

GFK-0467J-F

[Buy GE Fanuc Series 90-30 NOW!](#)

GE Fanuc Manual Series 90-30

Logiciel de programmation pour API 90-30/90-20/
Micro Manuel de reference

1-800-360-6802
sales@pdfsupply.com



GE Fanuc Automation

Automates Programmables Industriels

*Logiciel de programmation
pour API 90-30/90-20/Micro*

Manuel de référence

GFK-0467J

Mai 1997

Utilisation de l'expression

"Attention danger" et des termes

"Attention" et "Remarque" dans ce document

Attention danger

L'expression "Attention danger" est utilisée pour mettre en évidence des risques de blessures dues aux tensions, aux courants, aux températures ou à d'autres grandeurs physiques.

Toutes les situations où un manque d'attention peut être source de blessures physiques ou de dommages pour l'équipement sont repérées par cette expression.

Attention

Le terme "Attention" est associé aux situations où un manque d'attention risque de conduire à des dégâts matériels.

Remarque

Les "Remarques" ont pour but d'attirer votre attention sur des informations particulièrement utiles à la compréhension et à la mise en oeuvre de l'équipement.

Ce document est basé sur des informations disponibles au moment de sa publication. Malgré nos efforts de précision, nous ne pouvons prétendre couvrir tous les détails et toutes les variations matérielles ou logicielles possibles, ni aborder tous les cas de figure de l'installation, du fonctionnement ou de la maintenance. Les caractéristiques décrites dans ce document peuvent être absentes de certains systèmes matériels ou logiciels. GE Fanuc Automation ne s'engage pas à avertir les possesseurs de ce document d'éventuelles modifications ultérieures.

GE Fanuc Automation ne fournit aucune garantie explicite, implicite ou statutaire, et décline toute responsabilité quant à la précision, à l'utilité, et au caractère complet ou suffisant des informations contenues dans ce document. GE Fanuc Automation ne donne aucune garantie de qualité marchande et d'aptitude à une utilisation donnée.

Les marques suivantes sont des marques déposées de GE Fanuc Automation North America, Inc. :

Alarm Master	GENet	PowerMotion	Series One
CIMPLICITY	Genius	ProLoop	Series Six
CIMPLICITY PowerTRAC	Genius PowerTRAC	PROMACRO	Series Three
CIMPLICITY 90-ADS	Helpmate	Series Five	VuMaster
CIMSTAR	Logicmaster	Series 90	Workmaster
Field Control	Modelmaster		

Ce manuel décrit le fonctionnement du système, le traitement des défauts et les instructions de programmation des automates programmables industriels (API) Série 90™-30, Série 90™-20 et Série 90™ Micro Logicmaster 90™. Les API 90-30, 90-20 et 90 Micro sont des membres de la gamme d'automates programmables industriels Série 90™ de GE Fanuc Automation.

Révisions de ce manuel

De nombreuses clarifications, améliorations et ajouts ont été apportés au texte de ce manuel afin de prendre en compte les nouvelles fonctionnalités des unités centrales (UC) Version 8. Les UC Version 8 351 et 352, par exemple, possèdent une nouvelle fonctionnalité qui améliore la protection de la mémoire ; pour plus d'informations à son sujet, voir la section "Protection améliorée de la mémoire des UC versions 8 et ultérieures" à la page 2-15. De nouveaux produits API Micro sont énumérés à la page 2-41. Afin de clarifier et décrire en détail la séquence de mise sous tension, nous avons revu son diagramme ainsi que le texte qui l'accompagne aux pages 2-30 et suivantes.

Ce manuel contient aussi d'autres clarifications, en particulier aux sections "Fenêtre de communication avec la console de programmation" et "Modes de la fenêtre de communication" du chapitre 2, la section "Défaut d'application" du chapitre 3 et la section PID du chapitre 4. Les sections de l'annexe E ont été réorganisées afin de les rendre plus lisibles. D'autres corrections et modifications mineures ont aussi été apportées aux endroits appropriés. Une partie du texte a en outre été rendue moins spécifique à Logicmaster de façon à ce que le manuel de référence s'applique plus particulièrement aux UC des API 90-30, 90-20 et 90 Micro, indépendamment du logiciel de programmation.

Contenu de ce manuel

Chapitre 1. Introduction: présente une description des API 90-30, 90-20 et 90 Micro ainsi que du jeu d'instructions 90-30/20/Micro.

Chapitre 2. Fonctionnement du système : décrit le fonctionnement des API 90-30, 90-20 et 90 Micro. Cela inclut un exposé sur les séquences du cycle de l'API, les séquences de mise sous/hors tension des API, les compteurs et temporisateurs, la protection par mot de passe, les E/S, et le traitement des défauts. Cela inclut également des informations générales pour comprendre le principe de la programmation d'un diagramme en échelle.

Chapitre 3. Description et correction des défauts : fournit les informations permettant de rechercher et de corriger les défauts des API 90-30, 90-20 ou Micro. Il décrit les défauts de la table des défauts automate et les catégories de défauts dans la Table des défauts des E/S.

Chapitre 4. Jeu d'instructions Série 90-30/20/Micro : décrit les instructions de programmation disponibles pour les API 90-30, 90-20 et 90 Micro. Les informations dans ce chapitre sont organisées en sections correspondant aux principaux groupes d'instructions de programmation.

Annexe A. Temps d'exécution des instructions : indique la taille (en octets) et le temps d'exécution (en microsecondes) de chaque instruction de programmation. La taille est le nombre d'octets occupés par une instruction dans un programme d'application par diagramme en échelle.

Annexe B. Interpréter les tables des défauts : décrit comment interpréter le format de structure des messages lorsqu'on lit les tables des défauts en utilisant le logiciel Logicismaster 90-30/20/Micro.

Annexe C. Mnémoniques des instructions : liste les mnémoniques pouvant être tapés afin d'afficher les instructions de programmations lors de la recherche ou de l'édition d'un programme.

Annexe D. Fonction des touches : liste les attributions spéciales des touches, utilisées pour le logiciel Logicismaster 90-30/20/Micro.

Annexe E. Utilisation des nombres à virgule flottante : décrit les éléments à prendre en compte lors des calculs de nombres à virgule flottante.

Autres manuels à consulter

- GFK-0466 Logicismaster™ 90 Series 90™-30/20/Micro Programming Software User's Manual .*
- GFK-0468 Logicismaster™ 90 Series 90-30 and 90-20 Important Product information.*
- GFK-0356 Série 90™-30 Automate Programmable - Manuel d'installation.*
- GFK-0551 Series 90™-20 Programmable Controller Installation Manual.*
- GFK-0898 Series 90™-30 I/O Module Specifications Manual.*
- GFK-0255 Series 90™ Programmable Coprocessor Module and Support Software User's Manual.*
- GFK-0487 Series 90™ PCM Development Software (PCOP) User's Manual.*
- GFK-0499 CIMPPLICITY™ 90-ADS Alphanumeric Display System User's Manual .*
- GFK-0641 CIMPPLICITY™ 90-ADS Alphanumeric Display System Reference Manual .*
- GFK-0521 Alphanumeric Display Coprocessor Module Data Sheet.*
- GFK-0402 Series 90™-30 and 90-20 PLC Hand-Held Programmer User's Manual .*
- GFK-0840 Power Mate APM for Series 90™-30 PLC —Standard Mode User's Manual.*
- GFK-0781 Power Mate APM for Series 90™-30 PLC —Follower Mode User's Manual.*
- GFK-0293 Manuel utilisateur du Compteur Rapide HSC pour API 90-30.*
- GFK-0412 Series 90™-30 Genius Communications Module User's Manual.*
- GFK-0272 Genius Communications Module Data Sheet.*
- GFK-1034 Series 90™-30 Genius™ Bus Controller User's Manual.*
- GFK-1038 Series 90™-70 FIP Bus Controller User's Manual.*
- GFK-1037 Series 90™-30 FIP Remote I/O Scanner User's Manual.*

GFK-0825 Field Control™ Distributed I/O and Control System Genius™ Bus Interface Unit User's Manual.

GFK-1065 Manuel utilisateur du module de communication CCM pour API 90-30 .

GFK-0582 Series 90™ PLC Serial Communications User's Manual.

Vos remarques et suggestions sont les bienvenues

GE Fanuc Automation s'efforce d'éditer des documentations techniques de qualité. Après avoir utilisé ce manuel, merci de consacrer quelques instants à la page suivante, "Page de remarques", pour la compléter et nous la renvoyer.

David D. Bruton

Sr. Technical Writer

Chapitre 1.	Introduction.....	1-1
Chapitre 2.	Fonctionnement du système.....	2-1
	Section 1 : Description d'un cycle API	2-2
	Cycle de programme normal.....	2-2
	Calcul du temps de cycle.....	2-7
	Gestion interne	2-8
	Acquisition des entrées	2-8
	Exécution du programme d'application.....	2-8
	Restitution des sorties	2-8
	Calcul de CHECKSUM du programme utilisateur.....	2-9
	Fenêtre de communication avec la console de programmation.....	2-9
	Fenêtre de communication système (modèles 331 et supérieurs).....	2-10
	Communication du PCM avec l'API (modèles 331 et supérieurs).....	2-12
	Variantes du cycle de programme	2-13
	Mode Temps de cycle constant.....	2-13
	PLC	2-13
	Modes de la fenêtre de communication	2-14
	Interrupteur à clé des UC 351/352 : Changement de mode et protection de la mémoire flash	2-15
	Utilisation de l'interrupteur à clé des versions 7 et ultérieures	2-15
	Effacement de la table des défauts à l'aide de l'interrupteur à clé	2-16
	Protection améliorée de la mémoire des UC versions 8 et ultérieures.....	2-16
	Section 2 : Organisation du programme et données/références utilisateur..	2-18
	Blocs de sous-programme (API Série 90-30 seulement).....	2-19
	Exemples d'utilisation des blocs de sous-programme.....	2-19
	Appel des blocs de sous-programme	2-20
	Sous-programmes périodiques.....	2-21
	Références utilisateur	2-21
	Transitions et forçages	2-23
	Rémanence des données.....	2-24
	Types de données	2-25
	Références d'état système	2-26
	Structure des blocs fonctionnels.....	2-28
	Format des instructions à diagramme en échelle.....	2-28
	Format des blocs fonctionnels de programme.....	2-29
	Paramètres des blocs fonctionnels.....	2-30
	Entrée et sortie du flux validant d'une instruction (validation d'un bloc fonctionnel).....	2-31

Section 3 : Séquences de mise sous/hors tension	2-32
Mise sous tension	2-32
Mise hors tension	2-34
Section 4 : Compteurs et temporisateurs.....	2-36
Compteur de temps de fonctionnement	2-36
Horloge interne	2-36
Temporisateur chien de garde	2-37
Temporisateur de cycle constant.....	2-37
Contacts d'impulsions d'horloge.....	2-37
Section 5 : Sécurité du système.....	2-38
Mots de passe	2-38
Demandes de changement de niveau d'autorisation	2-39
Verrouillage/déverrouillage des sous-programmes.....	2-39
Verrouillage permanent d'un sous-programme	2-40
Section 6 : Série 90-30, Série 90-20 et Série Micro	2-41
Modules d'E/S Modèle 30.....	2-42
Formats des données d'E/S	2-44
Conditions par défaut des modules de sortie Modèle 30.....	2-44
Données de diagnostic	2-45
Données globales	2-45
Chapitre 3. Description et correction des défauts.....	3-1
Section 1 : Gestion des défauts	3-2
Processeur d'alarmes	3-2
Catégories de défauts	3-2
Réponse du système aux défauts.....	3-3
Table des défauts	3-3
Intervention défaut	3-4
Références de défauts	3-4
Définitions des références de défauts.....	3-5
Effets secondaires des défauts.....	3-5
Affichage de la Table des défauts de l'API	3-5
Affichage de la Table des défauts d'E/S	3-6
Accéder aux informations additionnelles concernant les défauts.....	3-6

Section 2 : Description du défaut dans la Table des défauts automate	3-7
Intervention défaut	3-8
Coprocesseur perdu ou manquant	3-8
Coprocesseur réinitialisé, additionnel ou non configuré	3-8
Incohérence dans la configuration système	3-9
Défaut logiciel du coprocesseur	3-10
Défaut de CHECKSUM des blocs de programme	3-10
Signal "seuil pile bas"	3-11
Dépassement du temps de cycle constant	3-11
Défaut d'application	3-12
Pas de programme utilisateur présent	3-12
Programme utilisateur altéré à la mise sous tension	3-13
Echec de l'accès par mot de passe	3-13
Défaut du système d'exploitation de l'UC de l'API	3-14
Défaut de communication pendant le stockage	3-16
Section 3 : Descriptions de la Table des défauts d'E/S.....	3-17
Module d'E/S perdu.....	3-17
Module d'E/S additionnel.....	3-18
Chapitre 4. Jeu d'instructions Serie 90-30/20/Micro.....	4-1
Section 1 : Instructions par diagramme en échelle	4-2
Utilisation des Contacts.....	4-2
Utilisation des bobines	4-3
Contact normalement ouvert — —.....	4-4
Contact normalement fermé — / —.....	4-4
Exemple :	4-4
Bobine —()—.....	4-4
Exemple :	4-4
Bobine inversée —(/)—	4-5
Exemple :	4-5
Bobine rémanente —(M)—.....	4-5
Bobine rémanente inversée —(/M)—.....	4-5
Bobine de transition Positive —(↑)—.....	4-5
Bobine de transition négative —(↓)—	4-6
Exemple :	4-6
Bobine de mise à "1" —(S)	4-6
Bobine de remise à "0" —(R)—	4-6
Exemple :	4-7
Bobine rémanente de mise à "1" —(SM)—.....	4-7

Bobine rémanente de remise à "0" —(RM)—	4-8
Liens	4-8
Exemple :	4-8
Bobines "suite" (———<+>) et Contacts (<+>———) "suite"	4-9
Section 2 : Instructions de temporisation et de comptage	4-10
Données de bloc fonctionnel nécessaires aux temporisateurs et compteurs	4-10
ONDTR	4-12
Paramètres :	4-13
Types de mémoires valides :	4-13
Exemple :	4-14
TMR	4-15
Paramètres :	4-16
Types de mémoires valides :	4-16
Exemple :	4-17
OFDT	4-18
Paramètres :	4-19
Types de mémoires valides :	4-20
Exemple :	4-20
UPCTR	4-21
Paramètres :	4-21
Types de mémoires valides :	4-22
Exemple :	4-22
DNCTR	4-23
Paramètres :	4-23
Types de mémoires valides :	4-24
Exemple :	4-24
Exemple :	4-25
Section 3 : Instructions arithmétiques	4-27
Arithmétiques (ADD, SUB, MUL, DIV)	4-28
Paramètres:	4-29
Types de mémoires valides :	4-29
Exemple :	4-30
Instructions arithmétiques et types de données	4-30
MOD (INT, DINT)	4-32
Paramètres :	4-32
Types de mémoires valides :	4-33
Exemple :	4-33
SQRT (INT, DINT, REAL)	4-34
Paramètres :	4-34
Types de mémoires valides :	4-35
Exemple :	4-35
Instructions de trigonométrie (SIN, COS, TAN, ASIN, ACOS, ATAN)	4-36
Paramètres :	4-37
Types de mémoires valides :	4-37
Exemple :	4-37

Instructions de logarithme/exposant (LOG, LN, EXP, EXPT)	4-38
Paramètres :	4-38
Types de mémoires valides :	4-39
Exemple :	4-39
Conversion radian (RAD, DEG).....	4-40
Paramètres :	4-40
Types de mémoires valides :	4-40
Exemple :	4-41
Section 4 : Instructions de comparaison.....	4-42
Paramètres :	4-43
Types de mémoires valides	4-43
Exemple :	4-44
RANGE (INT, DINT, WORD).....	4-45
Paramètres :	4-46
Types de mémoires valides :	4-46
Exemple 1:.....	4-46
Exemple 2 :	4-47
Section 5 : Opérations Logiques	4-48
AND et OR (WORD).....	4-50
Paramètres :	4-50
Types de mémoires valides :	4-51
Exemple :	4-51
XOR (WORD)	4-52
Paramètres :	4-52
Types de mémoires valides :	4-53
Exemple :	4-53
NOT (WORD)	4-54
Paramètres :	4-54
Types de mémoires valides :	4-55
Exemple :	4-55
SHL et SHR (WORD).....	4-56
Paramètres :	4-57
Types de mémoires valides :	4-57
Exemple :	4-58
ROL et ROR (WORD).....	4-59
Paramètres :	4-59
Types de mémoires valides :	4-60
Exemple :	4-60
BTST (WORD).....	4-61
Paramètres :	4-61
Types de mémoires valides :	4-62
Exemple :	4-62
BSET et BCLR (WORD).....	4-63
Paramètres :	4-63
Types de mémoires valides :	4-64
Exemple :	4-64

BPOS (WORD).....	4-65
Paramètres :.....	4-65
Types de mémoires valides :.....	4-66
Exemple :	4-66
MSKCMP (WORD, DWORD)	4-67
Si tous les bits d'I1 et d'I2 sont identiques.....	4-67
En présence d'une incohérence.....	4-67
Paramètres :.....	4-68
Types de mémoires valides :.....	4-68
Exemple :	4-69
Section 6 : Instructions de transfert de données	4-71
MOVE (BIT, INT, WORD, REAL).....	4-72
Paramètres :.....	4-73
Types de mémoires valides :.....	4-73
Exemple 1:.....	4-74
Exemple 2:.....	4-74
BLKMOV (INT, WORD, REAL).....	4-75
Paramètres	4-75
Types de mémoires valides :.....	4-76
Exemple :	4-76
BLKCLR (WORD)	4-77
Paramètres :.....	4-77
Types de mémoires valides :.....	4-78
Exemple :	4-78
SHFR (BIT, WORD).....	4-79
Paramètres :.....	4-80
Types de mémoires valides :.....	4-80
Exemple 1:.....	4-81
Exemple 2:.....	4-81
BITSEQ (BIT)	4-82
Occupation mémoire d'un séquenceur binaire.....	4-83
Paramètres :	4-84
Types de mémoires valides :	4-84
Exemple :	4-85
COMMREQ.....	4-86
Bloc de commande.....	4-86
Paramètres :.....	4-87
Types de mémoires valides :.....	4-88
Exemple :	4-88
Section 7 : Instructions sur tableau	4-89
ARRAY_MOVE (INT, DINT, BIT, BYTE, WORD)	4-90
Paramètres	4-91
Types de mémoires valides	4-91
Exemple 1 :.....	4-92
Exemple 2 :.....	4-92
Exemple 3 :.....	4-93

SRCH_EQ et SRCH_NE (INT, DINT, BYTE, WORD) SRCH_GT et SRCH_LT SRCH_GE et SRCH_LE	4-94
Paramètres :	4-95
Types de mémoires valides :	4-95
Exemple 1 :	4-96
Exemple 2 :	4-96
Section 8 : Instructions de conversion	4-97
—>BCD-4 (INT)	4-98
Paramètres :	4-98
Types de mémoires valides :	4-98
Exemple :	4-99
—>INT (BCD-4, REAL)	4-100
Paramètres :	4-100
Types de mémoires valides :	4-100
Exemple :	4-101
—>DINT (REAL)	4-102
Paramètres :	4-102
Types de mémoires valides :	4-102
Exemple :	4-103
—>REAL (INT, DINT, BCD-4, WORD)	4-104
Paramètres :	4-104
Types de mémoires valides :	4-104
Exemple :	4-105
—>WORD (REAL)	4-106
Paramètres :	4-106
Types de mémoires valides :	4-106
Exemple :	4-107
TRUN (INT, DINT)	4-108
Paramètres :	4-108
Types de mémoires valides :	4-109
Exemple :	4-109
Section 9 : Instructions de commande	4-110
CALL	4-111
Exemple :	4-111
DOIO	4-112
Paramètres :	4-113
Types de mémoires valides :	4-113
Exemple d'entrée 1 :	4-114
Exemple d'entrée 2 :	4-114
Exemple de sortie 1 :	4-115
Exemple de sortie 2 :	4-115
Instruction DO I/O évoluée pour les UC modèles 331 et 341	4-116
END	4-118
Exemple :	4-118

MCR	4-119
Différences entre les instructions MCR et JUMP	4-120
Exemple :	4-121
ENDMCR	4-122
Exemple :	4-122
JUMP	4-123
Exemple :	4-124
LABEL	4-125
Exemple :	4-125
COMMENT	4-126
SVCREQ	4-127
Paramètres :	4-128
Types de mémoires valides :	4-128
Exemple :	4-128
SVCREQ #6: Modifier/lire l'état de la tâche de CHECKSUM et le nombre de mots qu'elle doit vérifier	4-129
Pour lire la valeur courante du nombre de bits :	4-129
Pour écrire un nouveau nombre de mots :	4-129
Exemple :	4-130
SVCREQ #7: Modifier/lire l'horloge interne	4-131
Exemple :	4-132
Contenu du bloc de paramètres	4-133
Pour modifier/lire la date et l'heure en utilisant le format DCB :	4-133
Pour modifier/lire la date et l'heure en utilisant le format ASCII compressé avec (:) intercalé	4-134
SVCREQ #13: Mettre hors service (arrêter) l'API	4-135
Exemple :	4-135
SVCREQ #14: Vider les tables des défauts	4-136
Exemple :	4-136
SVCREQ #15: Lire la dernière entrée enregistrée dans la table des défauts	4-137
Exemple 1:	4-138
Exemple 2 :	4-139
SVCREQ #16: Lire le compteur de temps de fonctionnement	4-141
Exemple :	4-141
SVCREQ #18: Lire l'état des forçages des E/S	4-142
Exemple :	4-142
SVCREQ #23: Lire Checksum maître	4-143
Exemple :	4-143
SVCREQ #26/30: Interroger les E/S	4-144
Exemple :	4-144
SVCREQ #29 : Lire le compteur de mise hors tension	4-145
Exemple :	4-145
PID	4-146
Paramètres :	4-147
Types de mémoires valides :	4-147
Bloc de paramètres PID :	4-148
Fonctionnement de l'instruction	4-150
Paramètres internes de RefArray	4-153
Sélection de l'algorithme PID (PIDISA ou PIDIND) et gains	4-154

	Limites d'amplitude et de coefficient de CV	4-155
	Période d'échantillonnage et planification du bloc PID	4-156
	Détermination des caractéristiques du procédé	4-156
	Définition des paramètres utilisateur et réglage des gains en boucle	4-158
	Définition des gains en boucle - approche de réglage de Ziegler et Nichols	4-158
	Exemple d'appel PID.....	4-159
Annexe A.	Temps d'exécution des instructions	A-1
Annexe B.	Interprétation des tables des défauts	B-1
	Table des défauts automate	B-1
	Table des défauts d'E/S	B-8
Annexe C.	Mnémoniques des instructions.....	C-1
Annexe D.	Fonction des touches	D-1
Annexe E.	Utilisation des nombres à virgule flottante.....	E-1
	Nombres à virgule flottante	E-1
	Format interne des nombres à virgule flottante.....	E-3
	Valeurs des nombres à virgule flottante.....	E-4
	Saisie et affichage des nombres à virgule flottante	E-5
	Erreurs dans les opérations et les nombres à virgule flottante	E-6

Figure 2-1. Cycle d'API	2-3
Figure 2-2. Organigramme de la fenêtre de communication de la console de programmation	2-10
Figure 2-3. Organigramme de communication système	2-11
Figure 2-4. Communication du PCM avec l'API.....	2-12
Figure 2-5. Séquence de mise sous tension	2-33
Figure 2-6. Chronogramme des contacts d'impulsions d'horloge	2-37
Figure 2-7. Structure d'E/S de l'API Série 90-30	2-41
Algorithme à termes indépendants (PIDIND).....	4-155

Tableau 2-1. Contribution au temps de cycle.....	2-4
Tableau 2-4. Références des registres.....	2-21
Tableau 2-5. Références logiques (Bits)	2-22
Tableau 2-5. Références logiques - Suite.....	2-23
Tableau 2-6. Types de données	2-25
Tableau 2-7. Référence d'état système	2-26
Tableau 2-7. Références d'état système - Suite	2-27
Tableau 2-7. Références d'état système - Suite	2-28
Tableau 2-8. Modules d'E/S Modèle 30.....	2-42
Tableau 2-9. Modules d'E/S Modèle 30 - Suite	2-43
Tableau 2-9. Modules d'E/S Modèle 30 - Suite	2-44
Tableau 3-1. Sommaire des défauts.....	3-3
Tableau 3-2. Interventions défaut.....	3-4
Tableau 4-1. Types de contacts.....	4-2
Tableau 4-2. Types de bobines.....	4-3
Tableau 4-3. Instructions de demande de service	4-127
Tableau 4-4. Présentation des paramètres PID.....	4-148
Tableau 4-4. Présentation des paramètres PID (suite)	4-149
Tableau 4-5. Détails des paramètres PID	4-151
Tableau 4-5. Détails des paramètres PID - suite	4-152
Tableau 4-5. Détails des paramètres PID - suite	4-153
Tableau A-1. Temps d'exécution des instructions	A-2
Tableau A-1. Temps d'exécution des instructions - suite	A-3
Tableau A-1. Temps d'exécution des instructions - suite	A-4
Tableau A-1. Temps d'exécution des instructions - suite	A-5
Tableau A-1. Temps d'exécution des instructions - suite	A-6
Tableau A-1. Temps d'exécution des instructions - suite	A-7
Tableau A-1. Temps d'exécution des instructions - suite	A-8
Tableau A-1. Temps d'exécution des instructions - suite	A-9
Tableau A-2. Taille des instructions pour les UC 351 et 352	A-10
Tableau B-1. Groupe de défauts automate.....	B-4
Tableau B-2. Interventions défaut automate	B-5
Tableau B-3. Codes d'erreur des défauts logiciels de l'UC de l'API.....	B-5
Tableau B-4. Codes d'erreur des défauts de l'UC de l'API	B-6
Tableau B-5. Informations défaut automate - Détection d'un code opération booléen interdit	B-7

Tableau B-6. Horodatage de défaut automate	B-7
Tableau B-7. Octet indicateur de format de la Table des défauts d'E/S.....	B-9
Tableau B-8. Adresse de référence des E/S.....	B-9
Tableau B-9. Type de mémoire des adresses de références	B-9
Tableau B-10. Groupes de défauts d'E/S	B-10
Tableau B-11. Intervention défaut d'E/S.....	B-11
Tableau B-12. Informations spécifiques de défaut d'E/S	B-11
Tableau B-13. Horodatage de défaut d'E/S	B-12

Les automates Série 90-30, Série 90-20 et Série Micro sont des membres de la gamme d'automates programmables industriels (API) Série 90™ de GE Fanuc. Ces API sont faciles à installer et à configurer, offrent des instructions de programmation avancées, et sont compatibles avec l'API 90-70.

