 1 bit qui peuvent être traitées comme des sorties; un indicateur peut être enclenché ou déclenché, et interrogé sur son état. Les indicateurs sont utilisés pour la mémorisation d'informations.

Temporisateurs (Timers) et Compteurs (Counters)

Temporisateurs et compteurs sont des registres programmables, ils peuvent contenir des valeurs de 31 bits (0 - 2.147.483.647 en décimal). Ils occupent la même plage d'adressage de 0..1599: le nombre de temporisateurs dépend de l'instruction DEFTC (par défaut: 32 temporisateurs aux adresses 0 à 31 et 1568 compteurs des adresse 32 à 1599)

La seule différence entre un temporisateur et un compteur est que le temporisateur est décrémenté automatiquement à intervalle régulier suivant une base de temps (pouvant être définie par l'instruction DEFTB, valeur par défaut: 1/10 sec)

Les temporisateurs et les compteurs ne peuvent contenir de valeurs négatives.

Quand un temporisateur ou un compteur contient une valeur différente de zéro, son état est haut (High (H)). Quand le contenu est nul, son état est bas (Low (L))

Registres (Registers)

Un registre est une cellule mémoire de 32 bits pouvant contenir n'importe quelle information en binaire, décimal, hexadécimal ou virgule flottante.
Vous pouvez faire des opérations mathématiques sur les registres, transférer d'autres informations (de ou vers: entrées, sorties, indicateurs, temporisateurs, compteurs, registres).
Les registres peuvent aussi bien contenir des valeurs positives que négatives.

Constantes (Constants)

Plusieurs types de constantes peuvent être utilisées (Voir page suivante).

TeXte (Text)

Les textes sont des chaînes de caractères ASCII mémorisées dans le PCD et destinées à être envoyées par la ligne série.

Bloc de données (Data Block)

Un bloc de données est une zone de la mémoire pour mémoriser des données 32 bits, pouvant être transférées de et vers les Registres, Compteurs et Tempos.

1.2 Constantes

Il y a plusieurs types de constantes utilisées dans les instructions du PCD, la plage valide dépend de l'instruction.

Les constantes de type K ne peuvent pas être utilisées pour les instructions LOAD (LD, LDH et LDL).

Binaire (BINARY)

La valeur est terminée avec un 'Q' ou 'Y', eg. 1001Q, 11111111Y. Max 12 bits.

DECIMAL

Défaut, pas de format spécial.

HEXADECIMAL

La valeur est terminée avec 'H', eg. ABCDH, 1234H, DEADH.

ASCII

Le caractère ASCII doit être entouré par des apostrophes, ex. 'A', 'z'.

Virgule flottante (FLOATING POINT (FFP))

Un point décimal doit être présent et/ou le signe exposant ('E'), ex. 1.2, 12E-1
Plage (FFP) : 2.710505E-20 … - 9.223371E+18.

K CONSTANTES

Utilisées quand un "médium control code" (mc) est requis pour éviter les confusions: pour montrer qu'il s'agit d'une constante le mc "K" est utilisé.
Les instructions LOAD (LD, LDL, LDH) n'utilisent jamais ce type de constante.
Les types binaires, hexadécimales ou Ascii peuvent être spécifiés, ex. K 10, K 11Q, K FFH, K 'A'.

Type de constante	**Valeur Minimum**	**Valeur Maximum**
Décimale	-2.147.483.648	2.147.483.647
Hexadécimale	0 H	FFFFFFFF H
Binaire	0 Q	111111111111 Q
ASCII	'a', 'B', '%', etc. (max 4 caractères/registre)	
Virgule flottante	-9.22337177 E+18	+9.22337177 E+18
	-2.71050535 E -20	+5.42101070 E-20

1.3 Les codes conditionnels (Condition Codes [cc])

Les flags arithmétiques (ARITHMETIC STATUS FLAGS)

Les flags arithmétiques sont positionnés par les opérations mathématiques (en entier ou en virgule flottante).
Le flag erreur (Error) peut être mis High par n'importe quelle instruction exécutée avec des données non valides.

P	Positif	High si le résultat d'une instruction arithmétique est positif
N	Negatif	High si le résultat d'une instruction arithmétique est négatif (le flag P a toujours l'état inverse du flag N)
Z	Zéro	High si le résultat d'une opération arithmétique est 0
E	Erreur	High si une instruction ne peut être exécutée High en cas de dépassement de capacité ou conversion fautive

Accumulateur (ACCUMULATOR)

Etant donné que le processeur "lit" une seule instruction à la fois, il ne peut interroger l'état des éléments que un par un.
Pour pouvoir exécuter une branche jusqu'à une instruction d'action, le résultat intermédiaire après chaque instruction est mémorisé dans l'Accumulateur (ACCU). A la fin d'une branche, le résultat final est présent dans l'ACCU (0 ou 1). Sur base de ce résultat un élément (une sortie, par exemple) pourra être déclenché (ACCU = 0) ou enclenché (ACCU = 1).

L' ACCUmulateur est positionné High/Low (1 ou 0) principalement par les instructions de Bits. L'état de l'ACCU peut être positionné suivant un état bien défini, ou suivant l'état des flags arithmétiques par l'instruction ACC.

Code conditionnel (CONDITION CODES [CCs])

L'exécution de certaines instructions peut être liée à l'état d'un flag de status, cette dépendance se fait à l'aide du code conditionnel [cc].
Si la condition est fausse, l'instruction n'est pas exécutée.

NOTE: Certaines instructions sont dépendantes de l'Accumulateur, et ne sont exécutées que quand l'Accumulateur (ACCU) est High (1).

blanc	Pas de condition
H	Si l'Accumulateur = H
L	Si l'Accumulateur = L
P	Si le flag Positif = H (flag Negatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur flag = H
(C)	Complémente utilisé seulement avec l'instruction ACC

1.4 Types de ressources et valeurs

Type	Description		Plage	Remarque	
I	Input	Partagent les	0..8191		
O	Output	mêmes adresses	0..8191		
F	Flag		0..8191	Volatile / Non volatile	Par Système
T	Timer	Partagent les	0..450	Volatile	
C	Counter	mêmes adresses	0..1599	Non Volatile	
R	Register		0..4095	Non Volatile	
K	K constant		0..16383		
COB	Cyclic Organisation Block		0..15		
XOB	Exception Organisation Block		0..31		
PB	Program Block		0..299		
FB	Function Block		0..999		
SB	Sequential Block		0..31		Par CPU
IST	Initial Step		0..1999		
ST	Step		0..1999		
TR	Transition		0..1999		
X	Text	Partagent les	0..7999	4000..7999	
DB	Data Block	mêmes adresses	0..7999	en mémoire étendue	
-	Semaphore		0..99		Par Système

Notes personnelles :

2. Instructions BIT

Les instructions BIT travaillent avec l'Accumulateur, les entrées, les sorties, les flags et l'état des temporisateurs ou compteurs.

STH	STart High	Démarrage d'une combinaison logique avec interrogation "H"
STL	STart Low	Démarrage d'une combinaison logique avec interrogation "L"
ANH	ANd High	Combinaison logique ET
ANL	ANd Low	Combinaison logique ET inverse
ORH	OR High	Combinaison logique OU
ORL	OR Low	Combinaison logique OU inverse
XOR	eXclusive OR	Combinaison logique OU exclusif
ACC	ACCumulator operation	Opérations sur l'ACCUmulateur
DYN	DYNnamic	Interrogation dynamique (détection de flancs)
OUT	Set element from ACCU	Positionnement d'une sortie / flag suivant l'état de l'ACCU
SET	SET element	Enclenchement d'une sortie / flag
RES	RESet element	Déclenchement d'une sortie / flag
COM	COMplement element	Inversion de l’état d'une sortie / flag
SETD	SET element Delayed	Enclenchement retardé d'une sortie / flag
RESD	RESet element Delayed	Déclenchement retardé d'une sortie / flag

Notes personnelles :

STH START HIGH

Démarrage d'une combinaison logique avec interrogation "H"

Description L'ACCU est positionné suivant l'état logique de l'élément adressé.

Ceci est le départ d'une nouvelle branche.

Le résultat de la branche précédente est effacé par l'instruction; simultanément, l'état "H" de l'élément adressé (I, O, F, T, C) est lu et le résultat est mémorisé dans l'ACCU.

Usage

STH[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

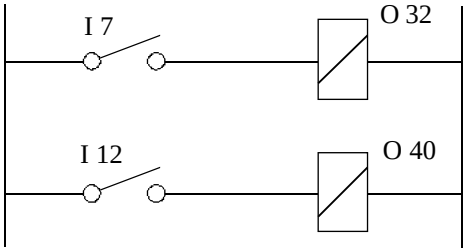
Exemple STH I 7 ; ACCU = état de I 7

Flags ACCU positionné sur l'état de l'élément adressé.

Note Si un temporisateur ou un compteur contient 0: son état est défini comme L, sinon son état est H.

Voir aussi STL, STHS.

Pratique



; Un programme PCD est toujours constitué d'au moins un COB.

```
COB       0     ; En-tête du COB
          0
STH       I 7   ; Si l'entrée 7 est H
OUT       O 32  ;    Alors enclenchement de la sortie 32
                ;    Sinon déclenchement de la sortie 32
STH       I 12  ; Si l'entrée 12 est H
OUT       O 40  ;    Alors enclenchement de la sortie 40
                ;    Sinon déclenchement de la sortie 40

ECOB            ; Fin du COB (End of COB)
```


STL START LOW

Démarrage d'une combinaison logique avec interrogation "L"

Description L'ACCU est positionné suivant l'état logique inverse de l'élément adressé.

Ceci est le départ d'une nouvelle branche.

Le résultat de la branche précédente est effacé par l'instruction; simultanément, l'état "L" de l'élément adressé (I, O, F, T, C) est lu, inversé et le résultat est mémorisé dans l'ACCU.

Usage

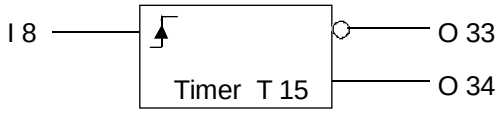
STL[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

Exemple STL I 9 ; ACCU = état inverse de I 9

Flags ACCU positionné sur l'état inverse de l'élément adressé.

Voir aussi STH.

Pratique




```
COB         0       ; En-tête du COB
            0

STH       I 8       ; Si l'entrée 8 devient haute
DYN       F 10
LD        T 15      ;    Alors charger la tempo avec 2 sec.
            20
STL       T 15      ; Si la tempo est écoulée
OUT       O 33      ;    Alors enclenchement de la sortie 33
                    ;    Sinon déclenchement de la sortie 33
STH       T 15      ; Si la tempo n'est pas écoulée
OUT       O 34      ;    Alors enclenchement de la sortie 34
                    ;    Sinon déclenchement de la sortie 34

ECOB                ; Fin du COB
```

ANH AND HIGH
Combinaison logique ET

Description Combinaison logique "ET" entre l'ACCU et l'état de l'élément adressé, le résultat est mémorisé dans l'ACCU.

Usage

ANH[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

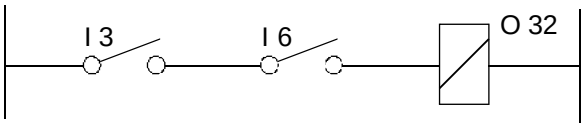
Exemple

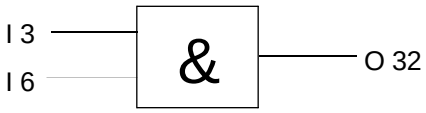
```
ANH     I 3     ; ET entre l'ACCU et l'entrée 3
ANHX    I 128   ; ET entre l'ACCU et l'entrée  (128+Registre d'Index)
```

Flags ACCU positionné suivant le résultat.

Voir aussi ANL.

Pratique






```
COB         0       ; En-tête du COB
            0

STH       I 3       ; Si l'entrée 3 est H
ANH       I 6       ;    Et l'entrée 6 est H
OUT       O 32      ;    Alors enclenchement de la sortie 32
                    ;    Sinon déclenchement de la sortie 32

ECOB                ; Fin du COB
```


ANL AND LOW
Combinaison logique ET inverse

Description Combinaison logique "ET" entre l'ACCU et l'état inverse de l'élément adressé, le résultat est mémorisé dans l'ACCU.

Usage

ANL[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

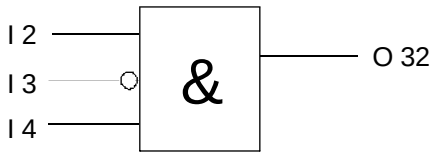
Exemple

```
ANL     I 4     ; ET entre l'ACCU et l'état inverse de l'entrée 4
ANHX    I 128   ; ET entre l'ACCU et l'état inverse de l'entrée
                ;       (128 + Registre d'Index)
```

Flags ACCU positionné suivant le résultat.

Voir aussi ANH.

Pratique




```
COB         0       ; En-tête du COB
            0

STH       I 2       ; Si l'entrée 2 est H
ANL       I 3       ;     Et l'entrée 3 est L
ANH       I 4       ;     Et l'entrée 4 est H
OUT       O 32      ;     Alors enclenchement de la sortie 32
                    ;     Sinon déclenchement de la sortie 32

ECOB                ; Fin du COB
```

ORH OR HIGH
Combinaison logique OU

Description

Combinaison logique "OU" entre l'ACCU et l'état de l'élément adressé, le résultat est mémorisé dans l'ACCU.

Les instructions OU sont utilisées pour la liaison en parallèle des éléments.

Une combinaison débute par un "start" (STH ou STL); chaque branche parallèle partielle débute par un ORH.

Si une branche parallèle partielle est remplie (ACCU=1) alors l'état logique des branches partielles suivantes n'exerce plus d'influence sur le résultat de la combinaison totale.

Usage

ORH[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

Exemple

```
STH     I 5     ; Si I 5 est H
ORH     I 13    ;    OU I 13 est H
                ; Alors ACCU = 1, Sinon ACCU = 0
```

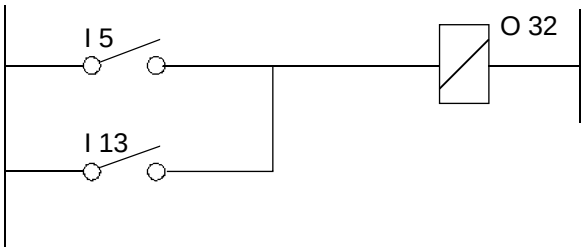
Flags

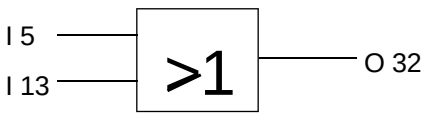
ACCU positionné suivant le résultat.

Voir aussi

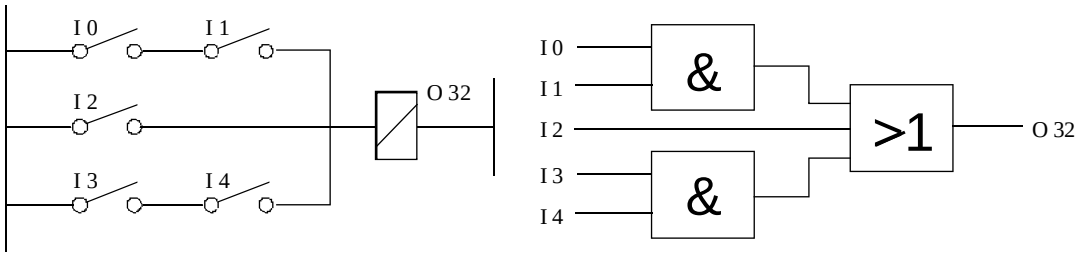
ORL.

Pratique






```
COB         0     ; En-tête du COB
            0
STH       I 5     ; Si l'entrée 5 est H
ORH       I 13    ;    Ou l'entrée 13 est H
OUT       O 32    ;    Alors enclenchement de la sortie 32
                  ;    Sinon déclenchement de la sortie 32
ECOB              ; Fin du COB
```

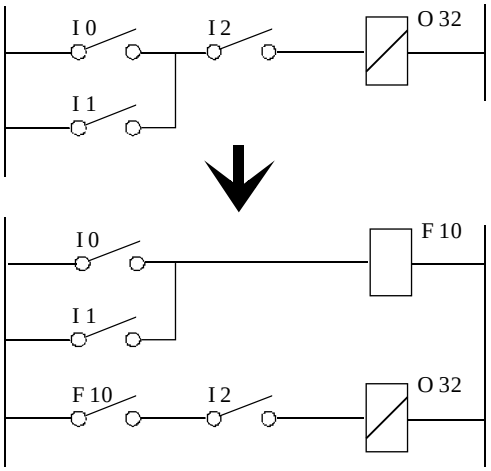



```
COB         0       ; En-tête du COB
            0

STH       I 0       ; Si l'entrée 0 est H
ANH       I 1       ;    Et l'entrée 1 est H
ORH       I 2       ;    Ou l'entrée 2 est H
ORH       I 3       ;    Ou l'entrée 3 est H
ANH       I 4       ;    Et l'entrée 4 est H
OUT       O 32      ;    Alors enclenchement de la sortie 32
                    ; Sinon déclenchement de la sortie 32

ECOB                ; Fin du COB
```

D'après l'exemple ci-dessus, l'instruction OR a "priorité" sur le AND.




```
COB         0       ; En-tête du COB
            0

STH       I 0       ; Si l'entrée 0 est H
ORH       I 1       ;    Ou si l'entrée 1 est H
OUT       F 10      ;    Alors enclenchement du Flag 10
                    ;    Sinon déclenchement du Flag 10
STH       F 10      ; Si le Flag 10 est H
ANH       I 2       ;    Et l'entrée 2 est H
OUT       O 32      ;    Alors enclenchement de la sortie 32
                    ;    Sinon déclenchement de la sortie 32

ECOB                ; Fin du COB
```

ORL OR LOW
Combinaison logique OU inverse

Description Combinaison logique "OU" entre l'ACCU et l'état inverse de l'élément adressé, le résultat est mémorisé dans l'ACCU.

Les instructions OU sont utilisées pour la mise en parallèle des éléments.

Usage

ORL[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

Exemple

```
STH     I 3     ; Si I 3 est H
ORL     I 7     ;    OU si I 7 est L
                ; Alors ACCU = 1, sinon ACCU = 0
```

Flags ACCU positionné suivant le résultat.

Voir aussi ORH.

XOR EXCLUSIVE OR

Combinaison logique OU EXCLUSIF

Description Combinaison logique "OU Exclusif " entre l'ACCU et l'état de l'élément adressé, le résultat est mémorisé dans l'ACCU.

Grâce à l'instruction XOR, les états logiques des deux éléments peuvent être comparés entre eux. Si ils sont identiques, l'ACCU contient 0; si ils sont différents, l'ACCU est 1.

Usage

XOR[X]	**élément (i)**	**; I 0-8191, O 0-8191, F 0-8191** **; T 0-450, C 0-1599**

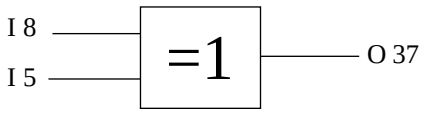
Exemple

```
XOR     I 5     ; ACCU = ACCU XOR I 5
```

Flags ACCU est positionné suivant le résultat.

Voir aussi OR, AND.

Pratique




```
COB       0       ; En-tête du COB
          0
STH       I 8     ; Si l'entrée 8 est H
XOR       I 5     ;    XOR l'entrée 5 est H
OUT       O 37    ;    Alors enclenchement de la sortie 37
                  ;    Sinon déclenchement de la sortie 37

ECOB              ; Fin du COB
```

I 8	I 5	O 37
L	L	L
L	H	H
H	L	H
H	H	L

ACC ACCUMULATOR OPERATIONS
Opérations sur l'ACCUmulateur

Description Modifie l'état de l'Accumulateur suivant le code dans l'opérande :

C	Complément	l'Accu est inversé
H	High	Accu = High (1)
L	Low	Accu = Low (0)
P	Positif	Accu = état du flag Positif (P)
N	Négatif	Accu = état du flag Négatif (N)
Z	Zéro	Accu = état du flag Zéro (Z)
E	Error	Accu = état du flag Error (E)

L'opérande ne peut pas être passée comme paramètre d'un bloc de fonction (FB).

Usage

```
ACC     code            ; code = C|H|L|P|N|Z|E
```

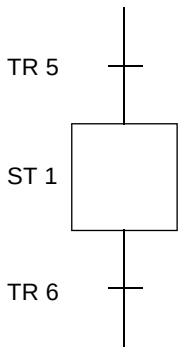
Exemple

```
ACC     H       ; ACCU = 1
ACC     E       ; ACCU = état du flag E
```

Flags L'ACCU est modifié.

Voir aussi OUT, Codes conditionnels.

Pratique




```
CMP     R 99    ; Compare R 99
        'x'     ;   avec le caractère ASCII 'x'
ACC     Z       ; Si égal, l'ACCU est mis H
                ;    (copie du flag Z dans l' ACCU)

STXT    1
        10

....
```

DYN DYNAMIC (EDGE DETECTION)

Interrogation dynamique, détection de flancs

Description Pour la détection de flancs montants ou descendants.

Le résultat dans l'ACCU est High seulement lorsque l'ACCU change de l'état Low à High lors des exécutions successives du DYN.
Le flag donné comme opérande permet de mémoriser l'état précédent de l'ACCU. Si l'ACCU est Low, il reste Low, et le flag est également Low. Le flag ne doit pas être nécessairement Low la première fois que l'instruction DYN est exécutée.
Pour la détection d'un flanc montant, utilisez l'instruction STH pour interroger l'élément; pour la détection d'un flanc descendant, utilisant STL.

Usage

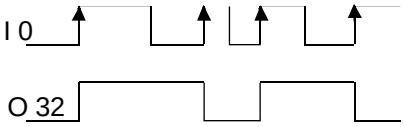
DYN[X]	**flag**	**(i)**	**; F 0-8191**

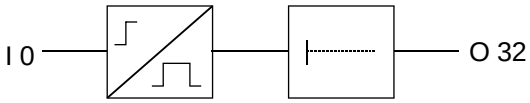
Exemple DYN F 100 ; le Flag 100 mémorise l'état de l'ACCU

Flags L'ACCU est positionné suivant le résultat.

Voir aussi STH, STL, ANH, ANL, ORH, ORL.

Pratique





; Solution avec l'instruction DYN

```
        COB     0       ; En-tête du COB
                0
        STH     I 0     ; Si l'entrée 0 devient H
        DYN     F 500   ;    (Détection du flanc)
        COM     O 32    ;    Alors inversion de la sortie 32
                        ;    Sinon rien
        ECOB            ; Fin du COB
```

; Solution sans l'instruction DYN

```
        COB     0       ; En-tête du COB
                0
        STH     I 0     ; Si l'entrée 0 est H
        ANL     F 500   ;    Et le flag 500 est L
        SET     F 500   ;    Alors flag 500 est mis H
        COM     O 32    ;    Inversion de la sortie 32
                        ;    Sinon rien
        STL     I 0     ; Si l'entrée 0 est L
        RES     F 500   ;    Alors flag 500 est mis L
                        ;    Sinon rien
        ECOB            ; Fin du COB
```

OUT SET ELEMENT FROM ACCUMULATOR

Positionnement d'une sortie / flag suivant l'état de l'ACCU

Description Positionne une sortie ou un Flag suivant l'état de l'ACCU.

Si l'ACCU est High, alors la sortie ou le Flag est mis High. Si l'ACCU est Low, alors la sortie ou le Flag est mis Low.

Usage

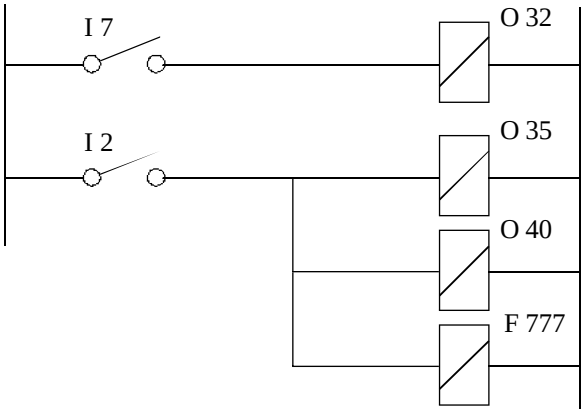
OUT[X] élément (i) ; O 0-8191, F 0-8191

Exemple OUT O 32 ; Positionne la sortie 32 suivant l'état de l'ACCU

Flags Inchangés.

Voir aussi OUTS.

Pratique




```
COB         0       ; En-tête du COB
            0

STH       I 7       ; Si l'entrée 7 est H
OUT       O 32      ;    Alors enclenchement de la sortie 32
                    ;    Sinon déclenchement de la sortie 32
STH       I 2       ; Si l'entrée 2 est H
OUT       O 35      ;    Alors enclenchement de la sortie 35
                    ;    Sinon déclenchement.
OUT       O 40      ;    Enclenchement de la sortie 40
                    ;    Sinon déclenchement
OUT       F 777     ;    Enclenchement du flag 777
                    ;    Sinon déclenchement

ECOB                ; Fin du COB
```

SET SET ELEMENT

Enclenchement d'une sortie / flag

Description La sortie ou le Flag est mis High **seulement si l'ACCU est High**.

Si l'ACCU est Low, l'instruction n'est pas exécutée.

Une sortie ou un flag positionné avec l'instruction SET reste enclenché (H) jusqu'au moment ou il est déclenché par une instruction RES (ou OUT). L'exécution, dans les deux cas, dépend du succès (ACCU = 1) de la branche qui précède. Lorsque SET ou RES sont utilisés dans un programme cyclique (diagramme logique), il faudra tenir compte de la priorité à donner à l'enclenchement ou au déclenchement.

Usage

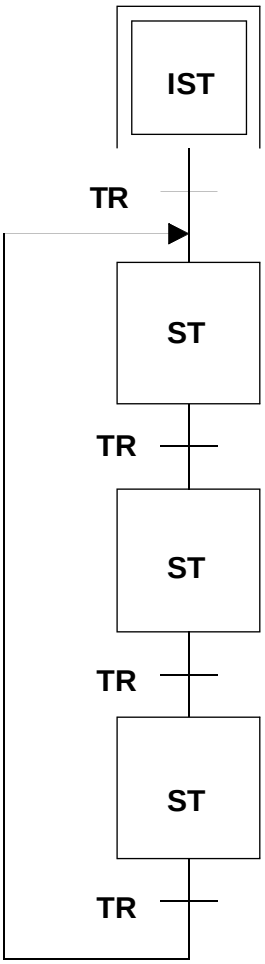
SET[X] élément (i) ; O 0-8191, F 0-8191

Exemple SET O 32 ; Si l'ACCU est 1 alors O 32 = H

Flags Inchangés.

Voir aussi RES, SETD, RESD.

Pratique Après l'enclenchement de l'entrée 7, les sorties 36 et 37 doivent clignoter.




```
STH   I   7      ; l'entrée est H ?

SET   O   36     ; enclenchement de la sortie 36
SET   O   37     ; enclenchement de la sortie 37
LD    T   1      ; chargement tempo n° 1
          5      ; avec 0.5 sec.

STL   T   1      ; tempo écoulé ?

RES   O   36     ; déclenchement de la sortie 36
RES   O   37     ; déclenchement de la sorite 37
LD    T   2      ; chargement tempo n° 2
          10     ; avec 1 sec.

STL   T   10     ; tempo écoulé ?
```

RES RESET ELEMENT

Déclenchement d'une sortie / flag

Description La sortie ou le flag est mis Low **seulement lorsque l'ACCU est High**.
Si l'ACCU est Low, l'instruction n'est pas exécutée.

Usage

RES[X] élément (i) ; O 0-8191, F 0-8191

Exemple
```
RES     O 13    ; Si l'ACCU est 1 alors O 13 = L
```
Flags Inchangés.

Voir aussi SET, SETD, RESD.

Pratique voir SET.

COM COMPLEMENT ELEMENT

Inversion de l'état d'une sortie / flag

Description L'état de la sortie ou du flag est complémenté (inversé) **seulement si l'ACCU est High.**

Si l'ACCU est Low, l'instruction n'est pas exécutée.

Note : Cette instruction est principalement utilisée pour l'activation du "Watch-dog" (chien de garde): il suffit de placer l'instruction "COM O 255" dans un programme cyclique. Il faut tenir compte du fait que l'instruction n'est exécutée que si l'ACCU = H. ("ACC H" peut être placé juste avant l'instruction ou COM peut être placé juste après l'instruction COB).

Usage

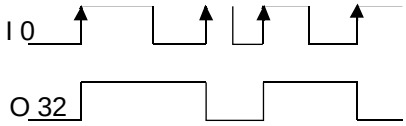
COM[X] élément (i) ; O 0-8191, F 0-8191

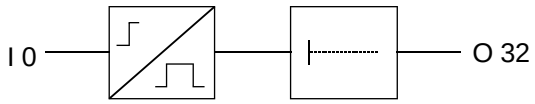
Exemple COM O 32 ; Si l'ACCU est 1, l'état de la sortie 32 est inversé

Flags Inchangés.

Voir aussi OUT, SET, RES, DYN

Pratique






```
COB       0      ; En-tête du COB
          0
STH       I 0    ; Si l'entrée 0 devient H
DYN       F 500  ;    (Détection du flanc)
COM       O 32   ;    Alors inversion de la sortie 32
                 ;    Sinon rien

ECOB             ; Fin du COB
```

SETD SET ELEMENT DELAYED

Enclenchement retardé d'une sortie / flag

Description La sortie ou le flag est enclenché (état High) après le délai donné dans l'opérande de la seconde ligne **seulement si l'ACCU est High**.

Si l'ACCU est Low, l'instruction n'est pas exécutée.
Le délai est donné en unité de base de temps, comme définie par l'instruction DEFTB.

Note : Un maximum de 16 actions retardées peuvent être exécutées simultanément. Dans un programme BLOCTEC, cette instruction doit toujours être utilisée en combinaison avec l'instruction DYN. Sans l'instruction DYN, un autre temporisateur sera activé à chaque cycle du programme, causant ainsi le positionnement du flag Error après le 16ème passage lorsque tous les temporisateurs auront été utilisés.

Usage

SETD[X]	**élément (i)**	**; O 0-8191, F 0-8191**
	délai	**; Délai en unités de base de temps (ex. 100ms)**

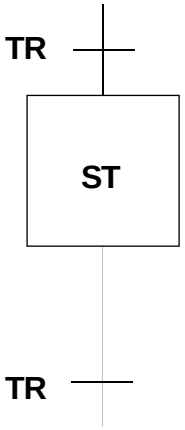
Exemple

```
SETD    O 32    ; Si l'ACCU est 1 alors sortie 32 = H
        100     ;    après 100 x 100mS = 10 secondes.
```

Flags Le flag **Error** est positionné si plus de 16 actions retardées sont actives simultanément.

Voir aussi RESD, DEFTB.

Pratique




```
STH ...          ; Condition de continuation du progr.

SET   O  35      ; O 35 est enclenchée immédiatement
SETD  O  40      ; O 40 est enclenchée après 120 sec.
         1200    ;    indépendamment de la
                 ;    continuation du programme
                 ;    GRAFTEC

STH ...          ; Condition de continuation du progr.
```

RESD RESET ELEMENT DELAYED

Déclenchement retardé d'une sortie / flag

Description La sortie ou le flag est déclenché (état Low) après le délai donné dans l'opérande de la seconde ligne **seulement si l'ACCU est High**.

Si l'ACCU est Low, l'instruction n'est pas exécutée.
Le délai est donné en unité de base de temps, comme définie par l'instruction DEFTB.

Note : Un maximum de 16 actions retardées peuvent être exécutées simultanément. Dans un programme BLOCTEC, cette instruction doit toujours être utilisée en combinaison avec l'instruction DYN. Sans l'instruction DYN, un autre temporisateur sera activé à chaque cycle du programme, causant ainsi le positionnement du flag Error après le 16ème passage lorsque tous les temporisateurs auront été utilisés.

Usage

```
RESD[X]   élément (i)    ; O 0-8191, F 0-8191
          délai          ; Délai en unités de base de temps (ex. 100ms)
```

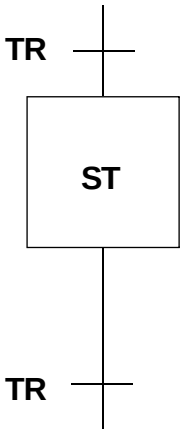
Exemple

```
RESD    O 32    ; Si l'ACCU est 1 alors la sortie 32 = L
        100     ;    après 100 x 100ms = 10 secondes.
```

Flags Le flag **Error** est positionné si plus de 16 actions retardées sont actives simultanément.

Voir aussi SETD, DEFTB.

Pratique




```
STH ...          ; Condition de continuation du progr.

SET   O  35      ; O 35 est enclenchée immédiatement
RESD  O  35      ; O 40 est déclenchée après 50 sec.
         50      ;  indépendamment de la
                 ;  continuation du programme
                 ;  GRAFTEC

STH ...          ; Condition de continuation du progr.
```

3. Instructions WORD

Les instructions WORD travaillent avec les Registres.

Les Registres contiennent des valeurs binaires, décimales, hexadécimales ou en virgule flottante. Pour des valeurs en virgule flottante, les instructions "Floating Point" doivent être utilisées.

Chargement de données :

LD	LoaD	Chargement d'une valeur de 32-bits
LDL	LoaD Low	Chargement d'une valeur de 16-bits (poids faible)
LDH	LoaD High	Chargement d'une valeur de 16-bits (poids fort)
DSP	load DiSPlay register	Chargement du registre d'affichage

Arithmétique élémentaire :

INC	INCrement register	Incrémentation (+1) d'un registre / compteur
DEC	DECrement register	Décrémentation (-1) d'un registre / compteur

Registre d'index :

SEI	SEt Index	Chargement du registre d'index
INI	Increment Index	Incrémentation (+1) du registre d'index
DEI	DEcrement Index	Décrémentation (-1) du registre d'index
STI	STore Index	Mémorisation du registre d'index
RSI	ReStore Index	Restitution du registre d'index

Copie de données :

MOV	MOVe data	Déplacement de données	
COPY	COPY data	Copie de données	Spécialement
GET	GET data	Transfert de données	utile pour
PUT	PUT data	Transfert de données	l'adressage indexé
TFR	TransFeR data	Transfert de données	
TFRI	TransFeR data Indirect	Transfert indirect de données	

Lecture / écriture binaire :

BITI	BIT In	Lecture d'une valeur binaire
BITIR	BIT In Reversed	Lecture inversée d'une valeur binaire
BITO	BIT Out	Sortie d'une valeur binaire
BITOR	BIT Out Reversed	Sortie inversée d'une valeur binaire

Lecture / écriture BCD :

DIGI	DIGit In	Lecture d'une valeur BCD
DIGIR	DIGit In Reversed	Lecture inversée d'une valeur BCD
DIGO	DIGit Out	Sortie d'une valeur BCD
DIGOR	DIGit Out Reversed	Sortie inversée d'une valeur BCD

Logique :

AND	AND registers	Combinaison logique ET entre deux registres
OR	OR registers	Combinaison logique OU entre deux registres
EXOR	EXclusive-OR registers	Combinaison logique OU exclusif entre deux registres
NOT	complement registers	Inversion logique du contenu d'un registre

Décalage et rotation :

SHIU	SHIft registers Up	Décalage vers le haut d'un bloc de registres
SHID	SHIft registers Down	Décalage vers le bas d'un bloc de registres
ROTU	ROTate registers Up	Rotation vers le haut d'un bloc de registres
ROTD	ROTate registers Down	Rotation vers le bas d'un bloc de registres
SHIL	SHIft register Left	Décalage vers la gauche d'un registre
SHIR	SHIft register Right	Décalage vers la droite d'un registre
ROTL	ROTate register Left	Rotation vers la gauche d'un registre
ROTR	ROTate register Right	Rotation vers la droite d'un registre

LD LOAD (32-BIT VALUE)

Chargement d'une valeur de 32-bits

Description Le Registre, Temporisateur ou Compteur adressé est chargé avec la valeur 32-bit.

Temporisateurs et Compteurs :

- Le chargement est effectué **seulement si l'ACCU est High**.
- Des valeurs négatives ou en virgule flottante ne peuvent être chargées (seulement les valeurs décimales, Hex, ASCII ou binaires).
- Si un temporisateur est chargé, celui-ci démarre immédiatement.
- L'état d'un temporisateur ou d'un compteur est H quand il contient une valeur différente de zéro, sinon son état est L.

Registres :

- Le chargement est indépendant de l'état de l'ACCU.
- La valeur peut être décimale, binaire, hexadécimale, ASCII ou virgule flottante.

Les valeurs binaires sont terminées par un 'Q' ou 'Y'.
Les valeurs hexadécimales sont terminées par un 'H'.
Les valeurs en virgule flottante doivent contenir un point décimal '.' ou un exposant 'E'. Les valeurs ASCII sont entourées d'apostrophes: 'a', 'A'. 'A'.
Comme la valeur est 32-bits, trois lignes de programme sont utilisées pour cette instruction.

Les opérandes ne peuvent pas être donnés comme paramètres d'un Bloc de Fonction (FB).

Usage

```
LD[X]     élément (i)     ; R 0-4095, T 0-450, C 0-1599.
          valeur          ; Décimal: -2.147.483.648 à +2.147.483.647
                          ; Hex: 0H à FFFFFFFFH
                          ; Binaire: 0Y à 111...111Y (32 bits)
                          ; Floating Point: ±5.42101E-20 à ±9.22337E+18
                          ; ASCII: 'A'-'Z', '0'-'9', '!', '?' etc.
```

Exemple

```
LD      R 0       ; Charge R 0 avec la valeur "Floating Point" 321
          3.21E1  ; (Temporisateurs et Compteurs :
                  ;       seulement des valeurs positives)
```

Flags Inchangés.

Voir aussi LDH, LDL (chargement 16-bit), Constantes.

Note LD occupe 3 lignes de programme

LD T/C: seulement exécuté si l'ACCU = H (1)
LD R: toujours exécuté

LDL LOAD LOW WORD (LOWER 16 BITS)

Chargement d'une valeur de 16-bits (de poids le plus faible)

Description

Charge les 16 bits de poids le plus faible (0-15) d'un Registre, Temporisateur ou Compteur avec une valeur de 16 bits (0-65535); les 16 bits de poids le plus fort sont mis à 0.

Temporisateurs et Compteurs :
LDL est exécuté **seulement si l'ACCU est High**.

Registres :
LDL est **toujours exécuté**.

LDL (et LDH "Load High") permet le passage de constantes de 16-bit comme paramètres des blocs de fonction.
LDH charge les 16 bits de poids le plus fort.

En utilisant ces instructions, une valeur de 32-bit peut être chargée. Pour charger tous les 32 bits, LDL doit être exécuté en premier lieu car il positionne les 16 bits de poids le plus fort à 0.

Les valeurs peuvent être chargées en décimal, hex, binaire ou ASCII; PAS en virgule flottante.

Usage

```
LDL[X]    élément (i)    ; R 0-4095, T 0-450, C 0-1599
          valeur         ; Décimal: 0-65535
                         ; Hexadécimal: 0H-FFFFH
                         ; Binaire: 16 bits
```

Exemple

```
LDL  R 100      ; Charge le Registre 100 avec FFFF Hex,
       0FFFFH   ; valant 65535 en décimal.
                ; R100 = 0000FFFFH
```

Flags Inchangés.

Voir aussi LDH, LD, Constantes.

Note Lorsque seules des valeurs < 65535 sont utilisées, l'instruction LDL peut être utilisée en standard pour charger les Compteurs, Temporisateurs et Registres.

LDH LOAD HIGH WORD (UPPER 16 BITS)

Chargement d'une valeur de 16-bits (de poids le plus fort)

Description Charge les 16 bits de poids le plus fort (16-31) d'un Registre, les 16 bits de poids le plus faible sont inchangés. LDH (et LDL "Load Low") permet de passer des constantes de 16 bits comme paramètres des blocs de fonction.
En utilisant ces instructions, une valeur de 32-bit peut être chargée. Pour charger tous les 32 bits, LDL doit être exécuté en premier lieu car il positionne les 16 bits de poids le plus fort à 0.
Les valeurs peuvent être chargées en décimal, hex, binaire ou ASCII; PAS en virgule flottante.
LDH ne peut pas être utilisé pour charger les Temporisateurs ou les Compteurs, où les 16 bits de poids le plus fort ne peuvent pas être chargés séparément.

Usage

```
LDH[X]    élément (i)     ; R 0-4095
          valeur          ; Décimal 0-65535
                          ; Hexadécimal 0H-FFFFH
                          ; Binaire: 16 bits
```

Exemple

```
LDH   R 100       ; Charge les bits 31-16 du Registre 100
        0FFFFH    ; avec FFFF Hex.  (R 100 = FFFFxxxx Hex.)
```

Flags Inchangés.

Voir aussi LDL, LD, Constantes.

Pratique Pour charger un Registre dans un "Function Block" avec une constante de 32-bit, vous ne pouvez utiliser directement l'instruction LD; à la place, vous devez utiliser LDL et LDH.
(Les 16 bits de poids fort et faible d'une constante peuvent être séparés en utilisant les instructions de l'Assembleur '&' (AND) et '/' (DIVIDE))
Une constante (12345678) doit être passée comme paramètre d'un bloc de fonction où elle est chargée dans un Registre.
Souvenez vous que LDL doit être effectué AVANT LDH.

```
COB          0        ; En-tête du COB
             0
CFB          0        ; Appel du bloc de fonction 0
12345678 & 0FFFFH     ; Paramètre 1  (16-bits inférieurs)
12345678 / 0FFFFH     ; Paramètre 2  (16 bits supérieurs)
ECOB                  ; Fin du COB

FB           0        ; En-tête du FB
LDL        R 0        ; Charge les 16 bits inférieurs du Register 0
           = 1        ;      avec le 1er paramètre  (16 bits inférieurs)
LDH        R 0        ; Charge les 16 bits supérieurs du Register 0
           = 2        ;      avec le 2ème paramètre  (16 bits supérieurs)
EFB
```


DSP LOAD DISPLAY REGISTER
Chargement du registre d'affichage

Description

L'état logique d'une entrée, sortie ou flag, ou le contenu d'un Registre, Temporisateur ou Compteur, ou une constante peut être chargée dans le "Display Register".

Cette valeur est affichée en décimal sur la console PCD8.P100 ou sur l'afficheur du PCD2.

Ceci peut être utile pour afficher un code d'erreur ou un état (status).

L'opérande ne peut pas être donnée comme paramètre d'un Bloc de Fonction (FB).

Usage

```
DSP     valeur      ; I 0-8191, O 0-8191, F 0-8191, T 0-450,
                    ; C 0-1599, R 0-4095, K 0-16383
```

Exemple

```
DSP     R 0     ; Display register = contenu de R 0
DSP     K 1234  ; Display register = constante 1234
```

Flags Inchangés.

Voir aussi Manuel - Matériel de la série PCD1 et PCD2.

Pratique Voir l'instruction INC.

INC INCREMENT REGISTER OR COUNTER

Incrémentation (+1) d'un registre / compteur

Description Le Registre ou Compteur est incrémenté d'une unité.
Les **Compteurs** sont incrémentés **seulement si l'ACCU est High (1)**.
Les **Registres** sont incrémentés **indépendamment de l'état de l'ACCU**.

Usage

INC[X] élément (i) ; R 0-4095, C 0-1599

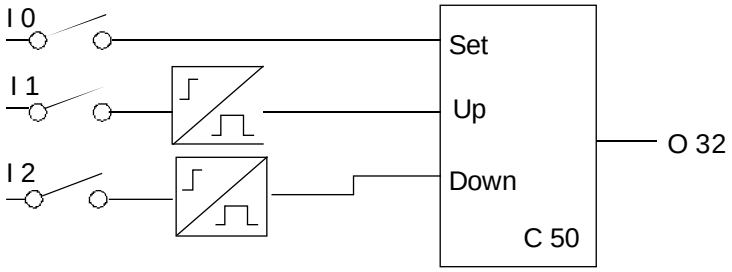
Exemple

```
INC     R 100   ; R 100 = R 100 + 1
```

Flags Les flags **Zéro** (Z), **Positif** (P) et **Négatif** (N) sont positionnés suivant le résultat.
Le flag **Error** (E) est enclenché dans le cas d'un dépassement de capacité.

Voir aussi DEC, ADD.

Pratique Compteur comptant/décomptant avec présélection et affichage de la valeur.




```
COB         0       ; En-tête du COB
            0

STH     I 0         ; Si l'entrée 0 est H
LD      C 50        ;     Alors chargement du C 50 avec 5
          5
                    ;     Sinon rien
STH     I 1         ; Si l'entrée 1 devient H
DYN     F 1         ;     (Détection du flanc)
INC     C 50        ;     Alors Compteur 50 + 1
                    ;     Sinon rien
STH     I 2         ; Si l'entrée 2 devient H
DYN     F 2         ;     (Détection du flanc)
DEC     C 50        ;     Alors Compteur 50 - 1
                    ;     Sinon rien
STH     C 50        ; Si le Compteur 50 > 0
OUT     O 32        ;     Alors enclenchement de la sortie 32
                    ;     Sinon déclenchement de la sortie 32

DSP     C 50        ; Affichage du Compteur 50

ECOB
```


DEC DECREMENT REGISTER OR COUNTER

Décrémentation (-1) d'un registre / compteur

Description Le Registre ou Compteur est décrémenté d'une unité.
Les **Compteurs** sont décrémentés **seulement si l'ACCU est High (1).**
Les **Registres** sont décrémentés **indépendamment de l'état de l'ACCU.**

Usage

DEC[X] élément (i) ; R 0-4095, C 0-1599

Exemple

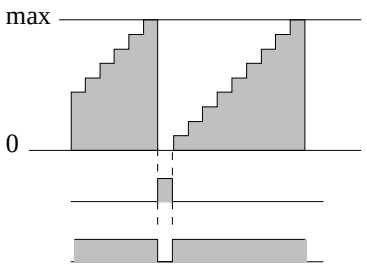
```
DEC     R 100   ; R 100 = R 100 - 1
```

Flags Les flags **Zéro** (Z), **Positif** (P) et **Négatif** (N) sont positionnés suivant le résultat.
Le flag **Error** (E) est enclenché dans le cas d'un dépassement de capacité.

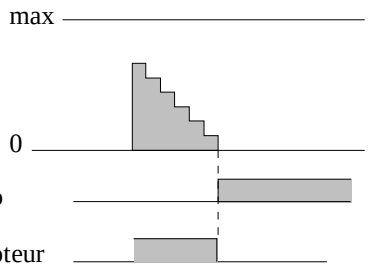
Voir aussi INC, SUB.