L'API 90-20 offre une plate-forme d'un bon rapport coût-performance pour les applications à faibles besoins d'E/S. Les principaux objectifs de l'API 90-20 sont les suivants :

- Offrir un petit automate facile à utiliser, à installer, à mettre à niveau et à entretenir.
- Offrir une compatibilité de gamme et un bon rapport coût-performance.
- Faciliter l'intégration système par des protocoles et un matériel de communication standard.

L'API 90 Micro fournit aussi une plate-forme rentable pour les applications de comptage d'E/S. Outre les principaux objectifs identiques à l'API 90-20, l'API Micro offre aussi les avantages suivants :

- Incorporation de l'UC, de l'alimentation, des entrées et des sorties à un seul petit périphérique.
- Incorporation d'un compteur rapide à la plupart des modèles.
- Configuration aisée en raison de l'incorporation de l'UC, de l'alimentation, des entrées et des sorties à un seul petit périphérique.

Les API 90-30 (à l'exception des modèles 351/352) et 90-20 utilisent une architecture logicielle qui gère la mémoire et la priorité d'exécution dans le microprocesseur 80188. Les modèles 351 et 352 de l'API 90 utilisent plutôt un microprocesseur 80386 EX tandis que l'API 90 Micro, elle, utilise un microprocesseur H8. Cette opération supporte les tâches d'exécution du programme et de gestion interne de base, telles que les sous-programmes de diagnostic, la scrutation des E/S et le traitement des alarmes. Le système d'exploitation contient également des sous-programmes assurant l'interface avec la console de programmation. Ces sous-programmes permettent de charger et de transférer des programmes d'application, de récupérer des informations d'état, et de commander l'API.

Dans l'API 90-30, le programme d'application (programme utilisateur), qui gère le processus final asservi par l'API, est commandé par un Séquenceur d'instructions dédié (coprocesseur ISCP). L'ISCP est un composant matériel dans les modèles 313 et supérieurs et un composant logiciel dans les modèles 311 et Micro. Le microprocesseur 80188 et l'ISCP fonctionnent simultanément, permettant ainsi au microprocesseur de prendre en charge les communications pendant que l'ISCP

exécute l'essentiel du programme d'application ; toutefois, le microprocesseur doit exécuter les blocs fonctionnels non-booléens.

Sur les API 90-30 et 90-20, il y a défaut lorsque certaines pannes ou conditions sont de nature à affecter le fonctionnement et les performances du système. Ces conditions peuvent affecter l'aptitude de l'API à contrôler une machine ou un procédé. D'autres conditions jouent simplement le rôle d'alarme, telles qu'un signal de "seuil pile bas", indiquant que la pile de sauvegarde de la mémoire doit être remplacée. Cette condition ou panne est appelée défaut.

Les défauts sont gérés par un processeur d'alarmes enregistrant les défauts dans la Table des défauts automate ou dans la Table des défauts d'E/S. (Les UC modèles 331 et 341 enregistrent également la date et l'heure des défauts.) Ces tables peuvent être affichées sous forme d'écrans dans le logiciel Logicmaster 90-30/20 en utilisant les instructions de commande et d'état.

Informations de référence complémentaires : voir les annexes à la fin de ce manuel.

L'annexe A indique la taille (en octets) et le temps d'exécution (en microsecondes) de chaque instruction de programmation.

L'annexe B décrit comment interpréter le format de structure des messages pour lire la Table des défauts d'API et la Table des défauts d'E/S.

L'annexe C donne une liste des mnémoniques des instructions utilisées avec le logiciel Logicmaster 90-30/20/Micro.

L'annexe D donne une liste des attributions spéciales des touches utilisées avec le logiciel Logicmaster 90-30/20/Micro.

L'annexe E fournit des instructions et des remarques relatives à l'utilisation des calculs de nombres à virgule flottante (disponibles uniquement sur les UC 90-30 351 et 352).

Ce chapitre décrit le fonctionnement des API Série 90-30, 90-20 et Micro. Le fonctionnement inclut :

- La description d'un cycle API (voir § 1).
- L'organisation du programme et données/références utilisateur (voir § 2).
- Les séquences de mise sous/hors tension (voir § 3).
- Les horloges et temporisateurs (voir § 4).
- La protection par mot de passe (voir § 5).
- Les modules d'E/S modèle 30 (voir § 6).

Section 1 : Description d'un cycle API

Le programme utilisateur des API Série 90-30, 90-20 et Micro est exécuté de manière répétitive jusqu'à ce qu'il rencontre une instruction provenant de la console de programmation ou d'un autre équipement. La séquence d'opérations nécessaire pour exécuter une fois un programme est appelée "cycle". En plus de l'exécution du programme utilisateur, le cycle comprend l'acquisition de données des modules d'entrée, la transmission de données aux modules de sortie, l'exécution de tâches interne, les communications avec la console de programmation et d'autres échanges de données.

Les API Série 90-30, 90-20 et Micro fonctionnent normalement en mode **CYCLE DE PROGRAMME NORMAL**. D'autres modes d'exploitation incluent le mode **STOP AVEC E/S INVALIDÉES**, le mode **STOP AVEC E/S VALIDÉES**, et le mode **CYCLE CONSTANT**, décrits dans ce chapitre. Chacun de ces modes est commandé par des événements externes et la configuration de l'application. Au début de chaque cycle, l'API décide de son mode d'exploitation.

Cycle de programme normal

Le mode **CYCLE DE PROGRAMME NORMAL** fonctionne généralement dans toutes les conditions. Le processeur exécute un programme d'application, rafraîchit les E/S et exécute les échanges de données et d'autres tâches. Cela se produit selon un cycle répétitif appelé "cycle du processeur". La séquence d'exécution du Cycle de programme normal comporte sept parties :

1. La gestion interne de début du cycle.
2. L'acquisition des entrées (lecture des données).
3. La résolution des équations du programme d'application.
4. La restitution des sorties (rafraîchissement des sorties).
5. Communication avec la console de programmation.
6. Communication avec les modules intelligents et les coprocesseurs.
7. Les diagnostics.

Toutes ces étapes, à l'exception du service demandé par la console de programmation, s'exécutent à chaque cycle. Le service demandé par la console de programmation se produit uniquement en cas de détection d'un défaut de carte ou si la console de programmation émet une demande de service. La séquence de Cycle de programme normal est présentée dans la figure suivante.

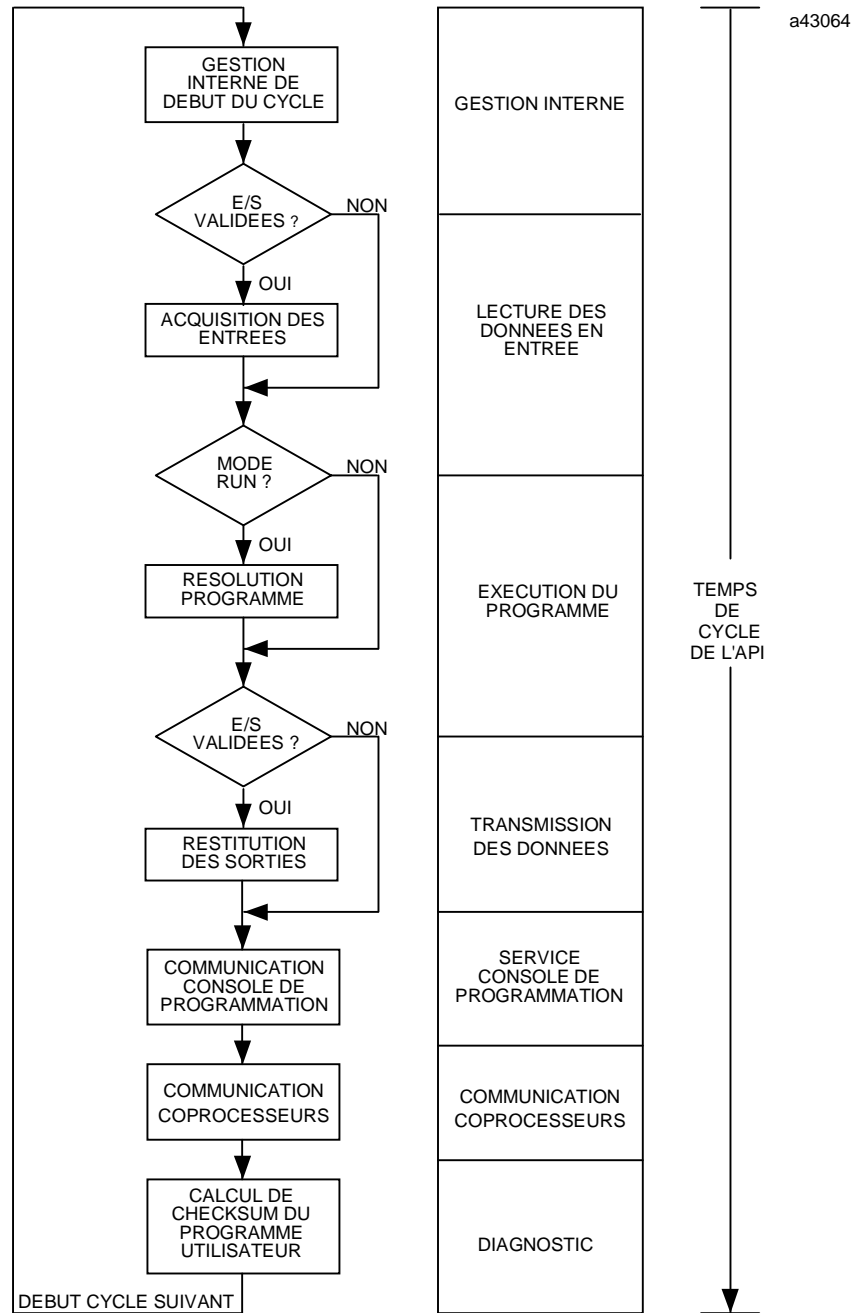


Figure 2-1. Cycle d'API

Comme indiqué dans la séquence de cycle de l'API, le cycle inclut plusieurs étapes. Ces étapes contribuent au temps total de cycle comme indiqué dans le tableau suivant.

Tableau 2-1. Contribution au temps de cycle

Éléments de Cycle	Description	Contribution au temps (ms) ⁴						
		Micro	211	311/313	331	340/341	351/352	
Gestion interne	<ul style="list-style-type: none"> • Calcule le temps de cycle. • Détermine le démarrage du cycle suivant. • Détermine le mode du cycle suivant. • Rafraîchit les tables de référence des défauts. • Remet à zéro le compteur chien de garde. 	0.368	0.898	0.714	0.705	0.424	0.279	
Lecture des données (entrées)	Réception des données venant des modules d'entrée et optionnels.	(5)	Voir Tableau 2-2 pour contribution au temps de cycle.					
Exécution du programme	Résolution du programme utilisateur.	Le temps d'exécution dépend de la longueur du programme et du type d'instructions qu'il utilise. Les temps d'exécution des instructions sont indiqués dans l'annexe A.						
Transmission des données (sortie)	Transmission des données aux modules de sortie et aux modules optionnels.	0.1656	Voir Tableau 2-2 pour contribution au temps de cycle					
Service des consoles externes	Les demandes de service émises par les consoles de programmation et les modules intelligents sont traitées. ¹	HHP	1.93	6.526	4.426	4.524	2.476	0.334
		LM-90	0.380	3.536	2.383	2.454	1.248	0.517
		PCM ²	N/A	N/A	N/A	3.337	1.943	0.482
Reconfiguration	Les emplacements contenant des modules défectueux ainsi que les emplacements vides sont surveillés.	N/A ⁶	N/A	0.458	0.639	0.463	0.319	
Diagnostics	Vérifient l'intégrité du programme utilisateur (la contribution au temps correspond au temps requis pour le checksum des mots de chaque cycle). ³	N/A ⁷	0.083	0.050	0.048	0.031	0.010	

1. La contribution au temps de cycle du service des consoles externes varie en fonction du mode de la fenêtre de communication dans laquelle le service est traité. En mode **LIMITED**, 6 ms maximum sont consacrées à cette fenêtre. En mode **RUN-TO-COMPLETION**, 50 ms maximum peuvent être consacrées à cette fenêtre, selon le nombre de demandes présentées simultanément.
2. Ces mesures ont été prises quand PCM était présent physiquement, mais pas configuré et lorsqu'aucune tâche d'application n'était en cours d'exécution sur PCM.
3. Le nombre de mots du calcul de la checksum au cours de chaque balayage peut être modifié à l'aide du bloc fonctionnel SVCREQ.
4. Ces mesures ont été prises avec un programme vide et la configuration par défaut. Les API 90-30 figuraient dans un bac à 10 emplacements vide auquel aucun bac d'extension n'était connecté.
5. La durée de lecture des données de l'API Micro peut être déterminée de la manière suivante : $0,365 \text{ ms. (scrutation fixe) + } 0,036 \text{ ms. (temps de filtre) } \times \text{(temps de cycle total)}/0,5\text{ms.}$
6. Comme l'API Micro possède un jeu d'E/S statique, la reconfiguration n'est pas nécessaire.
7. Comme le programme utilisateur de l'API Micro est en mémoire flash, son intégrité n'est pas vérifiée.

Tableau 2-2. Contributions au temps de scrutation des E/S pour les modules 90-30 311–341 (en millisecondes)

Type de module		Modèle d'UC						
		311/313	331			340/341		
			Bac principal	Bac d'extension	Bac distant	Bac principal	Bac d'extension	Bac distant
Entrée discrète, 8 points		0,076	0,054	0,095	0,255	0,048	0,089	0,249
Entrée discrète, 16 points		0,075	0,055	0,097	0,257	0,048	0,091	0,250
Entrée discrète, 32 points		0,094	0,094	0,126	0,335	0,073	0,115	0,321
Sortie discrète, 8 points		0,084	0,059	0,097	0,252	0,053	0,090	0,246
Sortie discrète, 16 points		0,083	0,061	0,097	0,253	0,054	0,090	0,248
Sortie discrète, 32 points		0,109	0,075	0,129	0,333	0,079	0,114	0,320
Entre/sortie combinée, 8 points		0,165	0,141	0,218	0,529	0,098	0,176	0,489
Entrée analogique, 4 canaux		0,151	0,132	0,183	0,490	0,117	0,160	0,462
Sortie analogique, 2 canaux		0,161	0,138	0,182	0,428	0,099	0,148	0,392
Compteur rapide		2,070	2,190	2,868	5,587	1,580	2,175	4,897
APM Power Mate (1 axe)		2,330	2,460	3,175	6,647	1,750	2,506	5,899
GCM	pas de périphériques	0,041	0,054	0,063	0,128	0,038	0,048	0,085
	8 périphériques à 64 points	11,420	11,570	13,247	21,288	9,536	10,648	19,485
GCM+	pas de périphérique	0,887	0,967	1,164	1,920	0,666	0,901	1,626
	32 périphériques à 64 points	4,120	6,250	8,529	21,352	5,043	7,146	20,052
PCM 311	non configuré ou sans tâche d'application	N/A	3,350	N/A	N/A	1,684	N/A	N/A
	lecture 128 %R aussi vite que possible	N/A	4,900	N/A	N/A	2,052	N/A	N/A
ADC 311		N/A	3,340	N/A	N/A	1,678	N/A	N/A
Entrée analogique, 16 canaux (courant ou tension)		1,370	1,450	1,937	4,186	1,092	1,570	3,796
Lien E/S maître	pas de périphérique	1,910	2,030	1,169	1,925	0,678	0,904	1,628
	16 périphériques à 64 points	6,020	6,170	8,399	21,291	4,992	6,985	20,010
Lien E/S esclave	32 points	0,206	0,222	0,289	0,689	0,146	0,226	0,636
	64 points	0,331	0,350	0,409	1,009	0,244	0,321	0,926

Tableau 2-3. Contributions au temps de scrutation des E/S pour les modules 351/352 90-30 (en millisecondes)

Type de module		UC		
		351/352		
		Bac principal	Bac d'extension	Bac distant
Entrée discrète, 8 points		0,030	0,055	0,206
Entrée discrète, 16 points		0,030	0,055	0,206
Entrée discrète, 32 points		0,043	0,073	0,269
Sortie discrète, 8 points		0,030	0,053	0,197
Sortie discrète, 16 points		0,030	0,053	0,197
Sortie discrète, 32 points		0,042	0,070	0,259
Entrée/sortie discrète combinée		0,060	0,112	0,405
Entrée analogique, 4 canaux		0,075	0,105	0,396
Sortie analogique, 2 canaux		0,058	0,114	0,402
Entrée analogique, 16 canaux (courant ou tension)		0,978	1,446	3,999
Sortie analogique, 8 canaux		1,274	1,988	4,472
Entrée/sortie analogique combinée		1,220	1,999	4,338
Compteur rapide		1,381	2,106	5,221
APM Power Mate (1 axe)		1,527	2,581	6,388
Processeur d'E/S		1,574	2,402	6,388
Interface Ethernet (pas de connexion)		0,038	0,041	0,053
GCM	pas de périphérique	0,911	1,637	5,020
	8 périphériques à 64 points	8,826	16,932	21,179
GCM+	pas de périphérique	0,567	0,866	1,830
	32 périphériques à 64 points	1,714	2,514	5,783
GBC	pas de périphérique	0,798	1,202	2,540
	32 périphériques à 64 points	18,382	25,377	70,777
PCM 311	non configuré ou sans tâche d'application	0,476	N/A	N/A
	lecture 128 %R aussi vite que possible	0,485	N/A	N/A
ADC (pas de tâche)		0,476	N/A	N/A
Lien E/S maître	pas de périphérique	0,569	0,865	1,932
	16 périphériques à 64 points	4,948	7,003	19,908
Lien E/S esclave	32 points	0,087	0,146	0,553
	64 points	0,154	0,213	0,789

Calcul du temps de cycle

Le tableau 2-1 liste les sept étapes composant le cycle de l'API. Le temps de cycle inclut des temps fixes (gestion interne et diagnostics) et des temps variables. Les temps variables varient en fonction de la configuration des E/S, de la taille du programme utilisateur et du type de console de programmation connectée à l'API.

Exemple de calcul du temps de cycle

Un exemple de calcul pour déterminer le temps de cycle d'un API Série 90-30 modèle 331 est présenté dans le tableau suivant.

Voici la liste des modules et des instructions utilisés pour effectuer ces calculs:

- Modules d'entrée : 5 modules d'entrée à 16 points modèle 30.
- Modules de sortie : 4 modules de sortie à 16 points modèle 30.
- Instructions de programmation : programme de 1200 pas constitués de 700 instructions booléennes (LD, AND, OR, etc.), 300 bobines de sortie (OUT, OUTM, etc.) et 200 instructions arithmétiques (ADD, SUB, etc.).

Élément du cycle	Calcul	Contribution au temps		
		sans cons. de prog.	avec HHP	avec LM90
Gestion interne	0,705 ms	0,705 ms	0,705 ms	0,705 ms
Lecture des données	$0,055 \times 5 = 0,275$ ms	0,275 ms	0,275 ms	0,275 ms
Exécution du programme	$700 \times 0,4 \mu\text{s} + 300 \times 0,5 \mu\text{s} + 200 \times 51,2 \mu\text{s} = 10,7$ ms	10,7 ms	10,7 ms	10,7 ms
Transmission des données	$0,061 \times 4 = 0,244$ ms	0,244 ms	0,244 ms	0,244 ms
Communication console de programmation	$0,4$ ms + temps cons. prog. + $0,6$ ms	0 ms	4,524 ms	2,454 ms
Communication coprocesseurs	Aucun dans cet exemple	0 ms	0 ms	0 ms
Reconfiguration	0,639 ms	0,639 ms	0,639 ms	0,638 ms
Diagnostics	0,048 ms	0,048 ms	0,048 ms	0,048 ms
Temps de cycle d'API	Gestion interne + Lecture des données + Exécution du programme + Transmission des données + Communication console de programmation + Communication coprocesseurs + Diagnostics	12,611 ms	17,135 ms	15,065 ms

Gestion interne

La partie "gestion interne" du cycle exécute toutes les tâches nécessaires en vue du début du cycle. Si l'API est en mode **CYCLE CONSTANT**, le cycle est différé jusqu'à ce que le temps de cycle imparti soit écoulé. Si ce temps imparti est déjà écoulé, le contact OV_SWP %SA0002 est mis à "1" et le cycle continue sans délai. Ensuite, le système rafraîchit les valeurs des temporisateurs (centièmes, dixièmes et secondes) en calculant la différence entre le début du cycle précédent et le temps du nouveau cycle. Pour garder la même précision, le début effectif du cycle est enregistré en incréments de 100 microsecondes. Chaque temporisateur comporte un champ "reste" contenant le nombre d'incréments de 100 microsecondes écoulés depuis sa dernière incrémentation.

Acquisition des entrées

L'acquisition des entrées se produit avant la résolution du programme. Au cours de cette partie du cycle, tous les modules d'entrée modèle 30 sont scrutés et leurs données stockées, selon le cas, dans l'emplacement mémoire %I (entrées logiques) ou %AI (entrées analogiques). Toutes les données globales reçues par un module de communication Genius sont stockées dans l'emplacement %G.

Les modules sont scrutés dans l'ordre croissant des adresses de références : le module de communication Genius en premier, puis le module d'entrées logiques, et enfin les modules d'entrées analogiques.

Si l'UC est en mode **STOP** et qu'elle est configurée pour ne pas scruter les E/S en mode **STOP**, l'acquisition des entrées est omise.

Exécution du programme d'application

L'exécution du programme commence toujours par la première instruction dans le programme d'application de l'utilisateur, immédiatement après la fin de l'acquisition des entrées. La résolution du programme détermine un nouvel ensemble de sorties. La résolution logique se termine lorsque l'instruction END est exécutée.

Le programme d'application est exécuté par le séquenceur d'instructions (ISCP) et le microprocesseur 80C188. Dans les UC modèles 313 et supérieurs, l'ISCP exécute les instructions booléennes et le microprocesseur 80C188 ou 80386 EX exécute les instructions de temporisation, de comptage et les blocs fonctionnels. Dans les UC modèle 311 et 90-20, le microprocesseur 80C188 exécute toutes les instructions booléennes, de temporisation, de comptage et les blocs fonctionnels. Sur les API Micro, le processeur H8 exécute toutes les instructions booléennes et les blocs fonctionnels.

Vous trouverez une liste des temps d'exécution pour chaque instruction de programmation dans l'annexe A.

Restitution des sorties

Les sorties sont restituées au cours de la partie "restitution des sorties" du cycle, immédiatement après la résolution logique. Les sorties sont rafraîchies, selon le cas, avec les données %Q (pour les sorties logiques) ou les données %AQ (pour les sorties analogiques). Si le module de

communication Genius (GCM) est configuré pour transmettre des données globales, les données %G sont envoyées au GCM. Les restitutions de sorties des API 90-20 et Micro comprennent uniquement les sorties discrètes.

Au cours de la restitution des sorties, tous les modules de sortie modèle 30 sont mis à jour dans l'ordre ascendant des adresses de références.

Si le processeur est en mode **STOP** et qu'il est configuré pour ne pas scruter les E/S en mode **STOP**, la mise à jour des sorties est omise. La restitution des sorties est exécutée lorsque toutes les données de sortie ont été envoyées aux modules de sortie modèle 30.

Calcul de CHECKSUM du programme utilisateur

La CHECKSUM du programme utilisateur est calculée à la fin de chaque cycle. Comme la CHECKSUM serait trop longue à calculer pour l'ensemble du programme, on peut spécifier, sur l'écran de configuration du processeur, le nombre de mots, entre 0 et 32, sur lesquels elle doit porter.

Si la CHECKSUM calculée ne correspond pas à la CHECKSUM de référence, l'indicateur d'erreur de CHECKSUM est mis à "1". Cela a pour effet d'enregistrer une nouvelle entrée dans la Table des défauts automate et de faire passer l'API en mode **STOP**. Ce défaut n'affecte pas la fenêtre de communication de la console de programmation.

Fenêtre de communication avec la console de programmation

Cette partie du cycle est consacrée à la communication avec la console de programmation. Si une console de programmation est connectée l'UC exécute le cycle de communication avec la console de programmation. Cette partie du cycle n'est pas exécutée s'il n'existe aucune console de programmation connectée ou de carte à configurer dans le système. Une seule carte est configurée à chaque cycle.

Une aide est fournie pour la miniconsole de programmation (HHP) et d'autres consoles de programmation pouvant se connecter au port série et utiliser le Series Ninety Protocol (SNP). Une aide est également fournie pour la communication entre la console de programmation et les coprocesseurs.

En mode Limited par défaut, l'UC exécute, à chaque cycle, une opération pour la console de programmation, c'est-à-dire qu'elle prend en charge une demande de service ou répond à l'appui d'une touche. Si le traitement de la demande du programmeur exige plus de 6 ms (ou 8 ms selon l'UC, voir Remarque), le traitement est réparti sur plusieurs cycles, si bien qu'aucun cycle ne dure plus de 6 ms (ou 8 ms selon l'UC, voir Remarque).

Remarque

La durée de la fenêtre de communication est limitée à 6 millisecondes pour les UC 340 et supérieures et à 8 millisecondes pour les modèles 311, 313, 323 et 331. En mode Run, toutefois, cette durée peut être portée jusqu'à 12 millisecondes (voir page 2-14).

La figure suivante présente un organigramme de la partie "communication de la console de programmation" du cycle.

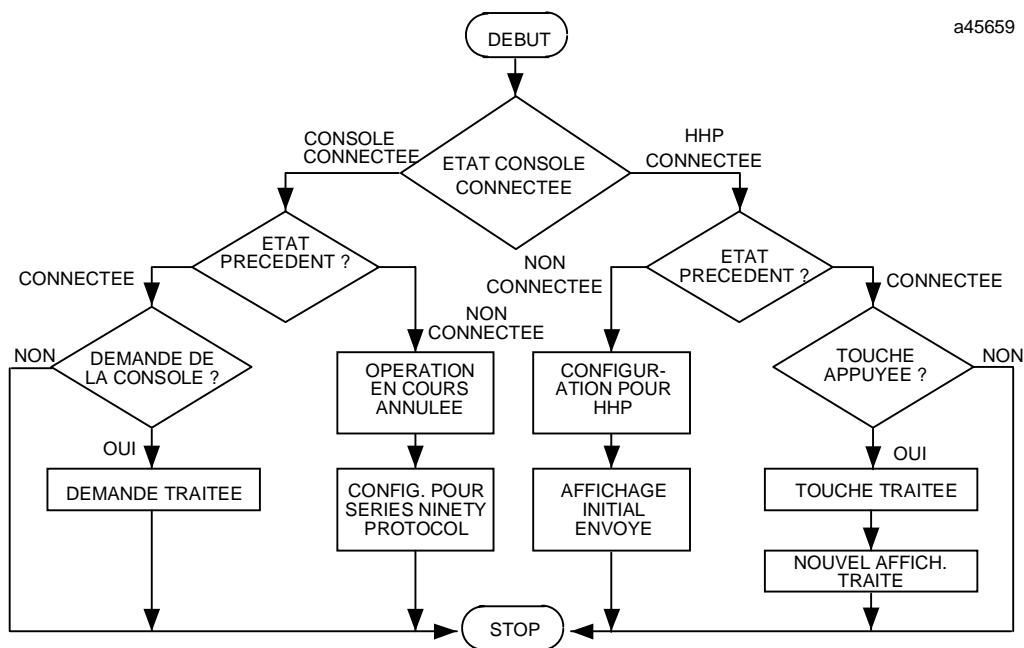


Figure 2-2. Organigramme de la fenêtre de communication de la console de programmation

Fenêtre de communication système (modèles 331 et supérieurs)

Il s'agit de la partie du cycle où les demandes de communication provenant des modules coprocesseurs, tel que le module coprocesseur programmable (PCM), sont traitées (voir organigramme). Les demandes sont prises en charge sur une base "premier arrivé premier servi". Toutefois, comme les coprocesseurs sont interrogés à tour de rôle, aucun d'entre eux n'a priorité sur un autre.

En mode **Run-to-Completion** par défaut, la durée de la fenêtre de communication système est limitée à 50 millisecondes. Si le traitement d'une demande émise par un coprocesseur exige plus de 50 ms, la demande est répartie sur plusieurs cycles, si bien qu'aucun cycle ne dure plus de 50 ms.

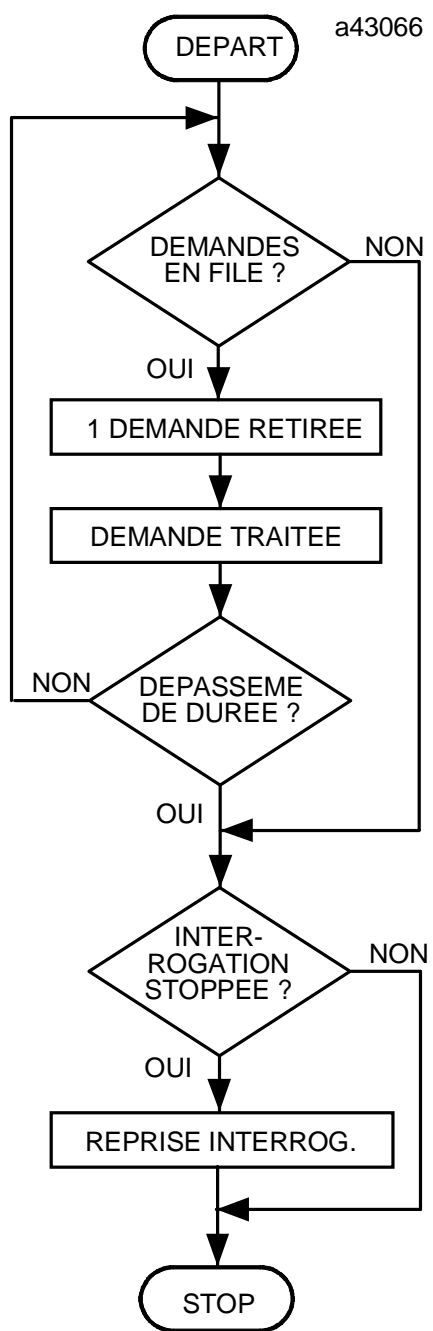


Figure 2-3. Organigramme de communication système

Communication du PCM avec l'API (modèles 331 et supérieurs)

Il n'existe aucun moyen, pour les coprocesseurs tels que le PCM, d'interrompre l'UC lorsqu'ils font une demande de service. L'UC doit interroger chaque coprocesseur afin de savoir s'il fait une demande de service. Cette interrogation s'effectue de manière asynchrone, comme tâche de fond au cours du cycle (voir l'organigramme ci-dessous).

Lorsqu'un coprocesseur est interrogé et qu'il envoie une demande de service à l'UC, la demande est mise en file d'attente en vue de son traitement pendant la fenêtre de communication système.

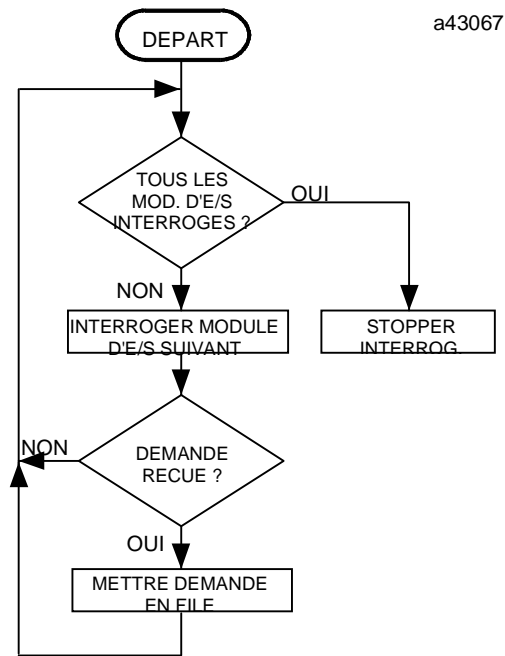


Figure 2-4. Communication du PCM avec l'API

Variantes du cycle de programme

En plus de l'exécution du cycle de programme normal, certaines variantes peuvent être rencontrées ou forcées. Ces variantes, décrites plus loin, peuvent être affichées et/ou modifiées depuis le menu d'état et de commande de l'API, dans le logiciel de programmation.

Mode Temps de cycle constant

Dans le cycle de programme normal, chaque cycle est exécuté aussi rapidement que possible avec une durée variant à chaque cycle. Un autre mode est le mode **TEMPS DE CYCLE CONSTANT**, où chaque cycle a la même durée. Utilisez un cycle constant lorsque des points d'E/S ou des valeurs de registre doivent être interrogées à une fréquence constante, notamment dans des algorithmes de contrôle. Pour cela, définissez le cycle constant configuré, qui sera ensuite utilisé comme mode de cycle par défaut et appliqué chaque fois que l'API passera du mode STOP au mode RUN. Il est possible de choisir une valeur comprise entre 5 et 200 millisecondes pour le temporisateur de cycle constant (par défaut : 100 millisecondes).

En raison des variations de temps pour chaque partie du cycle d'API, le temps de cycle constant doit être réglé à une valeur d'au moins 10 millisecondes supérieure au temps de cycle affiché sur la ligne d'état lorsque l'API est en mode de **CYCLE STANDARD**, afin d'éviter l'apparition intempestive de défauts pour cause de dépassement de cycle.

L'une des raisons de l'utilisation du mode **Temps de cycle constant** serait, par exemple, de garantir que les E/S sont rafraîchies à intervalles réguliers. Une autre raison serait de garantir qu'un certain temps s'écoule entre la restitution des sorties et l'acquisition des entrées du cycle suivant, pour permettre aux entrées de se stabiliser après avoir reçu les données de sortie du programme.

Si le temporisateur de cycle constant expire avant la fin du cycle, l'ensemble du cycle est exécuté, fenêtres comprises. Toutefois, un défaut de dépassement de cycle est enregistré au début du cycle suivant.

Remarque

Contrairement au cycle constant actif qui ne peut être modifié qu'en mode RUN, le mode de cycle constant configuré ne peut être modifié qu'en mode STOP. Il faut en outre stocker la configuration de la console de programmation vers l'API avant que la modification entre en vigueur. Une fois la configuration stockée, ce mode sera utilisé comme mode de cycle par défaut.

PLC

Lorsque l'API est mode **STOP**, le programme d'application n'est pas exécuté. On peut choisir si les E/S doivent être scrutées ou non. La scrutation des E/S peut avoir lieu en mode **STOP** si le paramètre **IOScan_Stop** (scrutation des E/S en mode **STOP**) sur l'écran de configuration du processeur est réglé sur **OUI**. Les communications avec la console de programmation et les coprocesseurs continuent. De plus, l'interrogation des cartes en défaut et l'exécution de la reconfiguration des cartes continuent en mode **STOP**. Pour des raisons d'efficacité, le système d'exploitation utilise des valeurs de tranches de temps plus grandes que celles utilisées en mode **RUN** (généralement d'environ 50 millisecondes par fenêtre).

Modes de la fenêtre de communication

Le mode par défaut de la fenêtre de communication de la console de programmation est le mode Limited. Autrement dit, quand la durée de traitement d'une demande est supérieure à 6 millisecondes, elle est répartie sur plusieurs cycles de façon à ce qu'aucun d'eux ne dure plus de 6 millisecondes. Pour les UC 313, 323 et 331, l'impact du cycle peut atteindre jusqu'à 12 millisecondes pendant un stockage en mode RUN. Le mode de la fenêtre active peut être modifié à l'aide de l'écran "Sweep Control" dans le logiciel Logicmaster. Pour obtenir des informations sur la modification du mode de la fenêtre active ou afficher l'aide en ligne du contrôle CIMPLICITY, voir le chapitre 5, "PLC Control and Status", du *Logicmaster 90™ Series 90™-30/20/Micro Programming Software User's Manual* (GFK-0466).

Remarque

Si vous remplacez le mode de la fenêtre système par le mode Limited, les modules en option tels que PCM ou GBC qui communiquent avec l'API à l'aide de la fenêtre système produiront moins d'impact sur le temps de cycle, mais les réponses à leurs demandes seront plus lentes.

Interrupteur à clé des UC 351/352 : Changement de mode et protection de la mémoire flash

Chaque UC 351 et 352 est équipée d'un interrupteur à clé à l'avant du module qui protège la mémoire flash contre tout écrasement de son contenu. Lorsque vous tournez la clé sur la position ON/RUN, la mémoire flash ne peut être modifiée qu'en ramenant préalablement la clé sur la position OFF.

Sur les versions 7 et ultérieures des UC 351 et 352, cet interrupteur à clé est doté d'une fonction supplémentaire qui permet de basculer l'API en mode STOP ou en mode RUN et de supprimer les défauts non fatals, comme décrit à la section suivante.

Enfin, sur les versions 8 et ultérieures des UC 351 et 352, cet interrupteur à clé offre aussi une fonction de protection améliorée de la mémoire qui fournit deux types de protection supplémentaires (voir la section "Utilisation de la protection de la mémoire des versions 8 et ultérieures").

Utilisation de l'interrupteur à clé des versions 7 et ultérieures

Contrairement aux fonctions de protection de mémoire flash de la version antérieure, l'UC ne possède le contrôle amélioré décrit ci-dessous que si vous activez l'interrupteur à clé à l'aide du paramètre Interrupteur à clé RUN/STOP dans l'écran de configuration de l'UC.

Avant que l'API passe en mode RUN, l'interrupteur à clé assure les mêmes protections et vérifications que la transition existante vers le mode RUN, ce qui signifie que l'API ne passe pas en mode RUN par l'intermédiaire de l'entrée de l'interrupteur à clé quand elle est en mode STOP/FAULT. Vous pouvez néanmoins effacer les défauts non fatals et placer l'API en mode RUN via l'interrupteur à clé.

Si les tables des défauts contiennent des erreurs *non fatales* (des erreurs qui n'entraînent pas le passage de l'UC au mode STOP/FAULT), l'UC passe en mode RUN la première fois que vous déplacez la clé de la position STOP à la position RUN et les tables des défauts ne sont PAS effacées.

Si les tables des défauts contiennent des erreurs *fatales* (passage de l'UC en mode STOP/FAULT), le voyant CPU RUN commence à clignoter à une fréquence de 2 Hz et un temporisateur de 5 secondes se met en marche la première fois que l'interrupteur à clé passe de la position STOP à la position RUN. Le clignotement du voyant RUN indique que les tables des défauts contiennent une ou plusieurs erreurs fatales. Dans ce cas, l'UC ne passe PAS à l'état RUN, même si l'interrupteur à clé se trouve sur la position RUN.

Effacement de la table des défauts à l'aide de l'interrupteur à clé

Lorsque vous déplacez la clé de la position RUN à la position STOP, puis que vous la ramenez sur la position RUN pendant les 5 secondes durant lesquelles le voyant RUN clignote, les défauts sont effacés et l'UC passe en mode RUN. A ce stade, le voyant cesse de clignoter et demeure allumé. Pour utiliser cette fonction, l'interrupteur doit demeurer sur la position RUN ou STOP pendant 1/2 seconde au moins avant d'être ramené sur l'autre position.

Remarque

Si vous laissez expirer le temporisateur de 5 secondes (arrêt du clignotement du voyant RUN), l'UC demeure dans son état d'origine, à savoir le mode STOP/FAULT, tandis que la table des défauts conserve les erreurs. Si vous tournez à nouveau la clé de l'interrupteur de la position STOP vers la position RUN à ce moment, le processus recommence en considérant cette action comme la première transition.

Le tableau suivant résume la manière dont les deux paramètres de l'UC affectant l'interrupteur à clé (R/S Switch et IOScan-Stop) et la position physique de celui-ci influencent l'API.

Paramètre R/S Key Switch de la configuration de l'UC	Position de l'interrupteur à clé	Paramètre IOScan-Stop de la configuration de l'UC	Fonctionnement de l'API
OFF	X	X	Tous les modes de la console de programmation de l'API sont autorisés.
ON	ON/RUN	X	Tous les modes de la console de programmation de l'API sont autorisés.
ON	OFF/STOP	X	L'API ne peut pas passer au mode RUN.
ON	Passage de l'interrupteur à clé de la position OFF/STOP à la position ON/RUN	X	L'API passe en mode RUN si aucun défaut fatal n'est présent ; sinon, le voyant RUN clignote pendant 5 secondes.
ON	Passage de l'interrupteur à clé de la position ON/RUN à la position OFF/STOP	NO	L'API passe en mode STOP-NO IO
ON	Passage de l'interrupteur à clé de la position ON/RUN à la position OFF/STOP	YES	L'API passe en mode STOP-IO

X=Ne produit aucun effet, quel que soit sa valeur

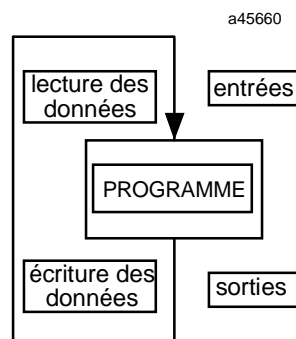
Protection améliorée de la mémoire des UC versions 8 et ultérieures

Outre les fonctionnalités décrites ci-dessus, l'interrupteur à clé des UC versions 8 et ultérieures peut aussi interdire toute modification de la mémoire RAM à partir du logiciel de programmation, à condition de définir préalablement le paramètre de programmation adéquat. Quand cette protection mémoire est activée, deux types d'opérations sont verrouillés : la configuration et le programme utilisateur ne peuvent pas être modifiés tandis que le forçage et le dépassement des données de point sont interdits. Vous activez ces fonctionnalités à l'aide du champ Mem Protect dans l'écran de configuration du module de l'UC 351 ou 352 du programme Logicmaster ou le champ Memory Protect de l'onglet Paramètres du module de l'UC 351 ou 352 dans l'écran de configuration du matériel du contrôle CIMPLICITY. Dans ces deux logiciels, cette fonctionnalité est désactivée par défaut.

Section 2 : Organisation du programme et données/références utilisateur

La taille totale de l'API Série 90-30 peut être de 6 Ko pour une UC modèle 311 ou 313, de 16 Ko pour un modèle 331, de 32 Ko pour un modèle 340 et de 80 Ko pour les UC modèles 341, 351 et 352. Pour l'API Série 90-20 avec UC modèle 211, la taille maximale d'un programme est de 2 Ko tandis que pour l'API Série 90 Micro avec UC Micro à 28 points, elle peut atteindre jusqu'à 12 Ko.

Le programme utilisateur contient les instructions exécutées lors de sa mise en route. Le nombre maximum de segments autorisé par bloc logique (programme ou sous-programme) est de 3000. L'API exécute le programme de façon cyclique.



Voir le document GFK-0356 Série 90t-30 Automate Programmable - Manuel d'installation ou le document GFK-0551 Series 90-20 Programmable Controller User's Manual pour la liste des tailles de programme et des limites de référence pour chaque modèle d'UC.

Tous les programmes commencent par une table de déclaration de variables. Cette table liste les symboles et les descriptions des références ayant été attribuées dans le programme utilisateur.

L'éditeur de déclaration de blocs liste les blocs de sous-programme déclarés dans le programme principal.

Blocs de sous-programme (API Série 90-30 seulement)

Un programme peut "appeler" les blocs de sous-programme lors de son exécution. Un sous-programme doit au préalable être déclaré dans l'éditeur de déclaration de blocs pour qu'une instruction CALL puisse être utilisée pour ce sous-programme. Chaque bloc logique du programme peut contenir jusqu'à 64 déclarations de blocs de sous-programme et jusqu'à 64 instructions CALL. La taille maximale d'un bloc de sous-programme est de 16 Ko ou de 3000 segments, mais le programme principal et l'ensemble des sous-programmes ne doivent pas excéder les limites de la taille logique pour ce modèle de processeur.

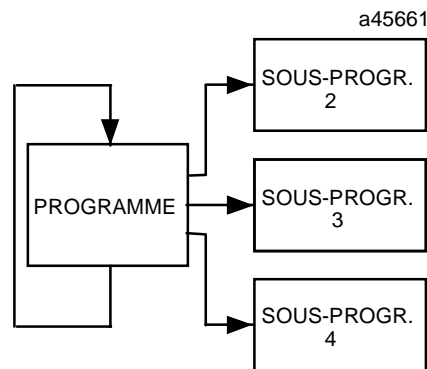
Remarque

Les blocs de sous-programme ne sont pas disponibles pour l'API Série 90-20 et Série Micro.

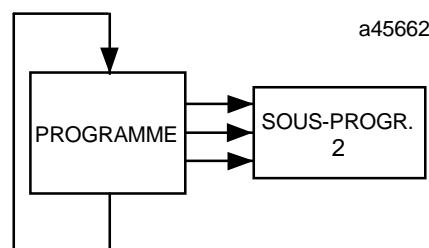
L'utilisation de sous-programmes est optionnelle. La division d'un programme en plusieurs sous-programmes plus petits peut simplifier la programmation et réduire le volume total du programme.

Exemples d'utilisation des blocs de sous-programme

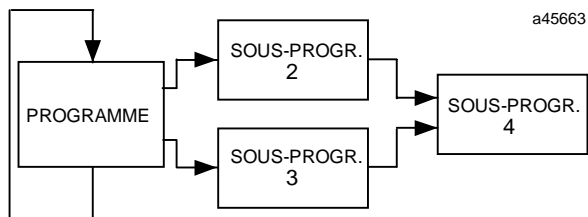
Un programme pourrait être divisé en trois sous-programmes, chacun d'eux pouvant être appelé par le programme selon les besoins. Dans cet exemple, le programme peut ne contenir qu'un minimum logiciel, chargé essentiellement de gérer le séquençement des blocs de sous-programme.