Pratique

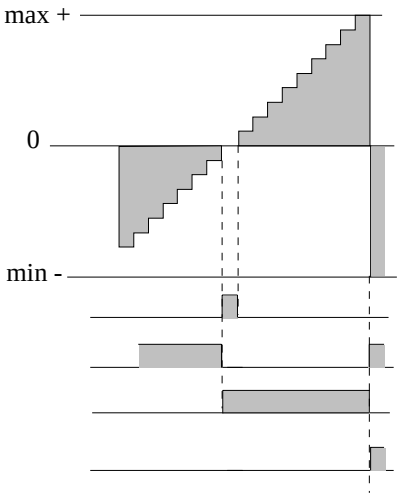
INC Compteur



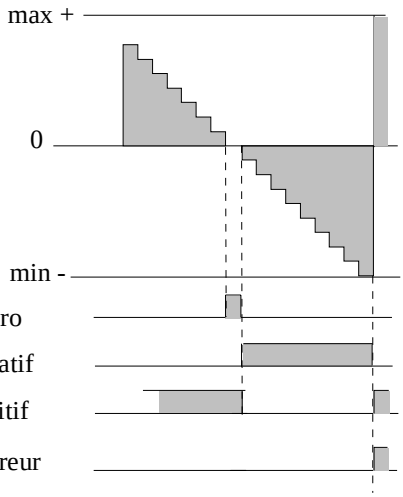
DEC Compteur



INC Registre



DEC Registre



SEI SET INDEX REGISTER

Chargement du registre d'index

Adressage Indexé :

Il est parfois nécessaire qu'une série d'entrées, sorties, flags, ... soient traités de la même manière (par exemple: remise à zéro de tous les flags ou registres). Dans un cas comme celui là, de longs programmes peuvent être réduits considérablement grâce à l'adressage indexé.
Chaque COB ou XOB possède son propre registre d'index (**Index register**).
Ce registre est utilisé pour l'adressage indexé: le contenu du Registre d'Index est ajouté à la valeur de l'opérande pour calculer l'adresse actuelle.
Les instructions indexées sont toujours terminées par un 'X', par exemple STHX, BITIX. Le Registre d'Index peut être chargé ou sauvé, incrémenté ou décrémenté jusqu'à une certaine valeur.

Description Le registre d'Index est chargé avec la constante (K 0-8191) ou avec le contenu du registre indiqué.

Note : Le Registre d'Index peut prendre des valeurs entre 0 et 8191 (13 bits). Si une valeur > 8191 est donnée, l'index est positionné sur 8191 et le XOB 12 est appelé.
Si une valeur < 0 est donnée, l'index est mis à 0 et le XOB 12 est appelé.

Usage

SEI	**valeur**	**; K 0-8191, R 0-4095**

Exemple

```
SEI     K 32    ; Charge le Registre d'Index avec 32
SEI     R 32    ; Charge le Registre d'Index avec le contenu du Registre 32
```

Flags Inchangés.

Voir aussi INI, DEI, STI, RSI.

Pratique L'état de l'entrée dont l'adresse est donnée par un codeur BCD doit être affiché sur la sortie 32.

```
COB       0
          0

DIGI      2     ; Lecture de 2 chiffres
        I 24    ;    A partir de l'entrée 24 (--> 31)
        R 500   ;    Et sauvegarde dans R 500
SEI     R 500   ; Charger l'Index avec le contenu de R 500
STHX    I 0     ; Si l'entrée (0 + Index) est H
OUT     O 32    ;    Alors enclenchement de la sortie 32
                ;    Sinon déclenchement de la sortie 32

ECOB
```


INI INCREMENT INDEX REGISTER

Incrémentation (+1) du registre d'index

Description Le Registre d'Index est comparé à la valeur de l'opérande (une constante de type K ou le contenu d'un registre)
Si le Registre d'Index est inférieur à cette valeur, le Registre d'Index est incrémenté et l'ACCU est mis à H (1).
Si le Registre d'Index est égal ou plus grand que la valeur de l'opérande, le Registre d'Index n'est pas incrémenté et l'ACCU est mis à L.
Si la valeur de l'opérande est > 8191 ou < 0, le XOB 12 est appelé.

Usage

INI	**valeur**	**; K 0-8191, R 0..4095**

Exemple

```
INI     K 100  ; Incrémente le Registre d'Index si il est inférieur à 100
INI     R 333  ; Incrémente le Registre d'Index si il est inférieur au
                   contenu du R 333.
```

Flags **ACCU** = 1 si le Registre d'Index est plus petit que la valeur de l'opérande.
ACCU = 0 si le Registre d'Index est plus grand ou égal à la valeur de l'opérande.

Voir aussi DEI, SEI.

Pratique A l'enclenchement, les registres 1500 à 1599 doivent être remis à zéro.

```
          XOB       16     ; XOB exécuté à la mise sous tension
          .....
          SEI       K 0    ; Positionne le Registre d'Index à 0
                           ; Répète
Reset:    LDX       R 1500 ; Charge le Registre (1500 + Reg Index)
                    0      ;     avec 0
          INI       K 499  ; Incrémente le Registre d'Index de 1
          JR        H Reset ;     Jusqu'à ce que le Registre d'Index > 499
          (ACC      H)
          .....
          EXOB
```


DEI DECREMENT INDEX REGISTER

Décrémentation (-1) du registre d'index

Description Le Registre d'Index est comparé à la valeur de l'opérande (une constante de type K ou le contenu d'un Registre).
Si le Registre d'Index est supérieur à cette valeur, le Registre d'Index est décrémenté et l'ACCU est mis à H.
Si le Registre d'Index est égal ou plus petit que la valeur de l'opérande, le Registre d'Index n'est pas décrémenté et l'ACCU est mis à L.
Si la valeur de l'opérande est > 8191 ou < 0, le XOB 12 est appelé.

Usage

DEI	**valeur**	**; K 0-8191, R 0..4095**

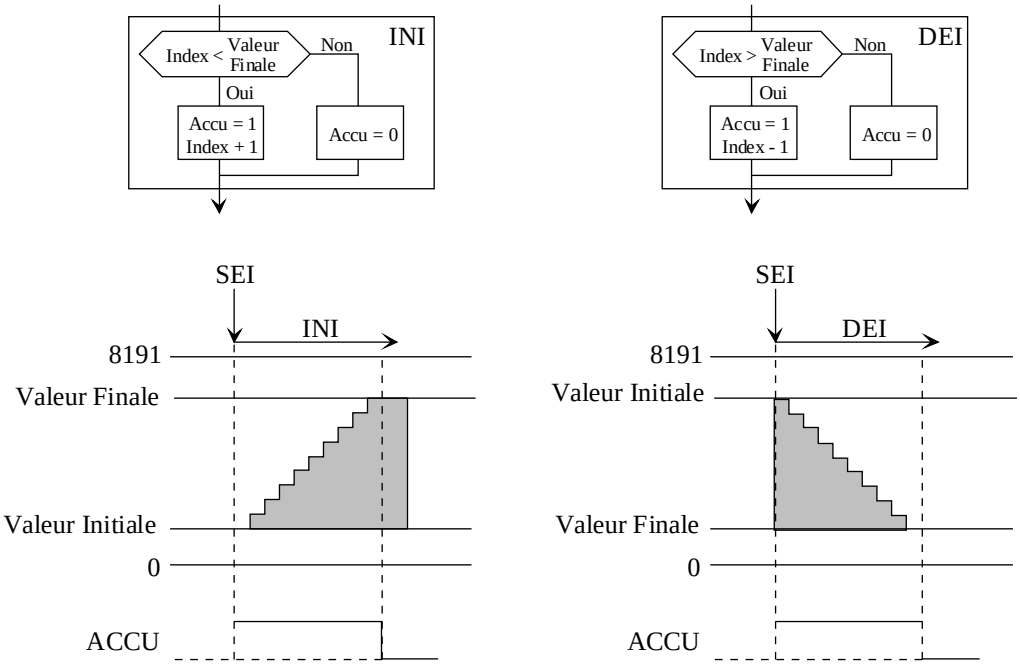
Exemple

```
DEI     K 100  ; Décrémente le Registre d'Index
               ;      si il est plus grand que la constante 100.
DEI     R 444  ; Décrémente le Registre d'Index
               ;      si il est plus grand que le contenu du R 444.
```

Flags **ACCU = 1** si le Registre d'Index est plus grand que la valeur de l'opérande.
ACCU = 0 si le Registre d'Index est plus petit ou égal à la valeur de l'opérande.

Voir aussi INI, SEI.

Pratique



Valeur Finale = valeur de l'opérande de l'instruction INI / DEI

STI STORE INDEX REGISTER
Mémorisation du registre d'index

Description La valeur du Registre d'Index est mémorisée dans le Registre de l'opérande.
Cette valeur peut être rechargée dans le Registre d'Index avec l'instruction RSI.
Le Registre d'Index n'est pas modifié.

Usage

STI	**reg**	**; R 0-4095**

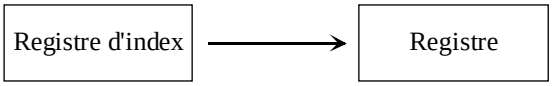
Exemple

```
STI     R 100   ; Mémorise le contenu du Registre d'Index dans le
                  Registre 100
```

Flags Inchangés.

Voir aussi RSI.

Pratique



RSI RESTORE INDEX REGISTER
Restitution du registre d'index

Description Charge le Registre d'Index avec le contenu du Registre donné dans l'opérande.

La valeur du Registre est en général une valeur d'un Registre d'Index ayant été sauvée par l'instruction STI.

La valeur maximum est 8191, seuls les 13 bits de poids le plus faible seront chargés.

Si un paramètre <0 ou > 8191 est entré: le XOB 12 est appelé.

Usage

RSI	**reg**	**; R 0-4095**

Exemple

```
RSI     R 100  ; Charge le Registre d'Index avec le contenu du
                 Registre 100.
```

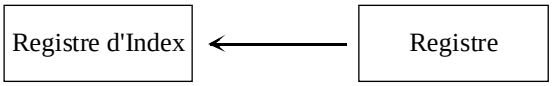
Ceci est identique à :

```
SEI     R 100
```

Flags Inchangés.

Voir aussi STI, SEI.

Pratique



MOV MOVE DATA

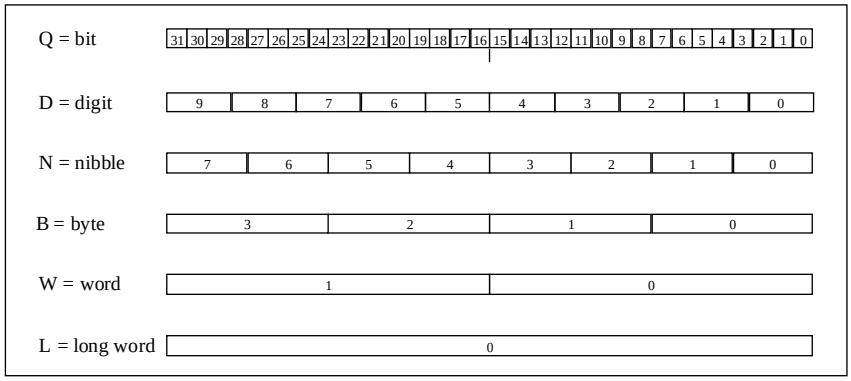
Déplacement de données

Description Copie des données d'un Temporisateur, Compteur ou Registre dans un Registre. C'est une instruction en 4 lignes :

- le premier et le troisième opérande sont la source et la destination.
- le second et quatrième opérande sont le type de données et la position :

Q = Bit (copie 1 bit)	0-31
D = Digit (4 Bits BCD)	0-9
N = Nibble (4 Bits Binaires)	0-7
B = Byte (8 Bits)	0-3
W = Word (16 Bits)	0/1
L = Long word (32 Bits)	0

Le type de données (Q, D, etc.) de la 2ème et 4ème opérande doivent être identiques; cependant, la position de la source et de la destination peuvent être différentes.



Usage

```
MOV[X]   source     (i)    ; R 0-4095, T 0-450, C 0-1599
         type position     ; Q|D|N|B|W|L  voir plus haut
         dest       (i)    ; R 0-4095
         type position     ; Q|D|N|B|W|L  voir plus haut
```

Exemple Copie le nibble de pois le plus fort (N7) du Registre 100 dans le nibble de poids le plus faible du Registre 101

```
; Avant :  R100:  1111 1010 1010 1010 1010 1010 1010 1010
           R101:  0001 0001 0001 0001 0001 0001 0001 0001
           MOV    R 100
                  N 7
                  R 101
                  N 0

; Après :  R100:  1111 1010 1010 1010 1010 1010 1010 1010
           R101:  0001 0001 0001 0001 0001 0001 0001 1111
```

Flags Inchangés.

Voir aussi COPY, GET, PUT, LD, LDH, LDL.

COPY COPY DATA
Copie de données

Description Les instructions COPY, GET et PUT ont la même fonction.
Elles sont utilisées pour le transfert indexé de données entre Registres, Compteurs et Temporisateurs. Dans chaque cas, l'entièreté du Registre, Compteur ou Temporisateur est copié.

Pour **COPYX**, le contenu du premier opérande est copié dans le second, les deux sont indexés.

Usage

COPY[X]	**source**	**(i)**	**; R 0-4095, T 0-450, C 0-1599**
	destination	**(i)**	**; R 0-4095, T 0-450, C 0-1599**

Exemple

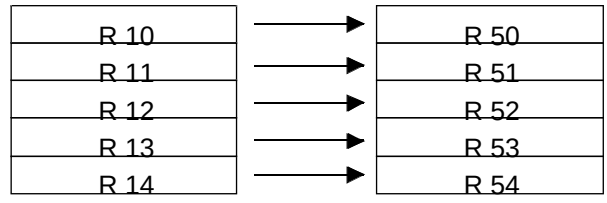
```
COPYX     R 10    ; Copie le contenu de R (10 + Index)
          R 50    ;      dans R  (50 + Index)
```

Flags Les flags **Zéro** (Z) et **Positif** ou **Négatif** sont positionnés suivant la valeur qui est copiée.

Voir aussi GET, PUT, MOV.

Pratique

```
      SEI     K 0
loop: COPYX   R 10
              R 50
      ...
      INI     K 4
      JR      H loop
```



GET GET DATA
Transfert de données

Description Les instructions COPY, GET et PUT ont la même fonction.
Elles sont utilisées pour le transfert indexé de données entre Registres, Compteurs et Temporisateurs. Dans chaque cas, l'entièreté du Registre, Compteur ou Temporisateur est copié.

Pour **GETX**, le contenu du premier opérande est copié dans le second, seul le premier opérande est indexé.

En plus, l'instruction GET permet le transfert de Textes / Data Blocks vers les Registres / Temporisateurs / Compteurs.

Usage

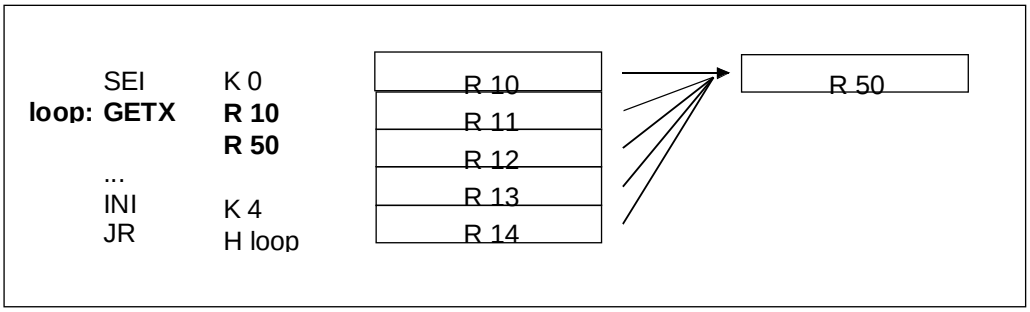
GET[X] source (i) ; R 0-4095, T 0-450, C 0-1599, X 0-3999, DB 0-3999
destination ; R 0-4095, T 0-450, C 0-1599

Exemple

```
GETX    R 10   ; Copie le contenu du R  (10 + index)
        R 50   ;     dans R 50
```

Flags Les flags **Zéro** (Z) et **Positif** ou **Négatif** sont positionnés suivant la valeur qui est copiée.

Voir aussi PUT, COPY, MOV.



Transfert entre Texte / Data Block et R/T/C :

L'instruction GET[X] peut transférer le Texte référencé dans des R/T/C consécutifs jusqu'à la fin du Texte (00 - NUL). Si le Texte ne se termine pas sur le remplissage total d'un R/T/C, alors le reste du R/T/C sera inchangé.
De la même façon, GET[X] peut transférer des données d'un Data Block dans des R/T/C consécutifs jusqu'à la fin du Data Block.
Un DATA BLOCKest une zone de la mémoire utilisateur où un grand nombre de Registres, Temporisateurs ou Compteurs peuvent être sauvegardés et rechargés par la suite lors du déroulement du programme. Les Data Blocks peuvent être utilisés pour mémoriser des valeurs spécifiques à un processus donné pour libérer les R/T/C lors du déroulement d'un autre processus.
Si l'instruction essaie de lire d'un Texte ou d'un Data Block qui n'existe pas, le XOB 13 (Error flag set) est appelé. Si le texte indexé ou le numéro du Data Block est supérieur à 3999, le XOB 12 est appelé (Index Register Overflow).

Formats des DATA BLOCKs et TEXTes :

Tous les Data Blocs et Textes doivent être définis dans le programme source de la manière suivante :

DB numéro '[' [longueur] ']' [valeur 1 [,valeur 2]...[,valeur n]]

TEXT numéro '[' [longueur] ']' "texte ligne 1" [;commentaire]
["texte ligne 2" [;commentaire]]
["texte ligne n" [;commentaire]]

où numéro, longueur peuvent être	décimal
	symbole
et valeur 1, ... peut être	décimale
	hexadécimale
	binaire
	virgule flottante (floating point)
	ASCII
	symbole

Les Textes et les Data Blocks sont numérotés de 0..3999 mais parce qu'ils partagent la même zone mémoire, ils ne peuvent avoir le même numéro. Par exemple, si un Data Block 10 est défini, le Texte 10 n'est plus disponible. Un Texte peut être défini comme normalement avec une longueur maximale possible de 3069 caractères.

Un Data Block peut contenir un maximum de 383 données, où une donnée peut être un Registre ou Temporisateur ou Compteur.

Note : Il est interdit d'avoir <254> ou <255> dans le premier byte d'un Texte normal. Si le PCD contient des EPROMs: les Data Blocs et Textes peuvent seulement être lus.

Exemples :

```
DB      100     [10]                ; Data Block numéro 100 de 10 données
                                    ; toutes les valeurs sont initialisées à zéro

DB      101     [] 1.0,konst,3.0,4  ; Data Block numéro 101 de 4 données
                                    ; Parmi lesquelles deux sont en virgule
                                    ;     flottante et une est un symbole

DB      102     [4] 1,2,3           ; Data Block n° 102 de 4 données 1, 2, 3, 0

DB      num     [data_len]          ; Data Block avec des symboles prédéfinis

TEXT    100     "TEXTE NORMAL"
TEXT    103     [5] "123"           ; Texte 103 qui est "123  " (avec 2 espaces)
```


GET Data Block /Texte :

Exemple 1 :

Data Block déclaré dans le programme source :

```
DB  100     [5] 0h,1h,2h,0a5a5a5a5h,720h
```

Instruction :

```
GET    DB 100      ; Transfert du Data Block 100
       R  1000     ;    dans le Registre 1000 et suivants
```

Résultat :

Registre	Valeur en Hex
1000	00000000
1001	00000001
1002	00000002
1003	a5a5a5a5
1004	00000720

Exemple 2 :

Texte déclaré dans le programme source :

```
TEXT    100          "THIS IS A TEXT 123"
```

Instruction :

```
GET    X  100      ; Transfert du Texte 100
       R  1000     ;    dans le Registre 1000 et suivants
```

Résultat :

(Les registres R 1000 à R 1004 étaient égal à zéro avant l'instruction)

Registre	Valeur ASCII	Valeur en Hex
1000	THIS	54484953
1001	IS	20495320
1002	A TE	41205445
1003	XT 1	58542031
1004	23	32332020

PUT PUT DATA
Transfert de données

Description Les instructions COPY, GET et PUT ont la même fonction.
Elles sont utilisées pour le transfert indexé de données entre Registres, Compteurs et Temporisateurs. Dans chaque cas, l'entièreté du Registre, Compteur ou Temporisateur est copié.
Pour **PUTX**, le contenu du premier opérande est copié dans le second, seul le second opérande (destination) est indexé.
En plus, l'instruction PUT permet le transfert des Registres, Temporisateurs, Compteurs vers les Textes et Data Blocks.

Usage

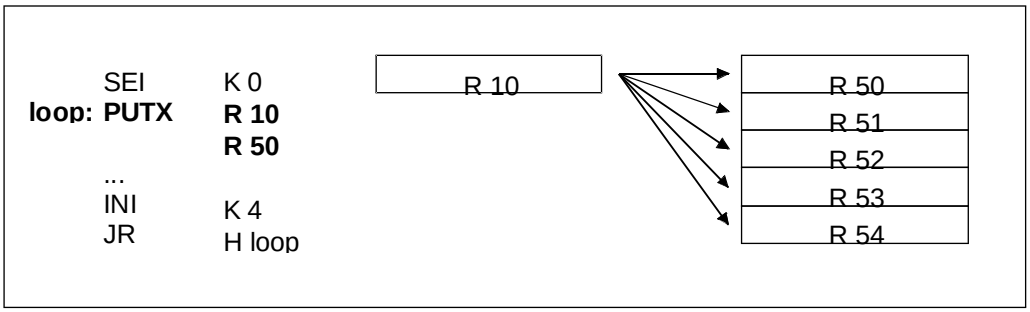
```
PUT[X]  source          ; R 0-4095, T 0-450, C 0-1599
        destination (i) ; R 0-4095, T 0-450, C 0-1599, X 0-3999, DB 0-3999
```

Exemple

```
PUTX    R 10    ; Copie le contenu du Registre 10
        R 50    ;    dans le Registre  (50 + Index)
```

Flags Les flags **Zéro** (Z) et **Positi**f ou **Négatif** sont positionnés suivant la valeur qui est copiée.

Voir aussi GET, COPY, MOV.



Transfert entre R/T/C et Text/Data Block :

L'instruction PUT[X] copie le contenu de R/T/C en commençant par le R/T/C référencé dans le Texte indiqué jusqu'à la fin du texte. Si un des R/T/C contient le caractère NUL (00h), l'instruction PUT changera sa valeur en un espace.
De la même façon, PUT[X] copie le contenu de R/T/C en commençant par le R/T/C dans le Data Block référencé jusqu'au moment ou celui-ci est rempli.
Un DATA BLOCK est une zone de la mémoire utilisateur où un grand nombre de Registres, Temporisateurs ou Compteurs peuvent être sauvegardés et re-chargés par la suite lors du déroulement du programme. Les Data Blocks peuvent être utilisés pour mémoriser des valeurs spécifiques à un processus donné pour libérer les R/T/C lors du déroulement d'un autre processus.
Si l'instruction essaie de lire d'un Texte ou d'un Data Block qui n'existe pas, le XOB 13 (Error flag set) est appelé.
Si l'instruction essaie de copier d'un R/T/C qui n'existe pas, le XOB 13 (Error flag set) est appelé. Si le texte indexé ou le numéro du Data Block est supérieur à 3999, le XOB 12 est appelé (Index Register Overflow).

PUT dans un Data Block /Texte :

Note : PUT ne peut étendre la longueur du texte ou du bloc de données. PUT ne peut transférer des valeurs dans un texte ou dans un bloc de données si des EPROMs ou des RAMs avec le pont en position "WP" sont utilisées.

Exemple 1 :

Data Block déclaré dans le programme source :

```
DB  100     [5]     ; Valeurs initiales = zéro
```

Contenu des Registres :

Registre	Valeur en Dec
1000	00000001
1001	00000002
1002	00000003
1003	01234567
1004	00000720

Instruction :

```
PUT     R 1000 ; Transfert du Registre 1000 et suivants
        DB 100 ;    dans le Data Block 100
```

Résultat comme affiché par le Debugger en représentation décimale :

```
DB 100 (0):   1    2    3    1234567    720
```

Exemple 2 :

Texte déclaré dans le programme source :

```
TEXT  100    [17]    ; Un texte de 17 espaces
```

Contenu des Registres :

Registre	Valeur ASCII	Valeur en Hex
1000	THIS	54484953
1001	IS	20495320
1002	A TE	41205445
1003	XT 1	58542031
1004	23	32332020

Instruction :

```
PUT     R 1000 ; Transfert des Registres 1000 et suivants
        X 100  ;    dans le Texte 100
```

Résultat comme affiché par le Debugger :

```
   TEXT 100: "THIS IS A TEXT 12"
```

(Le texte se termine après 17 caractères)

TFR TRANSFER DATA
Transfert de données

Description Cette instruction permet le transfert indexé de données ou de valeurs individuelles d'un Data Block ou d'un texte vers les Registres, Temporisateurs ou Compteurs; et inversement.

Usage **TRANSFERT DATA BLOCK (ou Texte) ⟶ R, T, C**
Copie une valeur (32 bits) d'un Data Block ou Texte vers un Registre, un Temporisateur ou un Compteur.

```
TFR[X]    source          ; DB (ou X) 0..3999, 4000..7999
          position        ; R 0..4095, K 0..382, K 0..16383
          dest     (i)    ; R 0..4095, T|C 0..1599
```

Le 1er opérande est le Data Block (ou Texte) contenant la valeur à transférer.
Le 2ème opérande est la position de la valeur à l'intérieur du Data Block (ou Texte); cette position peut être donnée comme une constante ou indirectement via un Registre.
Le 3ème opérande est la destination du Registre, Temporisateur ou Compteur.

TRANSFERT R, T, C ⟶ DATA BLOCK (ou Texte)
Copie un Registre, Temporisateur ou Compteur dans un Data Block (ou Texte) :

```
TFR[X]    source     (i)  ; R 0..4095, T|C 0..1599
          destination     ; DB (ou X) 0..3999, 4000..7999
          position        ; R 0..4095, K 0..382, K 0..16383
```

Le 1er opérande est un Registre, Temporisateur ou Compteur contenant la valeur à transférer (source).
Le 2ème opérande est la destination du Data Block (ou Texte).
Le 3ème opérande est la position dans le Data Block (ou Texte) où la valeur doit être transférée, la position peut être donnée comme une constante ou via un Registre.

Remarque :

La longueur d'un Data Block est dépendante du type de mémoire du PCD :

Mémoire	**Adresse**	**Longueur maximum DB**
Standard	DB 0..3999	383 valeurs
Etendue	DB 4000..7999	16383 valeurs

Exemple

```
TFR     DB  4010    ; Copie du Data Block 4010
        K   13      ;    La valeur à la position 13
        R   26      ;    Vers le Registre 26
TFR     R   120     ; Copie le Registre 120
        DB  4025    ;    Vers le Data Block 4025
        K   6       ;    A la position 6
```

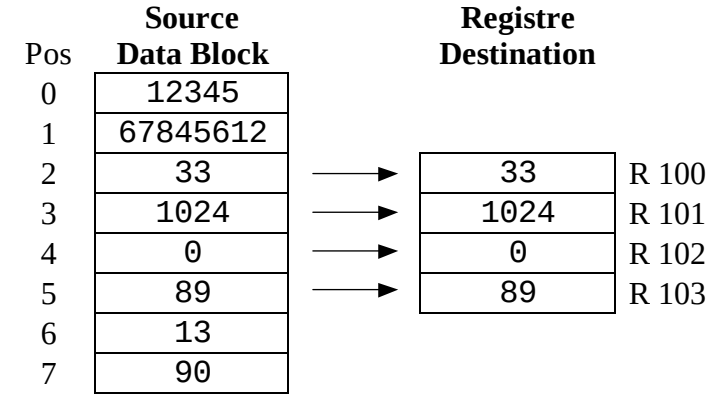

Flags Les flags **Zero** (Z) et **Signe** (P ou N) sont positionnés suivant la valeur copiée.

Voir aussi PUT, GET.

Notes Pour des raisons d'organisation mémoire, l'accès aux DBs 4000..7999 est plus rapide que celui aux DBs 0..3999. Il est dès lors recommandé d'utiliser cette instruction principalement avec les DBs 4000..7999.

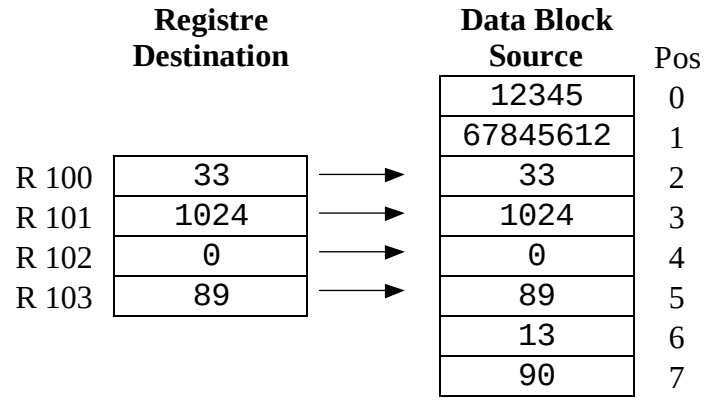
Pratique Les 4 valeurs (position 2..5) du Data Block 4010 sont copiées vers les Registres 100..103.

```
….
LD      R   999
            2
SEI     K   0
LOOP:   TFRX    DB  4010
        R   999
        R   100
INC     R   999
INI     K   3
JR      H   LOOP
…
```



Les Registres 100..103 sont copiés aux positions 2..5 du Data Block 4010 :

```
….
LD      R   999
            2
SEI     K   0
LOOP:   TFRX    R   100
        DB  4010
        R   999
INC     R   999
INI     K   3
JR      H   LOOP
…
```



TFRI TRANSFER DATA INDIRECT
Transfert indirect de données

Description Cette instruction permet le transfert indirect de valeurs individuelles d'un Data Block ou Texte vers les Registres, Temporisateurs ou Compteurs; et inversement.

Cette instruction ne fonctionne pas en mode paramétrisé.

Usage **TRANSFERT DATA BLOCK (ou Texte) ⟶ R, T, C**
Copie une valeur (32 bits) d'un Data Block ou Texte vers un Registre, un Temporisateur ou un Compteur.

```
TFRI    source        ; DB ou X   x
        position      ; R 0..4095, K 0..382, K 0..16383
        destination   ; R ou T|C  y
```

Le 1er opérande spécifie que la source est un Data Block ou un Texte; la variable x est le numéro d'un Registre contenant l'adresse du DB ou Texte.
Le 2ème opérande est la position de la valeur à l'intérieur du Data Block (ou Texte); cette position peut être donnée comme une constante ou indirectement via un Registre.
Le 3ème opérande spécifie le type de la destination (R ou T|C). La variable y est un numéro de Registre contenant l'adresse de destination du média.

TRANSFERT R, T, C ⟶ DATA BLOCK (ou Texte)
Copie un Registre, Temporisateur ou Compteur dans un Data Block (ou Texte) :

```
TFRI    source        ; R ou T|C  x
        destination   ; DB ou X   y
        position      ; R 0..4095, K 0..382, K 0..16383
```

Le 1er opérande spécifie que la source est un R ou T|C; la variable x est le numéro d'un Registre contenant l'adresse du média.
Le 2ème opérande spécifie le type de la destination (DB ou Texte); la variable y est un numéro de Registre contenant l'adresse de destination du média.
Le 3ème opérande est la position de la valeur à l'intérieur du Data Block (ou Texte); cette position peut être donnée comme une constante ou indirectement via un Registre.

Remarque :

La longueur du Data Block est dépendante du type de mémoire du PCD :

Mémoire	**Adresse**	**Longueur maximum DB**
Standard	DB 0..3999	383 valeurs
Etendue	DB 4000..7999	16383 valeurs

Exemples Transfert l'élément de la position 10 du Data Block 4000 vers le Registre 4095.

```
LD      R  100      ; Initialisation de l'adresse du DB
           4000
LD      R  101      ; Initialisation de l'adresse du Registre
           4095
TFRI    DB 100      ; Transfert DB
        K  10
        R  101
```

Transfert le Compteur 1000 à la position 50 du Data Block 4000

```
LD      R  100      ; Initialisation de l'adresse du DB
           4000
LD      R  101      ; Initialisation de la position
           50
LD      R  102      ; Initialisation de l'adresse du Compteur
           4095
TFRI    C  102      ; Transfert Compteur
        DB 100
        R  101
```

Flags Les flags **Zero** (Z) et **Signe** (P ou N) sont positionnés suivant la valeur copiée.

Voir aussi TFR, PUT, GET.

Notes Pour des raisons d'organisation mémoire, l'accès aux DBs 4000..7999 est plus rapide que celui aux DBs 0..3999. Il est dès lors recommandé d'utiliser cette instruction principalement avec les DBs 4000..7999.

BITI BIT IN

Lecture d'une valeur binaire

Description Copie un nombre de bits binaires d'une suite d'entrées, sorties, flags, ou d'un Temporisateur ou d'un Compteur dans un Registre.

Le 1er opérande est le nombre de bits à copier (1-32).
Le 2ème opérande est la source (I, O, F, T ou C).
Le 3ème opérande est le Registre de destination.

Si la source est des entrées, sorties ou flags: l'adresse donnée est l'élément le plus bas de la suite.
L'élément le plus bas devient le bit de poids le plus faible dans le Registre de destination (LEAST SIGNIFICANT).

C'est le contraire du SAIA PCA.

Usage

```
BITI[X]   bits          ; Nombre de bits à lire 1-32
          source        ; Adresse de la source I, O, F, T, C
          dest    (i)   ; Registre de destination R 0-4095
```

Exemple

```
BITI       16   ; Lit 16 bits
         I 32   ;    des entrées 32-47 et
         R 0    ;    mémorisation dans le Registre 0  (bits 0-15)
```

Flags Les flags **Zéro** (Z) et **Positif** ou **Négatif** sont positionnés suivant la valeur lue.

Voir aussi BITIR, DIGI, DIGIR.

Pratique Quand l'entrée 8 devient haute, une valeur de 8 bits binaires est lue des entrées 0 à 7 et mémorisée dans le Registre 500.

```
          COB        0     ; En tête du COB
                     0
          .....
          STH      I 8     ; Si l'entrée 8 devient Haute
          DYN      F 100
          JR       L NEXT

          BITI       8     ; Alors lire 8 bits binaires
                   I 0     ;    à partir de l'entrée 0  (jusqu'à 7)
                   R 500   ;    et mémorisation dans R 500

NEXT:     ....

          ECOB
```


BITIR BIT IN REVERSED

Lecture inversée d'une valeur binaire

Description Copie un nombre de bits binaires d'une suite d'entrées, sorties, flags, ou d'un Temporisateur ou d'un Compteur dans un Registre.

Le 1er opérande est le nombre de bits à copier (1-32).
Le 2ème opérande est la source (I, O, F, T ou C).
Le 3ème opérande est le Registre de destination.

Si la source est des entrées, sorties ou flags: l'adresse donnée est l'élément le plus bas de la suite.
L'élément le plus bas devient le bit de poids le plus fort dans le Registre de destination (MOST SIGNIFICANT).

C'est identique au SAIA PCA.

Usage

```
BITIR[X]   bits          ; Nombre de bits à lire 1-32
           source        ; Adresse de la source I, O, F, T, C
           dest    (i)   ; Registre de destination R 0-4095
```

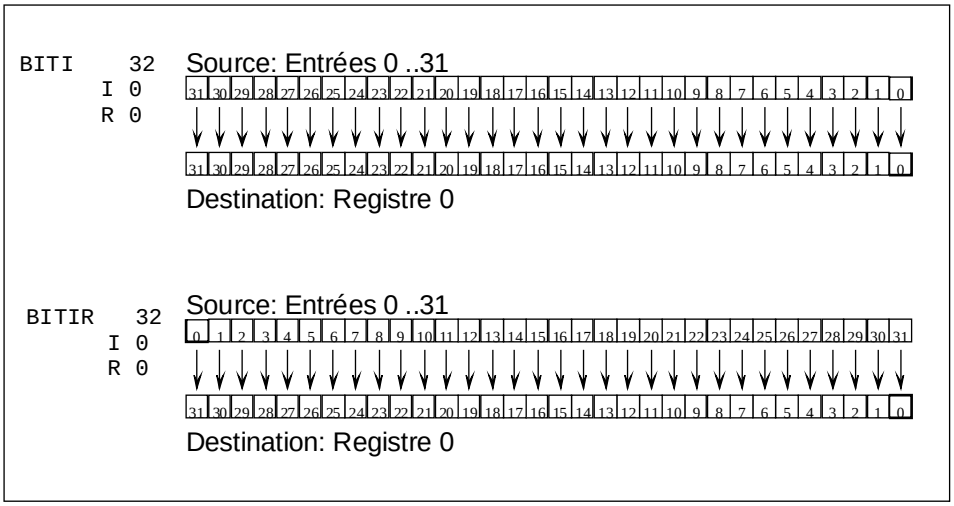
Exemple

```
BITIR      16   ; Lit 16 bits
         I 32   ;     des entrées 32-47 et
         R 0    ;     mémorisation dans le Registre 0  (bits 15-0)
```

Flags Les flags **Zéro** (Z) et **Positif** ou **Négatif** sont positionnés suivant la valeur lue.

Voir aussi BITI, DIGI, DIGIR.

Pratique



BITO BIT OUT

Sortie d'une valeur binaire

Description Copie un nombre de bits binaires d'un Registre vers une suite de sorties, flags ou bits d'un Temporisateur ou d'un Compteur.

Le 1er opérande est le nombre de bits à copier (1-32).
Le 2ème opérande est l'adresse du Registre source.
Le 3ème opérande est l'adresse de destination des sorties, flags, Temporisateur ou Compteur.

Si la destination est des sorties ou flags, l'adresse donnée est celle de l'élément le plus bas de la suite.
Le bit de poids le plus faible du Registre est copié dans l'élément le plus bas.

Ceci est le contraire du SAIA PCA.

Usage

```
BITO[X]   bits          ; Nombre de bits à copier 1-32
          source  (i)   ; Registre source R 0-4095
          dest          ; Adresse de destination  O, F, T, C
```

Exemple

```
BITO      8     ; Copie 8 bits
        R 10    ;    du Registre 10  (bits 0-7)
        O 48    ;    vers les sorties 48-55
```

Flags Inchangés.

Voir aussi BITOR, DIGO, DIGOR.

Pratique Copier l'état des entrées 0 à 15 sur les sorties 32 à 47

```
COB       0     ; En tête du COB
          0

BITI      16    ; Lecture de 16 bits
        I 0     ;    des entrées 0 (à 15)
        R 0     ;    mémorisation dans le R 0

BITO      16    ; Ecriture de 16 bits
        R 0     ;    de R 0
        O 32    ;    vers les sorties 32 (à 47)

ECOB
```


BITOR BIT OUT REVERSED

Sortie inversée d'une valeur binaire

Description

Copie un nombre de bits binaires d'un Registre vers une suite de sorties, flags ou bits d'un Temporisateur ou d'un Compteur.

Le 1er opérande est le nombre de bits à copier (1-32).
Le 2ème opérande est l'adresse du Registre source.
Le 3ème opérande est l'adresse de destination des sorties, flags, Temporisateur ou Compteur.

Si la destination est des sorties ou flags, l'adresse donnée est celle de l'élément le plus bas de la suite.
Le bit de poids le plus faible du Registre est copié dans l'élément le plus haut.

Ceci est identique au SAIA PCA.

Usage

```
BITOR[X]  bits          ; Nombre de bits à copier 1-32
          source  (i)   ; Registre source R 0-4095
          dest          ; Adresse de destination O, F, T, C
```

Exemple

```
BITOR     8     ; Copie 8 bits
        R 10    ;     du Registre 10  (bits 0-7)
        O 48    ;     vers les sorties 55-48
```

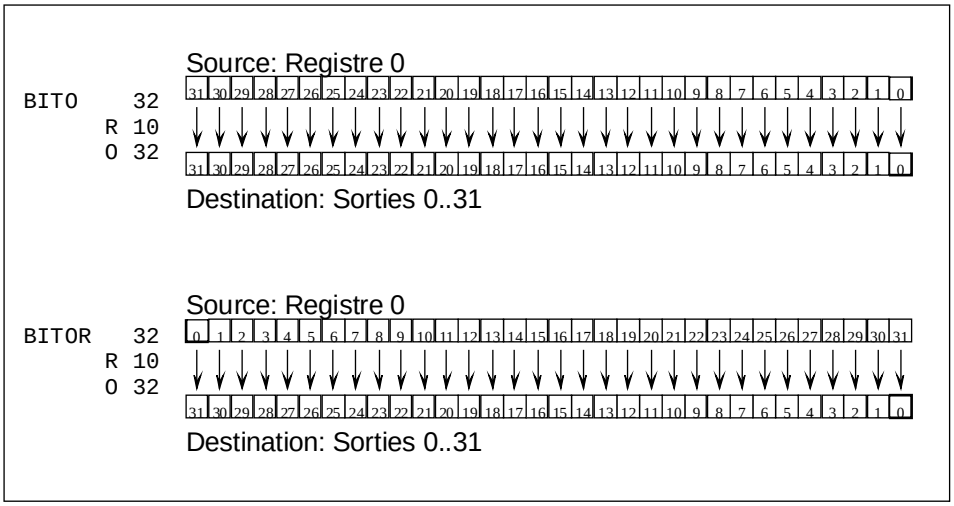
Flags

Inchangés.

Voir aussi

BITO, DIGO, DIGOR.

Pratique



DIGI DIGIT IN
Lecture d'une valeur BCD

Description Transfert de chiffres BCD ("Binary Coded Decimal") des entrées, sorties ou flags dans un Registre.
Un chiffre BCD est un ensemble de 4 bits (ex. 4 entrées) qui représente un chiffre décimal (0-9).

Le 1er opérande est le nombre de chiffre à transférer (1-10).
Le 2ème est l'adresse des entrées, sorties ou flags.
Le 3ème opérande est le Registre de destination.

L'entrée, sortie ou flag avec l'adresse la plus basse devient le bit de poids le plus faible du chiffre le moins significatif du Registre de destination.

Ceci est le contraire du SAIA PCA.

Usage

```
DIGI[X]   digits          ; Nombre de chiffres BCD 1-10
          source          ; Elément source I 0-8191, O 0-8191, F 0-8191
          dest    (i)     ; Registre de destination R 0-4095
```

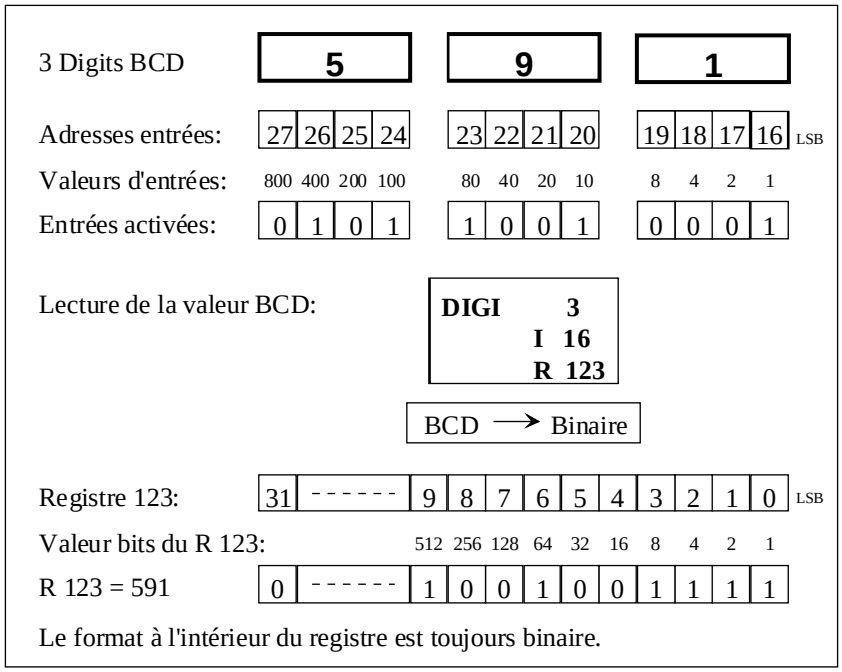
Exemple

```
DIGI      2     ; Lecture de 2 chiffre BCD
        I 32    ;     des entrées 39-36 et 35-32
        R 100   ;     dans le Registre 100
```

Flags Les flags **Zéro** (Z) et **Positif** ou **Négatif** sont positionnés suivant la valeur lue.

Voir aussi DIGIR, DIGO, DIGOR, BITI, BITIR.

Pratique



DIGIR DIGIT IN REVERSED

Lecture inversée d'une valeur BCD

Description Transfert de chiffres BCD ("Binary Coded Decimal") des entrées, sorties ou flags dans un Registre.
Un chiffre BCD est un ensemble de 4 bits (ex. 4 entrées) qui représente un chiffre décimal (0-9).

Le 1er opérande est le nombre de chiffre à transférer (1-10).
Le 2ème est l'adresse des entrées, sorties ou flags.
Le 3ème opérande est le Registre de destination.

L'entrée, sortie ou flag avec l'adresse la plus basse devient le bit de poids le plus fort du chiffre le plus significatif du Registre de destination.

Ceci est identique au SAIA PCA.

Usage

```
DIGIR[X]  digits          ; Nombre de chiffres BCD 1-10
          source          ; Elément source I 0-8191, O 0-8191, F 0-8191
          dest      (i)   ; Registre de destination R 0-4095
```

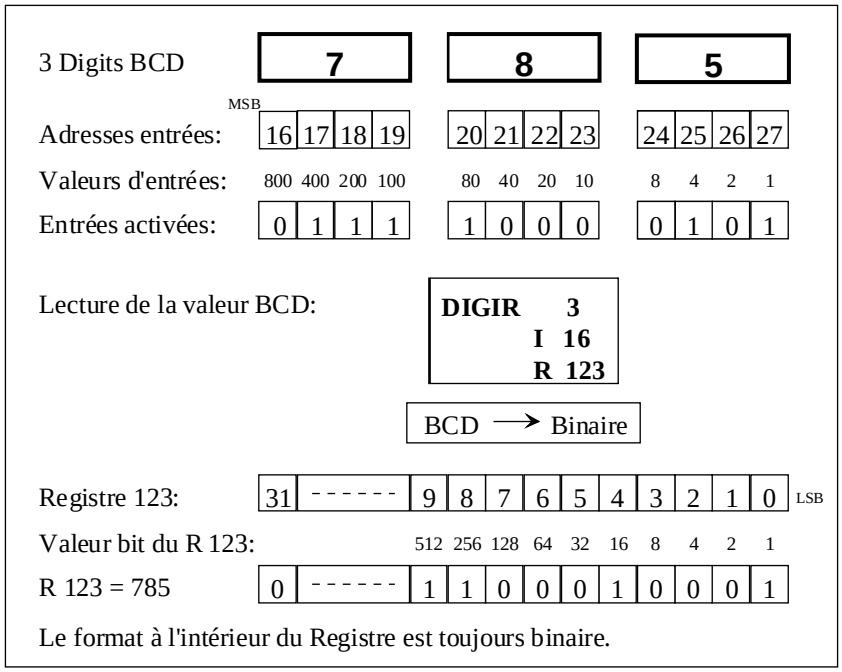
Exemple

```
DIGIR     2     ; Lecture de 2 chiffres BCD
        I 32    ;     des entrées 32-35 et 36-39
        R 100   ;     dans le Registre 100
```

Flags Les flags **Zéro** (Z) et **Positif** ou **Négatif** sont positionnés suivant la valeur lue.

Voir aussi DIGI, DIGO, DIGOR, BITI, BITIR.

Pratique



DIGO DIGIT OUT
Sortie d'une valeur BCD

Description Transfert de chiffres BCD d'un Registre vers une suite de sorties ou flags.
Un chiffre BCD consiste en 4 bits binaires.
Le 1er opérande est le nombre de chiffres à transférer.
Le 2ème opérande est le Registre source.
Le 3ème opérande est l'adresse de base des sorties ou flags.
L'adresse de la sortie ou flag la plus basse devient le bit de poids le plus faible du chiffre BCD le moins significatif.

Ceci est le contraire du SAIA PCA.

Usage

```
DIGO[X]   digits          ; Nombre de chiffres BCD 1-10
          source   (i)    ; Registre source R 0-4095
          dest            ; Elément de destination O 0-8191, F 0-8191
```

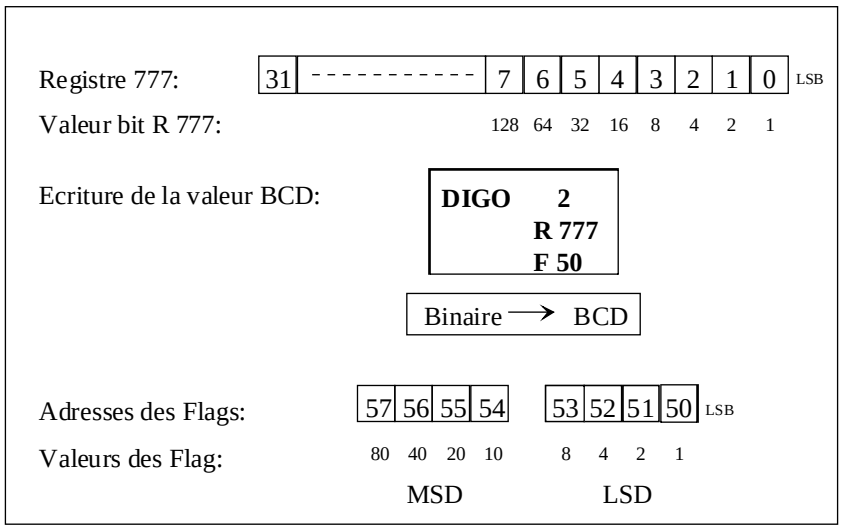
Exemple

```
DIGO      2     ; Transfert de 2 chiffres BCD
        R 123   ;     du Registre 123
        O 40    ;     vers les sorties 47-44 et 43-40
```

Flags Le flag **Error** (E) est positionné si un chiffre BCD est invalide (> 9).

Voir aussi DIGOR, DIGI, DIGIR, BITO, BITOR.

Pratique



DIGOR DIGIT OUT REVERSED

Sortie inversée d'une valeur BCD

Description Transfert de chiffres BCD d'un Registre vers une suite de sorties ou flags.
Un chiffre BCD consiste en 4 bits binaires.

Le 1er opérande est le nombre de chiffres à transférer.
Le 2ème opérande est le Registre source.
Le 3ème opérande est l'adresse de base des sorties ou flags.

L'adresse de la sortie ou flag la plus basse devient le bit de poids le plus fort du chiffre BCD le plus significatif.

Ceci est identique au SAIA PCA.

Usage

```
DIGOR[X]  digits          ; Nombre de chiffres BCD 1-10
          source   (i)    ; Registre source R 0-4095
          dest            ; Elément de destination O 0-8191, F 0-8191
```

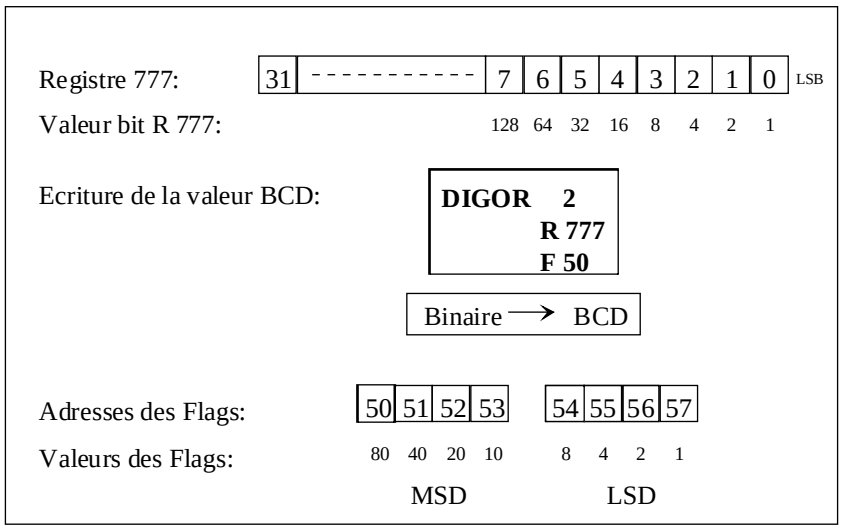
Exemple

```
DIGOR     2     ; Transfert de 2 chiffres BCD
        R 123   ;    du Registre 123
        O 40    ;    vers les sorties 40-43 et 44-47
```

Flags Le flag **Error** (E) est positionné si un chiffre BCD est invalide (> 9).

Voir aussi DIGO, DIGI, DIGIR, BITO, BITOR.

Pratique



AND AND REGISTERS

Combinaison logique ET entre deux registres

Description Le contenu du 1er Registre est combiné logiquement en une fonction ET (AND) avec le contenu du 2ème Registre. Le résultat est placé dans le 3ème Registre.

Usage

AND [X]	**valeur1**	**(i)**	**; R 0-4095**
	valeur2		**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

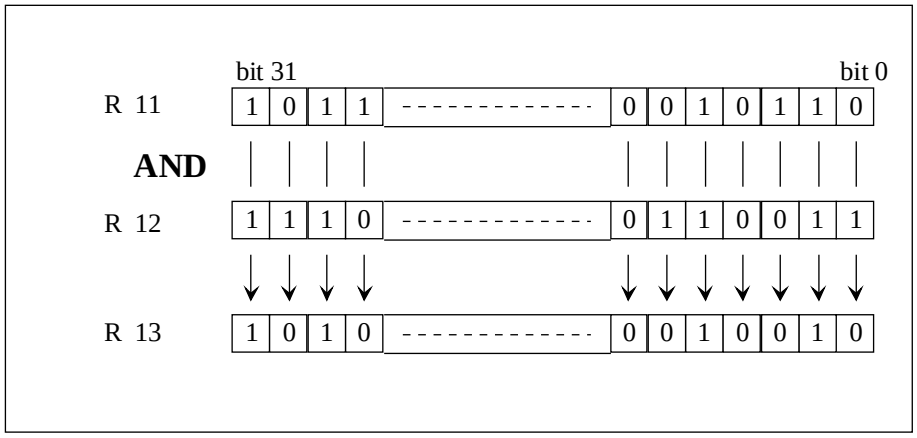
```
AND     R 11    ; ET (AND) entre Registre 11
        R 12    ;      et Registre 12
        R 13    ;      résultat dans Registre 13
```

R 13 contient 1 bit pour chaque bit à 1 simultanément dans R 11 ET R 12.

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat. Le flag **Error** (E) est toujours mis Low.

Voir aussi OR, NOT, EXOR.

Pratique



OR OR REGISTERS

Combinaison logique OU entre deux registres

Description Le contenu du 1er Registre est combiné logiquement en une fonction OU (OR) avec le contenu du 2ème Registre. Le résultat est placé dans le 3ème Registre.

Usage

OR[X]	**valeur1**	**(i)**	**; R 0-4095**
	valeur2		**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

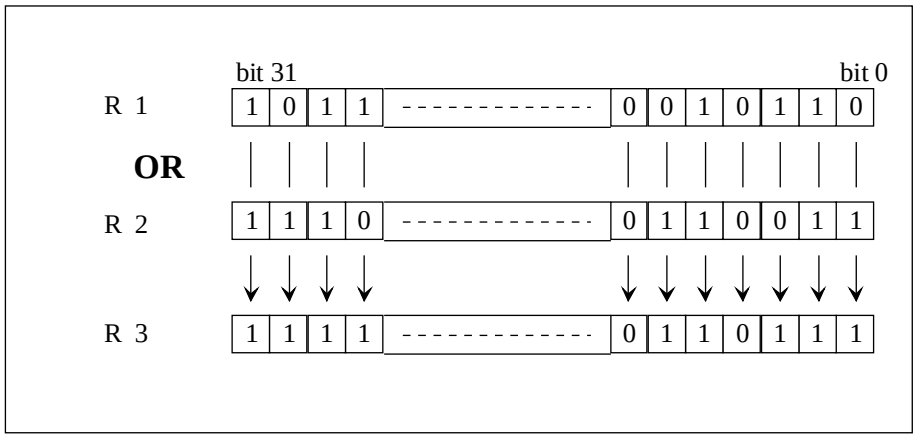
Exemple

```
OR      R 1     ; OU (OR) entre Registre 1
        R 2     ;    avec Registre 2
        R 3     ;    et résultat dans le Registre 3
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat. Le flag **Error** (E) est toujours mis Low

Voir aussi EXOR.

Pratique



EXOR EXCLUSIVE-OR REGISTERS

Combinaison logique OU exclusif entre deux registres

Description Le contenu du 1er Registre est combiné logiquement en une fonction OU exclusif (XOR) avec le contenu du 2ème Registre. Le résultat est placé dans le 3ème Registre.

Usage

```
EXOR[X]   valeur1   (i)    ; R 0-4095
          valeur2          ; R 0-4095
          résultat  (i)    ; R 0-4095
```

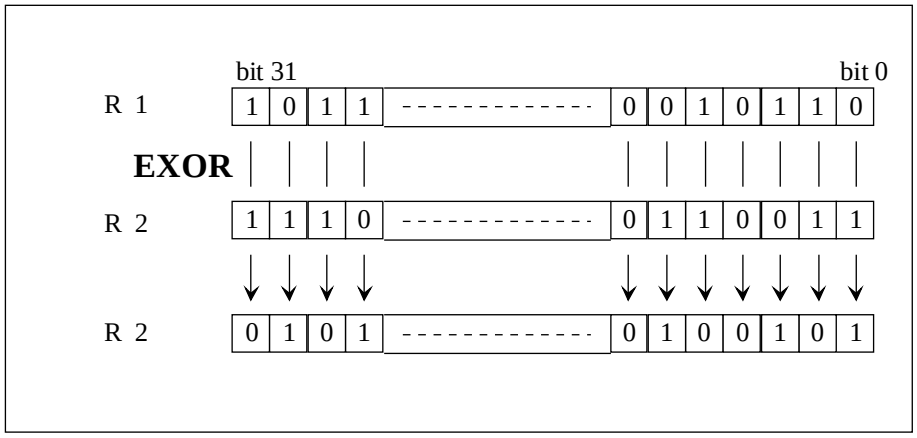
Exemple

```
EXOR    R 1     ; OU exclusif (XOR) entre Registre 1
        R 2     ;     et Registre 2
        R 2     ;     et le résultat est placé dans le Registre 2
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat. Le flag **Error** (E) est toujours mis Low.

Voir aussi OR.

Pratique



NOT COMPLEMENT REGISTER

Inversion logique du contenu d'un registre

Description Le contenu du 1er Registre est inversé (complément à 1) et le résultat est placé dans le 2ème Registre.

Usage

NOT[X]	**valeur**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

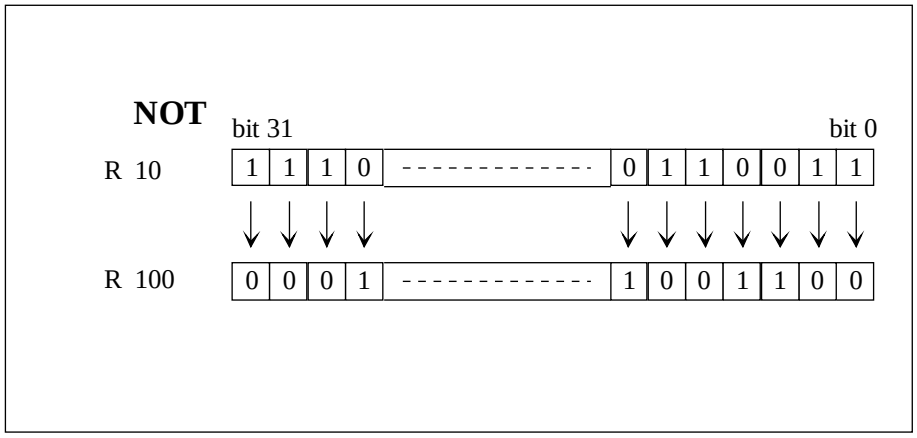
Exemple

```
NOT     R 10    ; Inversion du Registre 10
        R 100   ;     résultat dans le Registre 100
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat. Le flag **Error** (E) est toujours mis Low.

Voir aussi

Pratique



SHIU SHIFT REGISTERS UP

Décalage vers le haut d'un bloc de registres

Description Décalage d'une place vers le haut du contenu d'un bloc de Registres.
Le 1er et 2ème opérande donnent respectivement le début et la fin du bloc de Registres à décaler.
Après le décalage, le Registre dont l'adresse est la plus basse contient zéro; le registre immédiatement après la fin du bloc est écrasé par le contenu du registre le plus haut.
Le registre avec l'adresse la plus basse ou la plus haute peut être spécifié en premier.

Usage

SHIU	**début**	**; R 0-4094**
	fin	**; R 0-4094**

Exemple

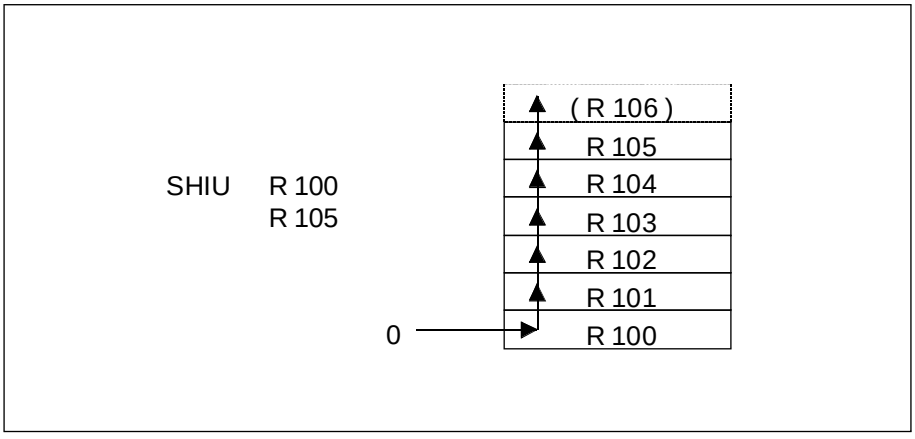
```
SHIU    R 100   ; Décale R 100 jusqu'à R 105 vers le haut
        R 105   ;       (d'une position)
                ; R 100 = 0, R 101 = R 100 ... R 106 = R 105
```

Flags Inchangés.

Note Cette instruction utilise un registre de plus que ceux qui sont spécifiés : le registre qui suit la fin du bloc est également utilisé.

Voir aussi SHID, ROTU, ROTD.

Pratique



SHID SHIFT REGISTERS DOWN

Décalage vers le bas d'un bloc de registres

Description Décalage d'une place vers le bas du contenu d'un bloc de Registres.
Le 1er et 2ème opérande donnent respectivement le début et la fin du bloc de Registres à décaler.
Après le décalage, le Registre dont l'adresse est la plus haute contient zéro; le registre précédant le début du bloc est écrasé par le contenu du registre le plus bas.
Le registre avec l'adresse la plus basse ou la plus haute peut être spécifié en premier.

Usage

SHID	**début**	**; R 0-4094**
	fin	**; R 0-4094**

Exemple

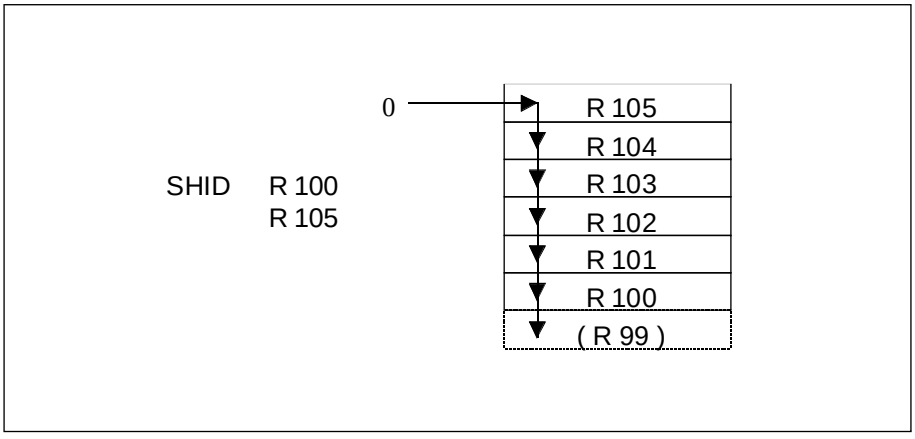
```
SHID    R 100  ; Décale R 100 jusqu'à R 105 vers le bas
        R 105  ;       (d'une position)
               ; R 99 = R 100 ... R 104 = R 105, R 105 = 0
```

Flags Inchangés.

Note Cette instruction utilise un registre de plus que ceux qui sont spécifiés : le registre qui précède le début du bloc est également utilisé.

Voir aussi SHIU, ROTU, ROTD.

Pratique



ROTU ROTATE REGISTERS UP

Rotation vers le haut d'un bloc de registres

Description Rotation d'une place vers le haut du contenu d'un bloc de Registres.
La 1er et 2ème opérande donnent respectivement le début et la fin du bloc de Registres.
Après la rotation, le Registre dont l'adresse est la plus basse contient la valeur du registre le plus haut.
Le registre avec l'adresse la plus basse ou la plus haute peut être spécifié en premier.

Usage

ROTU	**début**	**; R 0-4094**
	fin	**; R 0-4095**

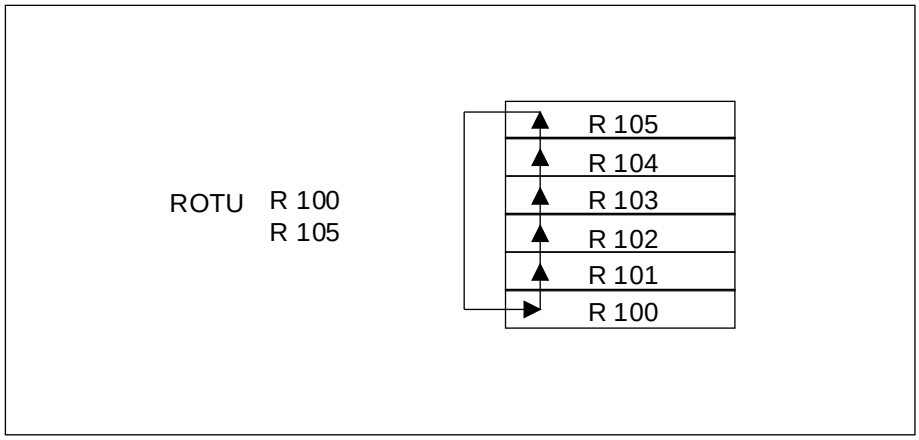
Exemple

```
ROTU    R 100   ; Rotation de R 100 jusqu'à R 105 vers le haut.
        R 105
                ; R 100 = R 105, R 101 = R100 ... R 105 = R 104
```

Flags Inchangés.

Voir aussi ROTD, SHIU, SHID.

Pratique



ROTD ROTATE REGISTERS DOWN

Rotation vers le bas d'un bloc de registre

Description Rotation d'une place vers le bas du contenu d'un bloc de Registres.
La 1er et 2ème opérande donnent respectivement le début et la fin du bloc de Registres.
Après la rotation, le Registre dont l'adresse est la plus haute contient la valeur du registre le plus bas.
Le registre avec l'adresse la plus basse ou la plus haute peut être spécifié en premier.

Usage

ROTD	**début**	**; R 0-4095**
	fin	**; R 0-4095**

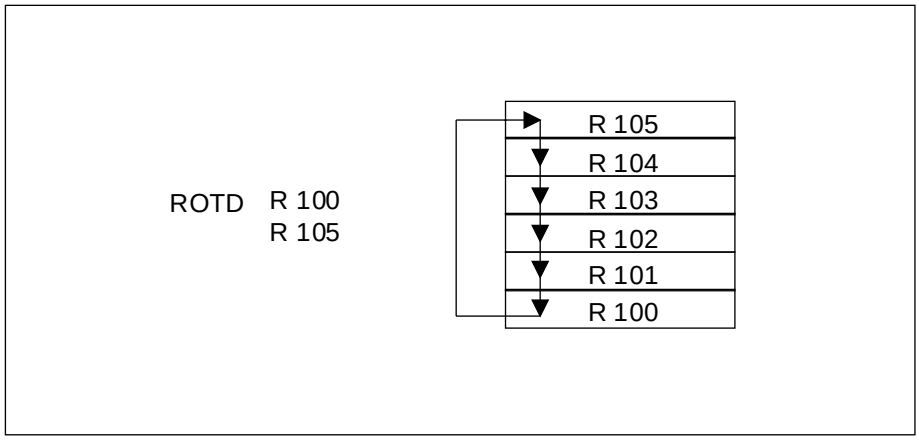
Exemple

```
ROTD    R 100   ; Rotation de R 100 jusqu'à R 105 vers le bas
        R 105
                ; R 100 = R 101 ... R 104 = R 105, R 105 = R 100.
```

Flags Inchangés.

Voir aussi ROTU, SHIU, SHID.

Pratique



SHIL SHIFT REGISTER LEFT

Décalage vers la gauche d'un registre

Description Le contenu du Registre est décalé vers la gauche du nombre de bits spécifié dans l'opérande de la seconde ligne.
Pour chaque bit décalé, le contenu de l'ACCU (1 ou 0) est entré depuis le bit 0 (bit de poids le plus faible).
A la fin du décalage, l'ACCU est positionné suivant la valeur du dernier bit sorti du Registre.

Usage

SHIL[X]	**reg**	**(i)**	**; R 0-4095**
	n bits		**; Nombre de bits 1-32**

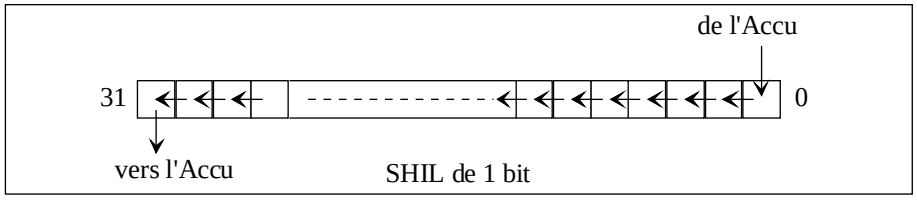
Exemple

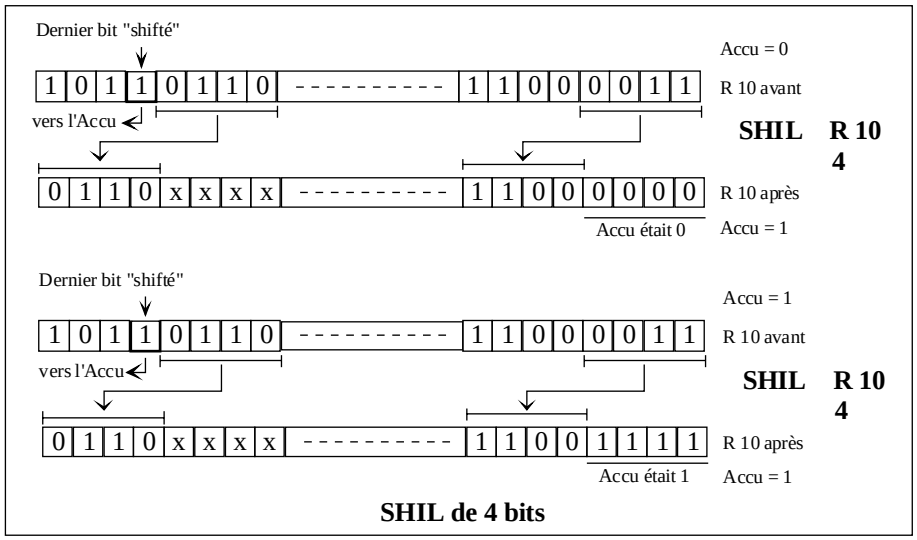
```
SHIL    R 10    ; Le Registre 10 est décalé vers la gauche
          4     ;      de 4 bits
                ; (multiplication par 16 si l'ACCU = 0)
```

Flags L' **ACCU** est positionné suivant la valeur du dernier bit sorti du Registre.

Voir aussi SHIR, ROTL, ROTR.

Pratique





SHIR SHIFT REGISTER RIGHT

Décalage vers la droite d'un registre

Description Le contenu du Registre est décalé vers la droite du nombre de bits spécifié dans l'opérande de la seconde ligne.
Pour chaque bit décalé, le contenu de l'ACCU (1 ou 0) est entré depuis le bit 31 (bit de poids le plus fort).
A la fin du décalage, l'ACCU est positionné suivant la valeur du dernier bit sorti du Registre.

Usage

```
SHIR[X]   reg      (i)    ; R 0-4095
          n bits          ; Nombre de bits 1-32
```

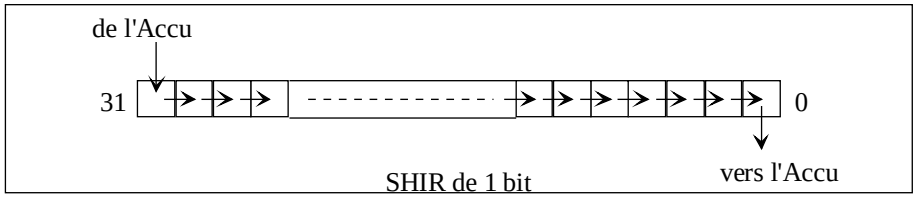
Exemple

```
SHIR    R 10    ; Le Registre 10 est décalé vers la droite
          16    ;      de 16 bits
                ; (division par 65536 si l'ACCU = 0)
```

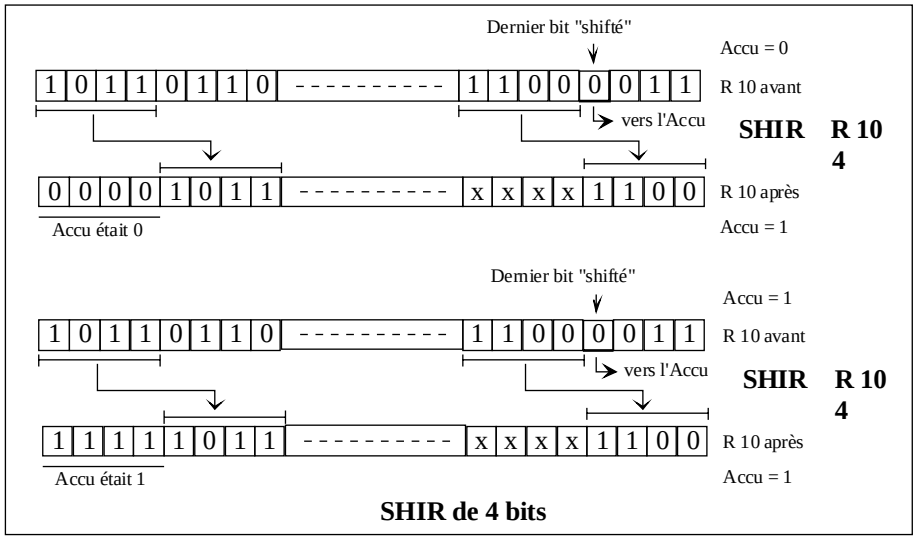
Flags L' **ACCU** est positionné suivant la valeur du dernier bit sorti du Registre.

Voir aussi SHIL, ROTL, ROTR.

Pratique



SHIR de 1 bit



SHIR de 4 bits

ROTL ROTATE REGISTER LEFT

Rotation vers la gauche d'un registre

Description Rotation du contenu du Registre vers la gauche du nombre de bits spécifié dans l'opérande de la seconde ligne.
Pour chaque rotation d'un bit, le bit le plus significatif (31) est copié dans le bit de poids le plus faible (0).
A la fin du décalage, l'ACCU est positionné suivant la valeur du dernier bit ayant effectué la rotation.

Usage

```
ROTL[X]   reg      (i)     ; R 0-4095
          n bits           ; Nombre de bits 1-32
```

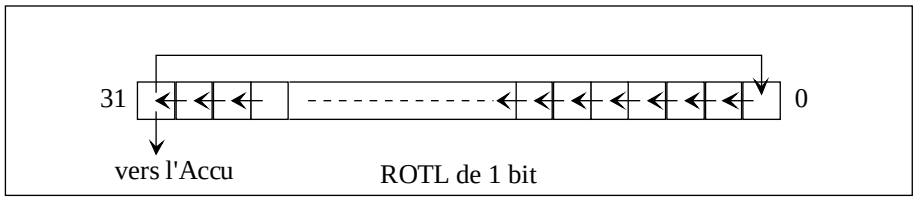
Exemple

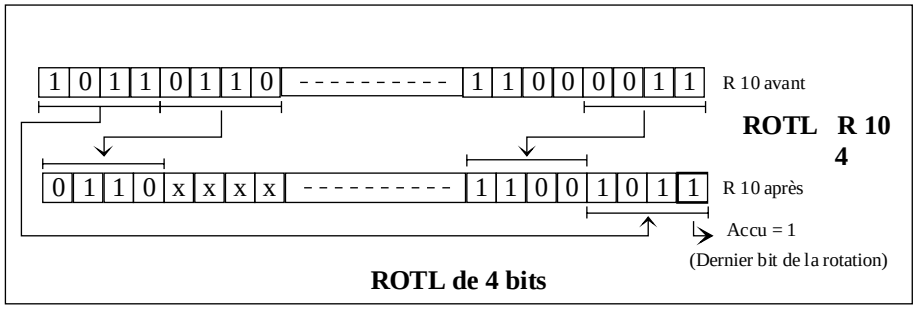
```
ROTL    R 10    ; Rotation du Registre 10 de 4 bits vers la gauche
          4
```

Flags L' **ACCU** est positionné suivant la valeur du dernier bit ayant effectué la rotation.

Voir aussi ROTR, SHIL, SHIR.

Pratique





ROTR ROTATE REGISTER RIGHT

Rotation vers la droite d'un registre

Description Rotation du contenu du Registre vers la droite du nombre de bits spécifié dans l'opérande de la seconde ligne.
Pour chaque rotation d'un bit, le bit le moins significatif (0) est copié dans le bit de poids le plus fort (31).
A la fin du décalage, l'ACCU est positionné suivant la valeur du dernier bit ayant effectué la rotation.

Usage

```
ROTR[X]   reg      (i)     ; R 0-4095
          n bits           ; Nombre de bits 1-32
```

Exemple

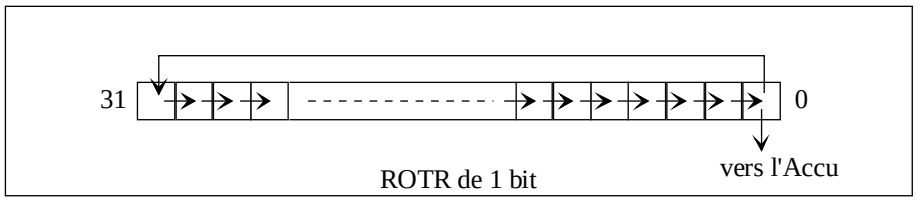
```
ROTR    R 10    ; Rotation du Registre 10 de 4 bits vers la droite
          4
```

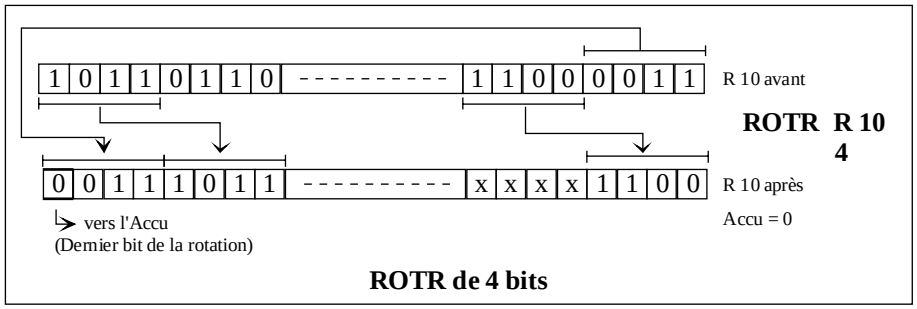
Flags L' **ACCU** est positionné suivant la valeur du dernier bit ayant effectué la rotation.

Note

Voir aussi ROTR, SHIL, SHIR.

Pratique





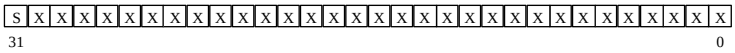
4. Instructions INTEGER (arithmétique)

Les instructions de calcul arithmétique en entier travaillent seulement avec les Registres.

ADD	ADD registers	Addition de registres
SUB	SUBtract registers	Soustraction de registres
MUL	MULtiply registers	Multiplication de registres
DIV	DIVide registers	Division de registres
SQR	SQare Root	Racine carrée
CMP	CoMPare registers	Comparaison de registre

Pour les valeurs en virgule flottante, les instructions "Floating Point" doivent être utilisées.

Le format "integer" est basé sur 32 bits suivant le schéma ci-dessous:



où X valeur du bit (0 ou 1)
S signe

Les bits 0 à 30 représentent la valeur entière en format binaire.
Le bit 31 est le bit de signe (0 = positif, 1 = négatif).

La plage de valeurs autorisées est:

Décimal	2.147.483.647	à	-2.147.483.648
Binaire (hexadécimal)	7FFF'FFFF	à	8000'0000

Format INTEGER

Notes personnelles :

ADD ADD REGISTERS
Addition de registres

Description Addition entière.
Additionne le contenu du 1er Registre ou constante au contenu du 2ème Registre ou constante; le résultat est mémorisé dans le 3ème Registre.

Usage

ADD	**valeur 1**	**; R 0-4095, K 0-16383**
	valeur 2	**; R 0-4095, K 0-16383**
	résultat	**; R 0-4095**

Exemple

```
ADD     R 20    ; Additionne 123 au Registre 20
        K 123
        R 20
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Error** (E) est positionné en cas de dépassement.

Voir aussi FADD (Floating Point Add).

Pratique Lecture de 2 nombres, addition de ceux-ci et mémorisation du résultat dans un registre.
Les deux nombres sont lus sur les codeurs BCD (2 chiffres) des entrées 16 à 23 et 24 à 31.

```
COB       0     ; En-tête du COB
          0

DIGI      2     ; Lecture de 2 chiffres
        I 16    ;     des entrées 16 (à 23)
        R 100   ;     et mémorisation dans R 100

DIGI      2     ; Lecture de 2 chiffres
        I 24    ;     des entrées 24 (à 31)
        R 200   ;     et mémorisation dans R 200

ADD     R 100   ; R 0 = R 100 + R 200
        R 200
        R 0

ECOB
```

SUB SUBTRACT REGISTERS
Soustraction de registres

Description Soustraction entière.
Soustrait le contenu du 2ème Registre ou de la constante du contenu du 1er Registre ou constante, et mémorise le résultat dans le 3ème Registre.

Usage

SUB	**valeur 1**	**; R 0-4095, K 0-16383**
	valeur 2	**; R 0-4095, K 0-16383**
	résultat	**; R 0-4095**

Exemple

```
SUB     R 1     ; Registre 3 = Registre 1 moins Registre 2
        R 2
        R 3
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Error** (E) est positionné en cas de dépassement.

Voir aussi ADD, FSUB.

Pratique Lecture de 2 nombres, soustraction de ceux-ci et mémorisation du résultat dans un registre.
Les deux nombres sont lus sur les codeurs BCD (2 chiffres) des entrées 16 à 23 et 24 à 31.

```
COB       0     ; En-tête du COB
          0

DIGI      2     ; Lecture de 2 chiffres
        I 16    ;     des entrées 16 (à 23)
        R 10    ;     et mémorisation dans R 10

DIGI      2     ; Lecture de 2 chiffres
        I 24    ;     des entrées 24 (à 31)
        R 11    ;     et mémorisation dans R 11

SUB     R 10    ; R 12 = R 10 - R 11
        R 11
        R 12

ECOB
```


MUL MULTIPLY REGISTERS
Multiplication de registres

Description Multiplication entière.
Le contenu du 1er Registre ou constante est multiplié par le contenu du 2ème Registre ou constante, le résultat est mémorisé dans le 3ème Registre.

Usage

```
MUL     valeur 1        ; R 0-4095, K 0-16383
        valeur 2        ; R 0-4095, K 0-16383
        résultat        ; R 0-4095
```

Exemple

```
MUL     R 0     ; Multiplication du Registre 0 par 10
        K 10
        R 0
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Error** (E) est positionné en cas de dépassement.

Voir aussi DIV, FMUL.

Pratique Lecture de 2 nombres, multiplication de ceux-ci et mémorisation du résultat dans un registre.
Les deux nombres sont lus sur les codeurs BCD (2 chiffres) des entrées 16 à 23 et 24 à 31.

```
COB       0     ; En-tête du COB
          0

DIGI      2     ; Lecture de 2 chiffres
        I 16    ;     des entrées 16 (à 23)
        R 50    ;     et mémorisation dans R 50

DIGI      2     ; Lecture de 2 chiffres
        I 24    ;     des entrées 24 (à 31)
        R 55    ;     et mémorisation dans R 55

MUL     R 50    ; R 4000 = R 50 * R 55
        R 55
        R 4000

ECOB
```


DIV DIVIDE REGISTERS
Division de registres

Description Division entière.
Division du contenu du 1er Registre ou constante par le contenu du 2ème Registre ou constante, et mémorisation du résultat dans le 3ème Registre.
Le reste de la division est placé dans le 4ème Registre.

Usage

```
DIV     valeur 1    ; R 0-4095, K 0-16383
        valeur 2    ; R 0-4095, K 0-16383
        résultat    ; R 0-4095
        reste       ; R 0-4095
```

Exemple

```
DIV     R 20    ; R 20 est divisé
        K 1000  ;    par 1000
        R 21    ;    le résultat est placé dans le Registre 21
        R 1     ;    et le reste dans le Registre 1
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Error** (E) est positionné en cas de division par zéro.

Voir aussi FDIV.

Pratique Lecture de 2 nombres, division de ceux-ci et mémorisation du résultat dans un registre.
Les deux nombres sont lus sur les codeurs BCD (2 chiffres) des entrées 16 à 23 et 24 à 31.

```
COB         0       ; En-tête du COB
            0

DIGI        2       ; Lecture de 2 chiffres
          I 16      ;    des entrées 16 (à 23)
          R 1       ;    et mémorisation dans R 1
DIGI        2       ; Lecture de 2 chiffres
          I 24      ;    des entrées 24 (à 31)
          R 2       ;    et mémorisation dans R 2
DIV       R 1       ; R 100 = R 1 / R 2
          R 2
          R 100     ;    résultat
          R 101     ;    reste
CPB       E 99      ; Si le flag Error Flag est positionné,
                    ;    alors appel du bloc de programme 99.

ECOB

PB          99
SET       O 47      ; Alarme en cas de division par zéro (sortie 47)
EPB
```


SQR SQUARE ROOT
Racine carrée

Description Racine carrée entière.
La racine carrée du contenu du 1er Registre ou constante est mémorisée dans le 2ème Registre.
Si le 1er Registre contient une valeur négative, le flag Error est positionné et le calcul n'est pas effectué.

Usage

SQR	**valeur**	**; R 0-4095**
	résultat	**; R 0-4095**

Exemple

```
SQR     R 0     ; La racine carrée du Registre 0 est
        R 100   ;     placée dans le Registre 100
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Error** (E) est positionné si le 1er Registre contient une valeur négative.

Voir aussi FSQR.

Pratique Extraction de la racine carrée d'un nombre lu sur les codeurs BCD (4 chiffres) des entrées 16 à 31.

```
COB         0       ; En-tête du COB
            0

DIGI        4       ; Lecture de 4 chiffres
          I 16      ;     des entrées 16 (à 31)
          R 100     ;     et mémorisation dans R 100

SQR       R 100     ; R 101 = √ R 100
          R 101

ECOB
```


CMP COMPARE REGISTERS
Comparaison de registres

Description Compare le contenu du 1er Registre ou constante avec le contenu du 2ème Registre ou constante.
La comparaison est effectuée par soustraction de la 2ème valeur de la 1ère, les flags de status sont positionnés suivant le résultat.
Le contenu des Registres sont inchangés.

Usage

```
CMP[X]   valeur 1  (i)    ; R 0-4095, K 0-16383
         valeur 2         ; R 0-4095, K 0-16383
```

Exemple

```
CMP      R 0     ; Compare le Registre 0 avec le Registre 1
         R 1     ; les status flags sont positionnés suivant le résultat
```

Flags Les Status flags sont positionnés comme suit :

	Z	P	N
Valeur 1 = Valeur 2	High	High	Low
Valeur 1 > Valeur 2	Low	High	Low
Valeur 1 < Valeur 2	Low	Low	High

Voir aussi AND, OR, EXOR, FCMP

Pratique Lire 2 nombres; si le 1er est plus grand, égal ou plus petit que le second : la sortie 32 (respectivement 33 ou 34) doit être activée. Les deux nombres sont lus sur les codeurs BCD (2 chiffres) des entrées 16 à 23 et 24 à 31.

```
COB         0      ; En-tête du COB
            0
DIGI        2      ; Lecture de 2 chiffres
          I 16     ;     des entrées 16 (à 23)
          R 1      ;     et mémorisation dans R 1
DIGI        2      ; Lecture de 2 chiffres
          I 24     ;     des entrées 24 (à 31)
          R 2      ;     et mémorisation dans R 2
CMP       R 1      ; Compare R 1 avec R 2
          R 2
ACC       Z        ; Si R1 = R2
OUT       O 33     ;     Alors enclencher la sortie 33 et le flag 0
OUT       F 0      ;     Sinon déclencher la sortie 33 et le flag 0
ACC       N        ; Si R 1 < R2
OUT       O 34     ;     Alors enclencher la sortie 34
                   ;     Sinon déclencher la sortie 34
ACC       P        ; Si R1 > R2
ANL       F 0      ;     (et différent)
OUT       O 32     ;     Alors enclencher la sortie 32
                   ;     Sinon déclencher la sortie 32
ECOB
```


5. Instructions FLOATING POINT (arithmétique)

Des valeurs en virgule flottante peuvent seulement être mémorisées dans les Registres.
Ces valeurs peuvent être chargées dans les Registres avec l'instruction LD.

Pour spécifier qu'il s'agit d'un nombre en virgule flottante, ce nombre devra comporter un point décimal '.' ou un exposant 'E'.

Par exemple : 1.2, 1E3, -4.656E-2.

La plage des nombres en Floating Point est :

± 5.42101E-20 ... ± 9.22337E+18 (précision de 5 chiffres significatifs)

IFP	Integer to Floating Point	Conversion entier → virgule flottante
FPI	Floating Point to Integer	Conversion virgule flottante → entier
FADD	Floating point ADD	Addition en virgule flottante
FSUB	Floating point SUBtract	Soustraction en virgule flottante
FMUL	Floating point MULtiply	Multiplication en virgule flottante
FDIV	Floating point DIVide	Division en virgule flottante
FSQR	Floating point SQuare Root	Racine carrée en virgule flottante
FCMP	Floating point CoMPare	Comparaison en virgule flottante
FSIN	Floating point SINe	Sinus
FCOS	Floating point COSine	Cosinus
FATAN	Floating point Arc TANgent	Arc tangente
FEXP	Floating point EXPonential	Exponentielle
FLN	Floating point Logarithm	Logarithme népérien
FABS	Floating point ABSolute value	Valeur absolue

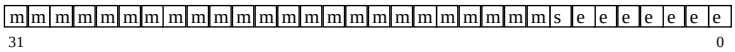
Note :

Les valeurs en virgule flottante sont mémorisées dans les Registres en utilisant un format binaire spécial, l'utilisation de cette valeur comme entier conduit à un résultat incorrect.

Le mélange de valeurs entières et virgule flottante dans les opérations mathématiques donne des résultats incorrects. Les valeurs entières doivent d'abord être converties en virgule flottante avec l'instruction **IFP.**

Les valeurs en virgule flottante peuvent être converties en entier avec l'instruction **FPI.**

Le format "floating point" est basé sur 32 bits suivant le schéma ci-dessous:



où m mantisse 24-bits
s signe du nombre
e exposant en 7-bits

Le bit de signe est 0 pour une valeur positive, et 1 pour une valeur négative.
La mantisse est considérée comme étant un nombre binaire normalisé avec virgule fixe; excepté pour 0 qui a le bit de poids le plus fort à 1.
L'exposant est la puissance de 2 nécessaire pour positionner la virgule dans la mantisse pour donner la valeur réelle à représenter. L'exposant est noté en excès à 64; ce qui signifie que 64 est ajouté au complément à 2 de la valeur.