Un bloc de sous-programme peut être utilisé plusieurs fois pendant l'exécution du programme. La partie logicielle devant être répétée plusieurs fois dans un programme peut être entrée dans un bloc de sous-programme. Pour y accéder, des instructions CALL permettent d'appeler ce bloc de sous-programme, réduisant ainsi la taille totale du programme.



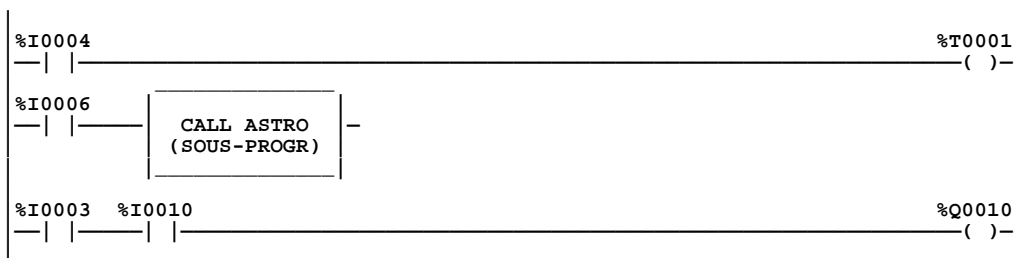
Non seulement les blocs de sous-programme peuvent être appelés par le programme, mais ils peuvent être également appelés par d'autres blocs de sous-programme. Un bloc de sous-programme peut aussi s'appeler lui-même.



L'API n'autorise que huit appels imbriqués avant de consigner un défaut "Application Stack Overflow" et de passer en mode **Arrêt/Défaut**. L'imbrication des niveaux d'appel compte le programme principal comme niveau 1.

Appel des blocs de sous-programme

Un bloc de sous-programme est exécuté chaque fois qu'il est appelé par le bloc principal du programme :



Ce exemple montre l'instruction CALL appelant un sous-programme, telle qu'elle apparaît dans le bloc appelant.

Sous-programmes périodiques

Les versions 4.20 et ultérieures des UC 340 et supérieures prennent en charge les sous-programmes périodiques. Vous devez cependant tenir compte des restrictions suivantes :

1. Les blocs fonctionnels du temporisateur (TMR, ONDTR et OFDTR) ne s'exécutent pas correctement au sein d'un sous-programme périodique. En effet, quand la plage de références d'un bloc fonctionnel DOIO au sein d'un sous-programme périodique contient des références attribuées à un module E/S intelligent (HSC, Power Mate APM, Genius, etc.), cela entraîne une perte de communications entre l'UC et le module. Les contacts FST_SCN et LST_SCN (%S1 et %S2) possèdent une valeur indéterminée pendant l'exécution du sous-programme périodique. Un sous-programme périodique ne peut pas appeler ou être appelé par d'autres sous-programmes.
2. La durée du sous-programme périodique (c'est-à-dire l'intervalle maximum entre le moment où il doit avoir été exécuté et celui où il est réellement exécuté) est d'environ 0,35 milliseconde lorsque le bac principal ne contient pas de module PCM, CMM ou ADC. Si le bac contient un de ces modules et même s'il n'est pas configuré ou utilisé, cette durée peut atteindre environ 2,25 millisecondes. Pour cette raison, nous vous recommandons de ne *pas* utiliser de sous-programme périodique avec des produits PCM.

Références utilisateur

Les données utilisées dans un programme d'application sont stockées sous forme de références logiques ou de références de type registre.

Tableau 2-4. Références des registres

Type	Description
%R	Le préfixe %R est utilisé pour attribuer des références registre système, qui stockent les données de programme, telles que les résultats de calculs.
%AI	Le préfixe %AI représente un registre d'entrée analogique. Ce préfixe est suivi de l'adresse de registre de la référence (par exemple, %AI0015). Un registre d'entrée analogique contient la valeur d'une entrée analogique ou une autre valeur.
%AQ	Le préfixe %AQ représente un registre de sortie analogique. Ce préfixe est suivi de l'adresse de registre de la référence (par exemple, %AQ0056). Un registre de sortie analogique contient la valeur d'une sortie analogique ou une autre valeur.

Remarque

Toutes les références de type registre sont sauvegardées en cas de coupure de l'UC.

Tableau 2-5. Références logiques (Bits)

Type	Description
%I	<p>Le préfixe %I représente les références d'entrée. Ce préfixe est suivi de l'adresse de la référence dans la table des entrées (par exemple, %I00121). Les références %I sont situées dans la table d'état des entrées, qui stocke l'état de toutes les entrées envoyées par les modules d'entrée au cours de la dernière acquisition des entrées.</p> <p>Les adresses de référence sont attribuées aux modules d'entrée logique avec le logiciel de configuration ou la miniconsole de programmation (HHP). Tant qu'une adresse de référence n'est pas attribuée, aucune donnée n'est reçue en provenance du module.</p>
%Q	<p>Le préfixe %Q représente les références des sorties physiques. La fonction de contrôle des bobines du logiciel Logicmaster 90-30/20/Micro recherche, dans les instructions, l'existence de l'utilisation multiple de références %Q à des bobines ou sorties à relais. On peut choisir le niveau désiré du contrôle des bobines (SIMPLE, AVERTIR MULTIPLE, ou MULTIPLE). Voir le document GFK-0466 Logicmaster 90-30/20/Micro Programming Software User's Manual pour plus d'informations sur cette fonction.</p> <p>Le préfixe %Q est suivi de l'adresse de la référence dans la table des sorties (par exemple, %Q00016). Les références %Q sont situées dans la table d'état des sorties, qui enregistre l'état des références des sorties, tel qu'il a été défini par le programme d'application. Les valeurs de cette table d'état des sorties sont envoyées aux modules de sortie à la fin de la scrutation du programme.</p> <p>Une adresse de référence est attribuée aux modules de sorties logiques en utilisant le logiciel de configuration ou la miniconsole de programmation (HHP). Tant qu'une adresse de référence n'est pas attribuée, aucune donnée n'est envoyée au module. Une référence %Q particulière peut être rémanente ou non rémanente. *</p>
%M	<p>Le préfixe %M représente les références internes. La fonction de contrôle des bobines du logiciel Logicmaster 90-30/20 recherche l'existence, dans les instructions, de l'utilisation multiple de références %M à des bobines ou sorties à relais. On peut choisir le niveau désiré du contrôle des bobines (SIMPLE, AVERTIR MULTIPLE, ou MULTIPLE). Voir le document GFK-0466 Logicmaster 90-30/20/Micro Programming Software User's Manual pour plus d'informations sur cette fonction. Une référence %M particulière peut être rémanente ou non rémanente. *</p>
%T	<p>Le préfixe %T représente les références temporaires. Le contrôle de l'utilisation multiple des références à des bobines n'est pas effectué sur ces références ; par conséquent, elles peuvent être utilisées plusieurs fois dans le même programme même lorsque le contrôle de l'utilisation des bobines est validé. %T peut être utilisé pour éviter des conflits d'utilisation des bobines lorsque les fonctions couper/coller et écrire/inclure fichier sont utilisées.</p> <p>Cet emplacement mémoire étant utilisé temporairement, il n'est jamais conservé en cas de coupure d'alimentation ou de transition RUN-STOP-RUN et ne peut pas être utilisé avec des bobines rémanentes.</p>

* La rémanence dépend du type de bobine. Pour plus d'informations, voir "Rémanence des données" page 2- 23.

Tableau 2-5. Références logiques - Suite

Type	Description
%S	<p>Le préfixe %S représente les références d'état système. Ces références sont utilisées pour accéder aux données spéciales de l'API, telles que les données des temporisateurs, des scrutations et des défauts. Les références système incluent les références %S, %SA, %SB et %SC.</p> <p>Les références %S, %SA, %SB et %SC peuvent être utilisées sur n'importe quels contacts.</p> <p>Les références %SA, %SB et %SC peuvent être utilisées sur les bobines rémanentes –(M)–.</p> <p>La référence %S peut être utilisée comme arguments d'entrée de chaîne binaire ou comme mot pour les instructions ou blocs fonctionnels.</p> <p>Les références %SA, %SB et %SC peuvent être utilisées comme arguments d'entrée ou de sortie de chaîne binaire ou comme mot pour les instructions ou blocs fonctionnels.</p>
%G	<p>Le préfixe %G représente les références des données globales. Ces références sont utilisées pour accéder aux données partagées par plusieurs API. Les références %G peuvent être utilisées sur les bobines rémanentes et les contacts car la mémoire %G est toujours rémanente. %G ne peut pas être utilisé sur les bobines non rémanentes.</p>

Transitions et forçages

Les références utilisateur %I, %Q, %M et %G ont des bits associés de transition et de forçage. Les références %T, %S, %SA, %SB et %SC ont des bits de transition, mais pas de bits de forçage. L'UC utilise les bits de transition pour les compteurs et les bobines de transition. A noter que les compteurs n'utilisent pas le même type de bits de transition que les bobines. Les bits de transition des compteurs sont stockés dans la référence de localisation.

Sur les UC modèles 331 et supérieurs, les bits de forçage peuvent être mis à "1". Lorsqu'ils sont à "1", les références associées ne peuvent pas être modifiées ni par le programme ni par le dispositif d'entrée ; elles ne peuvent être modifiées que sur une instruction de la console de programmation. Les UC modèles 323, 321, 313 311 et 211 ainsi que les UC Micro ne prennent pas en charge le forçage des références discrètes.

Rémanence des données

Les données sont dites "rémanentes" si elles sont sauvegardées par l'API lorsque celui-ci est stoppé. Les API Série 90 conservent le programme, les tables des défauts et les diagnostics, les forçages et les sorties forcées, les données de mot (%R, %AI, %AQ), les données des bits (%I, %S, %G, les bits de défauts et les bits réservés), les données %Q et %M (sauf si elles sont utilisées avec les bobines non rémanentes), et les données de mot stockées dans %Q et %M. Les données de %T ne sont pas sauvegardées. Cependant, comme nous l'avons décrit plus haut, les données des bits %SC sont rémanentes, mais pas celles des bits %S, %SA et %SB.

Les références %Q et %M ne sont pas rémanentes (c'est-à-dire qu'elles sont effacées à la mise sous tension lorsque l'API passe du mode STOP au mode RUN) chaque fois qu'elles sont utilisées avec les bobines non rémanentes. Les bobines non rémanentes incluent les bobines —()—, les bobines inversées —(/)—, les bobines de mise à "1" —(S)—, et les bobines de remise à "0" —(R)—.

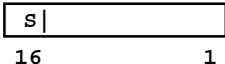

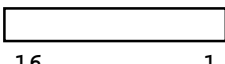
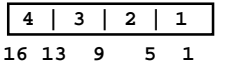

Lorsque des références %Q ou %M sont utilisées avec les bobines rémanentes, ou comme sorties de blocs fonctionnels, le contenu est conservé en cas de coupure d'alimentation ou de transition **RUN-STOP-RUN**. Les bobines rémanentes incluent les bobines rémanentes —(M)—, les bobines rémanentes inversées —(/M)—, les bobines rémanentes de mise à "1" —(SM)—, et les bobines rémanentes de remise à "0" —(RM)—.

La dernière programmation d'une référence %Q ou %M dans une instruction de bobine détermine si la référence %Q ou %M est rémanente ou non selon le type de bobine. Par exemple, si %Q00001 a été programmé en dernier comme référence d'une bobine rémanente, les données %Q00001 sont rémanentes. Par contre, si %Q00001 a été programmé en dernier sur une bobine non rémanente, les données %Q00001 ne sont pas rémanentes.

Types de données

Les types de données sont les suivants :

Tableau 2-6. Types de données

Type	Nom	Description	Format des données
INT	Entier signé	Les entiers signés utilisent des emplacements mémoire de 16 bits, et sont représentés en complément à deux. La plage valide des données de type INT est de -32 768 à +32 767.	<p>Registre 1</p>  <p>(emplacement à 16 bits)</p>
DINT	Entier signé en double précision	Les entiers signés en double précision sont stockés dans des emplacements mémoire de 32 bits (en fait, dans 2 emplacements mémoire de 16 bits) et sont représentés en complément à deux. (le bit 32 est le bit de signe). La plage valide des données de type DINT est de -2 147 483 648 à +2 147 483 867.	<p>Registre 2 Registre 1</p>  <p>(Valeur en complément à deux)</p>
BIT	Bit	Les données de type BIT sont les plus petites unités de mémoire. Elles ont deux états : 1 ou 0. Une chaîne de bits peut avoir une longueur N.	
BYTE	Octet	Les données de type OCTET ont une valeur de 8 bits. La plage valide est comprise entre 0 et 255 (0 à FF au format hexadécimal).	
WORD		Les données de type MOT utilisent 16 bits consécutifs d'emplacement mémoire ; mais, au lieu de représenter un nombre, dans l'emplacement mémoire, les bits sont indépendants les uns des autres. Chaque bit représente son propre état binaire (1 ou 0), et les bits ne sont pas considérés ensemble pour représenter un nombre entier. La plage valide des valeurs de mot est de 0 à FFFF.	<p>Registre 1</p>  <p>(emplacement à 16 bits)</p>
BCD-4	Décimal codé binaire à 4 chiffres	Les nombres DCB à 4 chiffres utilisent des emplacements mémoire à 16 bits. Chaque chiffre DCB utilise 4 bits et peut représenter des nombres entre 0 et 9. Ce codage DCB des 16 bits a une plage de valeurs valides de 0 à 9999.	<p>Registre 1</p>  <p>(4 chiffres BCD)</p>
REAL	Virgule flottante	Les nombres réels utilisent 32 bits consécutifs (deux emplacements mémoire à 16 bits consécutifs). La plage des nombres susceptibles d'être stockés dans ce format est comprise entre ± 1,401298E-45 et ± 3,402823E+38.	<p>Registre 2 Registre 1</p>  <p>(Valeur en complément à deux)</p>

S = bit de signe (0 = positif, 1 = négatif).

Références d'état système

Les références d'état système dans l'API Série 90 sont attribuées aux emplacements mémoire %S, %SA, %SB et %SC. Elles ont chacune un symbole. Des références d'impulsion d'horloge peuvent être, par exemple, T_10MS, T_100MS, T_SEC, et T_MIN. Des références pratiques peuvent être, par exemple, FST_SCN, ALW_ON et ALW_OFF.

Remarque

Les bits %S sont des bits à lecture seulement ; ne pas écrire dans ces bits. Vous pouvez toutefois écrire dans les bits %SA, %SB et %SC.

Ci-dessous se trouve une liste des références d'état système disponibles, pouvant être utilisées dans un programme d'application. Lorsqu'on entre le programme, on peut utiliser soit la référence soit le symbole. Voir le chapitre 3, "Description et correction des défauts", pour plus d'informations sur la description des défauts et le moyen de les corriger.

Vous ne pouvez pas utiliser ces noms spéciaux dans un autre contexte.

Tableau 2-7. Référence d'état système

Référence	Symbole	Définition
%S0001	FST_SCN	Mis à "1" si le cycle actuel est le premier.
%S0002	LST_SCN	Remis de "1" à "0" si le cycle actuel est le dernier.
%S0003	T_10MS	Contact temporisé de 0,01 s.
%S0004	T_100MS	Contact temporisé de 0,1 s.
%S0005	T_SEC	Contact temporisé de 1 s.
%S0006	T_MIN	Contact temporisé de 1 min.
%S0007	ALW_ON	Toujours à "1".
%S0008	ALW_OFF	Toujours à "0".
%S0009	SY_FULL	Mis à "1" lorsque la table des défauts automate est pleine. Remis à "0" lorsqu'une entrée est retirée dans cette table ou lorsque la table est vidée.
%S0010	IO_FULL	Mis à "1" lorsque la table des défauts d'E/S est pleine. Remis à "0" lorsqu'une entrée est retirée de la Table des défauts d'E/S ou lorsque la table est vidée.
%S0011	OVR_PRE	Mis à "1" lorsqu'un forçage existe dans l'emplacement mémoire %I, %Q, %M, ou %G.
%S0013	PRG_CHK	Mis à "1" lorsque le contrôle du programme de fond est actif.
%S0014	PLC_BAT	Mis à "1" pour indiquer un "seuil pile bas" dans une UC v. 4 ou ult. La référence du contact est rafraîchie une fois par cycle.

Tableau 2-7. Références d'état système - Suite

Référence	Symbole	Définition
%S0017	SNP_XACT	Le système central SNP_X est activement connecté à l'UC.
%S0018	SNPX_RD	Le système central SNP_X a lu des données provenant de l'UC.
%S0019	SNPX_WT	Le système central SNP_X a écrit des données dans l'UC.
%S0020		Mis à "1" lorsqu'une instruction de comparaison utilisant des données REAL s'exécute correctement. Il est mis à "0" lorsque l'une ou l'autre entrée n'est pas un nombre (de type NaN).
%S0032		Réservé au logiciel de programmation.
%SA0001	PB_SUM	Mis à "1" lorsqu'une CHECKSUM calculée dans le programme d'application ne correspond pas à la CHECKSUM de référence. Si le défaut résulte d'une panne temporaire, le bit logique peut être remis à "0" en chargeant le programme dans l'UC. Si le défaut résulte d'une panne permanente de la RAM, l'UC doit être remplacée.
%SA0002	OV_SWP	Mis à "1" lorsque l'API détecte que le cycle précédent a duré plus longtemps que le temps spécifié par l'utilisateur. Remis à "0" lorsque l'API détecte que la durée du cycle précédent n'a pas excédé le temps spécifié. Il est également mis à "0" pendant la transition du mode STOP au mode RUN . Valide uniquement si l'API est en mode Cycle constant.
%SA0003	APL_FLT	Mis à "1" lorsqu'un défaut de l'application s'est produit. Remis à "0" lorsque l'API passe du mode STOP en mode RUN .
%SA0009	CFG_MM	Mis à "1" lorsqu'une incohérence de la configuration est détectée à la mise sous tension du système ou pendant le chargement de la configuration. Remis à "0" à la mise sous tension de l'API si aucune incohérence n'existe, ou pendant le chargement de la configuration qui correspond au matériel.
%SA0010	HRD_CPU	Mis à "1" lorsque le diagnostic détecte un problème matériel de l'UC. Remis à "0" en remplaçant l'UC.
%SA0011	LOW_BAT	Mis à "1" lorsque le "seuil pile bas" est atteint. Remis à "0" en remplaçant la pile et en s'assurant qu'à la mise sous tension de l'API, la condition de la pile est normale.
%SA0014	LOS_IOM	Mis à "1" lorsqu'un module d'E/S cesse de communiquer avec l'UC de l'API. Remis à "0" en remplaçant le module et en remettant le bac principal sous tension.
%SA0015	LOS_SIO	Mis à "1" lorsqu'un coprocesseur cesse de communiquer avec l'UC de l'API. Remis à "0" en remplaçant le module et en remettant le bac principal sous tension.
%SA0019	ADD_IOM	Mis à "1" lorsqu'un module d'E/S est ajouté à un bac. Remis à "0" en remettant le bac principal sous tension et si la configuration enregistrée correspond au matériel.
%SA0020	ADD_SIO	Mis à "1" lorsqu'un coprocesseur est ajouté à un bac. Remis à "0" en remettant le bac principal sous tension et si la configuration enregistrée correspond au matériel.
%SA0027	HRD_SIO	Mis à "1" lorsqu'une panne matérielle est détectée dans un coprocesseur. Remis à "0" en remplaçant le module et en remettant le bac principal sous tension.
%SA0031	SFT_SIO	Mis à "1" lorsqu'un défaut logiciel irrémédiable est détecté dans un coprocesseur. Remis à "0" en remettant le bac principal sous tension et si la configuration correspond au matériel.
%SB0010	BAD_RAM	Mis à "1" lorsque l'UC détecte un défaut de la mémoire RAM à la mise sous tension. Remis à "0" si l'UC détecte que la RAM est valide à la mise sous tension.

Tableau 2-7. Références d'état système - Suite

Référence	Symbole	Définition
%SB0011	BAD_PWD	Mis à "1" lorsque se produit une violation d'accès par mot de passe. Remis à "0" lorsque la table des défauts de l'API est vidée.
%SB0013	SFT_CPU	Mis à "1" lorsque l'UC détecte une erreur irrémédiable dans le logiciel. Remis à "0" en remettant l'UC sous tension.
%SB0014	STOR_ER	Mis à "1" lorsqu'une erreur se produit quand la console de programmation effectue une opération de stockage. Mis à "0" lorsque cette opération est menée à bien.
%SC0009	ANY_FLT	Mis à "1" lorsqu'un défaut se produit. Mis à "0" lorsque les deux tables de défauts sont vides (aucune entrée).
%SC0010	SY_FLT	Mis à "1" lorsqu'un défaut est enregistré dans la table des défauts automate. Mis à "0" lorsque la Table des défauts automate est vide (aucune entrée).
%SC0011	IO_FLT	Mis à "1" lorsqu'un défaut est enregistré dans la Table des défauts des E/S. Mis à "0" lorsque la Table des défauts d'E/S est vide (aucune entrée).
%SC0012	SY_PRES	Mis à "1" tant qu'il existe au moins un défaut enregistré dans la Table des défauts automate. Mis à "0" lorsque la Table des défauts automate est vide (aucune entrée).
%SC0013	IO_PRES	Mis à "1" tant qu'il existe au moins un défaut enregistré dans la Table des défauts des E/S. Mis à "0" lorsque la Table des défauts d'E/S est vide (aucune entrée).
%SC0014	HRD_FLT	Mis à "1" lorsqu'un défaut matériel se produit. Mis à "0" lorsque les deux tables de défauts sont vides (aucune entrée).
%SC0015	SFT_FLT	Mis à "1" lorsqu'un défaut logiciel se produit. Mis à "0" lorsque les deux tables de défauts sont vides (aucune entrée).

Structure des blocs fonctionnels

Chaque segment de logique est constituée d'une ou plusieurs instructions de programmation. Elles peuvent être de type diagramme en échelle ou de fonctions plus complexes.

Format des instructions à diagramme en échelle

Le logiciel Logicmaster 90-30/20 inclut plusieurs types d'instructions par diagramme en échelle. Ces instructions assurent un déroulement et un contrôle de base du programme. Cela inclut, par exemple, un contact de relais normalement ouvert et une bobine inversée. Chaque bobine et contact de relais comporte une entrée et une sortie. Ensemble, ils fournissent un flux validant à travers la bobine ou le contact. Leur association en grand nombre permet de résoudre les équations logiques du programme.

Chaque bobine ou contact de relais doit posséder une référence, entrée lorsque le relais est sélectionné. Pour un contact, la référence correspond à un emplacement dans la mémoire, déterminant un niveau logique 1 ou 0 selon que le flux validant entre ou non dans le contact. Dans l'exemple suivant, si la référence %I0122 est à "1", le flux validant passera par ce contact de relais (sélection du relais).

```
%I0122
-| |-
```

Pour une bobine, la référence correspond à un emplacement mémoire contrôlé par le flux validant entrant dans la bobine. Dans cet exemple, si le flux validant entre par le côté gauche de la bobine, la référence %Q0004 est mise à "1".

```
%Q0004
-( )-
```

Le logiciel de programmation et la miniconsole de programmation (HHP) ont tous deux une fonction de contrôle des bobines, recherchant l'existence, dans les instructions, de l'utilisation multiple de références %Q ou %M aux sorties ou bobines de relais.

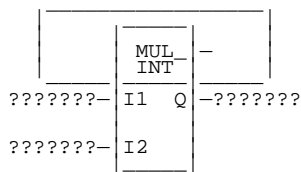
Format des blocs fonctionnels de programme

Certaines instructions sont très simples, telles que les instructions MCR (relais de contrôle maître), indiquées entre crochets par leur abréviation :

```
-[ MCR ]-
```

D'autres sont plus complexes. Les informations qui seront utilisées par l'instruction peuvent être entrées à plusieurs endroits.

Le bloc fonctionnel présenté ci-dessous est une multiplication (MUL). Les parties qui le composent sont communes à de nombreuses instructions de programmation. La partie supérieure du bloc fonctionnel indique le nom de l'instruction. Il peut également indiquer un type de données ; dans le cas présent, il s'agit d'un nombre entier signé.