Format Floating Point

IFP INTEGER TO FLOATING POINT

Conversion entier -> virgule flottante

Description Converti la valeur entière du Registre spécifié en format virgule flottante.
Le 2ème opérande donne la puissance de 10 à laquelle la valeur entière est élevée; ceci contrôle la position du point décimal.
Par exemple, si la puissance de 10 est +3, le contenu du Registre est multiplié par 1000 (10^3), et le résultat est mémorisé dans le Registre en format virgule flottante.
Si le Registre contenait 12, le résultat serait 12000.00.
Si la conversion n'est pas possible (nombre trop grand ou trop petit), le flag Erreur est positionné et la conversion n'est pas effectuée.

Usage

```
IFP[X]    reg       (i)    ; R 0-4095
          puissance          ; Puissance de dix -20 à +18
```

Exemple

```
IFP     R 0     ; R 0 = Valeur en virgule flottante de R 0 * 10^3
          3
```

Flags Le flag **Erreur** (E) est positionné si la conversion n'est pas possible.

Voir aussi FPI.

Pratique

R 500 Avant	Instruction	Conversion	R 500 Après
123	IFP R 500 0	R 500 * 10^0	1.23E+2
	IFP R 500 -2	R 500 * 10^{-2}	1.23E+0
	IFP R 500 3	R 500 * 10^3	1.23E+5

FPI FLOATING POINT TO INTEGER

Conversion virgule flottante -> entier

Description Converti la valeur en virgule flottante du Registre spécifié en format entier. Le 2ème opérande donne la puissance de 10 à utiliser dans la conversion. Le résultat est un nombre entier résultant du contenu du Registre multiplié par 10 exposant le 2ème opérande.
Par exemple, si le Registre contient 1234.56 et la puissance de 10 est -2, le résultat en entier sera 12.
Si la conversion n'est pas possible, le flag Erreur est positionné et la conversion n'est pas effectuée.

Usage

```
FPI[X]    reg        (i)    ; R 0-4095
          puissance         ; Puissance de dix -20 à +18
```

Exemple

```
FPI     R 0     ; Si R 0 contient 1234.56, il est converti
          0     ; en la valeur entière 1234 (puissance de dix est zéro)
```

Flags Le flag **Erreur** (E) est positionné si la conversion n'est pas possible.

Voir aussi IFP.

Pratique

R 500 Avant	Instruction	Conversion	R 500 Après
123.456	FPI R 500 0	R 500 * 10^{0}	123
	FPI R 500 -2	R 500 * 10^{-2}	1
	FPI R 500 3	R 500 * 10^{3}	123456

FADD FLOATING POINT ADD
Addition en virgule flottante

Description Additionne le contenu du 1er Registre au 2ème Registre, le résultat est mémorisé dans le 3ème Registre.
Les Registres doivent contenir des valeurs en format Floating Point.

Usage

```
FADD    reg1        ; R 0-4095
        reg2        ; R 0-4095
        résultat    ; R 0-4095
```

Exemple

```
FADD    R 100  ; R 500 = R 100 + R 101
        R 101
        R 500
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Erreur** (E) est positionné en cas de dépassement.

Voir aussi ADD.

FSUB FLOATING POINT SUBTRACT

Soustraction en virgule flottante

Description Soustrait le contenu du 2ème Registre du 1er Registre, le résultat est mémorisé dans le 3ème Registre.
Les Registres doivent contenir des valeurs en format Floating Point.

Usage

```
FSUB    reg1        ; R 0-4095
        reg2        ; R 0-4095
        résultat    ; R 0-4095
```

Exemple

```
FSUB    R 0     ; R 0 = R 0 - R 1
        R 1
        R 0
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Erreur** (E) est positionné en cas de dépassement.

Voir aussi SUB.

FMUL FLOATING POINT MULTIPLY

Multiplication en virgule flottante

Description Multiplie le contenu du 1er Registre avec le 2ème Registre, le résultat est mémorisé dans le 3ème Registre.
Les Registres doivent contenir des valeurs en format Floating Point.

Usage

```
FMUL    reg1        ; R 0-4095
        reg2        ; R 0-4095
        résultat    ; R 0-4095
```

Exemple

```
FMUL    R 20    ; R 0 = R 20 * R 30
        R 30
        R 0
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Erreur** (E) est positionné en cas de dépassement.

Voir aussi MUL.

FDIV FLOATING POINT DIVIDE

Division en virgule flottante

Description Divise le contenu du 1er Registre par le 2ème Registre, le résultat est mémorisé dans le 3ème Registre.
Les Registres doivent contenir des valeurs en format Floating Point.
Une division par zéro positionne le flag Erreur et l'opération n'est pas effectuée.
Note : Dans le cas de la division en Floating Point, il n'y a pas de reste.

Usage

FDIV	**reg**	**; R 0-4095**
	diviseur	**; R 0-4095**
	résultat	**; R 0-4095**

Exemple

```
FDIV    R 1     ; R 3 = R 1 / R 2
        R 2
        R 3
```

Flags Les flags **Zéro** (Z) et de **Signe** (P ou N) sont positionnés suivant le résultat.
Le flag **Erreur** (E) est positionné en cas de division par zéro.

Voir aussi DIV.

FSQR FLOATING POINT SQUARE ROOT
Racine carrée en virgule flottante

Description Racine carrée du contenu du 1er Registre, le résultat est mémorisé dans le 2ème Registre.
Si le 1er Registre contient une valeur négative, le flag Erreur est positionné, et la racine carrée de la valeur absolue (+ve) est calculée.

Usage

FSQR	**reg**	**; R 0-4095**
	résultat	**; R 0-4095**

Exemple

```
FSQR    R 0     ; R 0 = Racine carrée de R 0
        R 0
```

Flags Le flag **Zéro** (Z) est positionné suivant le résultat.
Le résultat est toujours **positif**, P est toujours H et N est toujours L.
Si la valeur était négative, le flag **Erreur** (E) est positionné.

Voir aussi SQR.

FCMP FLOATING POINT COMPARE
Comparaison en virgule flottante

Description Compare le contenu du 1er Registre avec le contenu du 2ème Registre, les flags de Status sont positionnés suivant le résultat.
Le contenu des Registres n'est pas modifié.
Les Registres doivent contenir des valeurs en format Floating Point.

Usage

```
FCMP[X]   reg1    (i)    ; R 0-4095
          reg2           ; R 0-4095
```

Exemple

```
FCMP    R 0     ; Compare R 0 et R 1, les flags de Status
        R 1     ;      sont positionné suivant le résultat.
```

Flags Les flags de Status sont positionnés comme suit :

	Z	P	N
Valeur 1 = Valeur 2	High	High	Low
Valeur 1 > Valeur 2	Low	High	Low
Valeur 1 < Valeur 2	Low	Low	High

Le flag **Erreur** (E) est mis Low.

Voir aussi CMP.

Note : Il ne faut JAMAIS tester l'égalité de valeurs Floating Point, pour éviter les erreurs d'arrondi : utilisez > ou <.

FSIN FLOATING POINT SINE
Sinus

Description Le sinus du contenu du 1er Registre est calculé et mémorisé dans le 2ème Registre.
Le 1er Registre doit contenir une valeur Floating Point en RADIANS dans la plage $\pm 10^6$.

Usage

FSIN[X]	**reg**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

```
FSIN    R 0     ; R 100 = Sinus de R 0
        R 100
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat.

Voir aussi FCOS, FATAN.

FCOS FLOATING POINT COSINE
Cosinus

Description Le cosinus du contenu du 1er Registre est calculé et mémorisé dans le 2ème Registre.
Le 1er Registre doit contenir une valeur Floating Point en RADIANS dans la plage $\pm 10^6$.

Usage

FCOS[X]	**reg**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

```
FCOS    R 100  ; R 20 = Cosinus de R 100
        R 20
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat.

Voir aussi FSIN, FATAN.

FATAN FLOATING POINT ARC TANGENT
Arc tangente

Description L'arc tangente du contenu du 1er Registre est calculé et mémorisé dans le 2ème Registre.
Le 1er Registre doit contenir une valeur Floating Point en RADIANS.
Le résultat dans le second Registre sera compris entre $-\pi/2$ et $+\pi/2$.

Usage

FATAN[X]	**reg**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

```
FATAN   R 1     ; R 0 = Arc tangente de R 1
        R 0
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat.

Voir aussi FSIN, FCOS.

FEXP FLOATING POINT EXPONENTIAL
Exponentielle

Description L'exponentielle du contenu du 1er Registre est calculée et mémorisée dans le 2ème Registre.
Le 1er Registre doit contenir une valeur en Floating Point.

Usage

FEXP[X]	**reg**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

```
FEXP    R 0     ; R 1 = e R0
        R 1
```

Flags Le flag **Zero** (Z) est positionné suivant le résultat, le flag **Négatif** (N) est toujours mis Low (P = 1).
Le flag **Erreur** (E) est positionné en cas de dépassement.

Voir aussi FPI, IFP.

FLN FLOATING POINT LOGARITHM
Logarithme népérien

Description Le logarithme népérien du contenu du 1er Registre est calculé et mémorisé dans le 2ème Registre.
Le 1er Registre doit contenir une valeur en Floating Point.
Si le logarithme népérien d'une valeur négative est calculé, le flag Erreur est positionné et le logarithme de la valeur absolue (+ve) est calculé.

Usage

FLN[X]	**reg**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

```
FLN     R 1     ; R 2 = ln R 1
        R 2
```

Flags Les flags **Zéro** (Z) et de **Signe** (N ou P) sont positionnés suivant le résultat.
Le flag **Erreur** (E) est positionné si le "ln" (logarithme népérien) de zéro ou d'une valeur négative est calculé.

Voir aussi FEXP.

FABS FLOATING POINT ABSOLUTE VALUE

Valeur absolue

Description La valeur absolue (conversion en positif si négatif) du 1er Registre est calculée et mémorisée dans le 2ème Registre.
Le 1er Registre doit contenir une valeur en Floating Point.

Usage

FABS[X]	**reg**	**(i)**	**; R 0-4095**
	résultat	**(i)**	**; R 0-4095**

Exemple

```
FABS    R 1     ; R 2 = valeur absolue de R 1
        R 2     ; Si R 1 contient -7.5 alors R 2 = 7.5
```

Flags Le flag **Zéro** (Z) est positionné suivant le résultat.

Voir aussi

6. Instructions BLOCTEC

Le BLOCTEC est une méthode de programmation structurée qui sépare un programme en différents blocs de code.

Un bloc d'organisation cyclique " COB " (**Cyclic Organisation Block**) est un bloc principal qui appellera des blocs de programme " PB " (**Program Blocks**), qui à leur tour appelleront des blocs de fonction " FB " (**Function Blocks**).
Un COB (COB 0) doit au minimum être présent dans un programme.
Les COBs peuvent appeler des PBs ou FBs (avec des paramètres optionnels).
PBs et FBs peuvent appeler d'autres PB ou FB jusqu'à 7 niveaux d'imbrication.

Les opérandes de ces instructions ne peuvent être passés comme paramètres de blocs de fonction.

Pour plus d'informations sur les méthodes de structuration de programmes à l'aide du BLOCTEC, référez-vous au chapitre "La programmation structurée" du Guide Utilisateur.

COB	Cyclic Organisation Block	Bloc d'organisation cyclique
ECOB	End of Cyclic Org'n Block	Fin d’un bloc d'organisation cyclique
XOB	Exception Organisation Block	Bloc d'exception
EXOB	End of Exception Org'n Block	Fin d’un bloc d'exception
PB	Program Block	Bloc de programme
EPB	End of Program Block	Fin d’un bloc de programme
CPB	Call Program Block	Appel d'un bloc de programme
CPBI	Call Program Block Indirect	Appel indirect d'un bloc de programme
FB	Function Block	Bloc de fonction
EFB	End of Function Block	Fin d'un bloc de fonction
CFB	Call Function Block	Appel d'un bloc de fonction
NCOB	Next Cyclic Org'n Block	Passage au bloc d'org'n cyclique suivant
SCOB	Stop Cyclic Org'n Block	Arrêt d'un bloc d'organisation cyclique
CCOB	Continue Cyclic Org'n Block	Continuation d'un bloc d'org'n cyclique
RCOB	Restart Cyclic Org'n Block	Redémarrage d'un bloc d'org'n cyclique

Note :

Les instructions BLOCTEC suivantes sont extrêmement dangereuses :

NCOB, SCOB, CCOB, RCOB et le temps de supervision COB

Si ces instructions sont employées dans un programme utilisant GRAFCET de sérieux problèmes peuvent survenir.

Si elles ne sont pas utilisées avec la plus grande prudence, ces instructions peuvent causer au mieux, le ralentissement du programme utilisateur, ou au pis, une complète désynchronisation du GRAFTEC et un CRASH.

Evitez l'utilisation de ces instructions dans une structure GRAFTEC.

COB CYCLIC ORGANISATION BLOCK
Bloc d'organisation cyclique

Description En-tête d'un bloc d'organisation cyclique "COB" (Cyclic Organisation Block).
Le 2ème opérande est le temps de supervision en incrément de 10 msec.
Si le temps de supervision est écoulé avant la fin de l'exécution du COB (ECOB), le bloc d'exception XOB 11 est exécuté si il est présent; si il est absent, le COB suivant est exécuté.
Si le temps de supervision est 0, le XOB 11 n'est jamais exécuté, le COB suivant est démarré seulement quand le COB actuel est terminé (ECOB est atteint).
Si plusieurs COB sont programmés, ils sont exécutés les uns après les autres dans l'ordre numérique.

L'ACCU est toujours High (1) au début d'un COB.

Note : L'instruction COB utilise 3 lignes de programme.

Usage

COB	**numéro**	**; COB numéro 0-15**
	temps	**; temps de supervision en 10ms (0-100000)**

Exemple

```
COB     0       ; Début du COB 0
        0       ;    Temps de Supervision = 0
....            ;    Corps du COB 0
....
....
ECOB            ; Fin du COB 0
```

Flags L' **ACCU** est mis High (1) au début du COB.

Voir aussi ECOB, NCOB, RCOB, SCOB, XOB, Guide Utilisateur.

ECOB END OF CYCLIC ORGANISATION BLOCK
Fin d'un bloc d'organisation cyclique

Description Termine le COB courant.

L'exécution continuera au COB suivant (si présent).

Un COB se termine toujours par l'instruction ECOB.

Usage

ECOB	**; Pas d'opérande**

Exemple

```
COB     0       ; Début du COB 0
        0
...             ;    Corps du COB
...
...
ECOB            ; Fin du COB
```

Flags Inchangés.

Voir aussi COB, Guide Utilisateur.

XOB EXCEPTION ORGANISATION BLOCK
Bloc d'exception

Description Marque le début d'un bloc d'exception "XOB" (Exception Organisation Blocks)

Usage

XOB	**numéro**	**; XOB numéro 0-30**

Exemple

```
XOB     16      ; XOB de démarrage à froid
        ...     ;     Corps du XOB
EXOB            ; Fin du XOB
```

Flags L' **ACCU** est mis High (1) au début du XOB.

Voir aussi EXOB, Guide Utilisateur.

Les blocs d'exception " XOB " (Exception Organisation Blocks) sont appelés lorsqu'une erreur ou un autre événement important se produit.
Le XOB contiendra un programme pour réagir face à cet événement.
Lors de l'exécution d'un XOB, l'appel d'un XOB de même niveau ou de niveau inférieur est ignoré.
Si le XOB associé n'est pas présent, aucune action n'aura lieu (l'événement est ignoré) et la LED d'erreur (ERROR) sera enclenchée. A la fin du XOB, la routine d'exception retourne à l'endroit d'où elle a été appelée.

Chaque XOB a sa fonction spécifique :

XOB	Description	Priorité
0	Chute de tension (Power down)	4
8	Instruction non valide	4
7	Surcharge du système	3
11	Temps de supervision du COB dépassé	3
14	XOB Cyclique	3
15	XOB Cyclique	3
17	XOB Demande d'interruption S-Bus	3
18	XOB Demande d'interruption S-Bus	3
19	XOB Demande d'interruption S-Bus	3
20	Entrée d'interruption INB1 (PCD1/2)	3
25	Entrée d'interruption INB2 (PCD1/2)	3
1	Chute de tension (rack d'extension)	2
2	Batterie déchargée	2
4	Erreur de parité sur le bus (PCD6)	1
5	Pas de réponse d'un module E/S	1
6	Erreur externe	1
9	Trop de tâches actives (GRAFTEC)	1
10	Dépassement du niveau PB / FB	1
12	Dépassement capacité registre d'index	1
13	Flag erreur positionné	1
16	Exécuté au démarrage du PCD	1
30	Connexion RIO maître ↔ esclave	1

Exceptions de niveau 4 :

Le niveau 4 est la plus haute priorité, seul les XOBs 0 et 8 peuvent interrompre l'exécution d'un autre XOB.

XOB 0 Chute de tension (Power Down)

Il peut s'écouler jusqu'à 10 msec. entre l'appel du XOB 0 et l'arrêt du CPU; pendant cet intervalle, l'utilisateur peut sauver certaines valeurs.
Si le XOB 0 est programmé, le message "XOB 0 START EXEC" est écrit dans la liste historique au moment ou celui-ci démarre; le message "XOB 0 EXECUTED" est inscrit à la fin de l'exécution, indiquant ainsi que le XOB était terminé avant l'arrêt du CPU.
Si le XOB n'est pas programmé, un "Restart Cold" est effectué immédiatement à la détection de la chute de tension. Si le XOB 0 est programmé, le "Restart Cold" est effectué à la fin du XOB si il reste encore du temps.

XOB 8 Instruction non valide (Invalid Opcode)

XOB 8 est appelé quand le firmware détecte une instruction non valide dans le programme utilisateur.

Exceptions de niveau 3 :

Si une exception de niveau 2 ou 3 survient pendant l'exécution d'un XOB de priorité plus faible, il sera traité directement après l'exécution du XOB courant. Les XOBs 20/25/11 ont un niveau de priorité plus élevé. c.-à-d. que si ils sont provoqués pendant l'exécution d'un XOB de priorité égale ou inférieure, ils seront traités directement après la fin du XOB courant.

XOB 7 Surcharge du système (System Overload)

Le mécanisme de queue pour les XOBs niveau 3 a dépassé sa capacité.

XOB 11 Temps de supervision du COB dépassé

Si la seconde ligne de l'instruction COB comprend un temps de supervision (en 1/100 sec.) et si le temps d'exécution du COB excède ce temps, le XOB 11 est appelé. Le temps d'exécution d'un COB est le temps qui s'écoule entre les instructions COB et ECOB.

XOB 14 XOB Cyclique
XOB 15

Les XOBs 14 et 15 sont appelés périodiquement avec une fréquence variant entre 5 msec. et 1'000 sec. Cette fréquence est définie avec l'instruction SYSWR.

XOB 17 XOB Demande d'interruption S-Bus
XOB 18
XOB 19

Ces trois XOBs peuvent être utilisés comme routines d'interruption. Leur exécution est démarrée par le réseau S-Bus; il est également possible de les exécuter grâce à l'instruction SYSWR.

XOB 20 Entrée d'interruption
XOB 25

Le XOB 20 (resp. 25) est appelé lorsque l'entrée d'interruption INB1 (resp. INB2) du PCD1 ou PCD2 détecte un flanc montant (consultez le manuel hardware PCD1 - PCD2 pour plus de détails).

Exceptions de niveau 2 :

XOB 1 Chute de tension dans un rack (ou boîtier) d'extension

La surveillance de tension d'un module d'alimention d'un rack (ou boîtier) d'extension (PCD6 ou PCD2) a détecté une chute de tension.
Dans ce cas, toutes les sorties du rack d'extension sont remises à zéro en moins de 2 msec. et le XOB 1 est appelé.
Si les sorties de ce rack "mort" continue d'être traitées (enclenchement, déclenchement, lecture) par le programme utilisateur, les XOB 4 et/ou XOB 5 seront aussi appelés.
Le XOB 1 est appelé une seule fois, 250 msec. après détection de l'erreur.

XOB 2 Batterie déchargée

La batterie est déchargée ou manquante.
Les informations des flags non-volatiles, des Registres ou du programme utilisateur en RAM, ainsi que de l'horloge peuvent être altérées.
Le XOB 2 est seulement appelé par le CPU 0 toutes les 250 msec. en cas d'erreur.

Exceptions de niveau 1 :

Si une exception de niveau 1 survient lors du traitement d'une autre exception, celle-ci ne sera pas traitée.

XOB 4 Erreur de parité sur le bus (Parity Error)

Le XOB 4 peut seulement être appelé par un PCD possédant des racks d'extension (seulement PCD6).
Le circuit de surveillance du bus d'adresse a détecté une erreur de parité. Ceci peut être causé par un câble, un rack d'extension ou un module d'extension de bus défectueux; ou simplement parce que le rack d'extension adressé n'est pas présent.

XOB 5 Pas de réponse d'un module d'E/S (I/O Quit Failure)

Les modules d'entrées/sorties des PCDs renvoient un signal au CPU qui les a adressés. Si le signal n'est pas retourné, le XOB 5 est appelé.
Généralement, cela se produit lorsque le module n'est pas présent, mais cela peut aussi survenir en cas de décodage fautif sur le module.
Pour un module PCD4 ne comprenant que 8 éléments, le XOB 5 n'est pas appelé si un des éléments absent est adressé.

Ce mécanisme n'est pas implémenté sur les PCD1 et PCD2.

XOB 6 Erreur externe

Pas utilisé. (prévu pour les modules intelligents du PCD6)

XOB 9 Trop de tâches actives Graftec

Plus de 32 branches GRAFTEC sont actives simultanément dans un bloc séquentiel (SB).

XOB 10 Plus de 7 niveaux d'imbrication des PBs/FBs

PBs et FBs peuvent être imbriqués jusqu'à 7 niveaux. Un appel supplémentaire (8ème niveau) provoquera l'exécution du XOB 10.
Le 8ème niveau ne sera pas appelé.

XOB 12 Dépassement du Registre d'Index

Le XOB 12 est appelé lorsque le programme utilisateur contient un élément indexé en dehors de la plage du Registre d'index : 0..8191.

XOB 13 Flag Erreur positionné (Error Flag)

Le XOB 13 est toujours appelé lorsque le flag Erreur est positionné quelqu'en soit la cause (erreur de calcul, transfert de données, communication, …).

XOB 16 Démarrage à froid (Cold Start)

Le XOB 16 est le XOB exécuté au démarrage du PCD (Cold Start XOB).
Cet XOB n'est exécuté que lorsque le PCD est mis sous tension ou lorsque la commande "Restart Cold" est exécutée.
Le XOB 16 sert à l'initialisation d'éléments avant que le programme commence.
Si une erreur se produit lors de l'exécution du XOB 16, le XOB 13 n'est pas appelé.

XOB 30 Connexion RIO maître ↔ esclave

La connexion est testée après chaque message envoyé par le maître (master) vers un esclave (slave). Le CPU maître appelle l'XOB 30 si l'esclave ne répond pas positivement au test. Ceci est essentiellement le cas quand, "ONLINE", une station est déconnectée du réseau ou plus sous tension.

EXOB END OF EXCEPTION ORGANISATION
Fin de bloc d'exception

Description Termine le XOB courant.

Avec l'instruction EXOB, le XOB retourne à l'endroit d'où il avait été appelé.

Usage

EXOB	**; Pas d'opérande**

Exemple

```
XOB     16      ; Début du XOB 16
                ...     ;       Corps du XOB 16
                ...
                ...
EXOB            ; Fin du XOB 16
```

Flags Inchangés.

Voir aussi XOB, Guide Utilisateur.

PB PROGRAM BLOCK
Bloc de programme

Description Marque le début d'un bloc de programme " PB " (Program Block), une sous-routine sans paramètres.

Usage

PB	**numéro**	**; PB numéro 0-299**

Exemple

```
PB      26      ; Début du PB 26
        ...     ;     Corps du PB 26
        ...
        ...
EPB             ; Fin du PB 26
```

Flags L' **ACCU** est mis High (1) au début du PB.

Voir aussi EPB, CPB, FB, Guide Utilisateur.

Pratique

EPB END OF PROGRAM BLOCK
Fin d'un bloc de programme

Description Fin du Program Block (PB) courant.
Le retour s'effectue à l'instruction suivant l'instruction Call Program Block (CPB).

Usage

EPB	**; Pas d'opérande**

Exemple

```
PB      0       ; Début du PB 0
        ...     ;     Corps du PB 0
        ...
        ...
EPB             ; Fin du PB 0
```

Flags L' **ACCU** est positionné sur la valeur qu'il possédait avant que le PB soit appelé.

Voir aussi PB, CPB, Guide Utilisateur.

Pratique

CPB CALL PROGRAM BLOCK

Appel d'un bloc de programme

Description Appelle conditionnellement ou non un Program Block. Si la condition n'est pas satisfaite, le PB n'est pas appelé.

Condition	Le Program Block est appelé :
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
CPB     [cc] numéro     ; PB numéro 0-299
                        ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
CPB     10      ; appel inconditionnel du PB 10
```

Flags L'**ACCU** est mis High au début du PB.
Dans le programme d'où le PB est appelé, l'Accu est remis sur l'état qu'il avait avant l'appel.

Voir aussi PB, EPB, CFB, Guide Utilisateur.

Pratique Structure IF .. THEN .. ELSE.

```
COB         0
            0
...
STH       I 15    ; SI l'entrée 15 est High
CPB       H 20    ;    ALORS appel du PB 20
CPB       L 25    ;    SINON appel du PB 25
...
ECOB

PB          20
.....
EPB

PB          25
.....
EPB
```

CPBI CALL PROGRAM BLOCK INDIRECT
Appel indirect d'un bloc de programme

Description Appelle conditionnellement ou non un Program Block dont le numéro est contenu dans un Registre.
Comme cette instruction utilise un code conditionnel, le type de média 'R' ne doit pas être spécifié.
Si le Registre contient un numéro de PB non valide (>299), ou si le PB n'existe pas, le flag Erreur est positionné et le XOB 13 est appelé (si il est présent).

Si la condition n'est pas satisfaite, le PB n'est pas appelé.

Condition	Le Program Block est appelé :
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
CPBI    [cc] registre    ; Numéro du registre contenant le numéro
                         ;      du PB a appeler
                         ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
CPBI    10      ; Appel du bloc de programme dont le numéro est
                ; contenu dans le Registre 10
```

Flags Le flag **Erreu**r est positionné si le Registre donné contient un numéro de PB erroné ou si le PB n'existe pas.
L'**ACCU** est mis High au début du PB.
Dans le programme d'où le PB est appelé, l'Accu est remis sur l'état qu'il avait avant l'appel.

Voir aussi PB, EPB, CFB, Guide Utilisateur.

FB FUNCTION BLOCK
Bloc de fonction

Description Début d'un bloc de fonction " FB " (Function Block). Un FB est une sous-routine avec des paramètres optionnels.
Une liste des paramètres du FB peut être définie, cette liste est donnée lorsque le FB est appelé.

Usage

FB	**numéro**	**; FB numéro 0-999**

Exemple

```
FB      0       ; Début du FB 0
...
STH     =1      ; Référence au 1er paramètre
...
EFB             ; Fin du FB 0
```

Flags L' **ACCU** est mis High (1) au début du FB.

Voir aussi EFB, CFB, Guide Utilisateur.

Pratique Calcul de la formule: Z = X * (X+Y)

```
FB        25    ; Function Block  X * (X+Y)
ADD     = 1     ; Z = X + Y
        = 2
        = 3
MUL     = 3     ; Z = Z  *  X
        = 1
        = 3
EFB

COB       7
          0
...
STH     I 1     ; Si l'entrée 1 devient H
DYN     F 1
CFB     H 25    ;     Alors R107 = R100 * (R100+330)
        R 100   ; Paramètre 1  (X)
        K 330   ; Paramètre 2  (Y)
        R 107   ; Paramètre 3  (Z)
STH     I 2     ; Si l'entrée 2 devient H
DYN     F 2
CFB     H 25    ;     Alors R107 = R200 * (R200+R201)
        R 200   ; Paramètre 1  (X)
        R 201   ; Paramètre 2  (Y)
        R 107   ; Paramètre 3  (Z)
ECOB
```

EFB END OF FUNCTION BLOCK
Fin d'un bloc de fonction

Description Termine le Function Block courant. Retourne à l'instruction suivant l'instruction Call Function Block (CFB).

Usage

EFB	**; Pas d'opérande**

Exemple

```
FB      0       ; Début du FB 0
                ...     ;       Corps du FB 0
                ...
                ...
EFB             ; Fin du FB 0
```

Flags L' **ACCU** est positionné sur la valeur qu'il possédait avant que le FB soit appelé.

Voir aussi FB, CFB, Guide Utilisateur.

CFB CALL FUNCTION BLOCK
Appel d'un bloc de fonction

Description

Appelle conditionnellement ou non un Function Block.
Si la condition n'est pas satisfaite, le FB n'est pas appelé.
Une liste optionnelle de paramètres peut suivre l'instruction CFB.
Les paramètres sont utilisés par les instructions à l'intérieur du Function Block.

Les paramètres sont référencés en utilisant '= n' comme opérande, où 'n' est le numéro du paramètre à utiliser (1-128).
La valeur du paramètre est substituée comme opérande.

Type	Description	Plage de valeurs
I	Input	0..8191
O	Output	0..8191
F	Flag	0..8191
C	Counter	0..1599
T	Timer	0..450
R	Register	0..4095
K	K constant	0..16383
X	teXt	0..7999
DB	Data Block	0..7999
S	Semaphore	0..99
W	Word	0..65535 (Utilisé pour les constantes sans type LDL, LDH)
M	MOV data type	Q 0..31 D 0..9 N 0..7 B 0..3 W 0..1 L 0

Usage

```
CFB     [cc] numéro     ; FB numéro 0-999
                        ; cc = code conditionnel: H|L|P|N|Z|E
        [param 1]       ; liste optionnelle de paramètres
        [param 2]
        ...
        [param n]
```

Exemple

```
CFB     H 10    ; Appelle le FB 10 si l'ACCU est High
        I 32    ; Paramètre 1
        R 10    ; Paramètre 2
```

Flags

L' **ACCU** est mis High au début du FB.

Voir aussi

FB, CPB, Guide Utilisateur.

NCOB NEXT CYCLIC ORGANISATION BLOCK

Passage au bloc d'organisation cyclique suivant

Description Force conditionnellement ou non le programme à continuer au COB suivant.
Si la condition n'est pas satisfaite, l'instruction NCOB est ignorée.
Des boucles d'attente peuvent être programmées en utilisant l'instruction NCOB et ceci sans interférer avec l'exécution des autres COBs.
Pour chaque boucle d'attente, une instruction NCOB doit être insérée; ceci autorise l'exécution "en parallèle" des COBs.

Note : De bons programmes BLOCTEC ou GRAFTEC ne doivent pas utiliser de boucles d'attente et dès lors l'instruction NCOB. Les programmes doivent utiliser l'état de l'ACCU pour contrôler leur exécution.
Un processus séquentiel peut être aisément programmé en GRAFTEC.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

NCOB	**[cc]**	**; cc = code conditionnel: H|L|P|N|Z|E**

Exemple

```
STH     I 15    ; Attendre tant que I 15 = L
NCOB    L
JR      L -2
```

Flags Inchangés.

Voir aussi RCOB, SCOB, CCOB, Guide Utilisateur.

Les instructions BLOCTEC suivantes sont extrêmement dangereuses :

NCOB, SCOB, CCOB, RCOB et le temps de supervision COB

Si ces instructions sont employées dans un programme utilisant GRAFCET de sérieux problèmes peuvent survenir.

Si elles ne sont pas utilisées avec la plus grande prudence, ces instructions peuvent causer au mieux, le ralentissement du programme utilisateur, ou au pis, une complète désynchronisation du GRAFTEC et un CRASH.

Evitez l'utilisation de ces instructions dans une structure GRAFTEC.

SCOB STOP CYCLIC ORGANISATION BLOCK (ancien)

Arrêt d'un bloc d'organisation cyclique

Description Arrête conditionnellement, ou non, le COB référencé.
L' exécution continue au COB suivant.
Le COB n'est plus exécuté jusqu'au moment ou un autre COB exécute l'instruction CCOB appropriée.
Un COB peut s'arrêter, mais il doit être redémarrer par un autre COB contenant l'instruction CCOB.
Si la condition n'est pas satisfaite, l'instruction SCOB est ignorée.

Note : Un programme bien structuré ne doit pas utiliser cette instruction. Celle-ci doit être utilisée dans votre application avec précaution.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
SCOB    [cc] cob        ; COB numéro 0-15
                        ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
SCOB    L 10    ; Stoppe le COB 10 si l'ACCU est Low
```

Flags Inchangés.

Voir aussi CCOB, NCOB, RCOB, Guide Utilisateur.

Tableau : versions firmware

Type de PCD	SCOB "ancien" : FW ≤ V..	SCOB "nouveau" : FW ≥ V..
PCD1.M1x0	---	V001
PCD2.M110/M120	V003	V004
PCD2.M150	---	V0A0
PCD4.Mxx0	V005	---
PCD4.Mxx5	V00B	V00C
PCD4.M445	V001	V00C
PCD6.M540	V004	---
PCD6.M1/M2	V00A	---
PCD6.M300	---	V001

SCOB STOP CYCLIC ORGANISATION BLOCK (nouveau)

Arrêt d'un bloc d'organisation cyclique

Description Arrête conditionnellement, ou non, le COB référencé.
Si l'on effectue un SCOB du COB actif, l'exécution continue au COB suivant, ou sinon le programme continu à la ligne d'instruction suivante.
Le COB n'est plus exécuté jusqu'au moment ou un autre COB exécute l'instruction CCOB appropriée.
Si la condition n'est pas satisfaite, l'instruction SCOB est ignorée.

Note : Un programme bien structuré ne doit pas utiliser cette instruction. Celle-ci doit être utilisée dans votre application avec précaution.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
SCOB    [cc]  cob       ; COB  numéro 0-15
                        ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
SCOB    L  10    ; Stoppe le COB 10 si l'ACCU est Low
```

Flags Inchangés.

Voir aussi CCOB, NCOB, RCOB, Guide Utilisateur.

Les instructions BLOCTEC suivantes sont extrêmement dangereuses :

NCOB, SCOB, CCOB, RCOB et le temps de supervision COB

Si ces instructions sont employées dans un programme utilisant GRAFCET de sérieux problèmes peuvent survenir.

Si elles ne sont pas utilisées avec la plus grande prudence, ces instructions peuvent causer au mieux, le ralentissement du programme utilisateur, ou au pis, une complète désynchronisation du GRAFTEC et un CRASH.

Evitez l'utilisation de ces instructions dans une structure GRAFTEC.

CCOB CONTINUE CYCLIC ORGANISATION BLOCK

Continuation d'un bloc d'organisation cyclique

Description Autorise la reprise de l'exécution, conditionnellement ou non, un COB qui avait été arrêté par l'instruction SCOB.
Si la condition n'est pas satisfaite, l'instruction est ignorée.
CCOB ne provoque pas la reprise immédiate de l'exécution du COB, il sera seulement exécuté à son tour dans la liste des COBs.

Note : Un programme bien structuré ne doit pas utiliser cette instruction. Celle-ci doit être utilisée dans votre application avec précaution.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
CCOB    [cc] numéro     ; COB numéro 0-15
                        ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
CCOB    L 10    ; COB 10 est repris si l'ACCU est Low (0)
CCOB      0     ; COB 0 est repris inconditionnelement
```

Flags Inchangés.

Voir aussi NCOB, RCOB, SCOB, Guide Utilisateur.

Les instructions BLOCTEC suivantes sont extrêmement dangereuses :

NCOB, SCOB, CCOB, RCOB et le temps de supervision COB

Si ces instructions sont employées dans un programme utilisant GRAFCET de sérieux problèmes peuvent survenir.

Si elles ne sont pas utilisées avec la plus grande prudence, ces instructions peuvent causer au mieux, le ralentissement du programme utilisateur, ou au pis, une complète désynchronisation du GRAFTEC et un CRASH.

Evitez l'utilisation de ces instructions dans une structure GRAFTEC.

RCOB RESTART CYCLIC ORGANISATION BLOCK

Redémarre un bloc d'organisation cyclique

Description Redémarre un COB, conditionnellement ou non, à partir d'une ligne de programme donnée.
Cette instruction peut être utilisée dans un COB ou XOB.
Si la condition n'est pas satisfaite, l'instruction RCOB est ignorée.
Le 1er opérande est le numéro du COB a redémarrer.
Le 2ème opérande est le numéro de ligne à partir de laquelle le COB doit redémarrer. Le numéro de ligne est un offset depuis le début du COB, ce n'est PAS l'adresse absolue d'une ligne de programme.

Note : Un programme bien structuré ne doit pas utiliser cette instruction. Celle-ci doit être utilisée dans votre application avec précaution.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
RCOB    [cc] cob   ; COB numéro 0-15
        ligne      ; Numéro de ligne pour redémarrer le COB (0-65535)
                   ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
RCOB    0       ; Redémarre le COB 0
        10      ; L'exécution commence à la 10ème ligne du COB 0
```

Flags Inchangés.

Voir aussi NCOB, SCOB, CCOB, Guide Utilisateur.

Les instructions BLOCTEC suivantes sont extrêmement dangereuses :

NCOB, SCOB, CCOB, RCOB et le temps de supervision COB

Si ces instructions sont employées dans un programme utilisant GRAFCET de sérieux problèmes peuvent survenir.

Si elles ne sont pas utilisées avec la plus grande prudence, ces instructions peuvent causer au mieux, le ralentissement du programme utilisateur, ou au pis, une complète désynchronisation du GRAFTEC et un CRASH.

Evitez l'utilisation de ces instructions dans une structure GRAFTEC.

Notes personnelles :

7. Instructions GRAFTEC

Le SAIA® GRAFTEC est une méthode de programmation graphique auto-documentée pour les processus séquentiels.

Un programme GRAFTEC consiste en une séquence alternée d'étapes, nommées ci-après " STeps " (ST) et de " TRansitions " (TR). Cette séquence de STs et TRs forme le corps d'un bloc séquentiel " SB " (Sequential Block), qui est appelé à partir d'un Cyclic Organisation Block (COB).

Les **STEPS** contiennent les actions à exécuter en utilisant des instructions telles que SET, RES, STXT, etc.

Les **TRANSITIONS** contiennent les conditions à remplir en utilisant des instructions telles que STH, ANL CMP, etc.
Une TR doit toujours être suivie d'un ST. Le ST est seulement exécuté si la TR qui le précède est vérifiée (ACCU = 1).

Pour la programmation facile du GRAFTEC, l'éditeur SAIA® GRAFTEC EDITOR (SGRAF) est recommandé. Cet éditeur permet de manipuler la structure d'un programme que vous créez graphiquement à l'écran. Grâce à cet éditeur, vous n'avez pas besoin d'utiliser les instructions du SAIA® GRAFTEC de ce chapitre.

Pour plus d'informations sur la programmation GRAFTEC, référez-vous au chapitre "La programmation structurée" du Guide Utilisateur.

Les opérandes de ces instructions ne peuvent être passées comme paramètres de blocs de fonction (FB).

SB	Sequential Block	Bloc séquentiel
ESB	End of Sequential Block	Fin d'un bloc séquentiel
CSB	Call Sequential Block	Appel d'un bloc séquentiel
RSB	Restart Sequential Block	Redémarrage d'un bloc séquentiel
IST	Initial STep	Etape initiale
ST	STep	Etape
EST	End of STep	Fin d'une étape
TR	TRansition	Transition
ETR	End of TRansition	Fin d'une transition

Notes personnelles :

SB SEQUENTIAL BLOCK
Bloc séquentiel

Description Début d'un bloc séquentiel " SB " (Sequential Block).

Un Sequential Block contient un programme GRAFTEC indépendant.

Le SB contient seulement des instructions GRAFTEC telles que IST, ST, TR, EST, ETR et ESB.

Usage

SB	**numéro**	**; SB numéro 0-31**

Exemple

```
SB      10      ; Début du SB 10
        ...     ;     Corps du SB 10, contient des STs et TRs
ESB             ; Fin du SB 10
```

Flags Inchangés.

Voir aussi ESB, CSB, RSB, IST, ST, TR, Guide Utilisateur.

ESB END OF SEQUENTIAL BLOCK
Fin d'un bloc séquentiel

Description Fin du Sequential Block (SB) courant.

Usage

ESB	**; Pas d'opérande**

Exemple

```
SB      10      ; Début du SB 10
        ...     ;     Corps du SB 10, contient des STs et TRs.
ESB             ; Fin du SB 10
```

Flags L' **ACCU** est repositionné tel qu'il était avant que le SB soit appelé.

Voir aussi SB, ST, TR, Guide Utilisateur.

CSB CALL SEQUENTIAL BLOCK
Appel d'un bloc séquentiel

Description Appelle conditionnellement ou non un Sequential Block.

Si la condition n'est pas satisfaite, le SB n'est pas appelé.

Un bloc séquentiel ne peut être appelé depuis un autre bloc séquentiel.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
CSB     [cc] numéro     ; SB numéro 0-31
                        ; cc = code conditionnel: H|L|P|N|Z|E
```

Exemple

```
CSB     L 10    ; Appelle le SB 10 si l'ACCU est Low (0)
```

Flags L' **ACCU** est mis High (1) au début du SB.
Dans le programme d'où le SB est appelé, l'ACCU est repositionné tel qu'il était avant que le SB ne soit appelé.

Voir aussi SB, CPB, Guide Utilisateur.

RSB RESTART SEQUENTIAL BLOCK
Redémarrage d'un bloc séquentiel

Description

Redémarre conditionnellement ou non un Sequential Block (SB).

Le 1er opérande est le numéro du SB à redémarrer.
Le 2ème opérande est le numéro du STep à partir duquel le SB doit être redémarré.

Si le redémarrage doit avoir lieu au niveau de plusieurs branches simultanées (programmes parallèles), l'instruction comportera autant de lignes supplémentaires que d'étapes à redémarrer.

Codes conditionnels	
blanc	Toujours (pas de condition)
H	Si l'Accumulateur = H (1)
L	Si l'Accumulateur = L (0)
P	Si le flag Positif = H (flag Négatif = L)
N	Si le flag Négatif = H
Z	Si le flag Zéro = H
E	Si le flag Erreur = H

Usage

```
RSB     [cc] numéro     ; SB  numéro  0-31
                        ; cc = code conditionnel: H|L|P|N|Z|E
        step            ; STep numéro 0-1999
        [step]          ; [STep numéro 0-1999]
        [...]           ; [ ... ]
        [step]          ; [STep numéro 0-1999]
```

Exemple

```
RSB     12      ; Redémarre le SB 12 au Step 1.
        1
```

Flags

L' **ACCU** est mis High (1) avant de redémarrer le SB.

Voir aussi

SB, CSB, ST, Guide Utilisateur.

Pratique

L'instruction RSB est extrêmement dangereuse. Il n'y a aucune vérification des paramètres spécifiés dans cette instruction; ce qui peut causer au mieux, une complète désynchronisation du programme, ou au pis, un CRASH si ces paramètres sont mal choisis ou de façon aléatoire.

IST INITIAL STEP
Etape initiale

Description

L'étape initiale (Initial Step) défini la première étape devant être exécutée lorsque le Sequential Block (SB) est appelé.

Chaque SB doit avoir au moins une étape initiale (Initial Step).
A part cela, une étape initiale est identique à une autre étape (voir ST).

IST est suivie d'une liste de Transitions entrantes " I " (incoming) et sortantes " O " (outgoing).

Usage

IST	**numéro**	**; Initial Step numéro 0-1999**
	liste	**; Liste des transitions incoming et outgoing**
		; (longueur variable)

Exemple

```
IST     1       ; Initial Step 1.
        I 900   ;       Incoming de la Transition 900.
        O 1     ;       Outgoing vers la Transition 1
        ...     ;       Corps du ST 1
        EST     ; Fin du ST 1
```

Flags

L' **ACCU** est mis High (1) au début de l'Initial Step.

Voir aussi

EST, SB, ST.

ST STEP
Etape

Description

Début d'une étape " ST " (Step).

ST est suivie d'une liste de Transitions entrantes " I " (incoming) et sortantes " O " (outgoing).

Un STep contient seulement des instructions d'action: SET, RES, OUT, LD, MOV, FADD etc.

Il ne doit **JAMAIS** contenir de boucle d'attente.

Les STeps peuvent appeler des Program Blocks (PBs) et Function Blocks (FBs), ceux-ci ne peuvent contenir de boucles d'attente.

Avec le SAIA® GRAFTEC, dès qu'un Step a été exécuté, le pointeur de programme va à la TRansition suivante.

Les STeps peuvent seulement apparaître à l'intérieur d'un SB.

Usage

```
ST      numéro      ; Step numéro 0-1999
        liste       ; Liste des incoming et outgoing Transitions
                    ;      (longueur variable)
```

Exemple

```
ST          10   ; Step 10
          I 9    ;      Incoming de la Transition 9
          O 10   ;      Outgoing vers la Transition 10
          ...    ;      Corps du Step
EST              ; Fin du Step
```

Flags

L' **ACCU** est mis High (1) au début du ST.

Voir aussi

EST, IST, TR, SB, Guide Utilisateur.

EST END OF STEP
Fin d'une étape

Description Termine le Step ou Initial Step (ST ou IST).

Usage

EST	**; Pas d'opérande**

Exemple

```
ST      0     ; Début du ST 0
      I 25    ;     Incoming de la Transition 25
      O 47    ;     Outgoing vers la Transition 47
      ...     ;     Corps du ST 0
EST           ; Fin du ST 0
```

Flags

Voir aussi ST, TR, SB, Guide Utilisateur.

Pratique

TR TRANSITION
Transition

Description

Début d'une Transition " TR ".

TR est suivie d'une liste de Steps entrants " I " (incoming) et sortants " O " (outgoing).

Une Transition contiendra des instructions logiques formant une combinaison dont le résultat final indiquera si le Step suivant doit être exécuté.
Si le résultat final de la TRansition est faux (ACCU = L), alors le STep suivant n'est pas exécuté; l'exécution continue avec la branche parallèle suivante ou COB. Lors du prochain appel, l'entièreté de la TRansition sera traitée à nouveau. Le STep suivant n'est exécuté que lorsque le résultat final de la TRansition est vrai (ACCU = H).

Dans un choix de séquence, l'ordre dans lequel les TRs parallèles est donné par l'ordre dans lequel sont définies les outgoing TRansitions dans le STep précédent.

Les TRs peuvent seulement apparaître à l'intérieur d'un SB.

Usage

```
TR      numéro      ; Transition numéro 0-1999
        liste       ; Liste des incoming et outgoing Steps
                    ;      (longueur variable)
```

Exemple

```
TR          10   ; Transition numéro 10.
          I 900  ;     Incoming du Step 900
          O 1    ;     Outgoing vers le Step 1
                 ;     Outgoing vers le Step 2
          ...    ;     Corps de la TR 10
ETR              ; Fin de la Transition 10
```

Flags

L' **ACCU** est mis High (1) au début de la TR.

Voir aussi

ETR, SB, ST, Guide Utilisateur.

ETR END OF TRANSITION
Fin d'une transition

Description Fin de la Transition (TR) courante.

Usage

ETR ; Pas d'opérande

Exemple

```
TR          0     ; Début de la TR 0
          I 12    ;     Incoming du Step 12
          O 14    ;     Outgoing vers le Step 14
          ...     ;     Corps de la TR 0
ETR               ; Fin de la TR 0
```

Flags

Voir aussi TR, ST, SB, Guide Utilisateur.

Notes personnelles :

8. Instructions de COMMUNICATION

Ces instructions fonctionnent seulement avec les modules CPU possédant un ou plusieurs ports sériels.

Avant qu'une communication puisse avoir lieu, l'instruction SASI doit être exécutée pour chaque ligne série. Cette instruction configure le port quant à son mode opératoire et à sa vitesse (baud rate).

Chaque canal peut travailler indépendamment dans un mode et à une des vitesses différents. Chaque canal possède également ses propres tampons (buffers) d'émission et de réception .

Modes	Fonction	Instruction(s)
C	Emission/Réception de caractères ASCII	STXD, SRXD
	Envoi de textes	STXT
MD / SD	Emission/Réception d'éléments	STXM, SRXM
MM4	Emission/Réception de registres sur réseau LAC	STXM, SRXM
S-BUS	Emission/Réception d'éléments via le SAIA BUS	STXM(I), SRXM(I)
PROFIBUS	Emission/Réception via PROFIBUS	STXM(I), SRXM(I) SCON(I)
OFF	Dé-assignation d'une interface série	-

Les signaux de contrôle (CTS, RTS, DSR, DTR et DCD) des lignes séries peuvent être lus ou écrits avec les instructions SICL et SOCL.

SCON permet d'ouvrir ou de fermer un canal virtuel PROFIBUS.
Des communications via le LAN 1 sont également possible grâce à l'instruction SCON.

SASI	Assign Serial Interface	Assignation d'une interface série
SASII	Assign Serial Interface Indirect	Assignation indirecte d'une interface série
SRXD	Serial Receive character	Réception série d'un caractère
STXD	Serial Transmit character	Transmission série d'un caractère
STXT	Serial Transmit Text	Transmission série d'un texte
SRXM	Serial Receive Media	Réception série de données
SRXMI	Serial Receive Media Indirect	Réception série indirecte de données
STXM	Serial Transmit Media	Transmission série de données
STXMI	Serial Transmit Media Indirect	Transmission série indirecte de données
SICL	Serial Input Control Line	Lecture d'un signal de contrôle
SOCL	Serial Output Control Line	Positionnement d'un signal de contrôle
SCON	Open communication channel to LAN 1 and PROFIBUS	Ouverture d'un canal de communication pour le LAN 1 et PROFIBUS
SCONI	Open communication channel to LAN 1 and PROFIBUS Indirect	Ouverture indirecte d'un canal de communication pour le LAN 1 et PROFIBUS

MODE C

Mode Caractère ou Texte :

- Des caractères isolés provenant de Registres ou un Texte peuvent être émis.
- Des caractères isolés peuvent être reçus et transférés dans des Registres.
- Couramment utilisé pour communiquer avec un terminal ou une imprimante.

MCO **Mode C sans handshaking**

L'utilisateur doit contrôler lui-même les signaux de contrôle avec les instructions SICL et SOCL.

MC1 **Mode C avec handshaking RTS/CTS**

Le signal RTS est automatiquement positionné par le PCD en fonction de la place restante dans le tampon de réception.
Le signal CTS influence la transmission du PCD.

RTS	Low	Le tampon de Réception contient plus de 450 caractères
	High	Le tampon de Réception contient moins de 300 caractères
CTS	Low	La Transmission est arrêtée
	High	La Transmission est reprise

MC2 **Mode C avec protocole Xon/Xoff**

Ce mode est similaire au handshaking RTS/CTS et est utilisé lorsqu'il n'y a pas de signaux de contrôle (ex. boucle de courant).
Deux caractères spéciaux Xon (CTRL/Q) et Xoff (CTRL/S) sont émis pour contrôler la transmission du partenaire.

Récepteur émet lorsque		
	Xoff	Le tampon de Réception contient plus de 450 caractères
	Xon	Le tampon de Réception contient moins de 300 caractères
Emetteur reçoit lorsque		
	Xoff	La Transmission est arrêtée
	Xon	La Transmission est reprise

MC3 **Mode C avec Echo**

Ce mode est utilisé lors de communications avec un terminal : tous les caractères reçus sont ré-émis vers l'écran du terminal.

MC4/MC5 **Mode C pour interface RS 485**

Les modes MC4/MC5 sont des modes qui enclenchent la porteuse d'une interface RS 485 seulement pendant la transmission d'information (caractère/texte) et retournent en réception en d'autres moments.

Cependant le retour en mode réception ainsi que le traitement des flags XBSY et TBSY diffèrent entre les deux modes.

Mode MC4:

- Le retour en mode réception s'effectue entre 0 et 1ms après la fin de transmission du dernier caractère.
- Les flags XBSY et TBSY sont remis à 0 entre 0 et 1ms après la fin de transmission du dernier caractère.

Mode MC5:

- Le retour en mode réception s'effectue un temps/bit (temps nécessaire à l'envoie d'un bit) après la fin de transmission du dernier caractère.
- Les flags XBSY et TBSY sont remis à 0 entre 0 et 1ms après le début de transmission du dernier caractère.

MODE D

Utilise des télégrammes en accord avec ISO 1745, IBM BSC et DIN 66019.

Des données spécifiques du SAIA® PCD peuvent être échangées entre deux PCDs ou entre un PCD et un autre système intelligent (IBM PC, etc.) directement connecté ou via le SAIA® LAN 1.

Les données peuvent être l'état des entrées, sorties ou flags, ou le contenu des Registres, Temporisateurs ou Compteurs.

<mode>	**Description**
MD0	Mode D master (maître)
SD0	Mode D slave (esclave)

Les deux modes sont équivalents quant à leur fonctionnalité; la seule différence est que si un conflit survient lors de la communication, la station Maître a toujours la priorité sur l'esclave (Slave) pour répéter sa requête.

Lorsque la communication se passe avec un PC, le PCD doit être mis en esclave (Slave: mode SD0).

Pour la description complète du protocole, consultez la documentation "Functional specification for the SAIA® P8 Protocol".

SAIA LAN 1

Le mode D peut également être utilisé avec le SAIA® LAN 1, dans ce cas la connexion de 2 stations peut être réalisée avec l'instruction SCON.

Le statut de la connexion est automatiquement écrit dans un registre par le LAN 1; l'adresse de ce registre sera défini dans l'instruction SASI.

Pour plus d'informations sur le réseau SAIA® LAN 1, consultez la documentation "SAIA® LAN 1".

MODE MM4

Le mode MM4 permet la connexion du PCD sur un réseau LAC/LAC2 de COMPEX. Le LAC/LAC2 est un réseau local industriel qui permet la connexion entre des systèmes intelligents hétérogènes. Le PCD est connecté au réseau via un communicateur qui assure les services de transmission.

Le mode MM4 consiste en l'échange de Registres de 32 bits :
soit 4 caractères ASCII.
Jusqu'à 64 Registres peuvent être échangés dans un télégramme.

La détection des erreurs de transmission est assurée par un CRC-16.

Ce mode permet également la communication en point à point de deux PCD.

Pour plus d'informations sur le réseau LAC/LAC2, consultez la documentation "Description of the LAC MM4 Protocol".

MODE S-BUS

Le protocole SAIA® S-BUS permet l'échange d'informations à travers un réseau de type maître-esclave (single-client multiple-server network). Le client (Master) peut être un PCD ou un ordinateur pouvant réaliser le protocole S-BUS.

Le Serveur (Slave) est définit comme étant le système pouvant interpréter les commandes ou requêtes du Client : il s'agit d'un PCD ou d'un compteur de la gamme SAIA.

Le S-BUS est développé pour fonctionner sur un réseau RS 485 comprenant un client et un maximum de 32 serveurs. Grâce à l'utilisation d'un répéteur, il est possible d'ajouter des serveurs jusqu'à concurrence de 255. Le protocole S-BUS peut également s'utiliser pour une liaison point à point. Il est aussi possible d'utiliser le S-BUS avec un modem via l'interface RS 232 du PCD.

S-Bus possède les caractéristiques principales suivantes :

- Facilité d'utilisation (installation, mise en service et programmation)
- Economique : le protocole S-Bus est inclus dans tous les PCDs; ce qui signifie qu'il ne faut pas de processeur de communication séparé.
- Sûreté des transmissions, grâce à l'utilisation du CRC-16
- Vitesse de transfert élevée grâce à un protocole binaire optimisé et à des vitesses jusqu'à 38,4 kbps.
- Accès aux données et au diagnostique à distance via modem sur le réseau téléphonique public ou privé.
- Drivers de communication disponibles pour les systèmes de supervision tels que Wizcon, InTouch, FactoryLink, Fix D-Macs, Genesis, …
- La console de programmation a accès à toutes les stations du réseau. Cela signifie que n'importe quelle station esclave connectée au réseau peut être controlée par la console depuis un point central.
- Multi-maîtres grâce au S-Bus gateway

<mode>	**Description**
SM2	S-Bus maître, avec mode data
SM1	S-Bus maître, avec bit de parité
SM0	S-Bus maître, pas de parité, avec caractère break
SS2	S-Bus esclave, avec mode data
SS1	S-Bus esclave, avec bit de parité
SS0	S-Bus esclave, pas de parité, avec caractère break
GS2	S-Bus Gateway esclave, avec mode data
GS1	S-Bus Gateway esclave, avec bit de parité
GS0	S-Bus Gateway esclave, pas de parité, avec caractère break
GM	S-Bus Gateway Maître
OFF	Dé-initialisation de la ligne série

Pour plus d'informations, consultez le "Manuel SAIA® S-BUS" (réf. 26/739).

MODE PROFIBUS

PROFIBUS est le réseau de terrain le plus répandu, il peut être utilisé dans une vaste gamme d'applications.

PROFIBUS permet que des appareils provenant de différents fournisseurs puisse communiquer ensemble sans qu'il y ait besoin d'adapter les interfaces.

PROFIBUS est standardisé dans une norme européenne (EN50170).

PROFIBUS est indépendant des constructeurs et des appareils sont disponibles chez un grand nombre de fournisseurs.

PROFIBUS consiste en un assortiment de produits compatibles. Il y a 3 variantes de PROFIBUS, chacune correspondant à un domaine d'application :

- PROFIBUS-DP Decentral Periphery
- PROFIBUS-FMS Fieldbus Message Specification
- PROFIBUS-PA Process Automation

Les SAIA® PCDs fonctionnent avec PROFIBUS-FMS et DP (master/slave)

PROFIBUS-FMS est la solution universelle pour les communications entre appareils intelligents et les automates programmables, ainsi que pour l'échange d'informations entre les automates programmables.
L'utilisation d'un réseau PROFIBUS-FMS avec les SAIA® PCDs nécessite un processeur dédicacé : PCD4.M445 ou la carte PCD7.F700.

PROFIBUS-DP, optimisé au niveau rapidité, répond parfaitement aux besoins de communication entre les systèmes d'automatisation et les appareils périphériques décentralisés.
L'utilisation d'un réseau PROFIBUS-DP (master ou slave) avec les SAIA® PCDs nécessite une carte dédicacée : PCD7.F750, resp. ..F770.

La définition et la configuration (paramètres de bus, liste des canaux, dictionnaire des objets) d'un réseau PROFIBUS peut être une tâche complexe, fonction de la taille du projet. Pour rendre ce travail facile, SAIA a développé un configurateur PROFIBUS fonctionnant sur Windows.

Le configurateur génère un fichier contenant les textes de définition pour tous les canaux PROFIBUS. Ce fichier texte est utilisé par l'instruction SASI pour ouvrir la communication.

Pour plus d'informations, consultez les manuels :

"SAIA® PROFIBUS-FMS" (réf. 26/742 F)
"SAIA® PROFIBUS-DP" (réf. 26/765 F).

SASI ASSIGN SERIAL INTERFACE

Assignation d'une interface série

Description Initialise une ligne série.
Le 1er opérande est le numéro de la ligne série.
Le 2ème opérande est le numéro d'un Texte contenant la définition du mode opératoire de la ligne série (voir pages suivantes).
Cette initialisation doit être répétée pour chaque ligne série que l'on désire utiliser.
Généralement, les instructions SASI seront placées dans le XOB 16.
Les 4 lignes séries peuvent travailler dans des modes et à des vitesses différents.
Pour PROFIBUS, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Usage

SASI	**canal**	**; Ligne série numéro 0-3, [PROFIBUS: 10-99]**
	texte no	**; Numéro du Texte de Définition 0-3999**

Exemple

```
SASI    0       ; Initialisation ligne série 0
        100     ; Définitions dans le texte 100
```

Flags Le flag **Erreur** (E) est positionné si le texte de définition est non valide ou absent.

Voir aussi Textes SASI.

Note Pour PROFIBUS, le numéro du canal (10..99) est le canal virtuel appelé aussi CREF (communication relationship). Les informations nécessaires pour initialiser le canal sont contenues dans un texte qui est généré automatiquement par le configurateur PROFIBUS.

Pratique Initialisation de la première ligne série (numéro 0) en mode TEXT avec une vitesse de 4800 bps, 7 bits de données, parité paire et 1 bit de stop.

L'instruction SASI est placée dans le XOB 16.

```
        XOB     16      ; Cold Start Exception Organisation Block
        SASI    0       ; Assignation de la ligne série 0
                10      ;      avec les paramètres du texte 10
        EXOB

TEXT 10 "UART:4800,7,E,1;"
        "MODE:MC0;"
        "DIAG:F1000,R4000;"
```

Les Flags 1000 .. 1007 sont utilisés comme flags de diagnostique et le Registre 4000 comme Registre de diagnostique.

Le TEXTE peut également être écrit en une ligne :

```
TEXT 10  "UART:4800,7,E,1;MODE:MC0;DIAG:F1000,R4000;"
```

SASI Textes

Un texte spécial de définition est nécessaire pour l'instruction " SASI " Assign Serial Interface.

Format :

```
TEXT  xxxx    "<uart_def>;"
              "<mode_def>;"
              "<diag_def>;"
              ["<rx_buf>;"]
              ["<tx_buf>;"]
```

où xxxx est le numéro d'un texte (0..3999)

Ce texte peut également être écrit en une ligne.

Les différents paramètres sont :

<uart_def>	Définit la vitesse (baud rate), longueur des données, parité, ...
<mode_def>	Définit le mode opératoire (C, D, ...).
<diag_def>	Adresse des Flags et du Registre de Diagnostique

Les 2 derniers paramètres sont optionnels et sont seulement utilisés en mode C :

<rx_buf>	Taille du tampon de Réception (défaut = 1).
<tx_buf>	Taille du tampon de Transmission.

Mode OFF

Ce mode diffère des textes SASI standards et est principalement utilisé pour le PCD4 équipé de deux CPUs, les 4 lignes sérielles doivent être considérées comme un ressource commune. Chaque CPU peut assigner n'importe laquelle des 4 interfaces séries. Pour éviter les contentions entre les interfaces : un mécanisme de sémaphores a été implémenté.

Quand une interface série est assignée : un sémaphore est positionné de telle façon que si une autre assignation a lieu sur la même interface, la led "error" est enclenchée, l'instruction abandonnée et la routine d'exception "Error Flag" est appelée. Une interface série ne peut être ré-assignée qu'après qu'elle soit dé-assignée; ceci est réalisé en exécutant l'instruction SASI avec le texte suivant :

TEXT xxxx "MODE:OFF"

De la même façon, si un CPU essaie de dé-assigner une interface série qui l'est déjà : la routine d'exception "Error flag" est également appelée.

<uart_def>

Définit la vitesse (baud rate), taille du caractère (char_len), parité, nombre de bit de stop, timeout.

Format : "UART:<baud-rate>,<char_len>,<parity>,<stop_bit>[,<timeout>];

<baud_rate>	<char_len>	<parity>	<stop_bit>	<time_out>	ou défaut
110	7	E (even)	1	10..15000 ms	15 s
150	8	O (odd)	2		9 s
300		L (low)			5 s
600		H (high)			3 s
1.200		N (none)			2 s
2.400					1 s
4.800					0,5 s
9.600					0,25 s
19.200					0,2 s
38.400					0,1 s

<time_out> :

Le timeout n'est pas utilisé dans le Mode C.
Les timeouts par défaut sont donnés en secondes et sont fonction de la vitesse.
Dans les autres modes, le timeout est le temps après lequel un message est répété si le partenaire n'a pas acquitté (acknowledge) le message reçu. Si après 3 essais (retry), le partenaire ne répond pas, il est déclaré "not responding".

Mode SBUS :

Le mode S-BUS utilise toujours 11 bits par caractères (10 bits avec le mode data) : il n'y a donc pas de définition pour <char_len>, <parity>, <stop_bit>

"UART:<Baudrate>[,<Timeout>][,<TS-Delay>][,<TN-Delay>][,<Break-Length>];"

<baud_rate>: 110 .. 38.400
<time_out>: 10 .. 15000 ms
<TS delay>: 10 .. 15000 ms
<Break-Length>: 4 .. 15 char

TimeOut, TS-Delay, TN-Delay et Break-Length sont optionnels et ne sont normalement utilisés que dans des applications spéciales.
Pour plus d'informations consultez le manuel "SAIA® S-BUS" (réf. 26/739).

Exemple de texte <uart_def> :

"UART:9600,7,E,1;"

9600 bps, 7 bits données, Parité paire (Even), 1 bit de stop et timeout par défaut.

Note : Tous les caractères doivent être introduits en MAJUSCULES. Lorsque le texte "SASI" est entouré des directives $SASI et $ENDSASI, l'assembleur teste la validité du texte et convertit tous les caractères en majuscules.

<mode_def>

Définit le mode opératoire de la ligne série.

Format : "MODE:<mode>[,<mode_opt>];"

<mode_opt> est une série optionnelle de paramètres dépendant du mode.

<mode>	<mode_opt>	Description
MC0	---	Mode C sans handshaking automatique
MC1	---	Mode C avec handshaking RTS/CTS
MC2	---	Mode C avec protocole Xon/Xoff
MC3	---	Mode C avec écho
MC4	---	Mode C pour interface RS 485
MD0	---	Mode D Maître
	R xxxx (1)	via LAN1; Registre = statut de SCON
SD0	-	Mode D Esclave
	R xxxx (1)	via LAN1; Registre = statut de SCON
SM2	R xxxx (2)	Mode SBUS Maître (client) mode Data station
SM1	R xxxx (2)	mode Parité éloignée
SM0	R xxxx (2)	mode Break (remote)
SS2	---	Mode SBUS Esclave (server) mode Data
SS1	---	mode Parité
SS0	---	mode Break
GM	---	Mode SBUS Gateway Maître
GS2	---	Mode SBUS Gateway Slave mode Data
GS1	---	mode Parité
GS0	---	mode Break
MM4	(3)	Mode MM4
OFF	-	Dé-assignation de la ligne série

(1) Mode D avec LAN 1

Lors de l'utilisation du LAN 1, un registre doit être fourni pour permettre à l'interface PCA2.T9x d'informer le PCD du statut de la connexion. Pour plus d'informations, voir l'instruction SCON.

(2) Mode SBUS Client

L'adresse de la station partenaire à laquelle sont destinés les messages est donnée dans un registre.

(3) MM4 <mode_opt> consiste en

<BCS_opt>,<trpartner>,<trinfo>,<repartner>,<reinfo>,<rechar>

<mode_opt>	Value	Description
<BCS_opt>	0 or 1	Block Check Sum (0: pas de BCS, 1: CRC-16
<trpartner>	R xxxx	Numéro de la station partenaire (Transmission)
<trinfo>	R xxxx	Remote ACK information
<repartner>	R xxxx	Numéro de la station partenaire (Réception)
<reinfo>	R xxxx	Information reçue
<rechar>	R xxxx	Nombre de caractères reçus

<diag_def>

Définit les médias de diagnostiques pour la communication

Format : "DIAG:<dia_elem>,<dia_reg>;"

	Type	Description
<dia_elem>	O xxxx F xxxx	Adresse de base de 8 Flags (ou Sorties) consécutifs
<dia_reg>	R xxxx	Adresse d'un registre de diagnostique

où xxxx est une adresse valide

Les 8 Flags fournissent des informations sur l'état de fonctionnement de la ligne série. En cas d'erreur lors de l'exécution d'une instruction de communication, plus d'informations quant au type de cette erreur peuvent être obtenues en examinant le contenu du registre de diagnostique.

Flags de diagnostique

L'adresse des Flags ou sorties suivant la définition DIAG du texte SASI est l'adresse de base de 8 flags ou sorties consécutives, utilisées comme suit :

Adresse	Nom	Description
xxxx	**RBSY**	Receiver busy
xxxx+1	**RFUL**	Receive buffer full
xxxx+2	**RDIA**	Receiver diagnostic
xxxx+3	**TBSY**	Transmitter busy
xxxx+4	**TFUL**	Transmit buffer full
xxxx+5	**TDIA**	Transmitter diagnostic
xxxx+6	**XBSY**	Text Busy
xxxx+7	**NEXE**	Not executed

RBSY **Receiver Busy (Récepteur occupé)**

Mode	
C	RBSYest High lorsque un caractère au moins se trouve dans le buffer de réception. Quand tous les caractères en attente dans le buffer de réception ont été lus avec l'instruction SRXD : RBSY est remis à 0.
D **MM4**	RBSY est High tant que le récepteur est occupé.
S-BUS	RBSY est High quand une station esclave reçoit un télégramme. La flag est remis à 0 dès que la réponse a été envoyée.
PROFIBUS	Non utilisé.

RFUL **Receive Buffer Full (Tampon de réception plein)**

Mode	
C	RFUL est High quand le nombre de caractères présents dans le buffer de réception du PCD est égal ou supérieur à la valeur de rx_buf (Longueur du buffer de réception). RFUL est Low lorsque le nombre de caractères restant dans le buffer est inférieur à la valeur de rx_buf. Le buffer de réception interne du PCD a toujours de la place pour 512 caractères.
D **MM4**	RFUL est High quand un paquet de données correctes a été reçu.
S-BUS	RFUL est High quand la station Maître a modifié des données dans la station esclave
PROFIBUS	RFUL est High lorsqu'un télégramme d'écriture a été reçu.

RDIA **Receiver Diagnostic (Diagnostique de réception)**

Mode	
C **D** **MM4** **S-BUS** **PROFIBUS**	RDIA est High si le PCD détecte une erreur lors de la réception d'un caractère : plus d'informations concernant l'erreur peuvent être obtenues en examinant le contenu de registre de diagnostique de communication. RDIA est remis à 0 lorsque tous les bits (0...15) de diagnostique de réception seront à 0.

TBSY **Transmitter Busy (Emetteur occupé)**

Mode	
C	TBSY est High lorsque le PCD transmet des caractères sur la ligne série. TBSY est remis Low lorsque tous les caractères du buffer de transmission ont été transmis.
D **MM4** **S-BUS** **PROFIBUS**	TBSY est High lorsque le PCD exécute le transfert de données. TBSY est remis Low lorsque le message a été acquitté ou lorsque le nombre de retry est atteint.

TFUL **Transmit Buffer Full (Tampon de réception plein)**

Mode	
C	TFUL est High lorsque le nombre de caractères présents dans le buffer de transmission du PCD est plus grand ou égal à la valeur déclarée pour tx_buf (Longueur du buffer de transmission). TFUL est remis à 0 lorsque le nombre de caractères restant dans le buffer de transmission est inférieur à la valeur de tx_buf. Le buffer de transmission du PCD a toujours de la place pour 512 caractères.
D	Non utilisé
MM4	TFUL est High lorsque la quittance du télégramme envoyé a été reçue.
S-BUS PROFIBUS	Non utilisé.

TDIA **Transmitter Diagnostic (Diagnostique de transmission)**

Mode	
C D MM4 S-BUS PROFIBUS	TDIA est High si le PCD détecte une erreur lors de la transmission d'un caractère : plus d'informations concernant l'erreur peuvent être obtenues en examinant le contenu de registre de diagnostique de communication.

XBSY **Text busy (Texte occupé)**

Mode	
C	XBSY est High lorsque le PCD transmet un texte (STXT); quand tout le texte a été transmit, XBSY est remis à 0. Note : XBSY est remis à 0 au début de la transmission du dernier caractère.
D	XBSY est High aussi longtemps qu'une connexion via le LAN 1 est ouverte.
MM4	XBSY est High lorsqu'il y a une activité sur le réseau LAC (instruction STXM).
S-BUS	XBSY est mis Low lorsque l'utilisateur a le droit de faire un SASI OFF.
PROFIBUS	XBSY est mis High lorsque le canal virtuel est ouvert.

NEXE **Not executed (Non exécuté)**

Mode	
C	
D MM4 S-BUS PROFIBUS	Si le PCD est incapable de réaliser l'opération demandée, NEXE est mis High; plus d'informations concernant l'erreur peuvent être obtenues en examinant le contenu de registre de diagnostique de communication.

Registre de diagnostique

L'adresse d'un registre de diagnostique doit être donnée avec la définition DIAG du texte SASI. Normalement, les 32 bits de ce registre de diagnostique sont Low (0). Si un bit est High (1), le tableau suivant en donne la signification (dépendante du Mode) :

Bit	Description	Cause	MODE				
			C	D	MM4	S-BUS	PROFI BUS
0	Overrun error	Ne doit jamais arriver (avertir SAIA)	●	●	●	●	
1	Parity error	Caractère reçu avec erreur de parité	●	●	●		
2	Framing error	Causé par une vitesse incorrecte	●	●	●	●	
3	Break	Coupure de ligne	●	●	●	●	
4	BCC error	Block Check Code incorrect (ou CRC-16)		●	●	●	
5	S-Bus PGU status	S-Bus PGU avec modems				●	
6	End of transmit	Transmission terminée (SASI OFF)		●		●	
7	Overflow error	Débordement du buffer de réception	●	●	●		
8	Length error	Longueur de télégramme non valide				●	
9	Format error	Format de télégramme non valide		●	●		●
10	Address error	Adresse du ACK non valide			●	●	
11	Status error	PCD dans un état erronné				●	
12	Range error	Adresse d'élément non valide		●		●	●
13	Value error	Erreur dans la valeur reçue				●	
14	Missing media err	Adresse de média non définie ou invalide				●	
15	Program error	Lecture d'un buffer de réception vide	●				
		LAN1 non assigné ou n° station non valide		●		●	
16	Retry count	Indique le nombre de retry (en binaire)				●	
17							
18	Transmission off	Transm. suspendue (CTS = L ou XOFF)	●				
19							
20	NAK response	Réception d'un NAK		●	●	●	●
21	No response	Pas de réponse après attente du timeout		●	●	●	
22	Multiple NAK	Réception de NAK après plusieurs essais		●	●	●	
23	TX buffer full	Plus de place dans le buffer de transmission	●				
	TS Delay	Pas de CTS après TS Delay				●	
24	Enquiry error	Pas de réponse à ENQ après plusieurs essais		●			
25	Format error	Texte de définition non valide	●				
		Commande non valide		●			●
26	Partner error	Un problème est survenu avec le partenaire			●		
27	Network error	Un problème est survenu avec le réseau			●		
28	Range error	Adresse d'élément non valide		●	●	●	●
29							
30	Receive error	Erreur de réception		●			●
31	Program error	Essai de transmission sans autorisation		●	●	●	●

Remarque : <rx_buf> et <tx_buf> sont seulement utilisés en mode C

<rx_buf>

Définit la limite du tampon de réception (Receive buffer).

Format : "RBUF: <rbuf_len>;

	Valeur	**Description**
<rbuf_len>	1.. 511	Longueur du tampon de réception

Le tampon de réception a toujours de la place pour 512 caractères (8 bits).

En mode C, la définition RBUF (1-511) détermine à quel moment il faut mettre à 1 le flag "Receive Buffer Full" (RFUL).

Dans les autres modes, RBUF n'est pas utilisé.

<tx_buf>

Définit la limite du tampon de transmission (Transmit Buffer).

Format : "TBUF:<tbuf_len>;

	Valeur	**Description**
<tbuf_len>	1.. 511	Longueur du Tampon de Transmission

Similaire au tampon de réception.

Le tampon de transmission a toujours de la place pour 512 caractères (8 bits).

En mode C, la définition TBUF (1-511) détermine à quel moment il faut mettre à 1 le flag "Transmit Buffer Full" (TFUL).

Dans les autres modes, TBUF n'est pas utilisé.

Exemples de Textes SASI :

Mode MC0 à 9600 bps, 7 bits de data, parité paire (Even), 1 bit de stop, utilisation des flags F 1000.. F 1007 et du registre R 4000 pour le diagnostique.

```
TEXT 10     "UART:9600,7,E,1;MODE:MC0;DIAG:F1000,R4000;"
```

Mode MC2 à 4800 bps, 8 bits de data, pas de parité (none), 1 bit de stop, utilisation des flags F 0.. F7 et du registre R 100 pour le diagnostique; tampon de réception de 25 caractères.

```
TEXT 20     "UART:4800,8,N,1;MODE:MC2;DIAG:F0,R100;"
            "RBUF:25;"
```

Mode SD0 (Slave) à 9600 bps, 7 bits de data, parité paire (Even), 1 bit de stop, utilisation des flags F 1000.. F 1007 et du registre R 4000 pour le diagnostique.

```
TEXT 30     "UART:9600,7,E,1;MODE:SD0;DIAG:F1000,R4000;"
```

Mode MD0 (Master) à 9600 bps, 7 bits de data, parité paire (Even), 1 bit de stop, utilisation des flags F 1000.. F1007 et du registre R 4000 pour le diagnostique. Le SAIA LAN 1 est utilisé et un timeout de 3 sec. est nécessaire :

```
TEXT 40     "UART:9600,7,E,1,3000;"
            "MODE:MD0,R1;"
            "DIAG:F1000,R4000;"
```

Le registre R 1 est utilisé pour l'état de la connexion du LAN 1.

Mode MM4 à 9600 bps, 8 bits de data, pas de parité (None), 1 bit de stop, Timeout: 300 ms, pas de BCS, les registres R 100.. R 104 sont utilisés pour le numéro du partenaire, ... Les flags F 1000.. F 1007 et le registre R 1000 sont utilisés pour le diagnostique.

```
TEXT 50     "UART:9600,8,N,300;"
            "MODE:MM4,0,R100,R101,R102,R103,R104;"
            "DIAG:F1000,R1000;"
```

Mode S-BUS mode parité, station maître à 9600 bps, le registre R 555 est utilisé pour le numéro du partenaire, utilisation des flags F 8000.. F 8007 et du registre R 4095 pour le diagnostique.

```
TEXT 60     "UART:9600;MODE:SM1,R555;DIAG:F8000,R4095;"
```

Mode S-BUS mode parité, station esclave à 9600 bps, utilisation des flags F 8000.. F8007 et du registre R 4095 pour le diagnostique.

```
TEXT 70     "UART:9600;MODE:SS1;DIAG:F8000,R4095;"
```

Mode S-BUS mode data, station esclave à 9600 bps, le registre 55 est utilisé pour le numéro du partenaire, utilisation des flags F 8000.. F 8007 et du registre R 4005 pour le diagnostique

```
TEXT 80     "UART:9600;MODE:SS2,R55;DIAG:F8000,R4005;"
```

Mode PROFIBUS

```
SASI        10          ; canal 10
            T_As_10     ; SASI texte  (créé par le configurateur PROFIBUS)
```

Utilisation de symboles dans les textes SASI

Des symboles peuvent aussi être utilisés dans les textes SASI.

symbole [. [[-] [0] largeur] [t | T]]	
symbole	Le nom symbolique. Ceci peut être n'importe quelle expression qui inclut un symbole, par exemple: MotorOn + 100. Des symboles avec des valeurs en virgule flottante ne sont pas autorisés.
.	Le point qui suit immédiatement le symbole indique qu'une largeur de champ et/ou un préfixe est présent.
Largeur	La largeur du champ : le nombre de chiffres ou espaces requis pour le nombre. Si la largeur commence par un 0, des zéros sont insérés à gauche.
t | T	Préfixe optionnel de type 't' ou 'T'. Si 't', la valeur est préfixée par le type du symbole en minuscule (o, f, r, ...); si 'T', le type du symbole est en majuscule (O, F, R,...)

Exemple :

```
BAUD      EQU       9600
D_FLAGS   EQU     F 500
D_REG     EQU     R 4095


          XOB       16
          SASI      1
                    3999

TEXT 3999"UART:",BAUD,",7,E,1;MODE:MC0;"
                  "DIAG:",D_FLAG.T,",",D_REG.T,";"
          .
          EXOB
```

Ceci crée le texte

```
"UART:9600,7,E,1;MODE:MC0;"
"DIAG:F500,R4095;"
```

Nouveau SASI avec $ (le texte SASI accepte le signe $) voir page suivante
peut être utilisé à partir des versions firmware suivantes :

PCD1 :	V070	PCD4.Mxx5 :	V0E0	PCD6.M3 :	V030
PCD2 :	V080	PCD4.M445 :	V0E0		

$SASI, $ENDSASI

Ces directives de l'Assembleur peuvent être utilisées pour délimiter les textes utilisés par l'instruction SASI.

Tous les textes entourés par ces deux directives sont testés par l'assembleur et les erreurs détectées.

Format : **$SASI**
<définition du texte SASI>
...
$ENDSASI

Si $SASI .. $ENDSASI ne sont pas utilisés, il est possible d'introduire un texte non valide qui peut causer une initialisation incorrecte de la ligne série.

EXEMPLE:

```
        XOB     16
        ...
        SASI    0       ; Initialisation du canal 0
                100     ; en utilisant le texte 100
        ...
        EXOB

$SASI   ; Le Texte 100 sera testé par l'assembleur comme texte SASI

TEXT 100 "UART:9600,7,E,1;MODE:MC0;DIAG:F1000,R4000;"

$ENDSASI
```

Nouveau SASI avec $ (le texte SASI accepte le signe $)

ex. : `"UART:$Ra,$Rb,$Rc,$Rd;MODE:$Re,$Rf;DIAG:F$Rg,R$Rh;"`

Ra	Baudrate	110 .. 38400 (numérique)
Rb	Bits	7, 8 (numérique)
Rc	Parity	E, O, N (codé ASCII)
Rd	Stop	1 or 2 (numérique)
Re	Mode	'MC0', 'SM2' etc. (codé ASCII)
Rf	Station	Registre avec station S-Bus (numérique)
Rg	Diagnostic flags	Registre avec le numéro du flag de diagnostique de base (0 .. 8191 numérique)
Rh	Diagnostic register	Registre avec le numéro du registre de diagnostique (0 .. 4095 numérique)

Attention cette nouvelle fonction est seulement implémentée sur les nouvelles plates-formes PCD, voir tableau des firmware page précédente.

SASII ASSIGN SERIAL INTERFACE INDIRECT
Assignation indirecte d'une interface série

Description Initialise une ligne série ou un canal PROFIBUS en mode indirect. L'instruction travaille de la même façon que l'instruction SASI. La différence étant qu'elle travaille en mode indirect, c'est à dire que le numéro du canal et le numéro du texte de définition peuvent être donnés dans un registre.

Usage

SASI	**canal**	**; Numéro du canal ou du Registre**
	text definition	**; Registre**

Canal :

Numéro du canal à initialiser.

Ce paramètre peut être donné directement ou indirectement :

0..3	numéro du canal
10..99	numéro du canal PROFIBUS
R 0..4095	registre qui contient le numéro du canal (0..3, 10..99)

Text_definition :

Ce paramètre est une adresse de registre (R 0..4095)

Ce registre contient le numéro du texte dont les paramètres pour la ligne série sont définis. Adresses valables pour le texte :

0..3999	dans la mémoire standard
4000..7999	dans la mémoire d'extension

Exemple

```
SASI    0    ; Initialisation ligne série 0
        R 1  ; le numéro du texte de définition est dans R 1.
```

Flags Le flag **Erreur** (E) est positionné si le texte de définition est non valide ou absent.

Voir aussi Textes SASI.

SRXD SERIAL RECEIVE CHARACTER (Mode C)

Réception série d'un caractère (Mode C)

Description Transfère le caractère ASCII suivant du tampon de réception du canal série donné par le 1er opérande dans le Registre donné par le 2ème opérande.

L'instruction SRXD ne doit être exécutée que lorsque au moins un caractère est présent dans le tampon de réception : indiqué par RBSY = H; sinon le flag Erreur est mis à 1.

Après l'exécution de SRXD, les 8 bits les moins significatifs du Registre contiennent le caractère, les autres bits du Registre sont mis à 0.

Le tampon de réception peut contenir jusqu'à 512 caractères; chaque fois que SRXD est exécuté, le caractère suivant est lu. Si le tampon de réception "déborde" (plus de 512 caractères), il y aura une erreur de réception (le flag RDIA et le bit correspondant du registre de diagnostique sont mis à 1).

Usage

```
SRXD[X]   canal        ; Numéro du canal série 0-3
          reg    (i)   ; Registre pour recevoir le caractère R 0-4095
```

Exemple

```
SRXD        3     ; Lit un caractère du canal 3
          R 100   ; et le mémorise dans le Registre 100
```

Flags Le flag **Erreur** (E) est positionné si l'instruction SRXD est exécutée alors que le tampon est vide ou que l'interface n'a pas été correctement assignée ou n'existe pas.

Voir aussi STXD, SRXM, Instructions de Communications, Flags de Diagnostique.

Pratique Application typique en structure Bloctec.

```
....
STH       F RBSY          ; Si il y a un caractère en attente
CFB       H READ_CHAR     ;     Alors lecture de ce caractère
....

FB          READ_CHAR     ; FB Lecture d'un caractère
[STH      H RDIA]         ; Si il y a une erreur de réception
[CFB      H RCV_ERROR]    ;     Alors traitement de l'erreur
SRXD        0             ; Lecture du caractère sur le canal 0
          R 999           ;     et mémorisation dans R 999
....
EFB
```

Note : Dans les applications simples, le traitement de l'erreur (entre crochets ci-dessus) peut être omis.

STXD SERIAL TRANSMIT CHARACTER (Mode C)

Transmission série d'un caractère (Mode C)

Description

Le caractère contenu sur les 8 bits les moins significatifs du registres donnés comme 2ème opérande est placé dans le tampon de transmission du canal série donné par le 1er opérande. La transmission est effectuée automatiquement.

Le tampon de transmission peut contenir jusqu'à 512 caractères. Si il est vide (tous les caractères ont été transmis), le flag TBSY est mis Low. Tant qu'il y a des caractères en attente, TBSY reste High.

Si le flag TDIA est High après l'exécution de STXD, cela indique un problème (le registre de Diagnostique peut être examiné).

Usage

```
STXD[X]   canal        ; Numéro du canal série 0-3
          reg    (i)   ; Registre contenant le caractère R 0-4095
```

Exemple

```
STXD      1     ; Transmission du caractère du
        R 100   ; Registre 100 (bits 7-0) sur le canal 1
```

Flags

Le flag **Erreur** (E) est positionné si la ligne série n'a pas été correctement assignée ou n'existe pas.

Voir aussi

SRXD, STXT, Instructions de Communication, Flags de Diagnostique.

Pratique

Application typique en structure Bloctec :

```
....
STL       F TFUL          ; Si il y a de la place dans le tampon TX
CFB       H SEND_CHAR     ;    Alors envoi d'un caractère
....

FB        SEND_CHAR       ; FB Envoi d'un caractère
STXD        0             ; Envoi sur le canal 0
          R 900           ;    du caractère contenu dans R 900

[STH      H TDIA]         ; Si il y a une erreur
[CFB      H SND_ERROR]    ;    Alors traitement de l'erreur
....

EFB
```

Note : Dans les applications simples, le traitement de l'erreur (entre crochets ci-dessus) peut être omis.

STXT SERIAL TRANSMIT TEXT (Mode C)

Transmission série d'un texte (Mode C)

Description

Transmet le Texte donné par le 2ème opérande sur la ligne série indiquée dans le 1er opérande (0-3). Le flag XBSY est mis High et le texte est transmis automatiquement. XBSY est remis à 0 lorsque le Texte a été transmis. La sortie d'un long texte peut prendre plusieurs secondes; l'exécution normale du programme continue pendant que le texte est émis en tâche de fond.

Le flag XBSY indique la fin de la tâche de fond. Quand XBSY est High, aucune autre instruction de communication ne peut être exécutée sur la même interface série.

Les textes peuvent contenir des séquences spéciales de caractères assurant la transmission formatée de valeurs d'éléments, voir Chaîne de contrôle.

Le flag de diagnostique NEXE est positionné si le Texte contient une séquence de contrôle non valide.

Usage

```
STXT[X]   canal          ; Numéro du canal série 0-3
          text    (i)    ; Numéro du Texte 0-3999, 4000-7999
```

Exemple

```
STXT    0       ; Transmission du Texte 123 sur le canal 0
        123
```

Flags

Le flag **Erreur** (E) est positionné si le Texte ou le canal n'existe pas, ou si la ligne série n'a pas été correctement initialisée.

Voir aussi

Textes, STXD, SRXD, Instructions de Communication, Flags de Diagnostique.

Pratique

Quand l'entrée 1 est High, le texte suivant doit être émis : "Hello world !"