Nom du bloc fonctionnel (MUL) et type de données (INT). INT (entier signé) représente le type et la taille des données sur lesquelles portera l'instruction.

De nombreuses instructions permettent de choisir le type de données sur lesquelles portera l'instruction une fois sélectionnée. Par exemple, on peut choisir le type de données "entier signé en double précision" pour l'instruction MUL. D'autres informations sur les types de données sont développées dans les pages précédentes de ce chapitre.

Paramètres des blocs fonctionnels

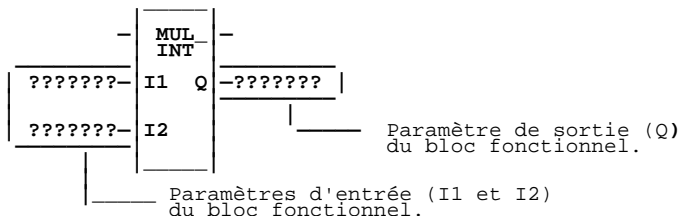
Chaque ligne entrant par la gauche d'un bloc fonctionnel représente une entrée dans cette instruction. Les deux formes d'entrée d'un bloc fonctionnel sont les constantes et les références. Une constante est une valeur explicite. Une référence est l'adresse d'une valeur.

Dans l'exemple suivant, le paramètre d'entrée I1 est entré dans le bloc fonctionnel ADD comme constante, et le paramètre d'entrée I2 comme référence.



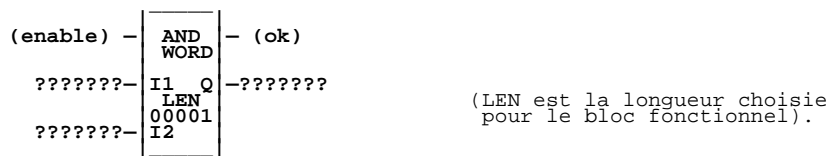
Chaque ligne quittant le côté droit du bloc fonctionnel représente une sortie. Il n'existe qu'une seule forme de sortie d'un bloc fonctionnel ou d'une référence. Les sorties ne peuvent jamais être écrites par des constantes.

Lorsqu'un point d'interrogation se trouve à gauche d'un bloc fonctionnel, on doit entrer la donnée proprement dite, l'adresse de la référence où se trouve cette donnée, ou encore une variable représentant cette adresse de référence. Lorsque des points d'interrogation se trouvent à droite d'un bloc fonctionnel, on entre généralement l'adresse de référence de la donnée définie par le bloc fonctionnel ou une variable représentant cette adresse de référence.

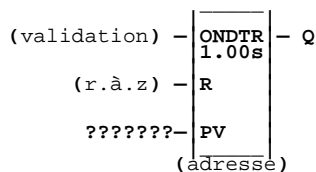


La plupart des blocs fonctionnels ne modifient pas les données d'entrée, mais placent le résultat de l'opération dans une référence de sortie.

Pour les instructions intervenant sur des tableaux, il est possible de choisir une longueur pour l'instruction. Dans le bloc fonctionnel suivant, une chaîne comportant jusqu'à 256 mots ou doubles mots peut être choisie pour l'instruction logique AND.

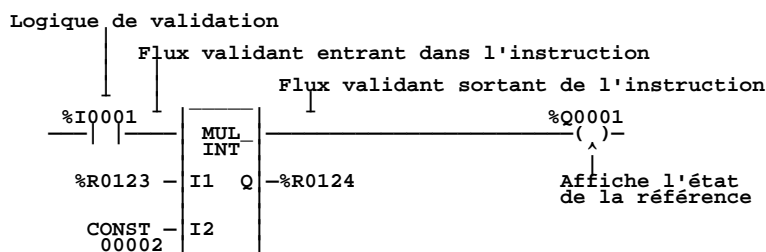


Les instructions de temporisation, de comptage, BITSEQ et ID exigent une adresse pour l'emplacement de trois mots (registres) qui stockent la valeur actuelle, la valeur présélectionnée, et un mot de contrôle de l'instruction.



Entrée et sortie du flux validant d'une instruction (validation d'un bloc fonctionnel)

L'entrée du flux validant s'effectue en haut à gauche du bloc fonctionnel. Souvent, la logique de validation est utilisée pour commander un bloc fonctionnel par le contrôle de flux ; sinon, le bloc fonctionnel est exécuté sans condition à chaque cycle d'UC.



Remarque

Les blocs fonctionnels ne peuvent pas être directement reliés au rail de flux gauche. Vous pouvez cependant utiliser %S7, le bit ALL_ON (toujours à "1") avec un contact normalement ouvert qui est lié au rail pour appeler une fonction à chaque cycle.

Le flux validant sort en haut à droite du bloc fonctionnel. Il peut être dirigé vers une autre logique de programme ou vers une bobine (optionnel). Les blocs fonctionnels transmettent le flux validant (niveau logique 1) s'ils ont été exécutés de façon satisfaisante. La description des instructions, dans ce manuel, explique les conditions dans lesquelles chaque instruction dirige le flux validant vers la droite.

Section 3 : Séquences de mise sous/hors tension

Il existe deux séquences possibles de mise sous tension de l'API Série 90-30 : la mise sous tension à chaud et la mise sous tension à froid. L'UC utilise généralement la séquence de mise sous tension à froid. Toutefois, dans une API modèle 331 ou supérieur, si l'intervalle de temps entre une mise hors tension et la mise sous tension suivante est inférieur à 5 secondes, la séquence de mise sous tension à chaud est utilisée.

Mise sous tension

Une séquence de mise sous tension à froid est composée de la séquence d'événements suivants :

Remarque

A l'exception de l'étape 1 qui est sautée, la séquence de mise sous tension à chaud est identique à la séquence de mise sous tension à froid.

1. L'UC exécute un autodiagnostic. Cela inclut le contrôle d'une partie de la RAM sauvegardée par pile pour déterminer si elle contient ou non des données valides.
2. Si une mémoire EPROM, EEPROM ou Flash est présente et si l'option de mise sous tension PROM dans PROM spécifie que le contenu PROM doit être utilisé, ce dernier est copié dans la mémoire RAM. Si aucune mémoire EPROM, EEPROM ou Flash n'est présente, la mémoire RAM demeure identique et n'est pas remplacée par le contenu PROM.
3. L'UC interroge chaque emplacement dans le système pour connaître les cartes présentes.
4. La configuration matérielle est comparée à la configuration logicielle pour s'assurer qu'elles sont identiques. Toute différence détectée est considérée comme défaut et signalée. En outre, si une carte est spécifiée dans la configuration logicielle et si un autre module est présent dans la configuration matérielle physique, cette condition est un défaut et est signalée.
5. S'il n'existe pas de configuration logicielle, l'UC utilise celle par défaut.
6. L'UC établit la communication entre lui-même et tous modules intelligents existants.
7. Dans l'étape finale de l'exécution, le mode du premier cycle est déterminé d'après la configuration de l'UC. S'il s'agit du mode RUN, le cycle est exécuté comme indiqué dans la "transition du mode STOP au mode RUN". La Figure 2.5, page suivante, présente la séquence de décision du processeur lorsqu'il décide de copier depuis la PROM ou de mettre sous tension en mode **STOP** ou **RUN**.

Remarque

Les étapes 2 à 6 ci-dessus ne s'appliquent pas à l'API Série 90 Micro. Pour plus d'informations sur les séquences de mise sous et hors tension de l'API Micro, voir la section "Séquences de mise sous et hors tension" du chapitre 5, "System Operation" dans le *Series 90 Micro PLC User's Manual* (GFK-1065).

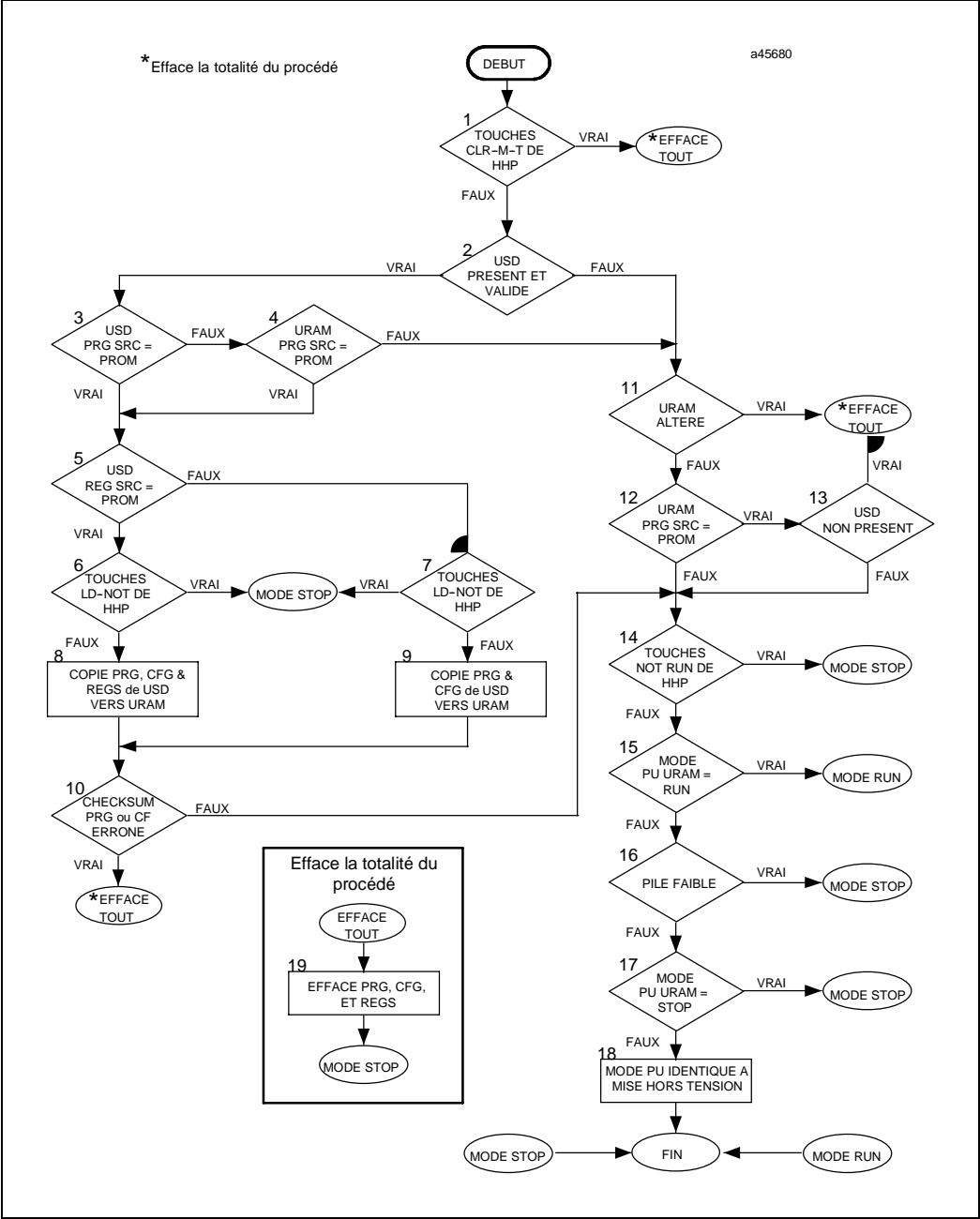


Figure 2-5. Séquence de mise sous tension

Avant l'instruction START (DEBUT) dans le diagramme de mise sous tension, l'UC exécute les diagnostics de mise sous tension pour tester la mémoire RAM ainsi que les différents périphériques qu'elle utilise. Une fois les diagnostics terminés, les structures de données internes et les périphériques utilisés par l'UC sont initialisés. L'UC détermine ensuite si la mémoire RAM utilisateur (URAM) a été altérée. Si c'est le cas, le programme utilisateur et la configuration sont effacés, les valeurs par défaut sont rétablies et tous les registres utilisateur sont supprimés.

TERMES DU DIAGRAMME :

PRG = programme utilisateur

CFG = configuration utilisateur

REGS = registres utilisateur (références %I, %Q, %M, %G, %R, %AI et %AQ).

USD = périphérique de stockage utilisateur, périphérique EEPROM ou Flash.

URAM = mémoire RAM utilisateur non volatile qui contient PRG, CFG et REGS.

DESCRIPTION DU DIAGRAMME :

- (1) Les touches <CLR> et <M_T> de HHP ont-elles été enfoncées pendant la mise sous tension pour effacer la totalité de URAM ?
- (2) USD est-il présent (ne peut être absent que sur les modèles qui utilisent le périphérique EEPROM) et les informations sur USD sont-elles valides ?
- (3) Le paramètre PRG SRC dans USD a-t-il la valeur Prom, signifiant que PRG et CFG doivent être chargés à partir du périphérique USD ?
- (4) Le paramètre PRG SRC dans URAM a-t-il la valeur Prom, signifiant que PRG et CFG doivent être chargés à partir du périphérique USD ?
- (5) Le paramètre REG SRC dans USD a-t-il la valeur Prom, signifiant que REGS doit être chargé à partir du périphérique USD ?
- (6 & 7) Les touches <LD> et <NOT> de HHP ont-elles été enfoncées pendant la mise sous tension pour empêcher le chargement de PRG, CFG et REGS à partir d'USD ?
- (8) Copie PRG, CFG et REGS d'USD vers URAM.
- (9) Copie PRG et CFG d'USD vers URAM.
- (10) Les checksums de PRG ou CFG venant d'être chargés à partir d'USD ne sont-ils pas valides ?
- (11) URAM est-il altéré ? Cela peut être dû au fait que le système a été mis hors tension alors que la pile était à plat ou non connectée. Cela peut aussi être dû à une mise à jour du micrologiciel.
- (12) Le paramètre PRG SRC dans URAM a-t-il la valeur Prom, signifiant que PRG et CFG doivent être chargés à partir du périphérique USD ?
- (13) USD est-il présent ? S'applique uniquement aux modèles qui utilisent le périphérique EEPROM.
- (14) Les touches <NOT> et <RUN> de HHP ont-elles été enfoncées pendant la mise sous tension pour effectuer la mise sous tension en mode Stop ?
- (15) Le paramètre PWR UP dans URAM a-t-il pour valeur RUN ?
- (16) La pile est-elle à plat ?
- (17) Le paramètre PWR UP dans URAM a-t-il pour valeur STOP ?
- (18) Attribue au mode de mise sous tension la valeur du mode de mise hors tension.
- (19) Efface PRG, CFG et REGS.

Remarque

La première partie du diagramme de la page précédente ne s'applique pas à l'API Série 90 Micro. Pour plus d'informations sur les séquences de mise sous et hors tension de l'API Micro, voir la section "Séquences de mise sous et hors tension" du chapitre 5, "System Operation" dans le *Series 90 Micro PLC User's Manual* (GFK-1065).

Mise hors tension

Le système est mis hors tension lorsque l'alimentation détecte que la tension secteur d'entrée a chuté pendant plus d'une période ou que la sortie d'alimentation 5 V a chuté au-dessous de 4,9 Vcc.

Section 4 : Compteurs et temporisateurs

Les compteurs et temporisateurs de l'API Série 90-30 incluent un compteur de temps de fonctionnement, une horloge interne (modèles 331, 340/341, 351/352 et Micro à 28 points), un temporisateur chien de garde et un temporisateur de cycle constant. Deux types de blocs fonctionnels de temporisation incluent un temporisateur suspensif et un temporisateur de démarrage-remise à zéro. Quatre contacts d'impulsions d'horloge s'ouvrent et se ferment pendant des intervalles de 0,01 s, 0,1 s, 1 s, et 1 min.

Compteur de temps de fonctionnement

Le compteur de temps de fonctionnement utilise des "impulsions" de 100 microsecondes pour suivre le temps écoulé depuis la mise sous tension de l'UC. Ce compteur redémarre à zéro après chaque coupure d'alimentation et mise sous tension. Une fois par seconde, le matériel interrompt l'UC pour permettre l'enregistrement des secondes. Le comptage des secondes peut continuer pendant environ 100 ans après le démarrage du compteur.

Comme le compteur de temps de fonctionnement est la base des opérations du système d'exploitation et des blocs fonctionnels de temporisation, il ne peut pas être remis à zéro depuis le programme utilisateur ou la console de programmation. Toutefois, le programme d'application peut lire la valeur actuelle du compteur de temps de fonctionnement en utilisant la demande de service 16.

Horloge interne

Les API Micro à 28 points et les API Série 90-30 modèles 331 et supérieurs sont équipées d'une horloge matérielle qui indique l'heure. L'horloge interne assure sept fonctions :

- Année (deux chiffres).
- Mois.
- Quantième du mois.
- Heure.
- Minute.
- Seconde.
- Jour de la semaine.

L'horloge interne est sauvegardée par pile et conserve sa valeur en cas de coupure d'alimentation. Toutefois, à moins d'initialiser l'horloge, les valeurs qu'elle contient sont sans signification. Le programme d'application peut lire et régler l'horloge interne en utilisant la demande de service 7. L'horloge interne peut être également lue et réglée à l'aide du logiciel de configuration de l'UC.

L'horloge interne est conçue pour gérer le passage d'un mois à l'autre et d'une année à l'autre. Elle tient également compte des années bissextiles jusqu'en 2079.

Temporisateur chien de garde

Le temporisateur chien de garde de l'API Série 90-30 est conçu pour tenir compte des conditions de panne catastrophique qui peuvent provoquer un cycle anormalement long. La temporisation du temporisateur chien de garde est de 200 millisecondes (500 millisecondes sur les UC 351 et 352) ; cette valeur étant fixe, elle ne peut pas être modifiée. Le temporisateur chien de garde démarre toujours à zéro au début de chaque cycle.

En cas de dépassement de durée du chien de garde, le voyant OK s'éteint ; l'UC est réinitialisée puis mise hors fonction, et les sorties prennent leur état par défaut. Toute communication devient impossible et tous les microprocesseurs de toutes les cartes sont stoppés. La reprise s'effectue en remettant sous tension le bac contenant l'UC.

Sur les UC 90-20 ainsi que les modèles 340 et supérieurs, la temporisation chien de garde réinitialise l'UC, exécute la logique de mise sous tension, génère un défaut de chien de garde et passe au mode STOP.

Temporisateur de cycle constant

Le temporisateur de cycle constant contrôle la durée d'un cycle de programme lorsque l'API Série 90-30 fonctionne en mode **TEMPS DE CYCLE CONSTANT**. Dans ce mode de fonctionnement, chaque cycle est exécuté pendant une durée identique. En règle générale, pour la plupart des programmes d'application, l'acquisition des entrées, la scrutation de la logique du programme d'application et la restitution des sorties n'exigent pas exactement le même temps d'exécution dans chaque cycle. La durée du temporisateur de cycle constant est paramétrée par la console de programmation entre 5 et 200 ms (la durée par défaut est de 100 ms).

Si le temporisateur de cycle constant expire avant la fin du cycle et que le cycle précédent n'a pas connu de dépassement de durée, l'API place une alarme de dépassement de cycle dans sa table des défauts. Au début du cycle suivant, l'API met à "1" le contact de défaut OV_SWP (dépassement de cycle). Le contact OV_SWP est remis à "0" lorsque l'API n'est plus en mode **TEMPS DE CYCLE** constant ou si le temps du dernier cycle n'a pas excédé la durée impartie par le temporisateur de cycle constant.

Contacts d'impulsions d'horloge

L'API Série 90-30 offre quatre contacts d'impulsion d'horloge de 0,01 s, 0,1 s, 1 s et 1 min. L'état de ces contacts reste inchangé pendant l'exécution du cycle. Ils offrent une impulsion ayant une durée égale d'activation et de désactivation. Ces contacts sont appelés T_10MS (0,01 s), T_100MS (0,1 s), T_SEC (1 s) et T_MIN (1 min.).

Le chronogramme suivant représente le temps d'ouverture/fermeture de ces contacts.

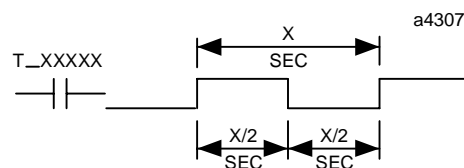


Figure 2-6. Chronogramme des contacts d'impulsions d'horloge

Section 5 : Sécurité du système

Sur les API Série 90-30, Série 90-20 et Micro, la sécurité est conçue pour interdire toute modification non autorisée du contenu d'un API. L'API offre quatre niveaux de sécurité. Le premier, qui est toujours disponible, permet seulement de lire les données de l'API et interdit toute modification du programme d'application. Les trois autres niveaux offrent un accès protégé par mot de passe.

Chaque niveau d'autorisation offre un plus grand nombre de possibilités que le(s) niveau(x) inférieur(s), cumulant à chaque fois les autorisations attribuées aux niveaux inférieurs. Les niveaux et leurs autorisations sont :

Niveau d'autorisation	Description
Niveau 1	Toutes les données peuvent être lues, à l'exception des mots de passe. Cela inclut toutes les mémoires de données (%I, %Q, %AQ, %R, etc.), les tables des défauts et tous les types de blocs de programme (données, valeur et constante). Aucune valeur ne peut être modifiée dans l'API.
Niveau 2	Ce niveau autorise l'accès en écriture aux emplacements mémoire (%I, %R, etc.).
Niveau 3	Ce niveau autorise l'accès en écriture au programme d'application en mode STOP seulement.
Niveau 4	Niveau par défaut des systèmes n'ayant aucun mot de passe défini. Le niveau par défaut d'un système comportant des mots de passe est le niveau non protégé le plus élevé. Ce niveau le plus élevé autorise les accès en lecture et écriture à toutes les mémoires ainsi qu'aux mots de passe dans les modes RUN et STOP . (Les données de configuration ne peuvent pas être modifiées en mode RUN .)

Mots de passe

Il existe un mot de passe pour chaque niveau d'autorisation dans l'API. (Aucun mot de passe ne peut être défini par l'accès de niveau 1.) Chaque mot de passe peut être unique, mais vous pouvez utiliser le même pour plusieurs niveaux. Les mots de passe comportent de 1 à 4 caractères ASCII. Ils ne peuvent être entrés ou modifiés qu'à l'aide du logiciel de programmation ou de la mini-console de programmation.

Un changement de niveau d'autorisation ne peut être effectué que si la liaison entre l'API et la console de programmation est correcte. Il n'est pas nécessaire qu'il y ait une activité, mais la liaison ne doit pas être interrompue. Si la liaison est interrompue pendant 15 minutes, le niveau d'accès retourne au niveau non protégé le plus élevé.

Lors de la connexion de l'API, le logiciel de programmation demande à l'API le degré de protection de chaque niveau d'autorisation. Le logiciel de programmation demande ensuite à l'API de passer au niveau non protégé le plus élevé, permettant ainsi au programmeur d'accéder au niveau non protégé le plus élevé sans avoir à demander un niveau particulier. Lorsque la HHP est connectée à l'API, ce dernier retourne au niveau non protégé le plus élevé.

Demandses de changement de niveau d'autorisation

Pour demander un changement de niveau d'autorisation, la console de programmation transmet le nouveau niveau d'autorisation et le mot de passe correspondant. Ce changement de niveau d'autorisation est refusé si le mot de passe envoyé par la console de programmation ne correspond pas au mot de passe du niveau demandé, stocké dans la table d'accès aux mots de passe de l'API. Le niveau d'autorisation actuel est conservé et aucun changement ne se produit. Si vous tentez d'accéder ou de modifier des informations dans l'API en utilisant la HHP sans le niveau d'autorisation correct, la HHP répond par un message d'erreur indiquant que l'accès est refusé.

Verrouillage/déverrouillage des sous-programmes

Les blocs de sous-programme peuvent être verrouillés ou déverrouillés en utilisant la fonction de verrouillage de bloc du logiciel de programmation. Deux types de verrouillage sont disponibles :

Type de verrou	Description
Lecture	Une fois ce verrou choisi, vous ne pouvez pas visualiser le sous-programme.
Edition	Une fois ce verrou choisi, les informations dans le sous-programme ne peuvent pas être modifiées mais peuvent être lues.

Un sous-programme précédemment verrouillé en Lecture ou Edition peut être déverrouillé dans l'éditeur de déclaration de blocs sauf s'il est verrouillé en permanence en Lecture ou Edition.