```
        XOB     16
        SASI    1       ; Initialisation canal série 1
                0       ;     avec les paramètres du Texte 0
        EXOB

$SASI
TEXT  0  "UART:9600,7,E,1;MODE:MC0;DIAG:F1000,R1000;"
$ENDSASI

        COB     0
                0
        STH     I 1     ; Si entrée 1 devient  High
        DYN     F 0
        ANL     F 1006  ; et pas de transmission en cours
        JR      L END   ;     (F 1006 = XBSY)
        STXT    1       ;  Alors envoi du Texte 10 sur le canal 1
                10
END:    ECOB

TEXT  10 "Hello world ! <10><13>"
```


Textes

Les textes peuvent être définis n'importe où dans le fichier programme (.SRC), mais ils sont placés dans un endroit pré-défini de la mémoire du PCD.
Un Texte peut être défini immédiatement après l'instruction qui l'utilise; ou tous les textes peuvent être définis dans un module source séparé.

Les règles suivantes doivent être observées :

- Un texte est défini par :

  TEXT n "Le texte proprement dit"

  où 'n' est le numéro du texte (0...3999)

- Le texte peut consister en plusieurs lignes, chaque lignes étant entourées de double apostrophes : " ..... "
  Les textes peuvent être de longueur quelconque.

- Les caractères de contrôles (code ASCII: 1..31), ou les caractères spéciaux (127..255) peuvent être introduits comme des valeurs décimales entourées de parenthèses angulaires : <nnn>.
  Par EXEMPLE : CR = <13>, LF = <10>, ESC = <27>, BELL = <7>, ...

- Les caractères ASCII standards (32..126) peuvent être introduits directement à partir du clavier.

Dans la mémoire du PCD, tous les textes sont terminés par le caractère NUL (code ASCII 0); ce caractère est ajouté automatiquement à la fin d'un texte par l'Assembleur. Pour cette raison, un texte ne peut contenir le caractère NUL. (Si le caractère NUL doit néanmoins être envoyé : il doit l'être indépendamment du texte en utilisant l'instruction STXD).

Exemples :

Les deux textes suivants donnent le même résultat :

```
TEXT 10     "The quick brown fox jumps over the lazy dog"

TEXT 11     "The quick brown fox"
            "jumps over the lazy dog"
```

Si après le texte l'imprimante doit aller à la ligne : les caractères line-feed et carriage return doivent être rajoutés :

```
TEXT 12     "The quick brown fox jumps over the lazy dog<CR><LF>"
```

Si vous possédez une imprimante EPSON, vous pouvez imprimer une partie du texte en gras en envoyant une séquence spéciale de caractères.
Par exemple pour imprimer le texte suivant :

`L'automate programmable` **`SAIA PCD4`** `n'a pas de limite.`

```
TEXT 13     "L'automate programmable"
            "<ESC>E SAIA PCD4 <ESC>F n'a pas de limite. <CR><LF>"
```

Note : Les caractères ESC E font imprimer une imprimante EPSON en gras, ESC F permet de retourner à l'impression normale.

Textes et variables

Les textes peuvent aussi contenir des variables comme l'horloge, le status d'une entrée, le contenu d'un registre, ...
Deux caractères ont une signification spéciale pour le PCD: $ et @

$ = ADRESSAGE DIRECT

L'adresse absolue de l'élément est donnée.

$H	Heure (Heure, Minute, Seconde) :	hh:mm:ss	
$HH	Heure (Heures seulement) :	hh	
$HM	Heure (Minutes seulement) :	mm	
$HS	Heure (Secondes seulement) :	ss	
$D	Date (Année, Mois, Jour) :	aa-mm-jj	
$d	Date (Jour, Mois, Année) :	jj.mm.aa	
$DD	Date (Jour seulement) :	dd	
$DM	Date (Mois seulement) :	mm	
$DY	Date (Année seulement) :	ss	
$W	Semaine (N° semaine, Jour de la semaine):	ss-jj	
$WN	Semaine (N° semaine seulement) :	ss	
$WD	Semaine (Jour seulement) :	jj	
$innnn	Etat logique d'une entrée (0, 1)		nnnn : adresse de l'élément (doit être 4 chiffres)
$onnnn	Etat logique d'une sortie (0, 1)		
$fnnnn	Etat logique d'un flag (0, 1)		
$Innnn	Etat logique de 8 entrées (nnnn à nnnn + 7)		nnnn : adresse du 1er élément (doit être 4 chiffres)
$Onnnn	Etat logique de 8 sorties (nnnn à nnnn + 7)		
$Fnnnn	Etat logique de 8 Flags (nnnn à nnnn + 7)		
$Cnnnn	Contenu d'un compteur		nnnn : adresse de l'élément (doit être 4 chiffres)
$Rnnnn	Contenu d'un registre		
$Tnnnn	Contenu d'un temporisateur		
$Lnnnn	incLusion d'un autre texte (3 niveaux maxi)		nnnn: numéro de texte (doit être 4 chiffres)
$xnn	Le caractère 'x' est répété 'nn' fois Le caractère ne peut pas être: (H|h|D|d|W|i|o|f|I|O|F|C|R|T|L|x)		nn : doit être 2 chiffres
$Annnn	Contenu d'un registre sous forme de caractères ASCII		nnnn : n° du registre (doit être 4 chiffres)

Exemple de $Annnn :

"$A0999"	avec R 999 = 00000000 hex	'NUL'
"$A0999"	avec R 999 = 00000061 hex	'a'
"$A0999"	avec R 999 = 00006162 hex	'ab'
"$A0999"	avec R 999 = 00616263 hex	'abc'
"$A0999"	avec R 999 = 61626364 hex	'abcd'

Les zéros à gauche ne sont pas envoyés. Un zéro ASCII est seulement envoyé si le byte de poids le plus faible est égal à 0.

@ = ADRESSAGE INDIRECT

L'adresse de l'élément est donné dans un Registre.

@innnn	Etat logique d'une entrée (0, 1)	nnnn : n° du registre (doit être 4 chiffres)
@onnnn	Etat logique d'une sortie (0, 1)	
@fnnnn	Etat logique d'un flag (0, 1)	
@Innnn	Etat logique de 8 entrées (add à add+7)	
@Onnnn	Etat logique de 8 sorties (add à add+7)	
@Fnnnn	Etat logique de 8 flags (add to add+7)	
@Cnnnn	Contenu d'un compteur	
@Rnnnn	Contenu d'un registre	
@Lnnnn	incLusion d'un autre texte (max 3 niveaux)	
@xnnnn	Le caractère 'x' est répété autant de fois que le contenu du Registre. Le caractère ne peut être: (H|D|W|i|o|f|I|O|F|C|R|L|x)	

Note : Pour imprimer un '$' il faut utiliser "$$", pour '@': "@@".

Exemple 1 :

```
TEXT  10    "Date: $D Heure: $H <LF><CR>"
            "$-50 <LF><CR>"
            "Entrées 0..7: $I0000 <LF><CR>"
            "Registre 100: $R0100 <LF><CR>"
            "$+20 <LF><CR>"
```

En supposant que ce texte est imprimé le 16 Août 90 à 9h, que les entrées 0 et 1 sont High et que le contenu du Registre 100 est 12345: ce qui suit sera imprimé:

```
Date: 90-08-16   Heure: 09:00:00
-------------------------------------------------
Entrées 0..7: 11000000
Registre 100: 12345
++++++++++++++++++
```

Exemple 2 :

Utilisation pratique de "$A...."
Le positionnement du curseur à l'écran peut être déterminé par 2 registres (coordonnées X et Y) : position X par le Registre R 1 (1..80)
position Y par le Registre R 2 (1..25)
La séquence "escape" pour le positionnement du curseur est:
<ESC><17><valeur de X><valeur de Y>
Il est donc possible de programmer: "....<ESC><17>$A0001$A0002...."

Par exemple pour donner une position fixe X=40 et Y=12, la séquence complète de 4 caractères peut être écrite dans un seul registre et envoyée avec $A...

Veuillez noter que les valeurs doivent être données en hexadécimal :
ESC =1B hex; 17 = 11hex; 40 = 28 hex (X); 12 = 0C hex (Y)
Chargement du registre R1000:

```
LD    R 1000
      1B11280CH
```

Texte pour le positionnement: `" ... $A1000 ..."`

Formatage

Le format d'impression du contenu de Registre ou de Compteur peut être spécifié dans le Texte. La largeur du champ et le nombre de décimales peut être spécifiés.

Une définition de format est introduite dans le texte par "$%xxxx", où 'xxxx' est le format requis, voir ci-dessous. Si une définition de format est utilisée, toutes les valeurs des Registres et Compteurs sont imprimés suivant ce format, jusqu'au moment ou une autre définition de format est rencontrée.

Dans les définitions de format suivant, le 'd/D' signifie 'décimal', le 'x/X' signifie 'hexadécimal' et le 'b/B' signifie 'binaire'. D'autre base pour le format comme o = Octal et f = floating point ne sont pas supportées. Si la valeur est trop large pour prendre place dans le champ défini, le format par défaut est utilisé (pas de formatage).

Définitions de format

Supposons que les Registres 10, 11 et 12 contiennent respectivement les valeurs suivantes : 123456, -7890 et 5

Pas de Format (defaut) :

La largeur du champ dépend de la valeur.

TEXT 0 "REGISTRE 10: $R0010 <LF><CR>"
"REGISTRE 11: $R0011 <LF><CR>"
"REGISTRE 12: $R0012"

Impression :
```
REGISTRE 10: 123456
REGISTRE 11: -7890
REGISTRE 12: 5
```

Champ de largeur fixe :

Utilisez la définition de format "$%xxd" ou "$%xxD", où 'xx' (1 - 99) est la largeur du champ.

"$%xxd": la valeur est cadrée à droite en ajoutant des espaces.

TEXT 1 "$%08dREGISTRE 10: $R0010 <LF><CR>"
"REGISTRE 11: $R0011 <LF><CR>"
"REGISTRE 12: $R0012"

Impression :
```
REGISTRE 10:  123456
REGISTRE 11:   -7890
REGISTRE 12:       5
```

"$%xxD": la valeur est cadrée à droite en ajoutant des zéros.

```
TEXT 1      "$%08DREGISTRE 10: $R0010 <LF><CR>"
            "REGISTRE 11: $R0011 <LF><CR>"
            "REGISTRE 12: $R0012"

Impression : REGISTRE 10: 00123456
             REGISTRE 11: -0007890
             REGISTRE 12: 00000005
```

Si le nombre comprend plus de chiffre que définit dans le format, il sera imprimé sans formatage.

Champ de largeur fixe et nombre de décimales fixes :

La valeur est cadrée à droite, mais le nombre de places décimales est toujours affichée, et est complétée si nécessaire par la droite avec des zéros.

Utilisez la définition de format "$%xx.yd", où 'xx' est la largeur totale du champ, et 'y' est le nombre de places à droite du point décimal.

```
TEXT 2      "$%07.3dREGISTRE 10: $R0010 <LF><CR>"
            "REGISTRE 11: $R0011"<LF><CR>"
            "REGISTRE 12: $R0012"

Impression : REGISTRE 10:   123.456
             REGISTRE 11:    -7.890
             REGISTRE 12:     0.005
```

Nombre de places décimales fixe :

Le nombre de places décimales est fixe mais la largeur du champ est dépendante de la taille du nombre.

Utilisez la définition de format "$%00.yd", où 'y' est le nombre de places décimales, complétées si nécessaire par la droite avec des zéros.

```
TEXT 2      "$%00.5dREGISTRE 10: $R0010 <LF><CR>"
            "REGISTRE 11: $R0011"<LF><CR>"
            "REGISTRE 12: $R0012"

Impression : REGISTRE 10:  1.23456
             REGISTRE 11: -0.00789
             REGISTRE 12:  0.00005
```

Suppression du formatage :

"$%00d" remet le format standard (pas de formatage).

Sauvegarde des définitions de format

Les définitions de format peuvent être sauvegardées en utilisant "$sn", où 'n' est le numéro de 'sauvegarde'. Jusqu'à 10 définitions de format peuvent être sauvegardées (0-9). Les formats sauvegardés sont activés en utilisant "$n", où 'n' est le numéro de 'sauvegarde' de la définition de format à activer.

Les formats peuvent être sauvés lors de la phase d'initialisation dans le XOB 16 (XOB de démarrage). Pour sauver un format, le texte contenant ce format doit être émis vers la ligne série avec l'instruction STXT.

Si on active un format qui n'a pas été sauvé, le format par défaut (pas de format) est utilisé.

Exemple :

```
          XOB       16
          ....
TEXT 991  "$%05.1d$s1"       ; Format 1  (nnn.n)
TEXT 992  "$%04.2d$s2"       ; Format 2  (n.nn)
TEXT 993  "$%08.3d$s3"       ; Format 3  (nnnn.nnn)

          SEI     K 0        ; Activation des définitions de format
DEF:      STH       XBSY
          JR      H DEF
          STXTX     0
                    991
          INI     K 2
          JR      H DEF
          ....
          EXOB

          COB       0
                    0
          ....
          STXT      1
                    10
TEXT 10   "Pompe  Litres    Prix/l    Total   <CR><LF>"
          "  1   $1$R0010  $2$R0011  $3$R0012 <CR><LF>
          "  2   $1$R0013  $2$R0014  $3$R0015 <CR><LF>
          ....
          ECOB
```

Ceci a le même effet que de formater le texte comme suit :

```
          " 1 $%05.1d$R0010  $%04.2d$R0011 $%8.3d$R0012"
          " 2 $%05.1d$R0013  $%04.2d$R0014 $%8.3d$R0015"
```

Résultats :

```
  Pompe  Litres    Prix/l    Total
    1      13.8     0.86      11.868
    2     158.2     0.95     150.290
```


Inclusion d'autres textes

La séquence "$Lnnnn" 'inclut' un autre texte qui est traité comme si il faisait partie du texte original.

Si ce texte inclus contient une nouvelle définition de format, ce format sera utilisé jusqu'à la fin du texte.

Au retour au texte original, le format original est ré-activé.

Exemple :

```
          COB     0
                  0
          ....
          STXT    1       ; Envoi du Texte 10
                  10
TEXT 10   "$L0100 Vitesse du moteur trop élevée<CR><LF>"
          ....
          STXT    1       ; Envoi du Texte 11
                  11
TEXT 11   "$L0100 Pression d'huile trop basse<CR><LF>"
          ...
          ECOB

TEXT 100 "ALARME moteur Diesel:"
```

Résultat :

```
ALARME moteur diesel: Vitesse du moteur trop élevée
ALARME moteur diesel: Pression d'huile trop basse
```

Utilisation de SYMBOLES dans les textes

Avec les "PCD Utilities" (V1.3 et après du PG3), des SYMBOLES peuvent être utilisés dans les textes. La valeur et optionnellement le type du symbole est inséré dans le texte. Le symbole est écrit à l'extérieur du texte ASCII qui est entouré de double apostrophes, et doit être séparé de celui-ci ou des autres symboles par une virgule. Après le symbole, une largeur de champ et un type peuvent éventuellement être donnés.

Format :

symbole [. [[-] [0] largeur] [t | T]]	
symbole	Le nom symbolique. Ceci peut être n'importe quelle expression qui inclut un symbole, par exemple: MotorOn + 100. Des symboles avec des valeurs en virgule flottante ne sont pas autorisés.
.	Le point qui suit immédiatement le symbole indique qu'une largeur de champ et/ou un préfixe est présent.
Largeur	La largeur du champ : le nombre de chiffres ou espaces requis pour le nombre. Si la largeur commence par un 0, des zéros sont insérés à gauche.
t | T	Préfixe optionnel de type 't' ou 'T'. Si 't', la valeur est préfixée par le type du symbole en minuscule (o, f, r, ...); si 'T', le type du symbole est en majuscule (O, F, R,...)

Exemples :

```
Flag    EQU     F 123
Output  EQU     O 32
Reg     EQU     R 999
TEXT 0  "$",Flag.04T        ; Texte 0: "$F0123"
TEXT 1  "",Flag             ; Texte 1: "123" (les "" sont nécessaires)
TEXT 2  "DIAG:",Output.T,",",Reg.T
                            ; Texte 2: "DIAG:O32,R999"
TEXT 3  "55:",Flag.T,"-",Flag+7,":",Output.T,"-",Output+7
                            ; Texte 3: "55:F123-130:O32-39"
TEXT 4  "FLAG Numéro: *",Flag.-8,"*"
                            ; Texte 4: "FLAG Numéro: *123    *"
```

Des symboles peuvent aussi être utilisée dans des textes SASI

```
D_FLAGS EQU     F 500
D_REG   EQU     R 4095

        XOB       16
        SASI      1
                  3999

TEXT 3999       "UART:9600,7,E,1;MODE:MC0;"
                "DIAG:",D_FLAG.T,",",D_REG.T,";"
```

Ceci crée le texte "DIAG:F500,R4095;"

Note : Les symboles (pour les textes) et les textes doivent être définis dans le même fichier.

SRXM SERIAL RECEIVE MEDIA (Mode D)

Réception série de données (Mode D)

Description Lit des éléments d'un PCD éloigné, et les copie dans les éléments destinations du PCD local. Les transferts peuvent être I|O|F vers O|F, ou R|T|C vers R|T|C.
Le 1er opérande est le numéro du canal.
Le 2nd opérande est le nombre d'éléments à transférer.
Le 3ème opérande est l'adresse la plus basse des éléments sources dans le PCD éloigné.
Le 4ème opérande est l'adresse la plus basse des éléments destinations dans le PCD local.
Après l'exécution de SRXM, le flag TBSY est mis High; il est remis Low lorsque l'opération est terminée.

Usage

```
SRXM[X]   canal          ; Numéro du canal série 0-3
          compte         ; Nombre d'éléments à recevoir 1-16
          source   (i)   ; Elément source I|O|F, R|T|C
          dest     (i)   ; Elément destination O|F, R|T|C
```

Exemple

```
SRXM      0     ; Lit le contenu de R 100-115
          16    ;      dans R 0-15 via canal série 0
        R 100
        R 0
```

Flags Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si SRXM est exécuté alors que le PCD était déjà en communication.

Voir aussi STXM, Instructions de Communication, Flags de Diagnostique.

Pratique Les entrées 0..15 du PCD éloigné doivent être copiées sur les sorties 32..47 du PCD local. (Les deux PCDs sont connectés via la ligne série).

PCD Local :

```
     XOB       16
     SASI      1
               0

TEXT 0   "UART:9600,7,E,1;MODE:MD0;"
         "DIAG:F1000,R1000;"

     EXOB

     COB       0
               0
     STH     F 1003   ; Si non occupé (TBSY)
     JR      H next
     SRXM      1      ; Alors réception sur canal 1
               16     ; 16 éléments
             I 0      ; I 0 (à 15) du PCD éloigné
             O 32     ; sur O 32 (à 47) du PCD local
next: ECOB
```

PCD éloigné :
(seule la ligne série doit être assignée)

```
     XOB       16
     SASI      1
               0

TEXT 0 "UART:9600,7,E,1;MODE:SD0;"
       "DIAG:F1000,R1000;"

     EXOB
```


SRXM SERIAL RECEIVE MEDIA (Mode MM4)

Réception série de données (Mode MM4)

Description

Copie le tampon de réception (télégramme reçu) dans une série de registres consécutifs du PCD. Lorsqu'un télégramme a été reçu correctement, RFUL est mis à 1; SRXM remet ce flag à 0.
Le 1er opérande contient le canal utilisé.
Le 2nd opérande est toujours à 0.
Le 3ème opérande est un registre ou un compteur qui contiendra après exécution de l'instruction le nombre de caractères lus.
Le 4ème opérande est l'adresse du premier registre où les caractères reçus doivent être copiés.
Chaque caractère reçu occupe 8 bits d'un registre et un registre contiendra 4 caractères au maximum.
Les caractères sont rangés dans les registres comme suit :

reg 1 :	11111111 22222222 33333333 44444444	1..4ème caractère
reg 2 :	55555555 66666666 77777777 88888888	5..8ème caractère

.....

Si le nombre de caractères reçus n'est pas un multiple de 4, le reste du dernier registre est mis à 0.

L'adresse du partenaire ayant envoyé le télégramme est contenu dans le registre <repartner> définit dans le texte de l'instruction SASI.

Usage

```
SRXM    canal     ; Numéro du canal série 0-3
        0         ; Non utilisé
        compte    ; Nombre de caractères lus R 0-4095 | C 0..1599
        reg       ; Adresse des registres de destination R 0-4095
```

Où :

compte — Registre ou compteur contenant (après exécution) le nombre de caractères lus.

reg — Adresse du premier registre dans lequel l'information sera copiée (un registre contient jusqu'à 4 caractères).
Le dernier registre utilisé sera : (reg 2 + [contenu de cnt/4]).

Exemple

```
SRXM    1       ; Transfère le télégramme reçu sur le canal 1
        0       ;    dans les registres R20 et consécutifs.
        C 100   ; C 100 : nombre de caractères reçus.
        R 20
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si SRXM est exécuté alors qu'aucun télégramme n'a été reçu.

Pour plus d'informations, consultez la documentation "Description of the LAC MM4 Protocol".

SRXM SERIAL RECEIVE MEDIA (Mode S-BUS)

Réception série de données (Mode S-BUS)

Pour plus d'informations, consultez le manuel "SAIA® S-BUS" (réf. 26/739).

Description

Lit des éléments d'une station PCD serveur, et les copie dans les éléments destinations du PCD client. L'adresse de la station serveur est donnée via un registre défini dans le texte SASI (voir page 8-11).
Les transferts peuvent être I|O|F vers O|F, ou R|T|C vers R|T|C.
Le 1er opérande est le numéro du canal.
Le 2ème opérande est le nombre d'éléments à transférer.
Le 3ème opérande est l'adresse la plus basse des éléments sources dans le PCD serveur.
Le 4ème opérande est l'adresse la plus basse des éléments destinations dans le PCD client.
Après l'exécution de SRXM, le flag TBSY est mis High; il est remis Low lorsque l'opération est terminée.

SRXM peut seulement être utilisé dans le PCD client.

Usage

```
SRXM[X]   canal          ; Numéro du canal série 0-3
          compte         ; Nombre d'éléments à recevoir
          source   (i)   ; Elément source I|O|F, R|T|C
          dest     (i)   ; Elément destination O|F, R|T|C
```

Où:

compte	1..32	nombre de R/T/C à lire	
	1..128	nombre de I/O/F à lire	
	0	Code de fonction spéciale	
	R nnnn	Utilisé pour le transfert de Data Block	
source	I/O/F	0..8191	
	R	0..4095	Adresse de base des éléments
	T/C	0..1599	dans le PCD esclave
	DB	0..7999	
	K	0..6000	Code de fonction spéciale
dest	I/O/F	0..8191	
	R	0..4095	Adresse de base des éléments
	T/C	0..1599	dans le PCD maître
	DB	0..7999	

Exemple

```
LD      R 100  ; Registre défini dans le texte SAS
          10   ;    pour le numéro de station serveur
SRXM      0    ; Lecture via le canal 0
          20   ;    de 20 éléments
        R 100  ;    de R 100-R 119 de la station 10
        R 0    ;    dans R 0-R 19
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si SRXM est exécuté alors que le PCD était déjà en communication.

Voir aussi

STXM, Instructions de Communication, Flags de Diagnostique.

La table suivante montre quels éléments peuvent être copiés de la station source vers les éléments appropriés de la station de destination.

		PCD Maître(destination)					
		O	**F**	**R**	**C**	**T**	**DB**
PCD Esclave (source)	**I**	•	•				
	O	•	•				
	F	•	•				
	R			•	•	•	•
	C			•	•	•	•
	T			•	•	•	•
	K			•			
	DB			•	•	•	

Fonctions spéciales

Code	Description		Exemples de résultat
K 0 ..7	**Lecture du statut du CPU**: 0..6: N° du CPU du PCD esclave 7: CPU local		R, C, H, S, D
K 1000	**Lecture de l'horloge**		format identique à RTIME
K 2000	**Lecture du "Display Register"**		
K 3000	**Lecture de la taille d'un Data Block**		
K 5000	**Lecture du type de PCD**	en ASCII	" D1" , " D2", " D4", …
K 5010		en décimal	1, 2, 4, …
K 5100	**Lecture du type de module**	en ASCII	" M1_", " M11", " M12", " M14",..
K 5110		en décimal	10, 11, 12, 13, 14, 24, …
K 5200	**Lecture version firmware**	en ASCII	" $4C", " 004", " X41", …
K 5210		en décimal	5, … or -1 dec for any '$', 'X', 'β'
K 5300	**Lecture n° CPU**	en ASCII	" 0", " 1"
K 5310		en décimal	0 .. 6
K 6000	**Lecture du numéro de station S-Bus** en mode BROADCAST Ceci ne fonctionne qu'en liaison point-à-point.		

Transfert de Data Blocks

Le format de l'instruction SRXM, lorsque l'on travaille avec les Data Blocks, diffère sensiblement du format normal. Pour adresser un élément d'un Data Block, il est nécessaire de spécifier le numéro du Data Block et la position de l'élément dans le Data Block.

SRXM **Canal**
Compte + Position
Source
Destination

Où Compte + Position : numéro de Registre.
Ce registre contient le nombre d'éléments à transférer (1..32) et la position dans le Data Block où le transfert doit avoir lieu. "Compte" doit être donné dans le mot de poids le fort et "Position" dans celui de poids le plus faible.

SRXM SERIAL RECEIVE MEDIA (PROFIBUS)

Réception série de données (Mode PROFIBUS)

Pour plus d'informations, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Description Lit des données (objets) de la station éloignée et les copie dans le PCD local.
Le 1er opérande est le numéro du canal.
Le 2ème opérande est le sous-index de l'objet source et destination.
Le 3ème opérande est le numéro de l'objet source (station éloignée).
Le 4ème opérande est le numéro de l'objet destination (station locale).

Usage

```
SRXM    canal     ; Numéro de canal 10-99
        compte    ; Sous-index 0-255
        source    ; Index objet source K 0-16383
        dest      ; Index objet destination K 100-499
```

Où:

canal	10..99	pour PCD4.M445
	10..19	pour PCD2 avec PCD7.F700
compte	0..255	Sous-index (0 = pas de sub-index)
source	Index de l'objet source (station éloignée) K 0..16383	
destination	Index de l'objet destination (station locale) K 100..499 PCD4.M445 K 100..199 PCD2 avec PCD7.F700	

Exemple

```
SRXM    13      ; Lit via le canal 10
        0       ;    pas de sous-index
      K 22      ;    copie l'objet 22 de la station éloignée
      K 150     ;    sur l'objet 150 de sa propre station
```

Flags Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si SRXM est exécuté alors que le PCD était déjà en communication.

Voir aussi STXM.

SRXMI SERIAL RECEIVE MEDIA INDIRECT (Mode S-BUS)

Réception série indirecte de données (Mode S-BUS)

Pour plus d'informations, consultez le manuel "SAIA® S-BUS" (réf. 26/739).

Description

Cette instruction fonctionne de la même manière que l'instruction SRXM. La différence est quelle travaille en mode indirect. Ce qui signifie que le nombre de média pour la source et la destination est donné via un registre.

SRXMI est seulement disponible pour le transfert de médias.
Le transfert de l'horloge, du Display-Registre, .. ne sont pas autorisés.

SRXMI ne peut pas être indexée ou paramétrisée.

Usage

SRXMI	**Canal**
	Compte ou Compte + Position
	Source-type et numéro de Registre
	Destination-type et numéro de Registrer

Canal

Ce paramètre est utilisé pour spécifier le numéro du canal (0...3).

Compte ou Compte + Position

Ce paramètre est un numéro de registre. Ce registre contient le nombre de médias ou le nombre et la position pour les Data Blocks.
Pour les Data-Blocks, "Compte" est donné dans le mot de poids le plus haut du registre et "Position" dans le mot de poids le plus faible.

Source-type et numéro de registre
Destination-type et numéro de registre

Ces paramètres spécifient la "Source" et la "Destination" du transfert. Chacun des paramètres est composé d'un caractère donnant le type de (I/O/F/R/T/C/DB) et d'un numéro de registre (0..4095). La source et la destination doivent respecter la compatibilité de la table pour les instructions SRXM/STXM.

Exemple

```
SRXMI    3      ; canal #3
         R 100  ; Nombre d'éléments dans R 100
         O 101  ; Source: sorties avec adresse de base dans R 101
         F 102  ; Destination: flags avec adresse de base dans R 102
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si SRXMI est exécuté alors que le PCD était déjà en communication.

Voir aussi

SRXM, Instructions de Communication, Flags de Diagnostique.

SRXMI SERIAL RECEIVE MEDIA INDIRECT (PROFIBUS)

Réception série indirecte de données (Mode PROFIBUS)

Pour plus d'informations, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Description Lecture de données (objets) en mode indirect de la station éloignée et les copie dans le PCD local.

Il est possible de sélectionner l'adressage direct ou indirect du canal.

SRXMI ne peut pas être indexée ou paramétrisée.

Usage

```
SRXMI   canal     ; Numéro du canal 10-99, R 0-4095
        compte    ; Sous-index R 0-4095
        source    ; Index objet source K [R 0-4095]
        dest      ; Index objet destination K [R 0-4095]
```

Où:

canal	10..99		pour PCD4.M445
	10..19		pour PCD2 avec PCD7.F700
	R 0..4095		pour l'adressage indirect
compte	R 0..4095		Registre contenant le sous-index (0 = pas de sous-index)
source	Registre contenant l'index de l'objet source (station éloignée)		
	K [R 0..4095]	0..16383	
destination	Registre contenant l'index de l'objet destination		
	K [R 0..4095]	100..499	PCD4.M445
		100..199	PCD2 pour PCD7.F700

Exemple

```
SRXMI   R 50    ; Numéro du canal dans R 50
        R 51    ;    sous-index dans R 50
        K 52    ;    index de l'objet source dans R 52
        K 53    ;    index de l'objet de destination dans R 53
```

Flags Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si SRXM est exécuté alors que le PCD était déjà en communication.

Voir aussi SRXM.

STXM SERIAL TRANSMIT MEDIA (Mode D)

Transmission série de données (Mode D)

Description

Transmet des éléments du PCD local vers des éléments du PCD éloigné.
Les transferts peuvent être I|O|F vers O|F, ou R|T|C vers R|T|C.
Le 1er opérande est le numéro du canal.
Le 2ème opérande est le nombre d'éléments à transférer.
Le 3ème opérande est l'adresse la plus basse des éléments sources dans le PCD local.
Le 4ème opérande est l'adresse la plus basse des éléments destinations dans le PCD éloigné.
Pendant l'exécution de STXM, le flag TBSY est mis High; il est remis Low lorsque l'opération est terminée.

Usage

```
STXM[X]   canal          ; Numéro du canal série 0-3
          compte         ; Nombre d'éléments à transmettre  1-16
          source   (i)   ; Elément source I|O|F, R|T|C
          dest     (i)   ; Elément destination O|F, R|T|C
```

Exemple

```
STXM       0    ; Transmet le contenu de R 100-115
           16   ;      dans R 0-15 via le canal série 0
         R 100
         R 0
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si STXM est exécuté alors que le PCD était déjà en communication.

Voir aussi

SRXM, Instructions de Communication, Flags de Diagnostique.

Pratique

Les entrées 0..15 du PCD local doivent être copiées sur les sorties 32..47 du PCD éloigné. Pour le PCD éloigné, seule la ligne série doit être assignée (voir SRXM)

PCD local :

```
          XOB       16
          SASI      1
                    15
$sasi
TEXT 15 "UART:9600,7,E,1;MODE:MD0;DIAG:F1000,R1000;"
$endsasi
          EXOB

          COB       0
                    0
          STH     F 1003   ; Si pas en train de transmettre (TBSY)
          JR      H NEXT
          STXM      1      ; Alors transfert via le canal 1
                    16     ;    16 éléments
                  I 0      ;    de l'entrée 0 (à 15) du PCD local
                  O 32     ;    vers la sortie 32 (à 47) du PCD éloigné
NEXT:     ECOB
```


STXM SERIAL TRANSMIT MEDIA (Mode MM4)

Transmission série de données (Mode MM4)

Description

Transfert des registres (caractères) en utilisant le protocole MM4.
Ce transfert peut se faire au travers d'un réseau LAC/LAC2 ou en point à point.
Le 1er opérande contient le canal utilisé.
Le 2ème opérande définit la fonction de transfert.
Le 3ème opérande est un registre ou un compteur qui contient le nombre de caractères à transmettre.
Le 4ème opérande est l'adresse du premier registre contenant les caractères à envoyer .
Chaque registre peut contenir 4 caractères; chaque caractère occupant 8 bits .
Les caractères doivent être rangés dans les registres comme suit :

reg 1 :	11111111 22222222 33333333 44444444	1..4ème caractère
reg 2 :	55555555 66666666 77777777 88888888	5..8ème caractère

.....

Le numéro du partenaire doit être contenu dans le registre <trpartner>, ce registre est défini dans le texte SASI.

Après l'exécution de STXM, le flag XBSY est mis High; il est remis Low lorsque l'opération est terminée (message acquitté par le partenaire).

Usage

```
STXM    canal     ; Numéro du canal série 0-3
        fct       ; Fonction à exécuter 0-4
        compte    ; Nombre de caractères à transmettre  R 0-4095
        reg       ; Adresse du registre source R 0-4095
```

Où :

fct	Fonction à exécuter		
	0 /2	Transmission de données	<trpartner> <> 0
	4	Diffusion	<trpartner> = 0
compte	Registre ou compteur contenant le nombre de caractères à transmettre.		
reg	Adresse du premier registre à partir duquel les informations doivent être transférées (un registre peut contenir jusqu'à 4 caractères).		

Exemple

```
STXM    1     ; Transmission sur le canal 1
        0     ;    indique une transmission
        C 100 ;    nombre de caractère à transmettre
        R 20  ;    1er registre contenant les informations
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si STXM est exécuté alors que le PCD est déjà en train de transmettre.

Pour plus d'informations, consultez la documentation "Description of the LAC MM4 Protocol"

STXM SERIAL TRANSMIT MEDIA (Mode S-BUS)

Transmission série de données (Mode S-BUS)

Pour plus d'informations, consultez le manuel "SAIA® S-BUS" (réf. 26/739).

Description

Transmet des éléments du PCD client vers des éléments du PCD serveur.
L'adresse de la station serveur est donnée via un registre défini dans le texte SASI (voir page 8-11).
Les transferts peuvent être I|O|F vers O|F, ou R|T|C vers R|T|C.
Le 1er opérande est le numéro du canal.
Le 2ème opérande est le nombre d'éléments à transférer.
Le 3ème opérande est l'adresse la plus basse des éléments sources dans le PCD serveur.
Le 4ème opérande est l'adresse la plus basse des éléments destinations dans le PCD client.
Pendant l'exécution de STXM, le flag TBSY est mis High; il est remis Low lorsque l'opération est terminée.

STXM peut seulement être utilisé dans le PCD client.

Usage

```
STXM[X]   canal          ; Numéro du canal série 0-3
          compte         ; Nombre d'éléments à transmettre
          source   (i)   ; Elément source I|O|F, R|T|C
          dest     (i)   ; Elément destination O|F, R|T|C
```

Où :

canal	0..3	Interface à utiliser	
compte	1..32	nombre de R/T/C à transférer	
	1..128	nombre de I/O/F à transférer	
	0	Code de fonction spéciale	
source	I/O/F	0..8191	
	R	0..4095	Adresse de base des éléments
	T/C	0..1599	dans le PCD maître
	DB	0..7999	
	K	4000	Fonction spéciale
destination	I/O/F	0..8191	
	R	0..4095	Adresse de base des éléments
	T/C	0..1599	dans le PCD esclave
	DB	0..7999	
	K	1000	Ecriture de l'horloge
	K	17, 18, 19	Fonction spéciale

Exemple

```
LD      R 100  ; Registre défini dans le texte SASI
          22   ;    pour le numéro de station de destination
STXM      0    ; Transmet via le canal 0
        100    ;    100 éléments
      F 100    ;    de F 100-F 199
      O 32     ;    vers sorties O 32-O 131 de la station 22
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si STXM est exécuté alors que le PCD était déjà en communication.

Voir aussi

SRXM, Instructions de Communication, Flags de Diagnostique.

La table suivante montre quels éléments peuvent être copiés de la station source vers les éléments appropriés de la station de destination.

		Slave PCD (destination)						
		O	**F**	**R**	**C**	**T**	**DB**	**Clock**
Master PCD (source)	I	•	•					
	O	•	•					
	F	•	•					
	R			•	•	•	•	•
	C			•	•	•	•	
	T			•	•	•	•	
	DB			•	•	•		

Pour l'écriture de l'horloge, deux registres sont émis. Pour le format des registres, voir l'instruction WTIME.

Fonction spéciale

Il est possible de provoquer l'exécution d'un XOB dans une station esclave en utilisant l'instruction STXM:

```
STXM      0..3          ; Numéro du canal série
          0             ; (doit être 0)
        K 4000          ; Utilisé pour indiquer une interruption XOB
        K 17|18|19      ; Numéro du XOB à exécuter
```

Il est possible d'utiliser cette instruction en mode "broadcast"; ceci permet de synchroniser des évènements.

Transfert de Data Blocks (Ecriture)

Le format de l'instruction STXM, lorsque l'on travaille avec les Data Blocks, diffère sensiblement du format normal. Pour adresser un élément d'un Data Block, il est nécessaire de spécifier le numéro du Data Block et la position de l'élément dans le Data Block.

STXM **Canal**
Compte + Position
Source
Destination

Où Compte + Position : numéro de Registre.

Ce registre contient le nombre d'éléments à transférer (1..32) et la position dans le Data Block où le transfert doit avoir lieu. "Compte" doit être donné dans le mot de poids le fort et "Position" dans celui de poids le plus faible.

STXM SERIAL TRANSMIT MEDIA (PROFIBUS)

Transmission série de données (Mode PROFIBUS)

Pour plus d'informations, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Description Ecrit des données (objets) de la station locale et les copie dans le PCD éloigné.
Le 1er opérande est le numéro du canal.
Le 2ème opérande est le sous-index de l'objet source et destination.
Le 3ème opérande est le numéro de l'objet source (station éloignée).
Le 4ème opérande est le numéro de l'objet destination (station locale).

Usage

```
STXM    canal     ; Numéro du canal 10-99
        compte    ; Sous-index 0-255
        source    ; Index objet source K 100-499
        dest      ; Index objet destination K 0-16383
```

Où:

canal	10..99	pour PCD4.M445
	10..19	pour PCD2 avec PCD7.F700
compte	0..255	Sous-index (0 = pas de sub-index)
source	Index de l'objet source (station locale)	
	K 100..499	PCD4.M445
	K 100..199	PCD2 avec PCD7.F700
destination	Index de l'objet destination (station éloignée)	
	K 0..16383	

Exemple

```
STXM    11      ; Transmet  via le canal 11
        3       ;    sous-index (élément) 3
      K 122     :    copie l'objet 122 de la station locale
      K 150     ;    sur l'objet 150 de la station éloignée
```

Flags Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si STXM est exécuté alors que le PCD était déjà en communication.

Voir aussi SRXM.

STXMI SERIAL TRANSMIT MEDIA INDIRECT (Mode S-BUS)

Transmission série indirecte de données (Mode S-BUS)

Pour plus d'informations, consultez le manuel "SAIA® S-BUS" (réf. 26/739).

Description

Cette instruction fonctionne de la même manière que l'instruction STXM. La différence est quelle travaille en mode indirect. Ce qui signifie que le nombre de média pour la source et la destination est donné via un registre.

STXMI est seulement disponible pour le transfert de médias.
Le transfert de l'horloge, du Display-Registre, .. ne sont pas autorisés.

SRXMI ne peut pas être indexée ou paramétrisée.

Usage

STXMI	**Canal**
	Compte ou Compte + Position
	Source-type et numéro de Registre
	Destination-type et numéro de Registrer

Canal

Ce paramètre est utilisé pour spécifier le numéro du canal (0...3).

Compte ou Compte + Position

Ce paramètre est un numéro de registre. Ce registre contient le nombre de médias ou le nombre et la position pour les Data Blocks.
Pour les Data-Blocks, "Compte" est donné dnas le mot de poids le plus haut du registre et "Position" dans le mot de poids le plus faible.

Source-type et numéro de registre
Destination-type et numéro de registre

Ces paramètres spécifient la "Source" et la "Destination" du transfert Chacun des paramètres est composé d'un caractère donnant le type de (I/O/F/R/T/C/DB) et d'un numéro de registre (0..4095). La source et la destination doivent respecter la compatibilité de la table pour les instructions SRXM/STXM.

Exemple

```
STXMI     1     ; Transmet via le canal 1
      R   100   ; Le nombre d'élément est contenu dans R 100
      DB  101   ; Source DB, adresse contenue dans le R 101
      R   102   ; Destination: R, adresse contenue dans le R 102
```

Flags

Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si STXMI est exécuté alors que le PCD était déjà en communication.

Voir aussi

STXM, Instructions de communication, Flags de Diagnostique.

STXMI SERIAL TRANSMIT MEDIA INDIRECT (PROFIBUS)

Transmission série indirecte de données (PROFIBUS)

Pour plus d'informations, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Description Ecriture de données (objets) en mode indirect dans la station éloignée et les copie dans le PCD local.

Il est possible de sélectionner l'adressage direct ou indirect du canal.

STXMI ne peut pas être indexée ou paramétrisée.

Usage

```
STXMI   canal     ; Numéro de canal 10-99, R 0-4095
        compte    ; Sous-index R 0-4095
        source    ; Index objet source K [R 0-4095]
        dest      ; Index objet destination K [R 0-4095]
```

Où:

canal	10..99		pour PCD4.M445
	10..19		pour PCD2 avec PCD7.F700
	R 0..4095		pour l'adressage indirect
compte	R 0..4095		Registre contenant le sous-index (0 = pas de sous-index)
source	Registre contenant l'index de l'objet source (station locale)		
	K [R 0..4095]	100..499	PCD4.M445
		100..199	PCD2 pour PCD7.F700
destination	Registre contenant l'index de l'objet destination		
	K [R 0..4095]	0..16383	

Exemple

```
STXMI   R 100  ; Numéro de canal dans R 100
        R 110  ;    sous-index dans R 110
        K 200  ;    index objet source dans R 200
        K 300  ;    index objet destination dans R 300
```

Flags Le flag **Erreur** (E) est positionné si le canal n'est pas correctement initialisé ou si STXMI est exécuté alors que le PCD était déjà en communication.

Voir aussi STXM.

SICL SERIAL INPUT CONTROL LINE

Lecture d'un signal de contrôle

Description Lit un signal de contrôle du canal série donné dans le 1er opérande et mémorise son état dans l'ACCU.

Le 2ème opérande est le signal à lire :

0 =	CTS	Clear To Send	(Prêt pour émettre)
1 =	DSR	Data Set Ready	(Appareil périphérique prêt)
2 =	DCD	Data Carrier Detect	(Porteuse détectée)

Pour le Port n° 0 (PGU) du PCD1, PCD2, PCD4, PCD6.M540 et PCD6.M300, l'instruction SICL est toujours autorisée, même si le port n'a pas été assigné ou configuré.

Pour tout autre port, l'instruction SICL est seulement autorisée pour un port configuré en S-Bus PGU (indépendamment que le port est assigné).

Autrement, l'instruction SICL est seulement autorisée après l'exécution de l'instruction SASI.

Usage

```
SICL    canal     ; Numéro du canal série 0-3
        signal    ; Numéro du signal CTS|DSR|DCD  (0|1|2)
```

Exemple

```
SICL     0      ; Si DSR du canal 0 est High
         1
CPB      H 25   ;    Alors appel du PB 25
```

Flags L'**ACCU** est positionné suivant l'état du signal de contrôle adressé.
Le flag **Erreur** (E) est positionné si le canal n'existe pas ou n'a pas été correctement initialisé.

Voir aussi SOCL, Instructions de Communication.

SOCL SERIAL OUTPUT CONTROL LINE

Positionnement d'un signal de contrôle

Description Positionne un signal de contrôle du canal série donné dans le 1er opérande suivant l'état de l'ACCU (H ou L).

Le 2ème opérande est le signal a positionner :

0 = RTS	Request To Send	(Demande pour émettre)
	(RS 485 : ACCU High	= transmission
	ACCU Low	= réception)
1 = DTR	Data Terminal Ready	(Appareil prêt)
2 =	Fonction spéciale	

Pour le Port n° 0 (PGU) du PCD1, PCD2, PCD4, PCD6.M540 et PCD6.M300, l'instruction SOCL est toujours autorisée, même si le port n'a pas été assigné ou configuré.

Pour tout autre port, l'instruction SOCL est seulement autorisée pour un port configuré en S-Bus PGU (indépendamment que le port est assigné).

Autrement, l'instruction SOCL est seulement autorisée après l'exécution de l'instruction SASI.

Usage

```
SOCL    canal     ; Numéro du canal série  0-3
        signal    ; Numéro du signal RTS|DTR (0|1) ou 2
```

Exemple

```
SOCL    0   ; Positionne le signal DTR du canal 0
        1   ;     suivant l'état de l'ACCU
```

Flags Le flag **Erreur** (E) est positionné si le canal n'existe pas ou n'a pas été correctement initialisé.

Voir aussi SICL, Instructions de Communication.

Fonctions spéciales :

Port 0 du PCD2

Une instruction SASI en mode SM1/SS1 dans le programme utilisateur configure le port n° 0 en RS 485. Si l'utilisateur désire utiliser le port n° 0 en RS 232, il doit alors exécuter ce qui suit *après l'instruction SASI :*

```
ACC    L        ; ACCU à 0
SOCL   0        ; Positionnement du signal « fonctions spéciales »
       2        ;     du port n° 0 selon l'état de l'ACCU (→ 0).
```


Basculement de RS 485 vers RS 422

L'interface série RS 422/RS 485 sur les modules de communication (..F..) PCD2.F520/F530 et PCD7.F110/F150 ainsi que sur le module de bus PCD4.C130 commute automatiquement vers RS 485 lorsque certains modes sont assignés :

Mode	**Type**
MC0 .. MC3, MD0 / SD0	RS 422
MC4, S-Bus	RS 485

Parfois, il est nécessaire de forcer le PCD pour utiliser S-Bus en RS 422; dans ce cas, on doit alors exécuter ce qui suit *après l'instruction SASI* :

```
ACC    L            ; ACCU à 0
SOCL   Port_nb      ; Positionnement du signal « fonctions
       2            ;     spéciales » du port n selon l'état
                    ;     de l'accumulateur (→ 0).
```

A l'inverse, il est aussi possible de forcer l'utilisation de RS 485 avec les modes MC0..MC3 ou MD0/SD0 :

```
ACC    H            ; ACCU à 1
SOCL   Port_nb      ; Positionnement du signal « fonctions
       2            ;     spéciales » du port n selon l'état
                    ;     de l'accumulateur (→ 1).
```

Basculement du mode réception à transmission en RS 485

Les instructions suivantes doivent être exécuter après le SASI :

- Commuter RS 485 en mode transmission

```
ACC    H            ; ACCU à 1
SOCL   Port_nb
       0
```

- Commuter RS 485 en mode réception

```
ACC    L            ; ACCU à 0
SOCL   Port_nb
       0
```


SCON SERIAL CONNECT TO LAN 1
Connexion au LAN 1

Description Etablit ou supprime une connexion virtuelle avec une autre station sur le SAIA® LAN 1.
Le 1er opérande est le numéro du canal série.
Le 2ème opérande est le numéro de station (1-250).
Le 3éme opérande est l'état de la connexion (0 = Fermeture, 1 = Ouverture).

L'état de la connexion est écrit dans le Registre défini dans le texte SASI à la ligne MODE (Exemple: "MODE:SD0,R4000;").

Etat de la connexion comme retourné dans le Registre:

Valeur	Description
0	Déconnecté
1	Connecté
2	Mis en attente
3	Destination occupée
4	Destination inconnue
6	PLC éloigné non connecté
10..2500	Si la connexion a été effectuée par un PCD éloigné : le registre contient le numéro du PLC multiplié par 10.

Pour plus d'informations, consultez la documentation du SAIA® LAN 1

Usage

```
SCON    canal     ; Numéro du canal série 0-3
        station   ; Numéro de la station LAN 1 1-250
        statut    ; Etat de la connexion (0=fermeture, 1=ouverture)
```

Exemple

```
SCON    0     ; Ouvre une connexion vers la station 100
        100   ;     sur le canal 0
        1
```

Flags Le flag **Erreur** (E) est positionné si le canal n'existe pas ou n'a pas été correctement initialisé.

Voir aussi SASI, Instructions de Communication.

SCON OPEN COMMUNICATION CHANNEL (PROFIBUS)

Ouverture d'un canal de communication (PROFIBUS)

Pour plus d'informations, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Description Ouvre ou ferme un canal virtuel de communication PROFIBUS.
Avant de pouvoir échanger des informations, il est nécessaire d'initialiser (ouvrir) un canal virtuel de communication avec l'instruction SCON.

Le 1er opérande est le numéro de canal.
Le 2nd opérande est toujours 0 (non utilisé).
Le 3ème opérande est le statut de la connexion (0 = Fermé, 1 = Ouvert).

Usage

```
SCON    canal    ; Numéro de Canal PROFIBUS 10-99
        0        ;
        statut   ; Etat de la connexion (0=fermeture,
1=ouverture)
```

Exemple

```
SCON    10      ; Ouverture du canal 10
        0       ;
        1       ;
```

Flags Le flag **Erreur** (E) est positionné si le canal n'existe pas.

SCONI SERIAL CONNECT TO LAN 1 INDIRECT
Connexion au LAN 1 indirecte

Description Etablit ou supprime de manière indirecte une connexion virtuelle avec une autre station sur le SAIA® LAN 1.
Le 1er opérande est le numéro du canal série ou un registre contenant le numéro de la ligne série.
Le 2ème opérande est un registre contenant le numéro de station (1-250).
Le 3éme opérande est un registre contenant l'état de la connexion
(0 = Fermeture, 1 = Ouverture).

SCONI ne peut pas être indexée ou paramétrisée.

L'état de la connexion est écrit dans le Registre défini dans le texte SASI à la ligne MODE (Exemple: "MODE:SD0,R4000;").

Etat de la connexion comme retourné dans le Registre:

Valeur	Description
0	Déconnecté
1	Connecté
2	Mis en attente
3	Destination occupée
4	Destination inconnue
6	PLC éloigné non connecté
10..2500	Si la connexion a été effectuée par un PCD éloigné: le registre contient le numéro du PLC multiplié par 10.

Pour plus d'informations, consultez la documentation du SAIA® LAN 1

Usage

```
SCONI     canal      ; Numéro de canal série 0-3 or R 0-4095
          station    ; Numéro de station LAN 1 R 0-4095
          statut     ; Statut de la Connexion R 0-4095
```

Exemple

```
SCONI     0   ;
        R 100 ;
        R 101 ;
```

Flags Le flag **Erreur** (E) est positionné si le canal n'existe pas ou n'a pas été correctement initialisé.

Voir aussi SCON, SASI, Instructions de Communication.

SCONI OPEN COMMUNICATION CHANNEL INDIRECT (PROFIBUS)

Ouverture indirecte d'un canal de communication (PROFIBUS)

Pour plus d'informations, consultez le manuel "SAIA® PROFIBUS" (réf. 26/742).

Description Etablit ou ferme de manière indirecte un canal virtuel de communication PROFIBUS.
Le 1er opérande est le numéro du canal ou un registre contenant le numéro du canal.
Le 2ème opérande est un registre qui doit contenir 0.
Le 3éme opérande est un registre contenant l'état de la connexion (0 = Fermeture, 1 = Ouverture).

SCONI ne peut pas être indexée ou paramétrisée.

Usage

SCONI	**canal**	**; N° de canal PROFIBUS nb 10-99 ou R 0-4095**
	station	**; R 0-4095 (toujours 0)**
	statut	**; Etat de la connexion R 0-4095**

Exemple

```
SCONI   R 10  ;
        R 11  ;
        R 12  ;
```

Flags Le flag **Erreur** (E) est positionné si le canal n'existe pas.

Voir aussi SCON.

9. Instructions pour le LAN 2

Le SAIA LAN 2 est un réseau local (Local Area Network) fonctionnant sur le principe du passage de jeton (token passing) et peut interconnecter jusqu'à 255 stations.

Un module LAN 2 (PCD6.T1.. ou PCD4.M340) est nécessaire dans chaque station.

Les états de n'importe quelles entrées (Inputs), sorties (Outputs) ou Flags, ainsi que les valeurs de n'importe quels Registres, Temporisateurs ou Compteurs, ou le statut du CPU peuvent être lus ou écrits via le LAN 2.

Les commandes du LAN 2 sont définies dans les textes de commandes LAN 2.

Il est possible de transférer n'importe quel élément binaire vers un autre élément binaire (ex. Input vers Flag), ou de Temporisateur/Compteur vers Temporisateur/Compteur, ou de Registre vers Registre.

IMPORTANT :

Les instructions telles que décrites ici nécessitent une version firmware V005 (ou supérieure) du LAN 2.

Instructions LAN 2 :

LRXD	Receive data via LAN 2	Réception de données via le LAN 2
LTXD	Transmit data via LAN 2	Transmission de données via le LAN 2
LRXS	Receive status via LAN 2	Réception du statut via le LAN 2
LTXS	Transmit status via LAN 2	Transmission d'un statut via le LAN 2

Notes personnelles :

LRXD RECEIVE DATA VIA LAN 2

Réception de données via le LAN 2

Description

Lecture de données dans une station éloignée via le LAN 2.
Le 1er opérande est la priorité (0 = haute, 1 = basse). Pour bénéficier d'une haute priorité, la longueur des données transférées ne doit pas être supérieure à 32 bytes.
Le 2ème opérande est le numéro du texte contenant l'adresse du partenaire, les éléments du partenaire devant être lus et où ces éléments doivent être copiés (station locale).
Le 3ème opérande est l'adresse de base de 10 Flags (ou sorties) de diagnostique. Le premier élément de diagnostique est le flag EXEC.
Initialement, il est mis Low par LRXD, et reste Low lors des exécutions successives de la même instruction LRXD jusqu'au moment ou le transfert est complet. Lorsque le transfert est complet, le flag EXEC est mis High.
Si l'instruction LRXD est exécutée à nouveau, le transfert de données est répété.
Le statut du flag EXEC est seulement modifié lorsque l'instruction LRXD est effectuée.

Usage

```
LRXD[X]   priorité      ; 0 = haute, 1 = basse
          texte   (i)   ; Numéro du texte contenant la commande 0-3999
          diag          ; Adresse de base flags de diagnostique F|O 0-8182
```

Exemple

```
LRXD        1     ; Transfert avec priorité basse
            900   ; Le Texte 900 contient la commande
          F 200   ; Flags de diagnostique F 200-209
```

Flags

Inchangés.

Voir aussi

LTXD, LRXS, LTXS, Flags de diagnostique, Textes de commande.

Pratique

Lorsque l'entrée I 1 du PCD local est enclenchée, les entrées I 0-7 du PCD éloigné numéro 3 sont lues et transférées sur les sorties O 32-39 du PCD local.

```
            COB         5
                        0
            STH       I 1     ; Si entrée I 1 devient High
            DYN       F 1
            ORL       F 100   ;    EXEC Flag
            CPB       H 50    ;    Alors appel du PB 50
            ECOB

            PB          50
            LRXD        1     ; Priorité
                        15    ; Numéro du texte
                      F 100   ; EXEC Flag
$LAN
TEXT 15     "3:I0-7:O32-39"
$ENDLAN
            EPB
```


LTXD TRANSMIT DATA VIA LAN 2

Transmission de données via le LAN 2

Description

Ecriture de données dans une station éloignée via le LAN 2.
Le 1er opérande est la priorité (0 = haute, 1 = basse). Pour bénéficier d'une haute priorité, la longueur des données transférées ne doit pas être supérieure à 32 bytes.
Le 2ème opérande est le numéro du texte contenant l'adresse du partenaire, les éléments de la station locale devant être transférés et où ces éléments doivent être copiés (partenaire).
Le 3ème opérande est l'adresse de base de 10 Flags (ou sorties) de diagnostique. Le premier élément de diagnostique est le flag EXEC.
Initialement, il est mis Low par LTXD, et reste Low lors des exécutions successives de la même instruction LTXD jusqu'au moment ou le transfert est complet. Lorsque le transfert est complet, le flag EXEC est mis High.
Si l'instruction LTXD est exécutée à nouveau, le transfert de données est répété.
Le statut du flag EXEC est seulement modifié lorsque l'instruction LTXD est effectuée.

Usage

```
LTXD[X]   priorité      ; 0 = haute,   1 = basse
          texte   (i)   ; Numéro du texte contenant la commande 0-3999
          diag          ; Adresse de base flags de diagnostique F|O 0-8182
```

Exemple

```
LTXD        1     ; Transfert avec priorité basse
            900   ; Le Texte 900 contient la commande
          F 200   ; Flags de diagnostique F 200-209
```

Flags

Inchangés.

Voir aussi

LRXD, LRXS, LTXS, Flags de Diagnostique, Textes de Commande.

Pratique

Lorsque l'entrée I 8 du PCD local est enclenchée, les entrées I 0-7 sont lues et transférées sur les sorties O 40-47 du PCD éloigné numéro 3.

```
            COB       5
                      0
            STH     I 8     ; Si l'entrée I 8 devient High
            DYN     F 8
            ORL     F 100   ;    Flag EXEC
            CPB     H 51    ;    Alors appel du PB 51
            ECOB

            PB        51
            LTXD      1     ; Priorité
                      16    ; Numéro du Texte
                    F 100   ; Flag EXEC
$LAN
TEXT 16     "3:I0-7:O40-47"
$ENDLAN
            EPB
```


LRXS RECEIVE STATUS VIA LAN 2
Réception du statut via le LAN 2

Description

Lecture du statut d'une station éloignée via le LAN 2 et mémorisation de celui-ci dans les Flags de diagnostique.
Cette instruction permet également de lire les statistiques (contrôle du trafic) de sa propre station.
Le statut peut être :

HALT	CPU en Halt
RUN	CPU en Run
DIS	Station déconnectée du LAN 2
CON	Station connectée au LAN 2

Le 1er opérande est le numéro du texte contenant l'adresse de la station dont le statut doit être lu.
Le 2ème opérande est l'adresse de base de 10 Flags (ou sorties) de diagnostique. Le premier élément de diagnostique est le flag EXEC.
Initialement, il est mis Low par LRXS, et reste Low lors des exécutions successives de la même instruction LRXS jusqu'au moment ou la lecture est complète. Lorsque la lecture est complète, le flag EXEC est mis High.
Le statut du flag EXEC est seulement modifié lorsque l'instruction LRXS est effectuée.
Si le statut est "disconnected" (déconnecté), seul le flag "Disconnected" est positionné à 1 (il ne s'agit pas ici d'une erreur).

L'instruction LRXS a toujours une haute priorité.

Usage

LRXS[X]	**texte**	**(i)**	**; Numéro du Texte contenant la commande 0-3999**
	diag		**; Adresse de base flags de diagnostique F|O 0-8182**

Exemple

```
LRXS        100   ; Le Texte 100 contient la commande
          O 32    ; Sorties utilisée pour le diagnostique O 32-41
TEXT 100 "020"    ; Lecture du statut de la station 20
```

Flags Inchangés.

Voir aussi LTXD, LRXD, LTXS, Flags de Diagnostique, Textes de Commande.

LTXS TRANSMIT STATUS VIA LAN 2

Transmission d'un statut via le LAN 2

Description

Modifie le statut d'un PCD éloigné.
Le statut peut être changé en :

HALT	Halt du CPU
RUN	Run du CPU
DIS	Déconnecte une station du LAN 2
CON	Connecte une station au LAN 2
TOUT:nnn	Valeur de Time Out où nnn = Valeur en 1/10 sec (10..250).

Le 1er opérande est le numéro du texte contenant l'adresse de la station dont le statut doit être changé ainsi que le nouveau statut.
Le 2ème opérande est l'adresse de base de 10 Flags (ou sorties) de diagnostique. Le premier élément de diagnostique est le flag EXEC.
Initialement, il est mis Low par LTXS, et reste Low lors des exécutions successives de la même instruction LTXS jusqu'au moment ou le transfert est complet. Lorsque le transfert est complet, le flag EXEC est mis High.
Le statut du flag EXEC est seulement modifié lorsque l'instruction LTXS est effectuée.

L'instruction LTXS a toujours une haute priorité.

Usage

```
LTXS[X]   texte   (i)   ; Numéro du texte contenant la commande 0-3999
          diag          ; Adresse de base flags de Diagnostique F|O 0-8182
```

Exemple

```
LTXS        1000        ; Le Texte 1000 contient la commande
          F 100         ; Flags de Diagnostique F 100-109
TEXT 1000 "035:DIS"     ; Déconnecte la station 35
```

Flags

Inchangés.

Voir aussi

LRXS, LRXD, LTXD, Flags de Diagnostique, Textes de Commande.

LAN 2: Flags de diagnostique

Pour chaque instruction SAIA LAN 2, une des opérandes donne l'adresse de base de 10 éléments de diagnostique (Sorties "Outputs" ou Flags).

Flag		**Statut High**	**Statut Low**
0	EXECuted	Commande exécutée	Commande en cours
+1	FAILure	Erreur	Pas d'erreur
+2	Statut du PCD local	Texte de commande non valide	Texte de commande valide
+3		Déconnecté	Connecté
+4		Time Out (Erreur de transmission)	Pas de Timeout
+5	Statut du PCD remote	Déconnecté	Connecté
+6		Non utilisé	Non utilisé
+7		Protégé contre l'écriture	Ecriture Ok
+8		HALT (CPU 0)	Run (CPU 0)
+9	Watch dog (1)	LAN reconfiguré	

(1) à partir de la version V003 (PCD4) et V007 (PCD6)

EXEC Flag : Le premier élément de diagnostique est le flag EXEC. Quand EXEC est Low, cela indique que l'exécution de l'instruction LAN 2 est toujours en cours d'exécution (réception ou émission de données). Le flag EXEC est mis initialement Low lorsque une instruction LAN 2 est exécutée pour la première fois; ce flag reste Low lors des exécutions successives de la même instruction jusqu'au moment ou le transfert est complet. Lorsque le transfert est complet, l'EXEC flag est mis High.

Le programme doit être structuré de telle manière que les instructions LAN 2 soient continuellement exécutées lorsque le flag EXEC est Low.

Si une instruction LAN 2 est exécutée à nouveau, le transfert de données est répétée. Le statut du flag EXEC est modifié seulement lorsqu'une instruction LAN 2 est exécutée.

PRIORITE : Le transfert d'informations par le LAN 2 est constitué de trames ("Frame"). Chaque trame est longue de 32 bytes (256 bits) et peut contenir 8 R|C|T ou 256 I|O|F.

Lorsque la station reçoit le jeton (" Token "), une seule trame est émise et le jeton est passé à la station suivante. Un transfert qui est plus long qu'une trame sera divisé en différentes trames successives et aura toujours une priorité basse (1); si une haute priorité est demandée, le flag +2 (Texte de commande non valide) est positionné et le télégramme n'est pas émis.

Un transfert qui est plus court qu'une trame pourra être émis avec une priorité haute (0) ou basse (1). Si la priorité est haute, la trame pourra être insérée entre les trames successives d'un transfert important (basse priorité).

Les télégrammes de statut ont toujours une haute priorité (0).

LAN 2: Textes de commande

Transfert de Données (LRXD / LTXD)

Les instructions LTXD et LRXD doivent avoir un texte de commande qui définit les données à envoyer ou à recevoir via le LAN 2.
Ces textes doivent toujours être en MAJUSCULES.

Format :

Chaque texte de commande contient le numéro de la station remote, le type de média et la liste d'adresses de la source, ainsi que le type de média et la liste d'adresses de la destination. Le numéro de station, ainsi que la liste d'adresses de source et de destination doivent être séparés par ':'.

La liste d'adresses peut être :

- une plage: deux adresses séparées par un '-'
  ex: "I100-200" (Adresses des entrées 100 à 200)
- une énumération de maximum 8 éléments simples séparés par des virgules ex : "R100,110,120".

Les listes de source et de destination doivent correspondre.

Note :
Toutes les combinaisons de source et de destination ne sont pas autorisées

Source	Destination					Adresses
	O	F	T	C	R	
I	●	●				0..8191
O	●	●				0..8191
F	●	●				0..8191
T			●	●		0..450
C			●	●		0..1599
R					●	0..4095

Exemples de lecture de données (RECEIVE DATA) (LRXD) :

TEXT 100 "015:O64-127:F1000-1063"
Transfère les valeurs des sorties O 64 à 127 de la station PCD (remote) numéro 15 sur les Flags F 1000 à 1063 du PCD local.

TEXT 101 "008:T11,13,25:C55,125,1231"
Transfère le contenu des Temporisateurs 11, 13 et 25 de la station PCD (remote) numéro 8 dans les Compteurs 55, 125 et 1231 du PCD local.

Exemple de transmission de données (TRANSMIT DATA) (LTXD) :

TEXT 75 "000:R11-22:R33-44"
Transfère le contenu des Registres 11 à 22 du PCD local dans les Registres 33 à 44 du PCD (remote) numéro 0.

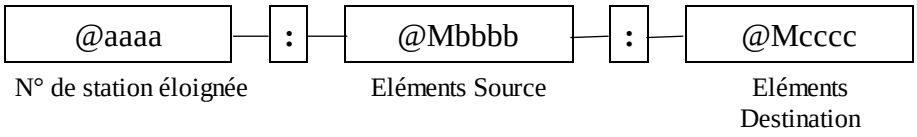
Adressage Indirect

Le nombre de textes à utiliser pour définir les transferts via le LAN 2 peut être réduit en utilisant l'adressage indirect : l'adresse effective est fournie par le contenu d'un registre.

Chaque constituant d'un texte de commande pour le LAN 2 (numéro de station, source et destination) peut être adressé indirectement en utilisant le caractère '@'.

Adressage indirect d'un élément

Format :



où : @ indique l'adressage indirect.
M est le medium code (I|O|F|T|C|R)
aaaa: Registre contenant l'adresse du partenaire.
bbbb: Registre contenant l'adresse de l'élément source
cccc: Registre contenant l'adresse de l'élément destination

Exemple :

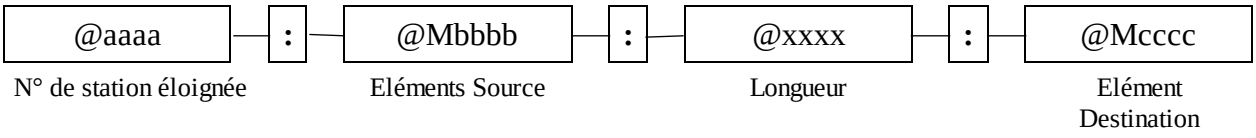
"@100:@I400:@F600"

Le numéro de station (remote) est dans le registre 100
L'entrée "Input" dont l'adresse est donnée par le registre 400 sera transférée dans le Flag dont l'adresse est donnée par le registre 600.

Adressage indirect d'éléments multiples

Le nombre d'éléments à transférer peut être donnés par le contenu d'un registre.

Format :



où : @ indique l'adressage indirect
M est le médium code (I|O|F|T|C|R)
aaaa: Registre contenant l'adresse du partenaire.
bbbb: Registre contenant l'adresse du 1er élément source
xxxx: Registre contenant le nombre d'éléments à transférer
cccc: Registre contenant l'adresse du 1er élément destination

Exemple :

"@200:@C100-@101:@C500"

Le numéro de station (remote) est contenu dans le registre 200.
L'adresse du premier compteur à transférer est donnée par le contenu du registre 100, le nombre de compteurs à transférer est dans le registre 101.
Ces compteurs seront copiés sur les compteurs dont la première adresse est donnée par le registre 500.

Adressage mixte

L'adressage direct et indirect peuvent être mélangés dans le même texte.

Dans l'adressage mixte, c'est la syntaxe de l'adressage indirect qui est d'application.

Exemples :

"@5:R100-50:@99"

Le registre 5 contient l'adresse de la station partenaire.
50 Registres à partir de R100 sont transférés sur les registres à partir de l'adresse contenue dans le Registre 99.

"25:I0-@55:F1000"

La station partenaire est la numéro 25. Les entrées à partir de I 0 sont transférées sur les Flags à partir de F 1000; le nombre d'entrées à transférer est contenu dans le Registre 55.

Remarque concernant l'adressage indirect et mixte

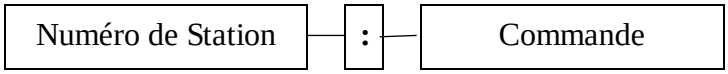
Le contenu des registres ne pouvant être testé par 'assembleur, l'utilisateur doit faire particulièrement attention à ne pas dépasser la plage des éléments adressés.

Un dépassement de la plage autorisée peut avoir des conséquences indéfinissables.

Transmit Status (LTXS)

Le statut d'une station peut être transféré via le LAN2.

Format :



Numéro de station	
0-254	La commande est émise vers la station spécifiée.
255	La commande est émise vers toutes les stations connectées exceptée la station propre.
Commande	
HALT	Met tous les processeurs de la station spécifiée en HALT. Le programme utilisateur est arrêté. Le Flag 8 des Flags de diagnostique est mis High.
RUN	Met tous les processeurs de la station spécifiée en RUN.
CON	Met la station LAN 2 en état "connecté". Les Flags de diagnostique 3 ou 5 sont mis Low.
DIS	Met la station LAN 2 en état "déconnecté". Dans ce cas aucune instruction n'est exécutée et aucune donnée n'est transférée. Les Flags de diagnostique 3 ou 5 sont mis High.
TOUT:NNN	Positionne le temps de "timeout" pour la station spécifiée. NNN est le nombre de stations sur le réseau (2-255).

Exemple de texte pour TRANSMIT STATUS (LTXS) :

```
TEXT  213     "018:HALT"
```

La station (remote) 18 est mise en HALT.

Read Status (LRXS)

Le statut d'une station peut être lu :

Format :

Le statut est donné par les Flags de diagnostique (Flag 3, 5 ou 8)

Exemple de texte pour RECEIVE STATUS (LRXS) :

```
TEXT 137      "015"
```

Le statut de la station 15 est lu et mémorisé dans les flags de diagnostique :

"Déconnecté" EXEC Flag + 5
"Halt" EXEC Flag + 8

Le flag "Failure" (EXEC Flag + 1) n'est pas modifié.

Si le statut est "déconnecté", alors seul le flag "déconnecté" est positionné car il n'y a pas d'erreur.

Note : Une communication LAN 2 est terminée seulement lorsque le flag EXEC = H après exécution de l'instruction LRXS

Contrôle du trafic réseau (Line traffic control) (LRXS)

Il est possible de vérifier ce qu'il se passe sur le réseau SAIA LAN 2 avec une instruction LRXS et un texte de commande spécial:

Format :

où xxxx est un registre (0-4095)

Ce texte de commande permet de lire les statistiques d'émission/réception du LAN 2; celles-ci sont mémorisées dans 4 registres consécutifs à partir du registre xxxx. Cette commande est particulièrement utile pour détecter les erreurs hardware sur le réseau.

Les valeurs retournées sont :

Registre	**Description**
xxxx	Nombre de trames reçues
xxxx + 1	Nombre de trames transmises
xxxx + 2	Nombre de ré-essais (= nombre de timeouts * 3)
xxxx + 3	Nombre de messages rejetés (= messages corrompus)

Note :

Pendant la première communication avec une station, le nombre de ré-essai est incrémenté : ceci implique que chaque station a au moins 1 ré-essai.
Les valeurs dans les registres statistiques sont mémorisées en format 16 bits (valeur maximum = 65535)

Exemple :

```
TEXT  123     "100:STAT:R20"
```

Lit les statistiques de sa propre station numéro 100 et mémorise la valeur dans les Registres 20 à 23.

Syntaxe des textes LAN 2 dans le fichier source

La syntaxe de textes LAN 2 écrites entres les directives $LAN ($LLAN) et $ENDLAN ($NOLAN, $NOLLAN) est vérifiée pendant l'assemblage.
Les caractères minuscules sont également convertis en majuscules.

De plus, l'Assembleur convertit les textes LAN 2 en format binaire, qui permet de rendre les communications LAN 2 plus rapides. Le gain de temps est de plus ou moins 15 msec. par commande; ce qui se remarque particulièrement pour des télégrammes courts.

Exemples :

```
$LAN
TEXT 0        "2:I0-255:F1000-1255"
TEXT 1        "5:r501-510:r101-110"
TEXT 2        "7:f0-31:o96-127"
TEXT 15       "2:con"
TEXT 37       "15"
TEXT 101      "8:t11,13,25:c55,125,1231"
TEXT 75       "0:R11-22:R33-44"
TEXT 200      "@100:@i400:@f600"
TEXT 1111     "@200:@C100-@101:@C500"
TEXT 99       "@5:R100-50:@R99"
TEXT 3999     "25:I0-@55:F1000"
$ENDLAN

TEXT 224      "Ceci est un texte ASCII normal <10><13>"
```

Note :

Les textes LAN 2 en format binaire ne peuvent pas être affichés ni édités à l'aide du debugger.

Utilisation de symboles dans les textes LAN 2

Des symboles peuvent également être utilisés dans les textes LAN 2.

La valeur et optionnellement le type du symbole est inséré dans le texte. Le symbole est écrit à l'extérieur du texte ASCII qui est entouré de double apostrophes, et doit être séparé de celui-ci ou des autres symboles par une virgule. Après le symbole, une largeur de champ et un type peuvent éventuellement être donnés.

Format :

symbole [. [[-] [0] largeur] [t | T]]	
symbole	Le nom symbolique. Ceci peut être n'importe quelle expression qui inclut un symbole, par exemple: MotorOn + 100. Des symboles avec des valeurs en virgule flottante ne sont pas autorisés.
.	Le point qui suit immédiatement le symbole indique qu'une largeur de champ et/ou un préfixe est présent.
Largeur	La largeur du champ : le nombre de chiffres ou espaces requis pour le nombre. Si la largeur commence par un 0, des zéros sont insérés à gauche.
t | T	Préfixe optionnel de type 't' ou 'T'. Si 't', la valeur est préfixée par le type du symbole en minuscule (o, f, r, ...); si 'T', le type du symbole est en majuscule (O, F, R,...)

Exemple :

```
SOURCE  EQU     R 55
DEST    EQU     R 66

$LAN
TEXT  25        "8:" , SOURCE.T , ":" , DEST.T
$ENDLAN
```

Texte résultant : `"008:R55:R66"`

10. Instructions de CONTROLE

Instructions de commande du programme.

Ces instructions commandent l'exécution du programme.

Les instructions de sauts sont de fréquentes sources d'erreurs (boucle sans fin, etc...); elles doivent dès lors être utilisées avec de grandes précautions. On utilisera de préférence les sauts dans les blocs de programme ou blocs de fonction plutôt que dans le programme principal.

Le GRAFTEC permet de programmer sans utiliser de sauts.

L'opérande de ces instructions ne peut être passé comme paramètre d'un Function Block.

JR	Jump Relative	Saut relatif
JPD	JumP Direct	Saut direct
JPI	JumP Indirect	Saut indirect
HALT	HALTs the cpu	Arrêt du CPU
LOCK	LOCK semaphore	Verrouillage d'un sémaphore
UNLOCK	UNLOCK semaphore	Déverrouillage d'un sémaphore

Notes personnelles :

JR JUMP RELATIVE
Saut relatif

Description Saute, conditionnellement ou non, vers l'avant ou l'arrière du programme du nombre de lignes spécifié dans l'opérande.

Le nombre de lignes pouvant être sautées doit être compris entre - 4096 (arrière) et + 4096 (avant); l'adresse vers laquelle le saut est effectué est obtenu en additionnant l'opérande au numéro de la ligne de programme contenant l'instruction JR.

Il est interdit de sauter en dehors du bloc courant (COB, PB, FB, ST, TR ou SB) : la destination DOIT être dans le bloc courant.

Les **codes conditionnels** suivants sont autorisés :

-	Saut inconditionnel (code conditionnel blanc)
H	Saute si ACCU = H (1)
L	Saute si ACCU = L (0)
P	Saute si Flag Positif = H (Flag Négatif = L)
N	Saute si Flag Négatif = H
Z	Saute si Flag Zéro = H
E	Saute si Flag Erreur = H

Si la condition n'est pas remplie, le saut n'est pas effectué; l'exécution continue avec l'instruction suivant "JR".

En programmant avec l'Assembleur, il est courant d'utiliser des "labels" ou étiquettes (noms symboliques) pour marquer la destination des sauts.
Les labels peuvent être de n'importe quelle longueur, mais seuls les 8 premiers caractères sont significatifs; les labels doivent toujours commencer par une lettre (A..Z).

Si l'on utilise un autre éditeur que SEDIT, il est nécessaire de faire suivre chaque label d'un ':'.

Usage

```
JR      [cc] offset     ; cc = code conditionnel (H|L|P|N|Z|E)
                        ; offset est le nombre relatif de lignes
                        ;      devant être sautées ( - 4096.. + 4096)
```

Exemple

```
JR      H -2            ; Saute 2 lignes en arrière
JR      H REPEAT        ; Saute au label "REPEAT"
```

Flags Inchangés.

Voir aussi JPD, JPI, LD.

Pratique Voir exemple à la page suivante.

Note Un programme bien structuré n'utilise pas de sauts.

JPD JUMP DIRECT
Saut direct

Description Saute, conditionnellement ou non, à une ligne de programme dont l'adresse est calculée à partir du début du bloc courant (COB, XOB, PB, FB, ST ou TR).

La ligne de destination est toujours un nombre positif compris entre 0 et le nombre total de lignes du bloc courant (max 8191 lignes).

Des "labels" peuvent être utilisés.

Les **codes conditionnels** suivants sont autorisés :

-	Saut inconditionnel (code conditionnel blanc)
H	Saute si ACCU = H (1)
L	Saute si ACCU = L (0)
P	Saute si Flag Positif = H (Flag Négatif = L)
N	Saute si Flag Négatif = H
Z	Saute si Flag Zéro = H
E	Saute si Flag Erreur = H

Si la condition n'est pas remplie, le saut n'est pas effectué; l'exécution continue avec l'instruction suivant "JPD".

Usage

```
JPD      [cc] offset      ; cc = code conditionnel (H|L|P|N|Z|E)
                          ; offset est l'offset à partir du début du
                          ;      block  (0..8191)
```

Exemple

```
JPD      L 10    ; Si l'ACCU est Low, le saut est effectué
                 ;      à la 10ème ligne du bloc courant
```

Flags Inchangés.

Voir aussi JR, JPI.

Pratique Concerne l'instruction JR décrite à la page précédente.

```
        STH  I 15
        ANL    XBSY
        DYN  F 15
        JR   H Next
        STXT   1
               57
        ...
Next:   ...
```

→

```
        STH  I 15
        ANL    XBSY
        CPB  H 25
        ...
        PB     25
        STXT   1
               57
        EPB
```

JPI JUMP INDIRECT
Saut indirect

Description

Similaire à JPD : saute, conditionnellement ou non, à une ligne de programme dont l'adresse est calculée à partir du début du bloc courant (COB, XOB, PB, FB, ST ou TR).

L'adresse de la ligne de programme est contenue dans le Registre donné dans l'opérande (seul les 8 bits de poids le plus faibles sont significatifs). Etant donné que l'instruction peut utiliser un code conditionnel, le médium code 'R' du registre n'est pas donné.

Les **codes conditionnels** suivants sont autorisés :

-	Saut inconditionnel (code conditionnel blanc)
H	Saute si ACCU = H (1)
L	Saute si ACCU = L (0)
P	Saute si Flag Positif = H (Flag Négatif = L)
N	Saute si Flag Négatif = H
Z	Saute si Flag Zéro = H
E	Saute si Flag Erreur = H

Si la condition n'est pas remplie, le saut n'est pas effectué; l'exécution continue avec l'instruction suivant "JPI".

La valeur d'un label peut être chargée dans un registre avec l'instruction LD.

Usage

```
JPI     [cc] reg     ; cc = code conditionnel (H|L|P|N|Z|E)
                     ; reg est le numéro du Registre contenant
                     ; l'offset à partir du début du block (0..8191)
```

Exemple

```
JPI     H 300  ; Si l'ACCU est High, le saut est effectué
               ;     à la ligne contenue dans le Registre 300
```

Flags

Inchangés.

Voir aussi

JR, JPI, LD.

Note

Il est de la responsabilité du programmeur de prendre des précautions afin que l'adresse de destination ne se situe pas en dehors du bloc courant.
En effet, comme l'adresse du saut est contenue dans un registre, l'Assembleur ne peut tester si celle-ci est correcte.

HALT HALTS THE CPU
Arrêt du CPU

Description Arrête le CPU, conditionnellement ou non.

L'état "Halt" n'est pas identique au "Stop". Après un HALT, le CPU ne peut être remis en Run que par un Restart ou par une remise sous tension.

Les **codes conditionnels** suivants sont autorisés :

-	Saut inconditionnel (code conditionnel blanc)
H	Saute si ACCU = H (1)
L	Saute si ACCU = L (0)
P	Saute si Flag Positif = H (Flag Négatif = L)
N	Saute si Flag Négatif = H
Z	Saute si Flag Zéro = H
E	Saute si Flag Erreur = H

Si la condition n'est pas remplie, le HALT n'est pas effectué; l'exécution continue avec l'instruction suivant HALT.

Note :

Après un HALT du CPU 0, un Restart Cold ne peut être effectué que sur tous les CPUs. Le status des sorties après un HALT est défini par la configuration hardware (jumpers).

Usage

HALT	**[cc]**	**; cc = code conditionnel: H|L|P|N|Z|E**

Exemple

```
HALT    E       ; HALT si le flag Erreur (E) est mis
```

Flags Inchangés.

Voir aussi Guide Utilisateur.

Pratique En cas d'erreur: arrêt du PCD et mémorisation dans un registre de certaines informations de diagnostique.

```
XOB       13
DIAG    R 1000
HALT
EXOB
```

Attention :

L'instruction HALT sera principalement utilisée pendant la phase de test. Cette instruction doit être utilisée avec la plus grande prudence dans un programme d'application !

LOCK LOCK SEMAPHORE
Verrouillage d'un sémaphore

Description

LOCK en conjonction avec UNLOCK, est utilisé pour éviter les conflits lorsque plusieurs CPUs lisent et écrivent les mêmes éléments.

100 Sémaphores (flags spéciaux) sont disponibles (0-99).

L'instruction LOCK teste le Sémaphore; si celui-ci est High (un autre CPU a exécuté un LOCK), l'ACCU est mis Low.
Si le Sémaphore est Low, l'ACCU et le Sémaphore sont mis High.

Il est de la responsabilité du programmeur de s'assurer qu'un CPU référence un élément si le Sémaphore associé est High (ACCU = L après le LOCK).

L'instruction UNLOCK efface le Sémaphore.

Une instruction UNLOCK doit suivre le plus rapidement possible une instruction LOCK de façon à ce qu'un CPU ne se voit pas refuser pendant trop longtemps l'accès d'un élément.

Usage

LOCK	**sémaphore**	**; Sémaphore 0-99**

Exemple

```
LOCK      1      ; Si le Sémaphore 1 est Low
                 ;      (les données ne sont pas accédées par un autre CPU)
CFB       H 100  ; Alors appel du FB 100.
                 ; Le Sémaphore 1 est utilisé pour protéger
                 ;      l'accès aux données
```

Flags

ACCU est mis High/Low.

Voir aussi

UNLOCK.

Pratique

voir UNLOCK.

UNLOCK UNLOCK SEMAPHORE

Déverrouillage d'un sémaphore

Description

UNLOCK en conjonction avec LOCK, est utilisé pour éviter les conflits lorsque plusieurs CPUs lisent et écrivent les mêmes éléments.

100 Sémaphores (flags spéciaux) sont disponibles (0-99).

L'instruction UNLOCK efface le Sémaphore.

Usage

UNLOCK	**sémaphore**	**; Sémaphore 0-99**

Exemple

```
UNLOCK    1      ; Le Sémaphore 1 est mis Low
```

Flags

Inchangés.

Voir aussi

LOCK.

Pratique

Un PCD est équipé de deux CPUs.
Le CPU 0 compare le contenu de 2 registres pendant que le CPU 1 transfère des informations BCD dans un de ces deux registres.

CPU 0

```
…
LOCK       1
CFB      H 10
….
FB         10
CMP      R 88
         R 89
UNLOCK     1
EFB
```

CPU 1

```
…
LOCK       1
CFB      H 100
….
FB       100
DIGI       2
         I 16
         R 88
DIGI       2
         I 24
         R 89
UNLOCK     1
EFB
```

L'utilisation du sémaphore 1 empêche le CPU 0 de comparer les 2 registres lorsque le CPU 1 exécute l'instruction DIGI et altère le contenu de ceux-ci. Si le CPU 0 comparait les registres a l'instant où le CPU 1 les modifie; il se pourrait qu'il compare une nouvelle valeur avec une ancienne.
Le Sémaphore 1 empêche également l'exécution par le CPU 1 de l'instruction DIGI avant que le CPU 0 n'aie terminé l'instruction CMP.

11. Instructions de DEFINITION

Ces instructions ne doivent être exécutées qu'une seule fois: lors de la mise sous tension. Si une de ces instructions est exécutée à nouveau, elle sera ignorée.

Normalement, ces instructions sont placées dans le bloc de démarrage à froid (XOB 16).

L'opérande de ces instructions ne peut pas être passé comme paramètre d'un Function Block parameter.

DEFVM	DEFine Volatile Memory (flags)	Définition de la mémoire non volatile (flags)
DEFTC	DEFine Timers/Counters	Définition des Temporisateurs/Compteurs
DEFTB	DEFine TimeBase	Définition de la base de temps
DEFTR	DEF Timer Resolution	Définition de la résolution des Temporisateurs
DEFWPR	DEFine Write Protected area in Run	Définition de la zone protégée en Run
DEFWPH	DEFine Write Protected area in Halt	Définition de la zone protégée à l'arrêt

Notes personnelles :

DEFVM DEFINE VOLATILE MEMORY (Flags)

Définition de la mémoire non volatile (flags)

Description Définit la zone des flags qui doivent être rendue non volatiles.

Les flags non volatiles retiennent leur valeur en cas de coupure de tension du PCD. Les flags volatiles sont mis à 0 par le PCD lors de la mise sous tension.

Tous les flags **AU DESSUS** du flag indiqué dans l'opérande sont définis comme étant non volatiles.

Si l'instruction n'est pas utilisée, TOUS les flags sont non volatiles par défaut.

Cette instruction est seulement exécutée par le CPU 0.

Usage

DEFVM	**flag**	**; 0-8190, répartition volatile/non volatile des flags**

Exemple

```
DEFVM   200     ; Les flags 0 - 199 sont volatiles,
                ;       200 - 8191 sont non volatiles
```

Flags Inchangés.

Voir aussi DEFTC, DEFTB, DEFWPR, DEFWPH.

Pratique Les flags 0..199 doivent être volatiles.

```
XOB     16      ; Cold start XOB

DEFVM   200     ; Flags 200..8191 sont non volatiles
....
EXOB
```


DEFTC DEFINE TIMERS/COUNTERS
Définition des Temporisateurs/Compteurs

Description Définit le nombre de Temporisateurs.

Les Temporisateurs et les Compteurs occupent la même plage d'adresses.

Tous les éléments **EN DESSOUS** de la valeur de l'opérande sont des Temporisateurs, tous les autres sont des Compteurs.

Si l'instruction n'est pas utilisée, les valeurs par défaut sont :

Temporisateurs : 0 - 31
Compteurs : 32 - 1599.

Note : Il est conseillé de ne pas définir plus de Temporisateurs que nécessaire. Le traitement de chaque Temporisateur affecte la vitesse d'exécution du programme.

Le nombre maximum de Temporisateurs est de 450.

Cette instruction est seulement exécutée par le CPU 0.

Usage

DEFTC	**cmpt**	**; Limite basse des Compteurs (0-450)**

Exemple

```
DEFTC   64      ; Temporisateurs 0-63, Compteurs 64-1599
```

Flags Inchangés.

Voir aussi DEFTB, DEFVM, DEFWPR, DEFWPH.

Pratique Une application nécessite 100 Temporisateurs.

```
XOB     16      ; Cold start XOB

DEFTC   100     ; 0..99 sont des Temporisateurs
                ; 100..1599 sont des Compteurs
....
EXOB
```


DEFTB DEFINE TIMEBASE
Définition de la base de temps

Description Définit la base de temps pour la décrémentation des Temporisateurs.

L'opérande donne la base de temps en dizaine de millisecondes. Des valeurs entre 1 et 1000 sont autorisées (10 msec à 10 sec).

Si la base de temps n'est pas explicitement définie, la valeur par défaut est de 100 msec (1/10 sec).

DEFTB est exécuté seulement par le CPU 0.

Pour les autres CPUs, l'instruction DEFTB défini la base de temps des Temporisateurs internes utilisés par les instructions SETD et RESD. Pour cette raison, les Temporisateurs internes du CPU 0 ont toujours la même base de temps que celle des Temporisateurs; la base de temps des Temporisateurs internes des autres CPUs peuvent différer de celles des Temporisateurs.

Note : Il faut tenir compte que lorsque l'on définit une base de temps faible (par ex.: 10 msec) et un nombre important de Temporisateurs, le programme utilisateur peut être ralenti.

Usage

```
DEFTB     base de temps     ; base de temps en dizaine de millisecondes
                            ;     (1-1000)
```

Exemple

```
DEFTB    100     ; Base de temps = 1 sec (100 * 10msec)
```

Flags Inchangés.

Voir aussi DEFVM, DEFTC, DEFWPR, DEFWPH, SETD, RESD.

Pratique Pour une application relativement lente, la base de temps peut être de 1 sec.

```
        XOB     16      ; Cold start XOB

        DEFTB   100     ; Base de temps: 100 * 10 ms = 1000 msec.
        ....
        EXOB
```


DEFTR DEFINE TIMER RESOLUTION

Définition de la résolution des Temporisateurs

Description Définit la résolution de décrémentation des Temporisateurs en **millisecondes**.

Par exemple, si "DEFTR 100" est spécifié, tous les Temporisateurs non-nuls sont décrémentés par 100 toutes les 100 msec. "DEFTR 1000" décrémente tous les Temporisateurs par 1000 toutes les 1000 msec. Si DEFTR et DEFTB sont utilisés dans le même programme, le message "**DOUBLE TIME BASE**" sera inscrit dans la liste historique et le CPU ira en "HALT".

L'avantage de l'instruction DEFTR (sur DEFTB) est que les valeurs spécifiées lors de l'utilisation des Temporisateurs sont indépendantes de la base de temps ou résolution et toujours introduites en multiple de 10 msec. Pour que l'instruction DEFTR ait une influence sur les Temporisateurs, l'instruction doit être programmée dans le CPU 0. L'instruction DEFTR autorise une résolution maximum de 10 msec ce qui signifie que la valeur spécifiée dans l'instruction est arrondie si nécessaire.

Note : DEFTR 25 : une base de temps de 20 msec sera définie.
L'instruction DEFTR, comme l'instruction DEFTB, agit également sur les instructions SETD, RESD et OUTD. Si l'instruction DEFTR est présente dans le programme utilisateur alors la base de temps de ces instructions est fixée à 10 msec indépendamment de la valeur spécifiée dans DEFTR.

Usage

DEFTR résolution ; résolution ≥ 10 msec

Exemple

```
DEFTR   100     ; Résolution des Temporisateurs  = 100 msec
```

Flags Inchangés.

Voir aussi DEFTB.

DEFWPR DEFINE WRITE PROTECTED AREA in RUN
Définition de la zone protégée en Run

Description Définit les éléments à protéger contre l'écriture par une station du LAN 2.

DEFWPR définit les éléments à protéger contre l'écriture lorsque le CPU est en RUN (DEFWPH définit les éléments à protéger quand le CPU est en HALT ou STOP).

Dans les deux cas, l'opérande définit le type d'élément et la limite supérieure de la plage à protéger. Les éléments de 0 jusqu'à la valeur de l'opérande sont protégés contre l'écriture.

Les instructions doivent être exécutées une fois par type d'éléments à protéger : O, F, T|C, R. Si tous les éléments doivent être protégés, DEFWPR doit être exécuté 4 fois.

Si ces instructions ne sont pas présentes, **AUCUN** élément n'est protégé contre l'écriture en mode RUN.

Cette instruction est seulement exécutée par le CPU 0.

Usage

```
DEFWPR   adds        ; O 0-8191, F 0-8191, T 0-450|C 0-1599, R 0-4095
```

Exemple

```
DEFWPR   F 999  ; Flags 0-999 sont protégés (RUN)
                ;     et ne peuvent dès lors être modifiés
                ;     par une autre station LAN 2
```

Flags Inchangés.

Voir aussi DEFWPH, DEFTC, DEFTB, DEFVM, LAN 2.

Pratique Dans une application utilisant un LAN 2, 1000 flags et 500 Registres doivent être protégés contre les modifications par une autre station quand le CPU est en RUN.

```
XOB          16     ; Cold start XOB

DEFWPR   F 999      ; Protection des flags 0..999
DEFWPR   R 499      ; Protection des Registres 0..499
....
EXOB
```


DEFWPH DEFINE WRITE PROTECTED AREA in HALT

Définition de la zone à protéger à l'arrêt

Description Définit les éléments à protéger contre l'écriture par une station du LAN 2.

DEFWPH définit les éléments à protéger contre l'écriture lorsque le CPU est en HALT (DEFWPR définit les éléments à protéger quand le CPU est en RUN).

Dans les deux cas, l'opérande définit le type d'élément et la limite supérieure de la plage à protéger. Les éléments de 0 jusqu'à la valeur de l'opérande dont protégés contre l'écriture.

Les instructions doivent être exécutées une fois par type d'éléments à protéger : O, F, T | C, R. Si tous les éléments doivent être protégés, DEFWPH doit être exécuté 4 fois.

Si ces instructions ne sont pas présentes, **AUCUN** élément n'est protégé contre l'écriture en mode RUN.

Cette instruction est seulement utilisée par le CPU 0.

Usage

```
DEFWPH   adds        ; O 0-8191, F 0-8191, T 0-450|C 0-1599, R 0-4095
```

Exemple

```
DEFWPH   C 79    ; Temporisateurs et Compteurs 0-79 sont
                 ;     protégés contre l'écriture (en Halt)
```

Flags Inchangés.

Voir aussi DEFWPR, DEFTC, DEFTB, DEFVM, LAN 2.

Pratique Dans une application utilisant un LAN 2, 1000 flags et 500 Registres doivent être protégés contre les modifications par une autre station quand le CPU est en RUN et en HALT.

```
XOB          16     ; Cold start XOB

; Définition de la protection en RUN
DEFWPR   F 999   ; Protection des flags 0..999
DEFWPR   R 499   ; Protection des Registres 0..499

; Définition de la protection en HALT
DEFWPH   F 999   ; Protection des flags 0..999
DEFWPH   R 499   ; Protection des Registres 0..499
....
EXOB
```

12. Instructions SPECIALES

NOP	No OPeration	Pas d'opération
RTIME	Read TIME	Lecture de l'horloge
WTIME	Write TIME	Ecriture de l'horloge
PID	P.I.D. control algorithm	Algorithme de contrôle PID
TEST	TEST hardware	Test du hardware (matériel) PCD
DIAG	read XOB DIAGnostic	Lecture du diagnostique d'un XOB
SYSRD	SYStem ReaD	Lecture de paramètres système
SYSWR	SYStem WRite	Ecriture de paramètres système
SYSCMP	SYStem CoMPare	Comparaison paramètres système

Les instructions suivantes ne doivent plus être utilisées, elles sont seulement présentes pour assurer la compatibilité.

ALGI	AnaLoGue Input	Lecture d'une valeur analogique
ALGO	AnaLoGue Output	Ecriture d'une valeur analogique

Ces instructions sont seulement utilisées pour la carte PCA2.W1x.
Pour la lecture des cartes analogiques PCD1, PCD2, PCD4 et PCD6, consultez le manuel hardware approprié.

STHS	STart High Slow	Interrogation lente d'un élément
OUTS	OUT Slow	Positionnement lent suivant l'état de l'ACCU

Ces instructions sont utilisées pour l'accès de modules lents : PCA2.W2xx/W3xx.

Notes personnelles :

NOP NO OPERATION
Pas d'opération

Description Instruction sans action.

Cette instruction est utilisée pour écraser (effacer) d'autres instructions, ou pour laisser de l'espace dans le programme pour des modifications ultérieures.

Usage

NOP	**; Pas d'opérande**

Exemple

```
NOP          ; Aucune action
```

Flags Inchangés.

RTIME READ TIME
Lecture de l'horloge

Description Transfère le contenu de l'horloge interne dans deux Registres.
Seul le premier des deux Registres est spécifié dans l'instruction.
Après l'instruction RTIME, les Registres contiennent :

Chiffre numéro	9	8	7	6	5	4	3	2	1	0
Registre	0	0	0	0	Heure		Minutes		Secondes	
Registre + 1	0	Semaine		Jour/S	Année		Mois		Jour	

Semaine	Numéro de la semaine	1..53	
Jour S	Jour de la semaine	1..7	Lundi = 1, Dimanche = 7
Année	Année	0..99	
Mois	Mois de l'année	1..12	
Jour	Jour du mois	1..28/29/30/31 (selon le mois)	
Heure	Heure	0..23	
Minute	Minute	0..59	
Seconde	Seconde	0..59	

Le contenu du Registre est en binaire, PAS en BCD, mais il peut être converti en BCD en utilisant l'instruction MOV et DIGO.

Usage

```
RTIME     reg              ; Numéro du Registre R 0-4094
```

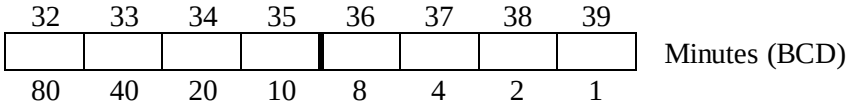
Exemple

```
RTIME     R 10     ; L'horloge est copiée dans les Registres 10 et 11
```

Flags Inchangés.

Voir aussi WTIME, MOV, DIGO.

Pratique Lorsque l'entrée I 3 est enclenchée, les minutes de l'horloge doivent être affichées en format BCD sur les sorties O 32-39.