Une fonction Recherche ou Recherche/Remplace peut être effectuée sur un sous-programme verrouillé en Lecture. Si la cible de la recherche est trouvée dans le sous-programme verrouillé en Lecture, l'un des messages suivants s'affiche, à la place de la logique :

Trouvé dans bloc verrouillé <nom_bloc> (Continuer/Quitter)

ou

Ne peut pas écrire ds bloc verrouillé <nom_bloc> (Continuer/Quitter)

On peut continuer ou annuler la recherche.

Les fichiers contenant des sous-programmes verrouillés peuvent être vidés ou effacés. Si un fichier contient des sous-programmes verrouillés, ces blocs restent verrouillés lorsque les fonctions Copier, Sauvegarder et Restaurer fichier du logiciel de programmation sont utilisées.

Verrouillage permanent d'un sous-programme

En plus des verrous LECTURE et EDITION, il existe deux types de verrouillages permanents. Si un verrouillage LECT-Permanent est validé, tous les accès en lecture des sous-programmes sont refusés. Si un verrouillage EDIT-Permanent est validé, toute tentative pour éditer le bloc est refusée.

Attention

A la différence des verrous LECTURE et EDITION, les verrouillages permanents ne peuvent pas être supprimés une fois validés.

Quand un verrou EDIT-PERMANENT est défini, il ne peut être remplacé que par un verrou LECT-PERMANENT. Un verrou LECT-PERMANENT, par contre, ne peut être remplacé par aucun autre type de verrou.

Section 6 : Série 90-30 , Série 90-20 et Série Micro

L'API constitue l'interface entre l'API Série 90-30 et les équipements et périphériques fournis par l'utilisateur. Les E/S Série 90-30 sont appelées modules d'E/S modèle 30. Les modules d'E/S modèle 30 se raccordent directement dans les emplacements de la plaque de base de l'UC ou dans les emplacements des platines d'extension des API Série 90-30 modèles 331 ou supérieurs. Les modèles 331, 340 et 341 supportent jusqu'à 49 modules d'E/S modèle 30. Les modèles 351 et 352, par contre, supportent jusqu'à 79 modules d'E/S modèle 30. La platine à 5 emplacements de l'API Série 90-30 modèle 311 ou 313 supporte jusqu'à 5 modules d'E/S modèle 30 et la platine à 10 emplacements, jusqu'à 10 modules d'E/S modèle 30.

La structure d'E/S de l'API Série 90-30 est présentée dans la figure suivante.

E/S de l'API

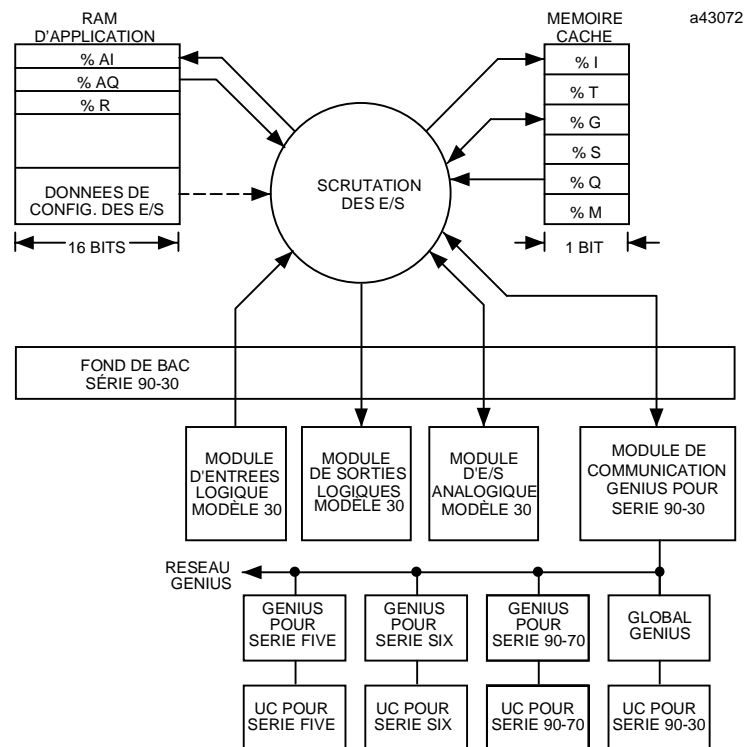


Figure 2-7. Structure d'E/S de l'API Série 90-30

Remarque

Le diagramme ci-dessus est spécifique à la structure d'E/S 90-30. Pour plus d'informations sur la structure d'E/S 90-20, voir le *Series 90™-20 Programmable Controller User's Manual* (GFK-0551) et pour plus d'informations sur la structure d'E/S de l'API Micro, voir le *Series 90™ Micro PLC User's Manual* (GFK-1065).

Modules d'E/S Modèle 30

Cinq types de modules d'E/S Modèle 30 sont disponibles : les modules d'entrées logiques, les modules de sorties logiques, les modules d'entrées analogiques, les modules de sorties analogiques, et les modules intelligents ou coprocesseurs. Le tableau suivant liste les modules d'E/S Modèle 30 par référence produit, nombre de points d'E/S, et les décrit brièvement.

Remarque

Au moment où ce manuel est imprimé, tous les modules d'E/S listés ci-dessous peuvent ne être pas disponibles. Pour connaître leur disponibilité actuelle, consulter votre distributeur local d'API GE Fanuc ou votre représentant GE Fanuc. Voir le document *GFK-0898 Série 90-30 Automate Programmable - Manuel d'installation* pour les spécifications et les informations de connexion de chaque module d'E/S Modèle 30.

Tableau 2-8. Modules d'E/S Modèle 30

Référence produit	Points	Description	Référence manuel
<i>Modules d'entrées logiques</i>			
IC693MDL230	8	120 Vca Isolé	GFK-0898
IC693MDL231	8	240 Vca Isolé	GFK-0898
IC693MDL240	16	120 Vca	GFK-0898
IC693MDL241	16	24 Vca/cc Logique positive/négative	GFK-0898
IC693MDL630	8	24 Vca/cc Logique positive	GFK-0898
IC693MDL632	8	125 Vcc Logique positive/négative	GFK-0898
IC693MDL633	8	24 Vcc Logique négative	GFK-0898
IC693MDL634	8	24 Vcc Logique positive/négative	GFK-0898
IC693MDL640	16	24 Vcc Logique positive	GFK-0898
IC693MDL641	16	24 Vcc Logique négative	GFK-0898
IC693MDL643	16	Vcc Logique positive, rapide	GFK-0898
IC693MDL644	16	24 Vcc Logique négative, rapide	GFK-0898
IC693MDL645	16	24 Vcc Logique positive/négative	GFK-0898
IC693MDL646	16	24 Vcc Logique positive/négative, rapide	GFK-0898
IC693MDL652	32	24 Vcc Logique positive/négative	GFK-0898
IC693MDL653	32	24 Vcc Logique positive/négative, rapide	GFK-0898
IC693MDL654	32	5/12 Vcc (TTL) Logique positive/négative	GFK-0898
IC693MDL655	32	24 Vcc Logique positive/négative	GFK-0898
IC693ACC300	8/16	Simulateur d'entrée	GFK-0898

Tableau 2-9. Modules d'E/S Modèle 30 - Suite

Référence produit	Points	Description	Référence manuel
<i>Modules de sorties logiques</i>			
IC693MDL310	12	120 Vca, 0,5A	GFK-0898
IC693MDL330	8	120/240 Vca, 2A	GFK-0898
IC693MDL340	16	120 Vca, 0,5 A	GFK-0898
IC693MDL390	5	120/240 Vcaisolated, 2 A	GFK-0898
IC693MDL730	8	12/24 Vcc Logique positive, 2 A	GFK-0898
IC693MDL731	8	12/24 Vcc Logique négative, 2 A	GFK-0898
IC693MDL732	8	12/24 Vcc Logique positive, 0,5 A	GFK-0898
IC693MDL733	8	12/24 Vcc Logique négative, 0,5 A	GFK-0898
IC693MDL734	6	125 Vcc Logique positive/négative, 2 A	GFK-0898
IC693MDL740	16	12/24 Vcc Logique positive, 0,5 A	GFK-0898
IC693MDL741	16	12/24 Vcc Logique négative, 0,5 A	GFK-0898
IC693MDL742	16	12/24 Vcc Logique positive, 1 A	GFK-0898
IC693MDL750	32	12/24 Vcc Logique négative	GFK-0898
IC693MDL751	32	12/24 Vcc Logique positive, 0,3A	GFK-0898
IC693MDL752	32	5/24 Vcc (TTL) Logique négative, 0,5A	GFK-0898
IC693MDL753	32	12/24 Vcc Logique positive/négative, 0,5A	GFK-0898
IC693MDL930	8	Relais, N.O. 4 A Isolé	GFK-0898
IC693MDL931	8	Relais, BC, Isolé	GFK-0898
IC693MDL940	16	Relais, N.O. 2 A	GFK-0898
<i>Modules d'entrées/sorties</i>			
IC693MDR390	8/8	Entrées 24 VDC Sorties à relais	GFK-0898
IC693MAR590	8/8	Entrées 120 VAC Sorties à relais	GFK-0898
<i>Modules analogiques</i>			
IC693ALG220	4 can.	Entrée analogique, Tension	GFK-0898
IC693ALG221	4 can.	Entrée analogique, Courant	GFK-0898
IC693ALG222	16	Entrée analogique, Tension	GFK-0898
IC693ALG223	16	Entrée analogique, Courant	GFK-0898
IC693ALG390	2 can.	Sortie analogique, Tension	GFK-0898
IC693ALG391	2 can.	Sortie analogique, Courant	GFK-0898
IC693ALG392	8 can.	Sortie analogique, Courant/Tension	GFK-0898
IC693ALG442	4/2	Entrée/sortie analogique combinée, Courant/Tension	GFK-0898

Tableau 2-9. Modules d'E/S Modèle 30 - Suite

Référence produit	Description	Référence manuel
<i>Modules optionnels</i>		
IC693APU300	Compteur rapide (HSC)	GFK-0293
IC693CMM311	Coprocésseur de communication (CMM)	GFK-0582
IC693PCM300	Coprocésseur programmable (PCM), 160K (35 K de programme utilisateur MegaBasic)	GFK-0255
IC693PCM301	Coprocésseur programmable (PCM), 192K Bytes (47 K de programme utilisateur MegaBasic)	GFK-0255
IC693PCM311	Coprocésseur programmable (PCM), 640K Bytes (190 K de programme utilisateur MegaBasic)	GFK-0255
IC693ADC311	Coprocésseur d'affichage alphanumérique	GFK-0521
IC693BEM331	Contrôleur de bus Genius	GFK-1034
IC693CMM301	Module de communication Genius (GCM)	GFK-0412
IC693CMM302	Module de communication Genius amélioré (GCM+)	GFK-0695
IC693BEM320	Module d'interface de lien d'E/S (esclave)	GFK-0631
IC693BEM321	Module d'interface de lien d'E/S (maître)	GFK-0823
IC693APU301	Module APM Motion Mate, Mode 1-Axis-Follower	GFK-0781
.. .. .	Module APM Motion Mate, Mode 1-Axis-Standard	GFK-0840
IC693APU302	Module APM Motion Mate, Mode 2-Axis-Follower	GFK-0781
.. .. .	Module APM Motion Mate, Mode 2-Axis-Standard	GFK-0840
IC693MCS001/2	Commande Power Mate J Motion (1 et 2 axes)	GFK-1256
IC693APU305	Module processeur d'E/S	GFK-1028
IC693CMM321	Communications Ethernet	GFK-1084

Formats des données d'E/S

Les données des E/S logiques sont stockées sous forme binaire dans la mémoire cache binaire (table des états). Les données des E/S analogiques sont stockées sous forme de mots et sont résidentes en mémoire dans une partie de la RAM d'application affectée à cet usage.

Conditions par défaut des modules de sortie Modèle 30

A la mise sous tension, les modules de sorties logiques Modèle 30 forcent par défaut leurs sorties à "0". Ils conservent cette condition par défaut jusqu'à la première restitution des sorties par l'API. Les modules de sorties analogiques peuvent être configurés à l'aide d'un cavalier situé sur le bornier du module, soit par défaut à "0" soit pour conserver leur dernier état. De plus, les modules de sorties analogiques peuvent être mis sous tension par une source d'alimentation externe, de telle sorte que, même si l'API n'est pas sous tension, le module de sorties analogiques continue à fonctionner dans son état choisi par défaut.

Données de diagnostic

Les bits de diagnostic sont disponibles à l'emplacement mémoire %S. Ils indiquent qu'un module d'E/S est perdu ou qu'il existe une incohérence dans la configuration des E/S. Les données de diagnostic ne sont pas disponibles pour les différents points d'E/S. Pour plus d'informations sur la gestion des défauts, voir le chapitre 3, "Description et correction des défauts".

Données globales

L'API Série 90-30 supporte le partage ultra-rapide des données entre plusieurs UC à l'aide des données globales. Le contrôleur de bus Genius, IC693BEM331 de l'UC (version 5 et ultérieures) ainsi que le module de communications Genius amélioré, IC693CMM302, peuvent transmettre jusqu'à 128 octets de données à d'autres API ou d'autres ordinateurs. Ils peuvent aussi recevoir jusqu'à 128 octets de chacun des 30 autres contrôleurs Genius maximum du réseau. Les données peuvent être transmises ou reçues dans n'importe quel type de mémoire et non uniquement les bits globaux %G. Le module de communications Genius original, IC693CMM301, était limité aux adresses %G fixes et ne pouvait échanger que 32 bits par adresse de bus série entre les SBA 16 et 23. Dans la mesure où le GCM amélioré possède des capacités 50 fois plus élevées, nous vous conseillons de ne plus utiliser cette carte d'origine.

Les données globales peuvent être partagées entre les API Série 90 existant sur le même réseau Genius. Il existe une méthode préconfigurée de partage des données globales. Aucune configuration n'est alors à effectuer par l'utilisateur.

Les données globales sont mises en œuvre par les modules de communication Genius en transmettant les données d'E/S d'après leurs adresses sur le réseau. Chaque module peut également lire les données globales transmises par sept autres modules de communication Genius.

Modules d'E/S Modèle 20

Les modules d'E/S suivants sont disponibles pour l'API Série 90-20. Chaque module est listé par référence produit, nombre de points d'E/S, et décrit brièvement. Les E/S sont intégrées dans un bac, avec l'alimentation. Pour les spécifications et les informations de connexion de chaque module, voir le chapitre 5 du *GFK-0551 Series 90-20 Programmable Controller User's Manual*.

Référence produit	Description	Points d'E/S
IC692MAA541	Module de base d'alimentation et d'E/S, 120 Vca Entrée/120 Vca Sortie/Alimentation 120 Vca	16 E/12 S
IC692MDR541	Module de base d'Alimentation et d'E/S 24 Vcc Entrée/Sortie à relais/Alimentation 120 Vca	16 E/12 S
IC692MDR741	Module de base d'Alimentation et d'E/S 24 Vcc Entrée/Sortie à relais/Alimentation 240 Vca	16 E/12 S
IC692CPU211	Module d'UC, Modèle 211	Non applicable

API Micro

Les API Série 90 Micro ci-dessous sont disponibles. Chaque API Micro est accompagnée de sa référence produit, du nombre de points d'E/S et d'une description succincte. L'UC, l'alimentation et les E/S font partie de la même unité. Pour plus d'informations sur le câblage et les caractéristiques de chaque module, voir le *Series 90 Micro Programmable Controller User's Manual*, GFK-1065.

Référence produit	Description	Points d'E/S
IC693UDR001	UC, alimentation et E/S (en une seule unité) Entrée CC Micro à 14 points/Sortie relais, alimentation secteur	8 E/6 S
IC693UDR002	UC, alimentation et E/S (en une seule unité) Entrée CC Micro à 14 points/Sortie relais, alimentation CC	8 E/6 S
IC693UAA003	UC, alimentation et E/S (en une seule unité) Entrée CA Micro à 14 points/Sortie CA, alimentation secteur	8 E/6 S
IC693UDR005	UC, alimentation et E/S (en une seule unité) Entrée CC Micro à 28 points/Sortie relais, alimentation secteur	16 E/11 S relais/1 S CC
IC693UAL006	UC, alimentation et E/S (en une seule unité) Entrée CC Micro à 23 points/Sortie CC, alimentation secteur	1 S CC/9 S relais/2 E analogiques/1 S analogique
IC693UAA007	UC, alimentation et E/S (en une seule unité) Entrée CA Micro à 28 points/Sortie CA, alimentation secteur	16 E/12 S
IC693UDR010	UC, alimentation et E/S (en une seule unité) Entrée CC Micro à 28 points/Sortie CC, alimentation CC	16 E CC/1 S CC/11 S relais
IC693UEX011	Unité d'extension à 14 points-Entrée CC à 14 points/Sortie relais, alimentation secteur	8 E/6 S

Chapitre 3

Description et correction des défauts

Ce chapitre permet de rechercher les défauts des API Série 90-30 et de les corriger. Il explique les descriptions des défauts qui apparaissent dans la table des défauts automate, et les catégories de défauts, qui apparaissent dans la table des défauts d'E/S.

Chaque explication de défaut, dans ce chapitre, indique la description du défaut pour la table des défauts automate, ou la catégorie de défaut pour la table des défauts d'E/S. Rechercher la description du défaut ou la catégorie de défaut correspondant à l'entrée de la table de défauts applicable (automate ou E/S), affichée sur votre console de programmation. Ci-dessous se trouve une description de la cause du défaut ainsi que les instructions à suivre pour le corriger.

Le chapitre 3 couvre les sujets suivants :

Section	Titre	Description	Page
1	Gestion des défauts	Décrit le type de défaut pouvant se produire dans l'API Série 90-30 ou Série 90-20 et comment ils sont affichés dans les tables des défauts. La description des écrans des tables des défauts (automate et E/S) est également incluse.	3-2
2	Description de la Table des défauts automate	Décrit chaque défaut de l'API et indique comment le corriger.	3-7
3	Description de la Table des défauts d'E/S	Décrit les catégories de défauts "module d'E/S perdu" et "module d'E/S additionnel".	3-17

Section 1 : *Gestion des défauts*

Des défauts se produisent dans l'API Série 90-30 lorsque certaines défaillances ou conditions affectent le fonctionnement et les performances du système. Ces conditions, telles qu'un module d'E/S ou un bac perdu, peuvent affecter la capacité de l'API à contrôler une machine ou un procédé. D'autres conditions peuvent signaler qu'un nouveau module est connecté et qu'il est à présent disponible. Ou bien, ces conditions peuvent servir seulement d'avertissement, tel qu'un signal de "seuil pile bas", indiquant que la pile de sauvegarde de la mémoire doit être remplacée.

Processeur d'alarmes

La condition ou la panne elle-même est appelée défaut. Lorsqu'un défaut est reçu et traité par l'UC, il est appelé "alarme". Le logiciel de l'UC qui traite ces conditions est appelé Processeur d'alarmes. L'interface entre l'utilisateur et le Processeur d'alarmes est le logiciel de programmation. Tout défaut détecté est enregistré dans une table des défauts et affiché dans la table des défauts automate ou la table des défauts d'E/S, selon le cas.

Catégories de défauts

Les API Série 90-30 et Série 90-20 détectent plusieurs catégories de défauts : les défauts internes, les défauts externes et les défauts de fonctionnement.

Classe de défauts	Exemples
Défauts internes	Pas de réponse des modules. Seuil pile bas. Erreur de CHECKSUM de la mémoire.
Défauts externes des E/S	Bac ou module perdu. Bac ou module additionnel.
Défauts de fonctionnement	Défauts de communication. Défauts de configuration. Défauts d'accès par mot de passe.

Réponse du système aux défauts

Généralement, les défauts matériels exigent la mise hors service de l'API ou le défaut est toléré. Les défauts d'E/S peuvent être tolérés par l'API, mais ils peuvent ne pas être tolérés par l'application ou le procédé contrôlé. Les défauts de fonctionnement sont normalement tolérés. Les défauts de l'API Série 90-30 possèdent deux attributs :

Attribut	Description
Table des défauts concernée	Table des défauts d'E/S Table des défauts automate
Intervention défaut	Fatal Diagnostic Informationnel

Table des défauts

Deux tables des défauts sont mises à jour dans l'API pour l'enregistrement des défauts : la table des défauts d'E/S pour l'enregistrement des défauts associés aux modules d'E/S, et la table des défauts automate pour l'enregistrement de tous les autres défauts. Le tableau suivant liste les groupes de défauts, les interventions défaut, les tables de défauts concernées, et le "nom" des points %S logiques du système affectés.

Tableau 3-1. Sommaire des défauts

Groupe de défauts	Intervention défaut	Table des défauts	Références spéciales des défauts discrets			
			ioflt	anyflt	iospres	losiom
Module d'E/S perdu ou manquant	Diagnostic	E/S	ioflt	anyflt	iospres	losiom
Module coprocesseur perdu ou manquant	Diagnostic	Automate	syflt	anyflt	sypres	losio
Configuration système incohérente	Fatal	Automate	syflt	anyflt	sypres	cfgmm
Défaut matériel de l'UC d'API	Fatal	Automate	syflt	anyflt	sypres	hrdcpu
Défaut de CHECKSUM	Fatal	Automate	syflt	anyflt	sypres	pbsum
Seuil pile bas	Diagnostic	Automate	syflt	anyflt	sypres	lowbat
Table des défauts automate pleine	Diagnostic	—	syfull			
Table des défauts d'E/S pleine	Diagnostic	—	iofull			
Défaut de l'application	Diagnostic	Automate	syflt	anyflt	sypres	aplflt
Pas de programme utilisateur	Informationnel	Automate	syflt	anyflt	sypres	noprog
RAM utilisateur altérée	Fatal	Automate	syflt	anyflt	sypres	badram
Défaut d'accès par mot de passe	Diagnostic	Automate	syflt	anyflt	sypres	badpwd
Défaut logiciel de l'API	Fatal	Automate	syflt	anyflt	sypres	sftcpu
Défaut de stockage d'API	Fatal	Automate	syflt	anyflt	sypres	storer
Dépassement du temps de cycle constant	Diagnostic	Automate	syflt	anyflt	sypres	ovswp
Défaut d'API inconnu	Fatal	Automate	syflt	anyflt	sypres	
Défaut d'E/S inconnu	Fatal	E/S	ioflt	anyflt	iospres	

Intervention défaut

Pour faciliter la recherche des défauts, deux tables des défauts sont prévues, évitant ainsi que l'une d'entre elles ne devienne trop longue. Ces tables sont : la table des défauts automate et la table des défauts d'E/S.

Lorsqu'un défaut de type fatal se produit, celui-ci est enregistré dans la table appropriée, les variables de diagnostic sont mises à "1", et le système est arrêté. Les défauts de type diagnostic sont enregistrés dans la table appropriée, et toute variable de diagnostic est mise à "1". Les défauts de type informationnel sont seulement enregistrés dans la table appropriée.

Les interventions défaut possibles sont indiquées dans la table suivante.

Tableau 3-2. Interventions défaut

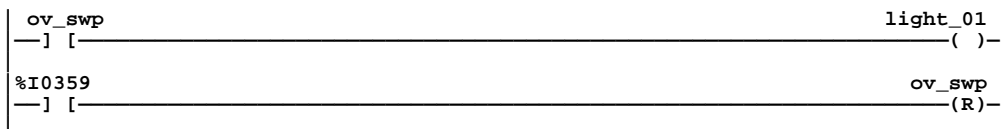
Intervention défaut	Réponse de l'UC
Fatal	Enregistre le défaut dans la table des défauts. Met à "1" les références de défauts. Passe en mode STOP .
Diagnostic	Enregistre le défaut dans la table des défauts. Met à "1" les références de défauts.
Informationnel	Enregistre le défaut dans la table des défauts.

Lorsqu'un défaut est détecté, l'UC utilise l'intervention défaut pour ce défaut. Les interventions défaut ne sont pas configurables dans l'API Série 90-30.