```
COB         0
            0
STH       I 3
DYN       F 3
CPB       H 25
...
ECOB
```

```
PB          25
RTIME     R 20
MOV       R 20
          D 2
          R 99
          D 0
MOV       R 20
          D 3
          R 99
          D 1
DIGOR       2
          R 99
          O 32
EPB
```

WTIME WRITE TIME

Ecriture de l'horloge

Description Copie le contenu de deux Registres dans l'horloge interne.
Seul le premier des deux registres est spécifié.
Le format est identique à l'instruction RTIME :

Chiffre numéro	9	8	7	6	5	4	3	2	1	0
Registre	0	0	0	0	Heure		Minutes		Secondes	
Registre + 1	0	Semaine		Jour/S	Année		Mois		Jour	

Les valeurs BCD peuvent être chargées dans les Registres à partir de Flags en utilisant l'instruction MOV et DIGI.

Usage

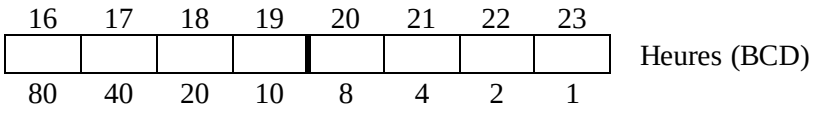
```
WTIME     reg                ; Registre source R 0-4094
```

Exemple WTIME R 500 ; Copie les Registres 500 et 501 dans l'horloge

Flags Inchangés.

Voir aussi RTIME, MOV, DIGI.

Pratique Après l'enclenchement de l'entrée I 4, les heures de l'horloge sont positionnées sur une nouvelle valeur. Cette valeur est lue sur les codeurs BCD des entrées I 16-23.




```
COB         0
            0
STH       I 4
DYN       F 4
CPB       H 26
...
ECOB
```

```
PB          26
RTIME     R 200
DIGIR       2
          I 16
          R 199
MOV       R 199
          D 0
          R 200
          D 4
MOV       R 199
          D 1
          R 200
          D 5
WTIME     R 200
EPB
```


PID PID CONTROL ALGORITHM
Algorithme de contrôle PID

Description Implémente un algorithme PID en utilisant les données contenues dans 13 Registres consécutifs.

Registre	Usage	Symbole		
+0	Nouveau résultat	Y_n	*	Valeur maxi : 'm' bits
+1	Résultat précédent	Y_{n-1}	*	
+2	Nouvelle valeur réglée	X_n	w	
+3	Valeur réglée précédente	X_{n-1}	*	
+4	Consigne	W_n	w	
+5	Consigne précédente	W_{n-1}	*	
+6	Facteur Proportionnel	F_p	w	* 256
+7	Facteur Intégral	F_i	w	* 256
+8	Facteur Dérivé	F_d	w	* 256
+9	Zone morte	D_r	w	
+10	Valeur de départ à froid	Y_s	w	Valeur de départ de Y_n
+11	Précision en bits	m	w	m = 8, 12 ou 16 bits
+12	Registre de travail	Z_s	*	Utilisé par le système

* Ces valeurs sont traitées par l'instruction PID.
w Ces valeurs doivent être écrites par le programme utilisateur.

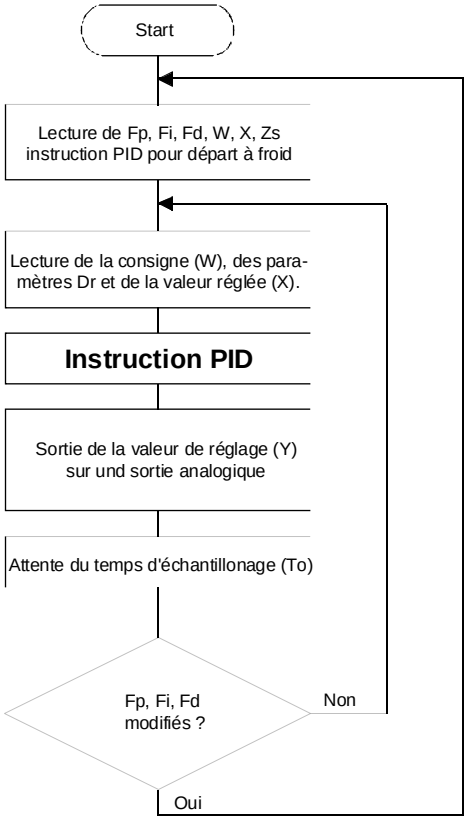
Usage

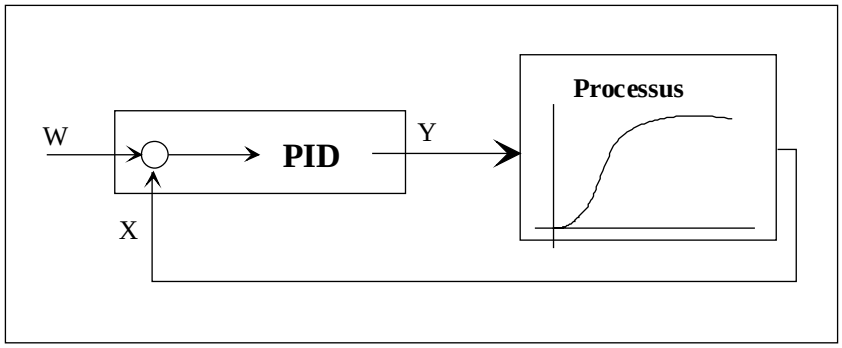
PID reg ; reg = adresse la plus basse de 13 Registres (R 0-4083)

Exemple
```
PID     R 1000  ; Utilise R 1000-1012 pour le PID
```

Flags Inchangés.

Pratique Une boucle PID classique consiste en :





Nouveau résultat Y_n :

C'est la nouvelle valeur de réglage à appliquer au processus, celle-ci est déterminée par l'équation avec : $Z_s = Z_s + (W_n - X_n)$:

$$Y_n = \frac{F_p}{256} * \{ \ (W_n - X_n) + \frac{F_i}{256} * Z_s + \frac{F_d}{256} * [(W_n - W_{n-1}) - (X_n - X_{n-1})] \ \}$$

Si le résultat dépasse la précision déclarée, il sera limité à sa valeur maximum (soit m bits) ou, en cas de résultat négatif à 0.

Résultat précédent Y_{n-1} :

Ancien résultat du calcul effectué lors de l'exécution précédente.

Nouvelle valeur réglée X_n :

La valeur réglée Xn doit être lue dans le processus et mémorisée dans le registre (R + 2) par le programme utilisateur.
La valeur réglée devra être de maximum 'm' bits.

Valeur réglée précédente X_{n-1} :

Ancienne valeur réglée, utilisée lors du précédent calcul.

Consigne W_n :

La consigne doit être écrite dans le registre (R + 4) par le programme utilisateur. La consigne ne doit pas dépasser 'm' bits.

Consigne précédente W_{n-1} :

Ancienne valeur de la consigne utilisée lors du calcul précédent.
Elle est utilisée pour détecter et évaluer l'ampleur d'un changement de consigne.

Facteur Proportionnel F_P :

Ce facteur détermine la caractéristique proportionnelle (amplification) du régulateur et doit être mémorisé dans le registre (R + 6) par le programme utilisateur.
Lors du calcul, seul les 16 bits de poids faible sont utilisés (0..65535)
Le facteur Proportionnel est déterminé comme suit :

$$F_p = \frac{1}{X_p} * 256 \qquad \text{avec} \qquad \text{Xp : bande proportionnelle}$$

Note : Pour entrer une bande proportionnelle de 5 %, le facteur F_p doit être :

$(1 / 0.05) * 256 = 5120$

Un démarrage à froid du PID doit être effectué après modification de F_p ou F_i

Facteur Intégral F_i :

Ce facteur détermine la caractéristique intégrale du régulateur et doit être mémorisé dans le registre (R + 7) par le programme utilisateur.
Lors du calcul, seul les 16 bits de poids faible sont utilisés (0..65535).
Le facteur Intégral est déterminé comme suit :

$$F_i = \frac{T_0}{T_i} * 256 \quad \text{with} \quad \begin{array}{l} T_0\text{: temps d'échantillonnage de l'instruction PID} \\ T_i\text{ : temps d'intégration} \end{array}$$

Un démarrage à froid du PID doit être effectué après modification de F_p ou F_i

Facteur Dérivé F_d :

Ce facteur détermine la caractéristique dérivée du régulateur et doit être mémorisé dans le registre (R + 8) par le programme utilisateur.
Lors du calcul, seul les 16 bits de poids faible sont utilisés (0..65535).
Le facteur Dérivé est déterminé comme suit :

$$F_d = \frac{T_d}{T_{0i}} * 256 \quad \text{with} \quad \begin{array}{l} T_0\text{: temps d'échantillonnage de l'instruction PID} \\ T_i\text{ : temps de dérivation} \end{array}$$

Zone morte D_r :

La zone morte définit la zone dans laquelle la valeur réglée peut varier sans modifier la valeur de réglage (Y_n).

Valeur de départ à froid Y_S (Cold Start) :

Cette valeur est utilisée par le programme système comme valeur de départ pour Y_n. Si le programme utilisateur écrit une valeur autre que 0 dans le registre de démarrage à froid, les opérations suivantes sont effectuées :

$$
\begin{aligned}
Y_n &= Y_s \\
Y_{n-1} &= Y_s \\
Z_s &= [(Y_s / F_p) - (W_n - X_n)] / F_i \\
W_{n-1} &= W_n \\
X_{n-1} &= X_n
\end{aligned}
$$

La valeur de Y_s est remise automatiquement à 0 par le programme système dès qu'elle a été utilisée.

Pour un démarrage à froid avec une valeur de sortie = 0, Y_s doit être mis à -1.

Quand $F_i = 0$, le PID ne peut pas être initialisé avec un Cold Start. Un Cold Start est néanmoins recommandé afin d'initialiser le registre de travail. Dans ce cas, la valeur Y_s est ignorée, le registre Z_s est mis à 0 et Y_n prend la valeur de la partie proportionnelle de l'algorithme.

Note : Le changement de contrôle manuel / automatique est une application typique du démarrage à froid. Pour assurer une transition sans choc, Y_s doit être positionné sur la valeur réglée courante lors du passage.

Résolution ou précision en bits m :

La valeur maximum de toutes les variables utilisées est déterminée par la résolution (précision en bits).
Si m = 8 : 8 bits sont utilisés dans les calculs (0..255)
Si m = 12 : 12 bits sont utilisés dans les calculs (0..4095)
Si m = 16 : 16 bits sont utilisés dans les calculs (0..65535)

La résolution est en général définie par le module analogique utilisé pour la sortie de la valeur de réglage.

Temps d'échantillonnage T_o :

Le temps d'échantillonnage T_o doit être programmé à l'extérieur de l'instruction PID en utilisant un Temporisateur.
En pratique : $T_o \approx 0{,}1$ fois les constantes de temps du processus; mais au minimum de 80 msec.

Capacité de calcul :

Le registre de travail Z_s a une capacité maximum de 2^{31}.
Quand on utilise des valeurs de 16 bits (m = 16), un dépassement peut survenir; dans ce cas, le PID ne fonctionne plus correctement.
Pour éviter ce problème, le facteur F_p doit être ≥ 2 si m = 16 (Il n'y a aucun problème quand m = 8 ou 12).

TEST TEST HARDWARE

Test du hardware (matériel) PCD

Description Effectue, conditionnellement ou non, un ou plusieurs tests du hardware (matériel) du PCD.

Si un test ne passe pas, l'exécution de l'instruction est arrêtée, et l'ACCU est mis Low. Si tous les tests sélectionnés sont réussis, l'ACCU est mis High.

Des tests individuels peuvent être sélectionnés comme suit :

Valeur	Bit numéro	Description du test
	11	
400	10	Perte de Mémoire
200	9	Mémoire d'extension corrompue
100	8	Mémoire RAM Système
	7	
40	6	Checksum du firmware
20	5	Ligne série
10	4	Horloge
	3	
4	2	Cheksum Programme Utilisateur / Texte
2	1	RAM Programme Utilisateur / Texte
1	0	RAM Publique (F, T, C, R, Mailbox)

Pour chaque bit positionné, le test correspondant est effectué.

Les tests 0 et 4 sont seulement effectués lorsque le CPU en test est le seul en Run; si un des autres CPU est en Run, ces tests ne sont PAS effectués.

Note: Certains tests sont très lents, et ne doivent dès lors pas être effectués pendant le déroulement normal du programme; ces tests peuvent être effectués lors du démarrage du PCD ou pendant une période idéale.

L'opérande ne peut pas être passée comme paramètre d'un bloc de fonction.

Usage

```
TEST   [cc] numéro    ; cc = code conditionnel (H|L|P|N|Z|E)
                      ; numéro = numéro qui détermine le test à effectuer
```

Exemple

```
TEST      50   ; Teste les EPROM système (40)
               ;      + horloge hardware (10).
TEST    L 4    ; Si l'ACCU = L, alors le programme utilisateur
               ;      et la mémoire Texte (EPROM) sont testés
```

Flags L' **ACCU** est mis High (1) si tous les tests passent, Low si un test rate.

Test de la RAM Publique (Valeur = 1)

Teste la RAM qui contient les F/R/T/C selon le procédé sauve-écrit-lit-compare et restaure.
Ce test n'est pas exécuté si un autre CPU est en RUN dans un environnement multiprocesseur.

ACCU	Flag Erreur	Description
0	1	Autre CPU en RUN
0	0	Erreur dans la RAM Publique
1	x	RAM Publique OK

Test de la RAM du Programme Utilisateur/Texte (Valeur = 2)

Teste la RAM qui contient le programme et les textes suivant le procédé sauve-écrit-lit-compare et restaure.
Si la mémoire n'est pas une RAM ou si la RAM est protégée contre l'écriture, alors le Test Checksum du Programme Utilisateur/texte est effectué.

ACCU	Flag Erreur	Description
0	1	En-tête du programme non valide
0	0	RAM défectueuse
1	0	RAM OK

Test Checksum du Programme Utilisateur/Texte (Valeur = 4)

Vérifie la checksum de l'entièreté du programme utilisateur/texte qui est utilisée pour ce CPU.

ACCU	Flag Erreur	Description
0	1	En-tête de programme non valide
0	0	Checksum fautive
1	0	Checksum OK

Test de l'horloge (Valeur = 10)

Teste l'existence de l'horloge " RTC " (Real Time Clock) et si elle s'incrémente correctement. Si un autre CPU accède à l'horloge durant ce test, il sera bloqué pendant 15 msec.

ACCU	Flag Erreur	Description
0	1	Horloge n'existe pas
0	0	Horloge défectueuse
1	0	Horloge OK

Test Lignes Séries (Valeur = 20)

Les lignes séries sont testées en initialisant le port en mode "loopback" et en envoyant une séquence de caractères qui sont vérifiés à la réception. Si une des lignes séries est assignée, le test n'est pas effectué.

ACCU	Flag Erreur	Description
0	1	Un des ports est assigné
0	0	Port défectueux
1	0	Lignes séries OK

Test du Firmware (Checksum) (Valeur = 40)

Les EPROMs contenant le firmware sont contrôlées (Cheksum).

ACCU	Flag Erreur	Description
0	0	Checksum non valide
1	0	Checksum OK

Test de la Mémoire RAM du système (Valeur = 100)

Les mémoires RAM du système sont contrôlées.

ACCU	Flag Erreur	Description
0	0	Checksum non valide
1	0	Checksum OK

Test de l'Extension Mémoire (Valeur = 200)

L'extension mémoire est testée durant la phase de démarrage de l'automate; si celle-ci est corrompue, un flag interne est positionné.

ACCU	Flag Erreur	Description
0	0	Extension Mémoire corrompue
1	0	Pas de corruption

Test de la Perte de Mémoire (Valeur = 400)

Si la séquence de caractères mémorisées dans le "mailbox" n'est pas valable lors du test, lors de la phase de démarrage: il est supposé que la RAM Publique a été corrompue pendant que l'automate était hors tension (batterie déchargée).

ACCU	Flag Erreur	Description
0	0	Mémoire Publique corrompue
1	0	Pas de corruption

DIAG READ XOB DIAGNOSTIC

Lecture du diagnostique d'un XOB

Description Mémorise dans un bloc de 12 Registres les informations de diagnostiques relatives au dernier bloc d'exception (XOB) exécuté ou du XOB courant. L'opérande est l'adresse la plus basse de 12 Registres.
L'instruction DIAG est normalement utilisée dans un XOB.

Utilisation des Registres:

Registre numéro		
0	XOB Numéro	Numéro du dernier XOB ou du XOB courant
+ 1	Ligne de Programme	Ligne de programme quand le XOB a été appelé
+ 2	Registre d'Index	Valeur du Registre d'Index lors de l'appel
+ 3	Ligne de programme du COB	Ligne de programme de l'appel niveau 0
+ 4	Imbrication niveau 1	Ligne de programme de l'appel niveau 1
+ 5	Imbrication niveau 2	Ligne de programme de l'appel niveau 2
+ 6	Imbrication niveau 3	Ligne de programme de l'appel niveau 3
+ 7	Imbrication niveau 4	Ligne de programme de l'appel niveau 4
+ 8	Imbrication niveau 5	Ligne de programme de l'appel niveau 5
+ 9	Imbrication niveau 6	Ligne de programme de l'appel niveau 6
+ 10	Imbrication niveau 7	Ligne de programme de l'appel niveau 7
+ 11	Non utilisé	Réserve

La ligne de programme de l'appel de bloc (Information de niveau d'imbrication) donne l'adresse de la ligne de programme où l'appel (CFB, CPB, ...) à eu lieu. Cette information permet de déterminer exactement où se situait le programme lorsque le COB a été appelé.
Les informations les plus importantes sont données par les registres R et R+1.

L'opérande ne peut pas être passée comme paramètre d'un bloc de fonction

Usage

DIAG	**reg**	**; R 0-4084, adresse la plus basse de 12 Registres**

Exemple

```
DIAG    R 1000  ; Mémorise les informations de diagnostique
                ;      dans les Registres 1000-1011
```

Flags Inchangés.

Voir aussi Guide Utilisateur.

Pratique L'adresse de la ligne où se produit une erreur doit être imprimée.

```
        XOB       13
        DIAG    R 1000
        STXT      1
                  100
TEXT 100  "$D $H: Flag Erreur positionné à l'adresse $R1001<CR><LF>"
        EXOB
```


SYSRD SYSTEM READ
Lecture de paramètres système

Description Lecture de paramètres systèmes tels que :

type de PCD, type de CPU, version firmware, nom du programme utilisateur, paramètres S-Bus, ...

Usage

```
SYSRD    Fonction    ; Code Fonction, K code, R 0-4095
         Résultat    ; Résultat de la lecture, R 0-4095
```

Fonction	K x or R x :	Constante ou registre contenant le code de fonction. Cette instruction peut être utilisée de manière directe (en utilisant une constante pour le code de fonction) ou indirecte en utilisant un registre.
Résultat	R 0..4095 :	Registre contenant le résultat.

Exemple

```
SYSRD   K 5000    ; Lit le type de PCD en ASCII
        R 20      ;      et met le résultat dans R 20
```

Flags Le flag **Erreur** est positionné si le code de fonction n'existe pas.

Voir aussi SYSWR.

Code de fonction

Code	Description de la fonction	Résultat
2000 2001 2002 2003 2004 2005 ¦ 2049	**Lecture EEPROM Utilisateur** Registre 0 Registre 1 Registre 2 Registre 3 Registre 4 Registre 5 Registre nn Registre 49 PCD1 Autres PCD	Valeur contenue dans l'EEPROM
5000 5010	**Lecture type PCD** en ASCII en décimal	ASCII / Décimal / Type " D1" / 1 / PCD1 " D2" / 2 / PCD2 " D4" / 4 / PCD4 " D6" / 6 / PCD6
5100 5110	**Lecture type CPU** en ASCII en décimal	ASCII / Décimal / Type " M1_" / 10 / PCD1.M1 " M1_" / 10 / PCD2.M12 " M15" / 15 / PCD2.M15 " M11" / 11 / PCD4.M11 " M12" / 12 / PCD4.M12 " M14" / 14 / PCD4.M14 " M24" / 24 / PCD4.M24 " M34" / 34 / PCD4.M34 " M44" / 44 / PCD4.M44 " M1_" / 10 / PCD6.M1 " M2_" / 20 / PCD6.M2 " M3_" / 30 / PCD6.M3 " M54" / 54 / PCD6.M5
5200 5210	**Lecture version Firmware** en ASCII en décimal	Exemples de réponses : " $4C", " 004", " X41" Ex : 5 dec pour Version 005 -1 dec pour '$', 'X', 'β'
5400	**Lecture du nom du Programme Utilisateur** en ASCII Le nom contient toujours 8 caractères ASCIIs	Rx contient les 4 premiers caractères du nom du programme en ASCII. Rx+1 contient les 4 caractères suivants en ASCII.
6000	**Lecture numéro de station S-Bus**	Exemple de résultat : 2 station numéro 2 -1 station non configurée
6010	**Lecture du S-Bus PGU TN delay**	Exemple de résultat : 10 Délai en msec. -1 S-Bus non configuré
6020	**Lecture du S-Bus PGU TS delay**	
6030	**Lecture du S-Bus PGU timeout**	

Code	Description de la Fonction	Résultat	
6040	**Lecture du S-Bus PGU baudrate**	Exemple de résultat : 9600 bps -1 S-Bus non configuré	
6050	**Lecture mode S-Bus PGU**	Statut	Déc.
		BREAK sans modems	0
		PARITY sans modems	1
		DATA sans modem	2
		BREAK avec modems	10
		PARITY avec modems	11
		DATA avec modem	12
		S-Bus non configuré	-1
6060	**Lecture port S-Bus PGU**	Exemple de résultat : 1 S-Bus PGU port configuré sur le port 1 -1 S-Bus non configuré	
6070	**Lecture niveau S-Bus**	Statut	Déc.
		S-Bus niveau 1 (réduit)	1
		S-Bus niveau 2 (complet)	2
		S-Bus non configuré	-1
6080	**Lecture processeur PGU** (S-Bus ou P8 protocole)	CPU 0	0
		CPU 1	1
6100	**Lecture du statut du modem** Lecture du statut de la connexion modem. Cette information permet à l'utilisateur de savoir à quelle étape se trouve le modem lors de son initialisation. Exemple de résultat : 2 PCD en attente d'une connexion avec le modem. 6 .. 39 PCD initialise le modem. 40 Ré-assignation du port série, mode SS1/SS0. 45..49 Connexion avec le modem perdue. Ceci est l'étape juste avant la ré-initialisation du modem. 50 Tout est OK et le PCD est "online" en mode SS0/1.		
6500 6510 6520	**Lecture du type de modem** **Lecture du reset string du modem** **Lecture de l'initialisation string du modem** . Lit la chaîne de caractères spécifiée depuis l'en-tête du programme et la place dans un bloc de registres commençant avec Rx.		
7000	**Lecture du compteur système**	0.. 2.147.483.647	
	Un compteur système interne est incrémenté chaque milliseconde. Ce compteur système est remis à 0 au démarrage seulement, un "Restart Cold" ne change pas sa valeur. La période du compteur système est exactement de **24 jours 20 heures 31 min 23 sec 647 ms** Pour un exemple, voir l'instruction SYSCMP.		

Lecture de l'horloge

Il est possible de lire chaque valeur séparément avec le code de fonction approprié. La valeur retournée est en format décimal .
Les codes de fonctions nécessaires se situent entre 7050 et 7090. Le code de fonction 7090 permet de savoir le nombre de secondes qui se sont écoulées depuis minuit (00:00:00), 01/01/1970, temps universel coordonné ; cette fonction dépend de l'horloge système (RTC).

Table des codes de fonction :

Code	**Pour**
7050	Secondes
7051	Minutes
7052	Heure
7053	Minutes et secondes
7054	Heure et minutes
7055	Heure, minutes et secondes
7060	Jour
7061	Mois
7062	Année
7063	Mois et jour
7064	Année et mois
7065	Année, mois et jour
7070	Jour de la semaine
7071	Semaine de l'année
7072	Semaine de l'année et jour de la semaine
7081	L'heure ("time") et la date (sur 2 registres)
7090	Secondes écoulées depuis 1970

Exemples :

```
1) SYSRD     7055     ; Lecture heure, minutes et secondes
           R 0
```

Résultat : R0 = 120203

```
2) SYSRD     7081     ; Lecture heure ("time") et date
           R 0
```

Résultat : R0 = 120203 ; R1 = 991130

Note : Si l'utilisateur utilise un code de fonction entre 7050 et 7090 qui ne se trouve pas dans la tabelle ci-dessus, le XOB 13 est appelé et le flag erreur positionné.

SYSWR SYSTEM WRITE

Ecriture de paramètres système

Description Cette instruction est le complément de SYSRD et autorise la modification des informations systèmes ou l'initialisation de fonctions systèmes via le programme utilisateur.

Usage

```
SYSWR    Fonction    ; Code Fonction, K code, R 0-4095
         Valeur      ; Valeur à écrire
```

Fonction	K x or R x :	Constante ou registre contenant le code de fonction Cette instruction peut être utilisée de manière directe (en utilisant une constante pour le code de fonction) ou indirecte en utilisant un registre.
Valeur	K y R 0..4095	Valeur à écrire Registre contenant la valeur à écrire.

Exemple

```
SYSWR   K 4005    ; Inhibe le XOB 5
        K 0
```

Flags Le flag **Erreur** est positionné si la fonction n'existe pas.

Voir aussi SYSRD.

Code de fonction

Code	Description de la fonction
1000	**System Watchdog** (seulement pour PCD1 et PCD2) Valeur autorisée pour K y ou R y : 0 Désactive le Watchdog 1 Active le Watchdog et fait un restart cold si il n'est pas rafraîchit endéans les 200msec. 2 Active le Watchdog et appelle le XOB 0 avant de faire un restart cold si il n'est pas rafraîchit endéans les 200msec. Quand le Watchdog a été activé, cette instruction doit être répétée continuellement au moins toutes les 200 msec. Le XOB 0 appelé par le Watchdog se différencie par le XOB 0 en cas de défaut d'alimentation par le message présent dans le 'History List" : Watchdog : "XOB0 WDOG START" Alimentation : "XOB 0 START EXEC".
2000 2001 2002 2003 2004 2005 ⋮ 2049	**Ecriture dans l' EEPROM** (pas sur tous les PCD, dépend du hardware) Le PCD est équipé d'une EEPROM dans laquelle jusqu'à 49 registres max. peuvent être écrits de la manière suivante : Code Fonction (2000 - 2049) indique le registre de l'EEPROM 0 .. 49. Valeurs permises pour K y ou R y : R y Registre contenant la valeur à écrire dans l'EEPROM. PCD1 : max. 5 registres (0 .. 5) Autres PCD : max. 49 registres (0 .. 49) <u>Attention :</u> Un maximum de 100.000 écritures est possible dans l'EEPROM; de ce fait, cette instruction ne doit pas être exécutée trop fréquemment. L'instruction SYSWR nécessite 20 msec. pour son exécution : elle ne doit donc pas être utilisée dans le XOB 0. Cette instruction peut être utilisée pour configurer des valeurs d'initialisation.
4000	**Longueur de la file d'attente des XOBs**. Les XOBs 14/15/17/18/19/20/25 utilisent un mécanisme de file d'attente (queue ou queuing mechanism). Si un XOB est actif alors que survient un autre XOB, celui-ci est placé dans une queue qui a une capacité de 127 entrées par XOB. Si cette limite est dépassée, le XOB 7 est appelée et la queue est effacée. Le message d'erreur "SYSTEM OVERLOAD" est inscrit dans la table historique. Une limite de 127 entrées est quelquefois trop importante pour des applications en temps réel; il est donc possible de re-définir cette limite à l'aide de cette instruction. Cette limite est commune à tous les XOBs qui utilisent ce mécanisme. Valeurs autorisées pour R y ou K y : 0 .. 127

Code	Description de la fonction
4005 4013	**Autorisation / Inhibition XOB 5/13** Inhibe les XOBs 5 ou 13. Dans certains cas, l'exécution de ces XOBs juste au moment où ils ont été provoqués, complique le déroulement du programme utilisateur. Si un XOB est invoqué une ou plusieurs fois alors qu'il est inhibé, celui-ci sera appelé une seule fois lorsqu'il sera à nouveau autorisé. Code de fonction : 4005 XOB 5 4013 XOB 13 Valeurs autorisées pour R y ou K y : 0 Inhibe le XOB 1 Autorise le XOB 2 Efface le flag Erreur dans le COB courant et dans le XOB actif (pour K 4013 seulement)
4014 4015	**Installation du XOB 14 /15** Configure un XOB appelé périodiquement à une fréquence définie par K y ou R y. Il est possible de configurer deux XOBs périodique avec une période variant de 5 msec à 1'000 sec. La valeur de K y ou R y est donnée en msec; si elle est 0, le XOB est désactivé. Cette instruction peut être exécutée à n'importe quel moment. Si un XOB est en cours quand un autre XOB est activé, celui-ci sera mis en attente jusqu'au moment où aucun XOB n'est actif et il sera alors exécuté. Les XOBs sont seulement exécutés si le CPU est en RUN ou en CONDITIONAL RUN. Code de Fonction : 4014 Configure le XOB 14 4015 Configure le XOB 15 Valeurs autorisées pour R y ou K y : 5 .. 1'000'000
4017 4018 4019	**Exécute le XOB 17 / 18 / 19** Exécute le XOB spécifié par Kx ou Rx sur le CPU désigné par K y ou R y. Les XOBs 17/18/19 sont des XOBs utilisateurs qui peuvent être activés via S-Bus ou par le programme utilisateur. Les XOBs sont seulement exécutés si le CPU est en RUN ou en CONDITIONAL RUN. Code de Fonction : 4017 Exécute le XOB 17 4018 Exécute le XOB 18 4019 Exécute le XOB 19 Valeurs autorisées pour R y ou K y : 0 .. 6 CPU où le XOB doit être exécuté 7 Exécution du XOB sur son propre CPU 8 Exécution du XOB sur tous les CPUs.

Code	Description de la fonction
6000	**Ecriture du numéro de station S-Bus** Change le numéro de station S-Bus suivant la valeur de K y ou R y (dans la RAM système et dans l'EEPROM). Cette instruction fonctionne que le programme soit en RAM (protégé en écriture), en EPROM ou en Flash EPROM. Valeurs autorisées pour K y ou R y : 0 .. 254 **Ecriture dans l' EEPROM** (pas sur tous les PCD, dépend du hardware) Attention : Un maximum de 100.000 écritures est possible dans l'EEPROM; de ce fait, cette instruction ne doit pas être exécutée trop fréquemment. L'instruction SYSWR nécessite 20 msec. pour son exécution : elle ne doit donc pas être utilisée dans le XOB 0.
7000	**Conversion FFP-IEEE** Convertit une valeur virgule flottante entre le format FFP (Fast Floating Point) et IEEE. Le format FFP est utilisé par toutes les instructions "virgule flottante" du SAIA PCD. Dès qu'une conversion en format IEEE a été réalisée sur une valeur, celle-ci ne peut plus être utilisée pour une opération arithmétique en virgule flottante. Code de Fonction : 7000 FFP vers IEEE 7001 IEEE vers FFP Valeurs autorisées pour R y : R y contient la valeur à convertir; le résultat est mémorisé dans le même registre.

Ecriture de l'horloge

Il est possible d'écrire chaque valeur séparément avec le code de fonction approprié. Chaque valeur utilise 2 digits, par ex. :

12 h 2 min. et 3 sec. sera écrit 120203.

Les codes de fonctions nécessaires se situent entre 7050 et 7081 selon la tabelle ci-dessous.

Table des codes de fonction :

Code	**Pour**
7050	Secondes
7051	Minutes
7052	Heure
7053	Minutes et secondes
7054	Heure et minutes
7055	Heure, minutes et secondes
7060	Jour
7061	Mois
7062	Année (< 100)
7063	Mois et jour
7064	Année et mois
7065	Année, mois et jour
7081	L'heure ("time") et la date (sur 2 registres)

Exemples :

```
1)  LD      R 0
              120203

    SYSWR       7055     ; Ecriture heure, minutes et secondes
            R 0


2)  LD      R 0
              120203
    LD      R 1
              991130

    SYSWR       7081     ; Ecriture heure ("time") et date
            R 0
```

Note : Si l'utilisateur utilise un code de fonction entre 7050 et 7081 qui ne se trouve pas dans la tabelle ci-dessus, le XOB 13 est appelé et le flag erreur positionné.

SYSCMP SYSTEM COMPARE

Comparaison de paramètres système

Description

L'instruction SYSCMP transforme n'importe quel registre en un pseudo Temporisateur.

SYSCMP compare la somme des deux opérandes au Compteur Système et positionne l'ACCU suivant le résultat.

Si le résultat de la somme est plus grand que le Compteur Système, l'ACCU est mis HIGH (1); si le résultat est plus petit ou égal, l'ACCU est mis LOW (0).

La combinaison de cette instruction avec SYSRD K 7000 permet de réaliser des temporisations ayant une résolution de 1 msec.

Il est également possible de mesurer le temps écoulé entre deux évènements avec une résolution de 1 msec.

Usage

```
SYSCMP   Rx            ; R 0..4095
         K y or R y    ; K 0..16383 ou R 0..4095
```

Exemple

```
SYSCMP   R 100    ; Compare le contenu des Registres (100 + 1500)
         K 1500   ;     au compteur système et positionne l'ACCU

SYSCMP   R 100    ; Compare le contenu de (R 100 + R 101)
         R 101    ;     au compteur système et positionne l'ACCU
```

Flags

L' **ACCU** est mis H/L suivant le résultat.

Voir aussi

SYSRD.

Pratique

Programmation d'un Temporisateur avec une résolution de 1 msec avec SYSRD et SYSCMP.

```
          COB       0
                    0
          …
          LD        R 100   ; Chargement du temps en ms (1500)
                    K 1500  ;     dans le R 100

          SYSRD     K 7000  ; Lecture du Compteur Système dans R 101
                    R 101

wait:     SYSCMP    R 101   ; Comparaison du Compteur Système avec
                    R 100   ;     (R 100 + R 101)
          JR        H wait  ; Si l'ACCU est HIGH alors bouclage
          …
          ECOB
```

ALGI ANALOGUE INPUT

Lecture d'une valeur analogique

Description

Lit une valeur 12-bits d'une carte analogique PCA2.W1xx, et la mémorise dans le Registre spécifié.

Le 1er opérande contient le numéro du canal A/D (0-7) et l'adresse de base du module.

Le 2ème opérande est le numéro du Registre de destination.

Si le premier opérande est donné comme le paramètre d'un FB, le canal A/D et l'adresse de base doivent être donnés sur la même ligne.

Usage

```
ALGI[X]    c    base          ; c = canal 0-7, base = 0-8176
           registre  (i)      ; Registre destination R 0-4095
```

Exemple

```
ALGI    2 64    ; Lit la valeur analogique du canal 2,
        R 10    ;      adresse de base du module: 64
                ;      et mémorisation dans R 10
```

Flags

Les flags **Zéro** (Z) et **Signe** (P ou N) sont positionnés en fonction de la valeur lue.
Le flag **Erreur** (E) est toujours mis Low.

Voir aussi

ALGO.

Note

Cette instruction ne peut pas être utilisée pour les modules PCD2, PCD4 et PCD6 (consultez les manuels hardware respectifs).

ALGO ANALOGUE OUTPUT

Ecriture d'une valeur analogique

Description Transfère une valeur 12-bits binaire d'un Registre vers une carte analogique PCA2.W1xx .

Le 1er opérande est le Registre à transférer.

Le 2ème opérande contient le numéro du canal D/A (0-7) et l'adresse de base du module.

Si le second opérande est donné comme le paramètre d'un FB, le canal D/A et l'adresse de base doivent être donnés sur la même ligne.

Usage

```
ALGO[X]   Registre  (i)      ; Registre Source R 0-4095
          c   base           ; c = canal 0-3, base = 0-8176
```

Exemple

```
ALGO      R 100  ; Transfère la valeur du R 100
          3 128  ;      sur le canal 3 du module à l'adresse de base 128
```

Flags Inchangés.

Voir aussi ALGI.

Note **Cette instruction ne peut pas être utilisée pour les modules PCD2, PCD4 et PCD6 (consultez les manuels hardware respectifs).**

STHS START HIGH SLOW

Interrogation lente d'un élément

Description

L'ACCU est positionné suivant l'état logique de l'élément adressé, en général une entrée (Input).

Cette instruction est identique à STH, excepté que le timing sur le bus du PCD est ralenti pour pouvoir accéder à des modules E/S lents.

La vitesse d'exécution du programme n'est pas affectée de manière significative.

Cette instruction doit être utilisée pour les modules analogiques PCA2.W2xx et PCA2.W3xx.

Usage

STHS[X] élément (i) ; I 0-8191, O 0-8191, F 0-8191

Exemple

```
STHS    I 25
```

Flags

L'**ACCU** est positionné suivant l'état logique de l'élément spécifié.

Voir aussi

OUTS, STH, Instructions BIT.

OUTS SET ELEMENT FROM ACCUMULATOR SLOW

Positionnement lent suivant l'état de l'ACCU

Description

L'élément spécifié, généralement une sortie, est positionné sur l'état de l'ACCU.

Cette instruction est identique à OUT, excepté que le timing sur le bus du PCD est ralenti pour pouvoir accéder à des modules E/S lents.

La vitesse d'exécution du programme n'est pas affectée de manière significative.

Cette instruction doit être utilisée pour les modules analogiques PCA2.W2xx et PCA2.W3xx.

Usage

OUTS[X] élément (i) ; I 0-8191, O 0-8191, F 0-8191

Exemple

```
OUTS    O 32
```

Flags

Inchangés.

Voir aussi

OUT, OUTD, STHS.

Pratique

La valeur analogique du canal 0 d'une carte PCA2.W2xx (adresse de base 96) doit être lue et mémorisée dans le Registre 100.
Après que la conversion soit déclenchée par l'instruction OUTS, les 8 bits binaires peuvent être lus à partir de l'adresse de base du module + 8 (= 104).

```
COB         0
            0
...
ACC       H           ; Pour être certain que l'ACCU = H
OUTS        96        ; Sélection du canal
...                     ;       Attendre 100 msec *)
CPB         RD_VAL    ; Appel du programme bloc RD_VAL
....
ECOB

PB          RD_VAL
BITIR       8         ; Lecture de 8 bits binaires en ordre inverse
          I 104       ;       des adresses 104 .. 111
          R 100       ;       dans le Registre 100
EPB
```

Le module analogique PCA2.W2x a un temps de conversion de ≤100 msec. Ce temps d'attente peut être réalisé avec un certain nombre de NOP consécutifs. (Le nombre de NOPs est dépendant du type de CPU).

Notes personnelles :

13. Liste Historique

Ce qui suit est une description détaillée de toutes les erreurs qui peuvent être inscrites dans le “HISTORY LIST” (liste historique) ou dans le registre “HALT REASON REGISTER”.
Cette liste peut être visualisée avec le debugger grâce à la commande : “Display History”.

Erreurs qui provoquent un XOB (si il est programmé)

Message	HALT	XOB	Système *)	Description
XOB START EXEC	N	0	Tous	XOB 0 a été démarré
XOB 0 EXECUTED	N	0	Tous	XOB 0 a été terminé complètement pendant le “power down”
XOB 0 WDOG START	N	0	1	Le watchdog système a été activé
EXTERN PWR FAIL	N	1	2, 5, 6	Problème d’alimentation dans un rack d’extension
PARITY FAILURE	N	4	5, 6	Erreur Rack PCD6 (Backplane)
SYSTEM OVERLOAD	N	7	Tous	Dépassement de la capacité du mécanisme de queue pour les XOBs niveau 3
ILLEGAL OPCODE	N	8	Tous	Exécution du XOB 8 suivi d’un “restart cold” suite à la détection d’une instruction non valide
>32 ST/TR ACTIVE	N	9	Tous	Trop de branches actives simultanément (GRAFTEC)
>7 CALL LEVELS	N	10	Tous	Dépassement de capacité du nombre de niveaux PB/FB

Les erreurs suivantes ont une ligne fixe dans la table des historiques et possèdent en plus un compteur d’erreur.

Message		HALT	XOB	Système	Description
BATT FAIL	000	N	2	2, 4, 5, 6	Batterie déchargée
IO QUIT FAIL	000	N	5	4, 5, 6	Un emplacement non équipé d’E/S a été adressé.
IR OVERFLOW	000	N	12	2, 4, 5, 6	Le registre d’index a été incrémenté au-delà de 8191
ERROR FLAG	000	N	13	2, 4, 5, 6	Flag Erreur positionné

*) Système:
1 : PCD1
2 : PCD2
4 : PCD4
5 : PCD6.M540
6 : PCD6.M1.., M2.., M3..

Erreurs lors du démarrage du PCD (System Startup Errors)

Ces erreurs sont détectées lors de la mise sous tension du PCD.

Message	HALT	Système	Description
RTC FAILURE	Oui	Tous	Horloge détectée mais ne fonctionne pas correctement.
DUART HW ERROR	Oui	Tous	Un des DUARTs est défectueux (communication)
CHECKSUM FAIL	Oui	Tous	Erreur de checksum EPROM Programme Utilisateur
BAD TXT/DB TABLE	Oui	Tous	Causée par un problème lors de l'initialisation des Textes/DBs
TXT/DB HW ERROR	Oui	Tous	Causée par un problème lors de l'initialisation des Textes/DBs
BAD MEM EXT INIT	Oui	Tous	Causée par un problème lors de l'initialisation des Textes/DBs dans la mémoire d'extension
USR MEM HW ERROR	Oui	6	Causée par un problème de la mémoire programme utilisateur
CPU SYNCH ERROR	Oui	6	Causée par un time-out du 2ème CPU
CPU FIRMWARE MIX	Oui	4, 6	Système multiprocesseurs équipé de firmware incompatibles

Erreurs systèmes

Ces erreurs sont mémorisées dans le registre "HALT REASON" qui peut être visualisé par le debugger quand un HALT est détecté. Ces erreurs peuvent survenir lors de la mise sous tension ou en RUN.

Message	HALT	Système	Description
BUS QUIT FAILURE	Oui	Tous	Le firmware a essayé d'accéder à une adresse non existante
68K INVALID OPC	Oui	Tous	Instruction assembleur 68000 non valide
68K ADDR ERROR	Oui	Tous	Accès à une adresse impaire
ZERO DIVIDE	Oui	Tous	Erreurs internes au firmware
68K CHK INSTR	Oui	Tous	…
68K TRAPV INSTR	Oui	Tous	…
PRIVILEGE VIOL	Oui	Tous	…
TRACE	Oui	Tous	…
ILLEGAL AUTO VEC	Oui	Tous	…
INTERRUPT ERROR	Oui	Tous	…
RESERVE INT	Oui	Tous	…

Erreurs de Programmation ou de Configuration

Les erreurs suivantes sont détectées à la mise sous tension du PCD.

Message	HALT	Système	Description
EVERYTHING IS OK	Non	Tous	Message normal
MODIFIED PROGRAM	Non	Tous	Le programme utilisateur a été modifié avec le Debugger. Seulement indiqué quand le programme est “write protect”.
CPU NUMBER > 6	Oui	6	Le numéro de CPU mis à l’aide d’interrupteurs DIL n’est pas valable.
CPU 0 START FAIL	Oui	4, 5, 6	CPU 1-6 : Aucun CPU ne peut être mis en RUN sans un programme dans le CPU 0
INIT-FAILURE	Oui	Tous	Plus de 32 étapes initiales dans le GRAFTEC
HEADER FAIL	Oui	Tous	L’en-tête du programme utilisateur est corrompu
NO PROGRAM	Oui	Tous	CPU sans programme
MEM-EXT CORRUPT	Oui	Tous	Mémoire d’extension en RAM corrompue
INVALID OPCODE	Oui	Tous	Instruction invalide chargée dans le CPU
MEDIA CORRUPTION	Oui/ Non	Tous	Causée par un problème de batterie
DOUBLE TIME BASE	Oui	Tous	Instructions DEFTB et DEFTR dans le même programme
BAD MODEM STRING	Oui	1	Chaîne de caractères pour le Modem trop longue EEPROM

Les erreurs suivantes sont détectées quand le PCD est en RUN, celles qui provoquent un HALT seront aussi mémorisées dans le registre HALT.

Message	HALT	Système	Description
BLOC NONEXISTENT	Oui	Tous	Appel d’un PB, FB, SB, ST, TR inexistant
HALTED BY LAN 2	Oui	4, 6	Le coprocesseur LAN 2 a mis le CPU en HALT
LAN 2 WATCHDOG	Non	4, 6	Activation du warchdog LAN 2
HALT INSTRUCTION	Oui	Tous	Exécution de l’instruction HALT
MANUAL HALT	Oui	4, 5, 6	CPU arrêté par l’interrupteur HALT
HALTED BY CPU 0	Oui	4, 6	Processeur(s) arrêté(s) par le CPU 0
SBUS-PGU ERROR	Oui	Tous	Assignation invalide sur un port S-BUS PGU

Notes personnelles :

Vos coordonnées :

Société :
Service :
Nom :
Adresse :

Téléphone :

Date :

A renvoyer à :

Saia-Burgess Controls SA
Rue de la Gare 18
CH-3280 Morat (Suisse)
http://www.saia-burgess.com

Guide des instructions SAIA® PCD

Vos commentaires seront les bienvenus pour améliorer la qualité et le contenu de cette documentation SAIA® PCD. Nous vous remercions par avance de votre collaboration.

Vos commentaires :