Références de défauts

Les références de défauts pour les API Série 90-30 et Série 90-20 sont d'un seul type : les références récapitulatives de défaut. Les références récapitulatives de défaut sont mises à "1" pour indiquer le défaut qui s'est produit. La référence de défaut est conservée jusqu'à ce que l'API soit réinitialisé ou que le programme d'application la mette à "0".

L'exemple ci-dessous montre un bit de défaut mis à "1" puis remis à "0". Dans cet exemple, la bobine light_01 (voyant) est mise à "0" lorsqu'un dépassement de cycle (OVSWP) se produit ; le voyant et le contact OV_SWP restent à "1" jusqu'à ce que le contact %I0359 soit fermé.



Définitions des références de défauts

Le processeur d'alarmes conserve l'état des 128 bits logiques système dans l'emplacement mémoire %S. Ces références de défauts peuvent être utilisées pour indiquer l'endroit où le défaut s'est produit et le type de défaut dont il s'agit. Les références de défauts sont attribuées aux emplacements %S, %SA, %SB et %SC, et elles ont chacune un symbole. Ces références peuvent être utilisées dans le programme d'application selon les besoins. Pour obtenir la liste des références d'état système, voir le chapitre 2, "Fonctionnement du système".

Effets secondaires des défauts

Deux défauts décrits précédemment ont des effets secondaires qui leur sont associés. Ces effets sont décrits dans le tableau suivant.

Effet secondaire	Description
Défaut logiciel de l'UC de l'API.	Chaque fois qu'un défaut logiciel de l'UC de l'API est enregistré, l'UC Série 90-30 ou Série 90-20 passe immédiatement en mode spécial de CYCLE D'ERREUR . Aucune activité n'est permise dans ce mode. La seule méthode pour supprimer cette condition est de réinitialiser l'API (c'est-à-dire de le mettre hors/sous tension).
Défaut pendant la période de stockage	Au cours d'un stockage (blocs de programme et autres données démarrant par l'instruction spéciale de début de séquence et se terminant par l'instruction de fin de séquence), si la liaison avec la miniconsole de programmation effectuant le stockage est interrompue, ou si un autre défaut se produit, interrompant le chargement, le Défaut de stockage est enregistré. Tant que ce défaut est présent dans le système, l'API ne passe pas en mode RUN .

Affichage de la Table des défauts de l'API

La table des défauts automate affiche les défauts de l'API, tels que les violations de mot de passe, les incohérences dans la configuration du système, les erreurs de parité et les erreurs de communication.

Le logiciel de programmation peut être dans n'importe quel mode de fonctionnement. S'il est en mode **OFFLINE**, aucun défaut n'est affiché. En mode **ONLINE** ou **MONITOR**, par contre, les données de défaut de l'API sont affichées. Enfin, en mode **ONLINE**, les défauts peuvent être effacés (cette fonction peut cependant être protégée par mot de passe).

Une fois mis à "0", les défauts encore présents ne sont pas réenregistrés dans la table (à l'exception du défaut "Seuil pile bas").

Affichage de la Table des défauts d'E/S

La table des défauts d'E/S affiche les défauts d'E/S tels que les défauts de circuit, les conflits d'adresse, les circuits forcés et les défauts du bus d'E/S. Par exemple :

Le logiciel de programmation peut être dans n'importe quel mode de fonctionnement. S'il est en mode **OFFLINE**, aucun défaut n'est affiché. En mode **ONLINE** ou **MONITOR**, les données relatives aux défauts d'E/S sont affichées. Enfin, en mode **ONLINE**, les défauts peuvent être effacés (cette fonction peut cependant être protégée par mot de passe).

Une fois mis à "0", les défauts encore présents ne sont pas réenregistrés dans la table.

Accéder aux informations additionnelles concernant les défauts

Les tables de défauts contiennent des informations élémentaires sur les défauts. D'autres informations relatives à chaque défaut peuvent être affichées par l'intermédiaire du logiciel de programmation. Ce dernier peut aussi fournir un affichage hexadécimal du défaut.

La dernière entrée, Correction, pour chaque description de défaut dans ce chapitre liste la/les mesure(s) corrective(s). A noter que la mesure corrective pour certains défauts inclut le message :

Afficher la Table des défauts automate sur la miniconsole de programmation. Prendre contact avec le service assistance-client de GE Fanuc, et lui communiquer toutes les informations indiquées par l'entrée de la Table des défauts.

Ce second message signifie que vous devez indiquer au Service assistance-client les informations lisibles directement dans la table des défauts et les informations hexadécimales. Le personnel du Service assistance-client vous indiquera alors la procédure à suivre.

Section 2 : Description du défaut dans la Table des défauts automate

Chaque explication de défaut contient une description du défaut et les mesures correctives le concernant. Plusieurs descriptions de défauts ont des causes multiples. Dans ce cas, le code d'erreur, affiché avec les informations additionnelles du défaut, sont utilisées pour distinguer différentes conditions de défaut partageant la même description. Le code d'erreur est indiqué par deux chiffres hexadécimaux dans le cinquième groupe de nombres, comme indiqué dans l'exemple suivant.

01	000000	01030100	0902	0200	000000000000
				Code d'erreur (deux premiers chiffres hex. dans le cinquième groupe)	

Certains défauts peuvent résulter d'une défaillance de la RAM de l'UC de l'API. Ces même défauts peuvent également se produire à la suite d'une coupure d'alimentation du système si la tension de la pile n'est pas suffisante pour sauvegarder la mémoire. Pour éviter une utilisation en double des instructions, lorsqu'une altération de mémoire est susceptible de provoquer une erreur, la correction indique simplement :

Corriger la mémoire altérée.

Autrement dit, vous devez :

1. Remplacer la pile si le système a été mis hors tension. La tension de la pile peut être suffisante pour conserver le contenu de la mémoire.
2. Remplacer l'UC de l'API. Les circuits intégrés de l'UC de l'API sont peut-être défectueux.

Le tableau suivant permet de trouver rapidement la description particulière d'un défaut d'API décrit dans cette section. Chaque entrée est listée telle qu'elle apparaît dans l'écran de la console de programmation.

Description des défauts	Numéro de page
Coprocasseur perdu ou manquant.	3-8
Coprocasseur réinitialisé, additionnel ou non configuré.	3-8
Incohérence dans la configuration du système.	3-8
Défaut logiciel du coprocasseur.	3-10
Défaut de CHECKSUM des blocs de programme.	3-10
Signal "seuil pile bas".	3-10
Dépassement du temps de cycle constant.	3-11
Défaut de l'application.	3-11
Pas de programme utilisateur présent.	3-12
Programme utilisateur altéré à la mise sous tension.	3-13
Echec d'accès par mot de passe.	3-13
Défaut du système d'exploitation de l'UC de l'API.	3-13
Défaut de communication pendant le stockage.	3-16

Intervention défaut

A la suite des défauts de type "**fatal**", l'API entre dans un type de mode **STOP** à la fin du cycle dans lequel l'erreur s'est produite. Les défauts de type "**diagnostic**" sont enregistrés et les contacts de défaut correspondants sont fermés. Les défauts de type "**informationnel**" sont simplement enregistrés dans la table des défauts automate.

Coprocasseur perdu ou manquant

Le groupe de défauts **Coprocasseur perdu ou manquant** intervient lorsqu'un module PCM, CMM ou ADC ne répond pas. Le défaut peut se produire à la mise sous tension si le module est manquant, ou pendant le fonctionnement si celui-ci ne répond pas. L'intervention défaut pour ce groupe est de type **Diagnostic**.

Code d'erreur :	1, 42
Nom :	Echec de la réinitialisation, par l'utilisateur, du coprocasseur
Description :	L'UC de l'API ne peut pas rétablir la communication avec le coprocasseur après sa réinitialisation utilisateur.
Correction :	<ol style="list-style-type: none"> (1) Essayer une seconde fois la réinitialisation utilisateur. (2) Remplacer le coprocasseur. (3) Mettre le système hors tension. Vérifier si le PCM est correctement placé dans le bac et si tous les câbles sont connectés et positionnés correctement. (4) Remplacer les câbles.
Code d'erreur :	Tous les autres
Nom :	Défaut du module pendant la configuration.
Description :	Le système d'exploitation de l'API génère cette erreur lorsqu'un module tombe en panne au cours d'une mise sous tension ou d'un stockage de la configuration.
Correction :	<ol style="list-style-type: none"> (1) Mettre le système hors tension. Remplacer le module situé dans ce bac et cet emplacement.

Coprocasseur réinitialisé, additionnel ou non configuré

Le Groupe de défauts **Coprocasseur réinitialisé, additionnel ou non configuré** intervient lorsqu'un coprocasseur (PCM, ADC, etc.) est mis sous tension, réinitialisé, ou si un module se trouvant dans le bac n'est pas spécifié dans la configuration. L'intervention défaut pour ce groupe est de type **Diagnostic**. Trois octets de données spécifiques fournissent un complément d'information concernant le défaut.

Correction :	<ol style="list-style-type: none"> (1) Mettre à jour le fichier de configuration en spécifiant le module. (2) Retirer le module du système.
---------------------	---

Incohérence dans la configuration système

Le Groupe de défauts **Incohérence dans la configuration** intervient lorsque le module occupant un emplacement est différent de celui spécifié dans le fichier de configuration. L'intervention défaut est de type **Fatal**.

Code d'erreur :	1
Nom :	Incohérence dans la configuration système
Description :	Le système d'exploitation (configurateur système) de l'API génère ce défaut lorsque le module occupant un emplacement n'est pas du même type que celui spécifié dans le fichier de configuration pour cet emplacement ou lorsque le type de bac configuré ne concorde pas avec celui qui est présent.
Correction :	Identifier l'incohérence et reconfigurer le module ou le bac.
Code d'erreur :	6
Nom :	Incohérence dans la configuration système
Description :	Ce code d'erreur est identique au code 1 car ce défaut survient lorsque le module occupant un emplacement n'est pas du même type que celui spécifié dans le fichier de configuration pour cet emplacement ou lorsque le type de bac configuré ne concorde pas avec celui qui est présent.
Correction :	Identifier l'incohérence et reconfigurer le module ou le bac.
Code d'erreur :	18
Nom :	Matériel non supporté
Description :	Un PCM ou un module de type PCM est présent dans 311, 313 ou 323 ou bien dans un bac d'extension.
Correction :	Remédier physiquement à la situation en enlevant le PCM ou le module de type PCM ou en installant une UC qui supporte le PCM.
Code d'erreur :	26
Nom :	Le module est occupé et n'a pas encore accepté la configuration
Description :	Le module ne peut pas accepter de nouvelle configuration pour le moment, car il est occupé avec un autre processus.
Correction :	Attendre que le module termine l'opération en cours et restaurer la configuration.
Code d'erreur :	51
Nom :	Exécution de l'instruction END à partir d'une action SFC
Description :	Ce défaut est dû au placement d'une instruction END dans la logique SFC ou la logique appelée par celle-ci.
Correction :	Supprimer l'instruction END de la logique SFC ou de la logique appelée par celle-ci.

Défaut logiciel du coprocesseur

Le Groupe de défauts **Défaut logiciel du coprocesseur** intervient lorsqu'un défaut logiciel non réparable se produit sur un module PCM ou ADC. L'intervention défaut pour ce groupe est de type **Fatal**.

Code d'erreur :	Tous
Nom :	La fréquence COMMREQ est trop élevée.
Description :	Les demandes de communication sont envoyées à un module plus rapidement qu'il ne peut les traiter.
Correction :	Modifier le programme d'API pour que les COMMREQ soit envoyées à ce module à une vitesse inférieure.

Défaut de CHECKSUM des blocs de programme

Le Groupe de défauts **Défaut de CHECKSUM des blocs de programme** intervient lorsque l'UC de l'API détecte des conditions d'erreur dans les blocs de programme reçus par l'API. Il intervient également lorsque l'UC de l'API détecte des erreurs de CHECKSUM pendant la vérification de la mémoire à la mise sous tension, ou au cours du contrôle de fond du mode **RUN**. L'intervention défaut pour ce groupe est de type **Fatal**.

Code d'erreur :	Tous
Nom :	Défaut de CHECKSUM des blocs de programme.
Description :	Le système (logiciel) d'exploitation de l'API génère cette erreur lorsqu'un bloc de programme est altéré.
Correction :	<ol style="list-style-type: none"> (1) Réinitialiser la mémoire de l'API et réessayer le stockage. (2) Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-AP de GE Fanuc, et lui communiquer toutes les informations contenues dans l'entrée de défaut.

Signal "seuil pile bas"

Le Groupe de défauts **Signal seuil pile bas** intervient lorsque l'UC de l'API détecte une pile faible dans l'Alimentation d'API ou dans un module, tel que le PCM, et signale une condition de "seuil pile bas". L'intervention défaut pour ce groupe est de type **Diagnostic**.

Code d'erreur :	0
Nom :	Signal de pile déchargée
Description :	La pile du module d'UC (ou d'un autre module équipé d'une pile) est déchargée.
Correction :	Remplacer la pile. Ne pas mettre le bac hors tension.
Code d'erreur :	1
Nom :	Signal de seuil pile bas
Description :	La pile du module d'UC ou d'un autre module est faible.
Correction :	Remplacer la pile. Ne pas mettre le bac hors tension.

Dépassement du temps de cycle constant

Le Groupe de défauts **Dépassement du temps de cycle constant** intervient lorsque l'UC de l'API fonctionne en mode **CYCLE CONSTANT** et détecte un dépassement du temporisateur de cycle constant. Les données de défaut additionnelles contiennent le temps réel du cycle dans les deux premiers octets et le nom du programme dans les huit octets suivants. L'intervention défaut pour ce groupe est **Diagnostic**.

- | | |
|---------------------|--|
| Correction : | (1) Augmenter le temps de cycle constant. |
| | (2) Supprimer la logique du programme d'application. |

Défaut d'application

Le Groupe de défauts **Défaut d'application** intervient lorsque l'UC de l'API détecte un défaut dans le programme utilisateur. L'intervention défaut pour ce groupe est de type **Diagnostic** ou de type **Fatal** en présence de l'erreur Dépassement de la pile d'appels de sous-programme.

Code d'erreur :	7
Nom :	Dépassement de la pile d'appels de sous-programme.
Description :	La profondeur d'appel est limitée à 8 niveaux : un sous-programme peut appeler un autre sous-programme qui, à son tour, peut appeler un autre sous-programme jusqu'à ce que le niveau 8 soit atteint.
Correction :	Modifier le programme pour que la profondeur d'appel de sous-programme n'excède pas 8 niveaux.
Code d'erreur :	1B
Nom :	Comm Req non traitée en raison des limitations de mémoire de l'API
Description :	Des demandes de communication sans attente peuvent être placées dans la file d'attente à une vitesse supérieure à celle du traitement (notamment une par cycle). Dans ce cas, lorsque l'accumulation des demandes de communications est supérieure à la quantité de mémoire minimale disponible de l'API, les demandes de communication ne sont pas traitées et des défauts sont générés.
Correction :	Réduire le nombre de demandes de communication ou la quantité de courrier échangée au sein du système.
Code d'erreur :	5A
Nom :	Demande d'arrêt de l'utilisateur
Description :	Le système d'exploitation de l'API (blocs fonctionnels) génère cette alarme informationnelle lorsque la demande de service 13 (arrêt de l'utilisateur) est exécutée dans le programme d'application.
Correction :	Aucune. Cette alarme est générée à titre d'information uniquement.

Pas de programme utilisateur présent

Le Groupe de défauts **Pas de programme utilisateur présent** intervient lorsque l'UC de l'API a reçu l'instruction de passer du mode **STOP** au mode **RUN**, ou en cas de stockage dans l'API alors que celui-ci ne contient aucun programme utilisateur. L'UC de l'API détecte l'absence d'un programme utilisateur à la mise sous tension. L'intervention défaut pour ce groupe est de type **Informationnel**.

Correction :	Charger un programme d'application avant de tenter de passer en mode RUN .
---------------------	---

Programme utilisateur altéré à la mise sous tension

Le Groupe de défauts **Programme utilisateur altéré à la mise sous tension** intervient lorsque l'UC de l'API détecte que la RAM utilisateur est altérée. L'UC de l'API reste en mode **STOP** jusqu'à ce qu'un programme utilisateur et un fichier de configuration valides soient chargés. L'intervention défaut pour ce groupe est de type **Fatal**.

Code d'erreur :	1
Nom :	Altération de la RAM utilisateur à la mise sous tension.
Description :	Le système (logiciel) d'exploitation de l'API génère cette erreur lorsqu'il détecte une altération de la RAM utilisateur à la mise sous tension.
Correction :	<ol style="list-style-type: none"> (1) Recharger le fichier de configuration, le programme utilisateur, et les références (s'il en existe). (2) Remplacer la pile de l'UC de l'API. (3) Remplacer la carte d'extension de mémoire de l'UC de l'API. (4) Remplacer l'UC de l'API.
Code d'erreur :	2
Nom :	Détection d'un code opération booléen interdit
Description :	Le système (logiciel) d'exploitation de l'API génère cette erreur lorsqu'il détecte une instruction erronée dans le programme utilisateur.
Correction :	<ol style="list-style-type: none"> (1) Restaurer le programme utilisateur et les références (s'il en existe). (2) Remplacer la carte d'extension de mémoire de l'UC de l'API. (3) Remplacer l'UC de l'API.

Echec de l'accès par mot de passe

Le Groupe de défauts **Défaut d'accès par mot de passe** intervient lorsque l'UC de l'API reçoit l'instruction de modifier un niveau d'accès et que le mot de passe inclus dans la demande n'est pas valide pour ce niveau. L'intervention défaut pour ce groupe est de type **Informationnel**.

Correction :	Réessayer la demande en utilisant le mot de passe correct.
---------------------	--

Défaut du système d'exploitation de l'UC de l'API

Les défauts dans le Groupe de défauts **Défaut du système d'exploitation de l'UC de l'API** sont générés par le logiciel d'exploitation de l'UC des API Série 90-30 ou Série 90-20. Ils peuvent se produire en différents points du fonctionnement du système. Lorsqu'un défaut de type **Fatal** se produit, l'UC de l'API passe **immédiatement** en mode spécial de **CYCLE D'ERREUR**. La seule activité autorisée lorsque l'API est dans ce mode est la communication avec la console de programmation. Aucune activité n'est autorisée lorsque l'API se trouve dans ce mode. La seule façon de supprimer cette condition est de réinitialiser l'alimentation de l'API. L'intervention défaut dans ce groupe est de type **Fatal**.

Code d'erreur :	1 à B
Nom :	La mémoire programme n'a pu être allouée.
Description :	Le système d'exploitation (gestionnaire mémoire) de l'API génère ces erreurs lorsque le logiciel demande au gestionnaire mémoire d'affecter ou de désaffecter un ou plusieurs blocs de mémoire de la RAM utilisateur qui sont interdits. Ces erreurs ne doivent pas se produire dans un système de production.
Correction :	Affiche la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et communiquer toutes les informations contenues dans l'entrée de défaut.
Code d'erreur :	D
Nom :	Mémoire système non disponible
Description :	Le système d'exploitation (scrutation des E/S) de l'API génère cette erreur lorsque le gestionnaire mémoire refuse sa demande de bloc de mémoire système du fait qu'aucun segment de mémoire système n'est disponible. Le défaut est de type informationnel si l'erreur se produit pendant l'exécution d'un bloc fonctionnel DO I/O (rafraîch. imm. des E/S). Il est de type fatal s'il se produit lors de l'initialisation à la mise sous tension ou lors de la configuration automatique.
Correction :	Affiche la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut.
Code d'erreur :	E
Nom :	La mémoire système n'a pu être libérée.
Description :	Le système d'exploitation (scrutation des E/S) de l'API génère cette erreur lorsqu'il demande au gestionnaire mémoire de désaffecter un bloc de mémoire système et que sa demande échoue. Cette erreur se produit uniquement pendant l'exécution d'un bloc fonctionnel DO E/S (rafraîch. imm. des E/S).
Correction :	(1) Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut. (2) Corriger la mémoire altérée.
Code d'erreur :	10
Nom :	Demande de scrutation invalide du scrutateur d'E/S
Description :	Le système d'exploitation (scrutation des E/S) de l'API génère cette erreur lorsque le système d'exploitation ou la scrutation du bloc fonctionnel DO I/O (rafraîch. imm. des E/S) ne demande ni scrutation complète ni partielle des E/S. Cela ne doit pas se produire dans un système de production.
Correction :	Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut.

Code d'erreur :	13
Nom :	Erreur du système d'exploitation de l'API
Description :	Le système d'exploitation de l'API génère cette erreur lorsqu'il rencontre certains problèmes. Cela ne doit pas se produire dans un système de production.
Correction :	(1) Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut. (2) Corriger la mémoire altérée.
Code d'erreur :	14, 27
Nom :	Altération de la mémoire programme de l'API.
Description :	Le système d'exploitation de l'API génère ces erreurs lorsqu'il rencontre certains problèmes. Cela <i>ne doit pas</i> se produire dans un système de production.
Correction :	(1) Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut. (2) Corriger la mémoire altérée.
Code d'erreur :	27 à 4E
Nom :	Erreur du système d'exploitation de l'API.
Description :	Le système d'exploitation de l'API génère ces erreurs lorsqu'il rencontre certains problèmes. Cela <i>ne doit pas</i> se produire dans un système de production.
Correction :	Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut.
Code d'erreur :	4F
Nom :	Echec de la communication
Description :	Le système d'exploitation (processeur de demandes de service) de l'API génère cette erreur lorsqu'il tente de répondre à une demande exigeant la communication avec le fond de bac et reçoit une réponse refusée.
Correction :	(1) Vérifier si le bus a une activité anormale. (2) Remplacer le coprocesseur vers lequel la demande était dirigée..
Code d'erreur :	50, 51, 53
Nom :	Erreurs dans la mémoire système
Description :	Le système d'exploitation de l'API génère ces erreurs lorsque le gestionnaire mémoire refuse sa demande de bloc mémoire système du fait qu'aucune mémoire n'est disponible ou qu'elle contient des erreurs.
Correction :	(1) Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut. (2) Corriger la mémoire altérée.
Code d'erreur :	52
Nom :	Echec de la communication avec le fond de bac
Description :	Le système d'exploitation (processeur de demandes de service) de l'API génère cette erreur lorsqu'il tente de répondre à une demande exigeant la communication avec le fond de bac et reçoit une réponse refusée.
Correction :	(1) Vérifier si le bus a une activité anormale. (2) Remplacer le coprocesseur vers lequel la demande était dirigée. (3) Vérifier si le câble parallèle de console de programmation est fixé correctement.

Code d'erreur :	Tous les autres
Nom :	Erreur système interne de l'UC de l'API
Description :	Une erreur système interne s'est produite qui ne doit pas se produire dans un système de production.
Correction :	Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut.

Défaut de communication pendant le stockage

Le Groupe de défauts **Défaut de communication pendant le stockage** intervient pendant le stockage, dans l'API, de blocs de programme et d'autres données. Le train d'instructions et de données pour le stockage des blocs de programme et des données commence par une instruction spéciale de début de séquence et se termine par une instruction de fin de séquence. En cas d'interruption de la communication avec la console de programmation effectuant le stockage, ou d'autre défaut mettant fin au chargement, ce défaut est enregistré. L'API ne passe pas en mode **RUN** tant que ce défaut est présent dans le système.

Ce défaut *n'est pas* remis à "0" automatiquement à la mise sous tension ; l'utilisateur doit ordonner expressément que cette condition soit supprimée. L'intervention défaut pour ce groupe est de type **Fatal**.

Correction :	Corriger le défaut et réessayer de charger le programme ou fichier de configuration.
---------------------	--

Section 3 : Descriptions de la Table des défauts d'E/S

La table des défauts d'E/S présente trois types d'information :

- La catégorie de défaut.
- Le type de défaut.
- La description du défaut.

Les défauts décrits dans la page suivante ont une catégorie de défaut, mais n'ont ni type ni groupe de défauts.

Chaque explication de défaut contient une description du défaut et les instructions à suivre pour le corriger. Plusieurs descriptions de défaut ont des causes multiples. Dans ce cas, le code d'erreur, affiché avec les informations additionnelles obtenues en appuyant sur CTRL-F, est utilisé pour distinguer différentes conditions de défaut partageant la même description de défaut. (Pour plus d'informations concernant l'utilisation de CTRL-F, se référer à l'annexe B, "Interpréter les Tables de défauts", dans ce manuel.) Le code d'erreur est indiqué par les deux premiers chiffres hexadécimaux dans le cinquième groupe de nombres, comme indiqué dans l'exemple suivant.

01	000000	01030100	0902	0200	000000000000
				Code d'erreur (deux premiers chiffres hexadécimaux du cinquième groupe)	

Le tableau suivant permet de trouver rapidement une description de défaut d'E/S dans cette section. Chaque entrée est listée telle qu'elle apparaît sur la console de programmation.

Module d'E/S perdu

La catégorie de défaut **Module d'E/S perdu** s'applique aux modules d'E/S logiques et analogiques Modèle 30. Il n'existe pas de types de défaut ou de descriptions de défaut associés à cette catégorie. L'intervention défaut est de type **Diagnostic**.

Description :	Le système d'exploitation de l'API génère cette erreur lorsqu'il détecte qu'un module d'E/S Modèle 30 ne répond plus aux commandes de l'UC de l'API, ou lorsque le fichier de configuration indique que l'emplacement devant être occupé par un module d'E/S est vide.
Correction :	<ol style="list-style-type: none"> (1) Remplacer le module. (2) Corriger le fichier de configuration. (3) Afficher la table des défauts automate sur la console de programmation. Prendre contact avec le Service assistance-API de GE Fanuc et lui communiquer toutes les informations contenues dans l'entrée du défaut.

Module d'E/S additionnel

La catégorie de défaut **Module d'E/S additionnel** s'applique aux modules d'E/S logiques et analogiques Modèle 30. Il n'existe pas de type ou de descriptions de défaut associés à cette catégorie de défaut. L'intervention défaut est de type **Diagnostic**.

Description :	Le système d'exploitation de l'API génère cette erreur lorsqu'un module d'E/S, qui était en défaut, refonctionne.
Correction :	(1) Aucune intervention n'est nécessaire si le module a été retiré ou remplacé ou si le bac déporté a été remis sous tension. (2) Mettre à niveau le fichier de configuration ou retirer le module.
Description :	Le système d'exploitation de l'API génère cette erreur lorsqu'il détecte qu'un module d'E/S Modèle 30 se trouve dans un emplacement, qui d'après le fichier de configuration, devrait être vide.
Correction :	(1) Retirer le module. (Il est peut-être dans le mauvais emplacement). (2) Mettre à niveau et restaurer le fichier de configuration en incluant le module additionnel.

La programmation consiste à créer un programme d'application pour Automate programmable industriel (API). Les API Série 90-30, 90-20 et 90 Micro ayant un jeu d'instructions commun, il est possible d'utiliser le même logiciel pour les programmer. Ce chapitre décrit les instructions de programmation pouvant être utilisées pour créer les programmes à diagramme en échelle pour les API Série 90-30 et Série 90-20.

Si le logiciel de programmation Logicmaster 90-30/20/Micro n'est pas encore installé, voir le document GFK-0466 Logicmaster 90-30/20/Micro Programming Software User's Manual pour plus d'informations. Ce manuel explique comment créer, transférer, modifier et imprimer les programmes.

La configuration consiste à attribuer des adresses logiques, ainsi que d'autres caractéristiques, aux modules physiques du système. Elle peut être effectuée avant ou après la programmation en utilisant le logiciel de configuration ou la miniconsole de programmation (HHP). Il est toutefois recommandé d'effectuer la configuration en premier. Si elle n'a pas été effectuée, voir le document GFK-0466 Logicmaster 90-30/20/Micro Programming Software User's Manual, afin de déterminer s'il est préférable de commencer la programmation dès à présent.

Ce chapitre contient les sections suivantes :

Section	Titre	Description	Page
1	Instructions par diagramme en échelle	Décrit les contacts, bobines et liens.	4-2
2	Instruction de temporisation et de comptage	Décrit les compteurs, décompteurs, temporisateurs suspensifs et chiens de garde.	4-10
3	Instructions arithmétiques	Décrit l'addition, la soustraction, la multiplication, la division, la division module, la racine carrée, les instructions de trigonométrie, les instructions de logarithme/exposant et les instructions de conversion radian. <i>Notez que les instructions de trigonométrie, de logarithme/ exposant et de conversion radian ne sont disponibles qu'avec l'UC modèle 352.</i>	4-27
4	Instructions de comparaison	Décrit les opérations égal à, différent de, supérieur à, supérieur ou égal à, inférieur à, inférieur ou égal à.	4-42
5	Instructions au niveau du bit	Décrit comment effectuer des opérations de comparaison et de transfert sur des chaînes binaires.	4-48
6	Instructions de transfert de données	Décrit les possibilités de base du transfert de données.	4-71
7	Instructions de tableau	Décrit comment utiliser les instructions de tableau pour entrer des valeurs dans un tableau et copier les données d'un tableau.	4-89
8	Instructions de conversion	Décrit comment convertir un nombre élémentaire d'un type à un autre.	4-97
9	Instructions de commande	Décrit comment limiter l'exécution du programme et modifier la façon dont l'UC exécute le programme d'application, en utilisant les instructions de commande.	4-110

Section 1 : *Instructions par diagramme en échelle*

Cette section explique l'utilisation des contacts, bobines et liens dans les segments du diagramme en échelle.

Fonction	Page
Contact N.O. et contact N.F.	4-3
Bobine et bobine inversée.	4-2
Bobine rémanente et bobine rémanente inversée.	4-5
Bobine de transition positive et bobine de transition négative.	4-5
Bobine de mise à "1" et bobine de remise à "0".	4-6
Bobine rémanente de mise à "1" et bobine rémanente de remise à "0".	4-7
Liens horizontaux et verticaux.	4-7
Bobines "suite" et contacts "suite".	4-8

Utilisation des Contacts

Un contact est utilisé pour contrôler l'état d'une référence. Le fait que le contact laisse passer le flux validant ou non dépend de l'état de la référence contrôlée et du type de contact. Une référence est activée si son état est à "1", et désactivée s'il est à "0".

Tableau 4-1. Types de contacts

Type de contact	Affichage	Le contact laisse passer le flux validant vers la droite :
Normalement ouvert	— —	Lorsque la référence est à "1".
Normalement fermé	— / —	Lorsque la référence est à "0".
Contact "suite"	<+>———	Si la bobine "suite" précédente est à "1".

Utilisation des bobines

Les bobines sont utilisées pour contrôler les références logiques. La logique conditionnelle doit être utilisée pour commander une bobine par contrôle de flux. Les bobines provoquent directement une action ; elles ne dirigent pas le flux validant vers la droite. Si une logique additionnelle doit être exécutée dans le programme comme conséquence de la condition de la bobine, une référence interne doit être utilisée pour cette bobine ou une combinaison bobine/contact "suite".

Les bobines sont toujours situées sur la position la plus à droite d'une ligne de logique. Un segment peut contenir jusqu'à 8 bobines.

Le type de bobine utilisé dépend du type d'exécution que l'on attend du programme. L'état des bobines rémanentes est sauvegardé à la réinitialisation de l'alimentation ou lorsque l'API passe du mode **STOP** ou au mode RUN. L'état des bobines non rémanentes est mis à zéro à la réinitialisation de l'alimentation ou lorsque l'API passe du mode **STOP** ou mode **RUN**.

Tableau 4-2. Types de bobines

Type de bobine	Affichage	Flux validant vers bobine	Résultat
N.O.	—()—	ACTIVE DESACT	Met la référence à "1". Met la référence à "0".
Inversée	—(/)—	ACTIVE DESACT	Met la référence à "0". Met la référence à "1".
Rémanente	—(M)—	ACTIVE DESACT	Met la référence à "1", rémanente. Met la référence à "0", rémanente.
Rémanente inversée	—(/M)—	ACTIVE DESACT	Met la référence à "0", rémanente. Met la référence à "1", rémanente.
De transition positive	—(↑)—	DESACT→ACT	Si la référence est à "0", la met à "1" pendant un cycle.
De transition négative	—(↓)—	ACT←DESACT	Si la référence est à "0", la met à "1" pendant un cycle.
De mise à "1"	—(S)—	ACT DESACT	Met la référence à "1" jusqu'à sa remise à "0" par —(R)—. Ne change pas l'état de la bobine.
De remise à "0"	—(R)—	ACT DESACT	Met la référence à "0" jusqu'à sa mise "1" par —(S)—. Ne change pas l'état de la bobine.
Rémanente de mise à "1"	—(SM)—	ACT DESACT	Met la référence à "1" jusqu'à sa remise à "0" par —(RM)—, rémanente. Ne change pas l'état de la bobine.
Rémanente de remise à "0"	—(RM)—	ACT DESACT	Met la référence à "0" jusqu'à sa mise à "1" par —(SM)—, rémanente. Ne change pas l'état de la bobine.
Bobine "Suite"	—<+>	ACTIVE DESACT	Met à "1" le contact "suite" suivant. Met à "0" le contact "suite" suivant.

Contact normalement ouvert —| |—

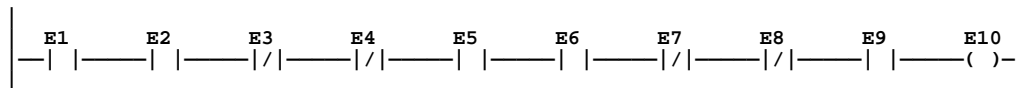
Un contact N.O. se comporte comme un interrupteur laissant passer le flux validant si la référence associée est à "1".

Contact normalement fermé —|/|—

Un contact N.F. se comporte comme un interrupteur laissant passer le flux validant si la référence associée est à "0".

Exemple :

L'exemple suivant présente un segment comportant 10 éléments ayant les symboles E1 à E10. La bobine E10 est à "1" lorsque les références E1, E2, E5, E6, et E9 sont à "1" et les références E3, E4, E7, et E8 sont à "0".

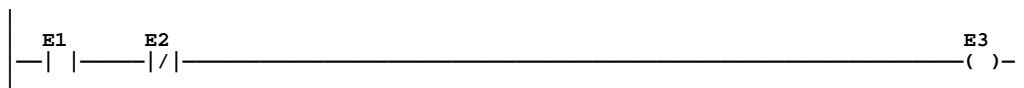


Bobine —()—

Une bobine met une référence logique à "1" pendant qu'elle reçoit le flux validant. Elle est non rémanente ; par conséquent, elle ne peut pas être utilisée avec les références d'état système (%SA, %SB, %SC ou %G).

Exemple :

Dans l'exemple suivante, la bobine E3 est à "0" lorsque la référence E1 est à "1" et la référence E2 est à "0".

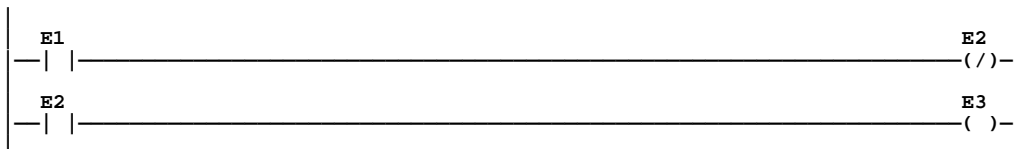


Bobine inversée —(I)—

Une bobine inversée met une référence logique à "1" lorsqu'elle ne reçoit pas de flux validant. Elle n'est pas rémanente ; par conséquent, elle ne peut être utilisée avec les références d'état système (%SA, %SB, %SC ou %G).

Exemple :

Dans l'exemple suivant, la bobine E3 est à "1" lorsque la référence E1 est à "0".



Bobine rémanente —(M)—

Comme une bobine N.O., la bobine rémanente met une référence logique à "1" pendant qu'elle reçoit le flux validant. L'état de la bobine rémanente est conservé pendant une coupure d'alimentation. C'est pourquoi elle ne peut pas être utilisée avec les références de la mémoire strictement non rémanente (%T).

Bobine rémanente inversée —(MI)—

La bobine rémanente inversée met une référence logique à "1" lorsqu'elle ne reçoit pas de flux validant. L'état de la bobine rémanente inversée est conservé pendant une coupure d'alimentation. C'est pourquoi elle ne peut pas être utilisée avec les références de la mémoire strictement non rémanente (%T).

Bobine de transition Positive —(↑)—

Si la référence associée à une bobine de transition positive est à "0", lorsque la bobine reçoit le flux validant, elle est mise à "1" jusqu'à ce que la bobine soit à nouveau exécutée. (Si le segment contenant la bobine est omise dans les cycles suivants, elle reste à "1".) Cette bobine peut être utilisée comme impulsion.

Afin de préserver la nature impulsionnelle de la bobine, chaque référence ne doit être utilisée comme bobine de transition qu'une seule fois dans le programme d'application.

Les bobines de transition peuvent être utilisées avec les références de mémoire rémanente ou non rémanente (%Q, %M, %T, %G, %SA, %SB ou %SC).

Bobine de transition négative —(↓)—

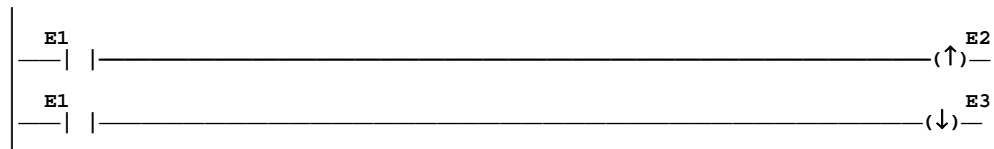
Si la référence associée à cette bobine est à "0", lorsque la bobine arrête de recevoir le flux validant, la référence est mise à "1" jusqu'à la prochaine exécution de la bobine.

Afin de préserver la nature impulsionnelle de la bobine, chaque référence ne doit être utilisée comme bobine de transition qu'une seule fois dans le programme d'application.

Les bobines de transition peuvent être utilisées avec les références de mémoire rémanente ou non rémanente (%Q, %M, %T, %G, %SA, %SB ou %SC).

Exemple :

Dans l'exemple suivant, lorsque la référence E1 passe de "0" à "1", les bobines E2 et E3 reçoivent le flux validant, mettant E2 à "1" pendant un cycle logique. Lorsque E1 passe de "1" à "0", le flux validant est supprimé de E2 et de E3, mettant la bobine E3 à "1" pendant un cycle.



Bobine de mise à "1" —(S)

Les bobines de mise à "1" et de remise à "0" sont des bobines non rémanentes pouvant être utilisées pour conserver (verrouiller) l'état d'une référence (par exemple, E1 soit à "1" soit à "0"). Lorsqu'une bobine de mise à "1" reçoit le flux validant, sa référence reste à "1" (que la bobine elle-même reçoive le flux validant ou non) jusqu'à ce la référence soit remise à "0" par une autre bobine.

Les bobines de mise à "1" écrivent un résultat non défini dans le bit de transition de la référence donnée. (Voir aux informations concernant les "Transitions et forçages" dans le chapitre 2, "Fonctionnement du système").

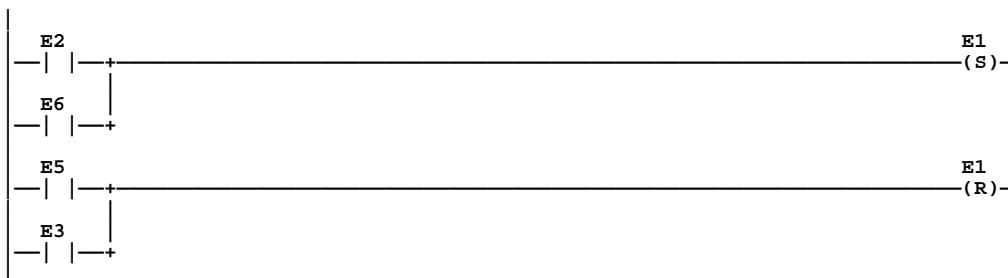
Bobine de remise à "0" —(R)—

La bobine de remise à "0" met une référence logique à "0" si la bobine reçoit le flux validant. La référence reste à "0" jusqu'à ce qu'elle soit mise à "1" par une autre bobine. La dernière bobine de mise à "1" ou de remise à "0" traitée dans une paire a la priorité.

Les bobines de remise à "0" écrivent un résultat non défini dans le bit de transition d'une référence donnée. (Voir les informations concernant les "Transitions et forçages" dans le chapitre 2, "Fonctionnement du système").

Exemple :

Dans l'exemple suivant, la bobine représentée par E1 est mise à "1" chaque fois que la référence E2 ou E6 est mise à "1". La bobine représentée par E1 est mise à "0" chaque fois que la référence E5 ou E3 est mise à "1".



Remarque

Lorsque le niveau de vérification de la bobine est UNIQUE, vous pouvez utiliser une référence %M ou %Q spécifique à l'aide d'une seule bobine, mais vous pouvez l'utiliser simultanément avec une bobine de mise à 1 et une bobine de remise à 0. Lorsque le niveau de vérification de la bobine est PREVENIR MULTIPLE ou MULTIPLE, chaque référence peut être utilisée avec plusieurs bobines, bobines de mise à 1 et bobines de remise à 0. En pareil cas, une référence doit être mise à 1 par une bobine de mise à 1 ou une bobine normale et peut être remise à 0 par une bobine de remise à 0 ou une bobine normale.

Bobine rémanente de mise à "1" —(SM)—

Les bobines rémanentes de mise à "1" et de remise à "0" sont similaires aux bobines de mise à "1" et de remise à "0", excepté qu'elles sont conservées en cas de coupure d'alimentation ou lorsque l'API passe du mode **STOP** au mode **RUN**. Une bobine rémanente de mise à "1" met une référence logique à "1" si la bobine reçoit le flux validant. La référence reste à "1" jusqu'à ce qu'elle soit remise à "0" par une bobine rémanente de remise à "0".

Les bobines rémanentes de mise à "1" écrivent un résultat non défini dans le bit de transition pour la référence donnée. (Voir les informations concernant les "Transitions et forçages" dans le chapitre 2, "Fonctionnement du système").

Bobine rémanente de remise à "0" —(RM)—

Cette bobine met une référence logique à "0" lorsqu'elle reçoit le flux validant. La référence reste à "0" jusqu'à ce qu'elle soit mise à "1" par une bobine rémanente de mise à "1". L'état de cette bobine est conservé pendant une coupure d'alimentation ou lorsque l'API passe du mode **STOP** au mode **RUN**.

La bobine rémanente de remise à "0" écrit un résultat non défini dans le bit de transition de la référence donnée. (Voir les informations concernant les "Transitions et forçages" dans le chapitre 2, "Fonctionnement du système").

Liens

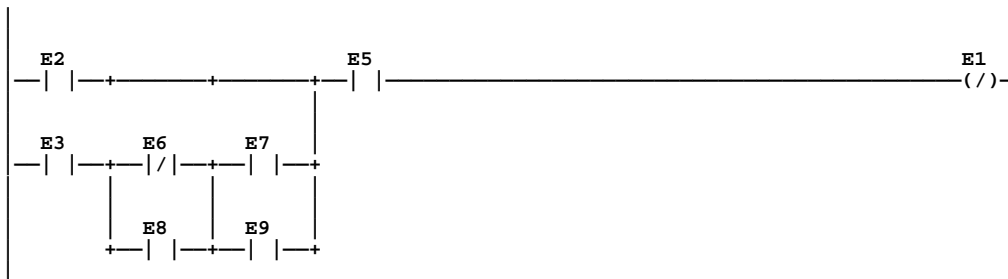
Les liens horizontaux et verticaux sont utilisés pour relier, entre les instructions, les éléments d'une ligne de diagramme en échelle. Leur rôle est de réaliser le flux validant de gauche à droite dans une ligne de diagramme.

Remarque

Vous ne pouvez pas utiliser un lien horizontal pour relier une fonction ou une bobine au rail de flux gauche. Vous pouvez cependant utiliser %S7, le bit système AWL_ON (toujours à 1), avec un contact normalement ouvert qui est lié au rail pour appeler une fonction à chaque cycle.

Exemple :

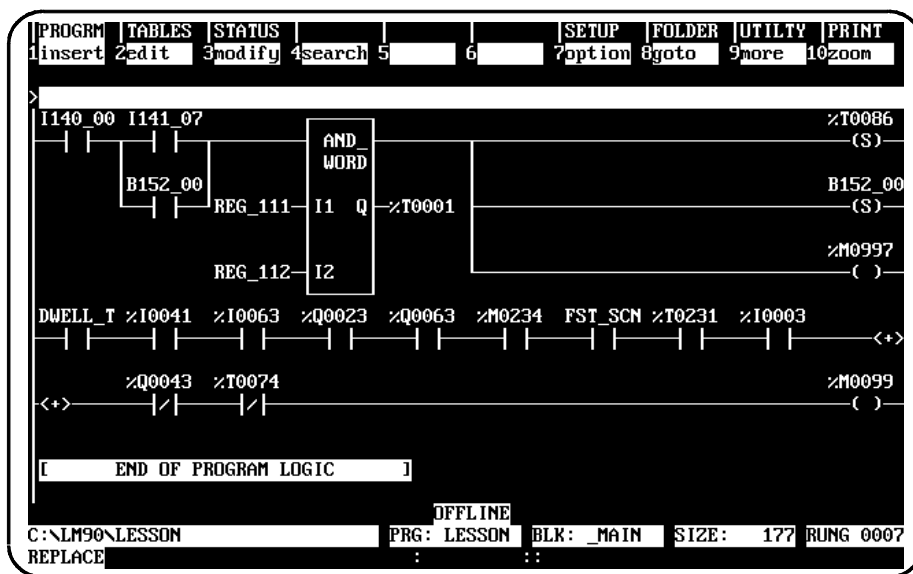
Dans l'exemple suivant, deux liens horizontaux sont utilisés pour relier les contacts E2 et E5. Un lien vertical est utilisé pour relier les contacts E3, E6, E7, E8 et E9 à E2.



Bobines "suite" (——<+>) et Contacts (<+>——) "suite"

Les bobines "suite"(——<+>) et les contacts "suite" (<+>——) sont utilisés pour continuer le diagramme en échelle au-delà de la limite des 10 colonnes. L'état de la bobine "suite" dernièrement exécutée est l'état du flux validant qui sera utilisé sur le prochain contact "suite" exécuté. Une bobine "suite" doit être présente pour que la logique puisse exécuter un contact "suite". L'état du contact "suite" est effacé lorsque l'API passe du mode STOP au mode RUN et le flux n'est rétabli que lorsque la bobine de transition est remise à 1 après le passage au mode RUN.

Chaque segment ne peut comporter qu'une seule paire bobine/contact "suite" ; le contact "suite" doit être dans la colonne 1, et la bobine "suite" dans la colonne 10. Un exemple de bobine et de contact "suite" est présenté ci-dessous.



Section 2 : *Instructions de temporisation et de comptage*

Cette section explique comment utiliser les temporisateurs suspensifs et chiens de garde, les compteurs et décompteurs. Les données associées à ces instructions sont rémanentes en cas de coupure d'alimentation.

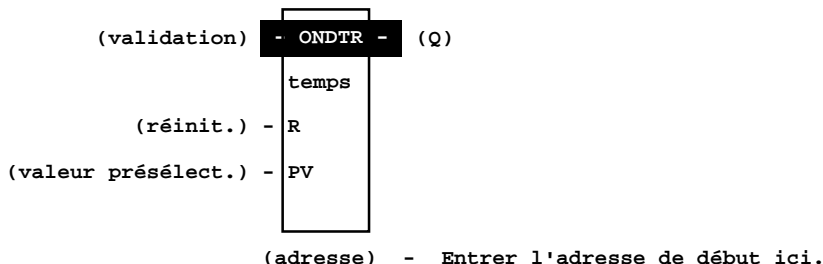
Abréviation	Fonction	Page
ONDTR	Temporisateur suspensif rémanent	4-12
TMR	Temporisateur suspensif simple	4-15
OFDT	Temporisateur à l'ouverture	4-18
UPCTR	Compteur	4-21
DNCTR	Décompteur	4-23

Données de bloc fonctionnel nécessaires aux temporisateurs et compteurs

Chaque temporisateur ou compteur utilise 3 mots (registres) de mémoire %R pour stocker les informations suivantes :

valeur courante (CV)	mot 1
valeur présélectionnée (PV)	mot 2
mot de contrôle	mot 3

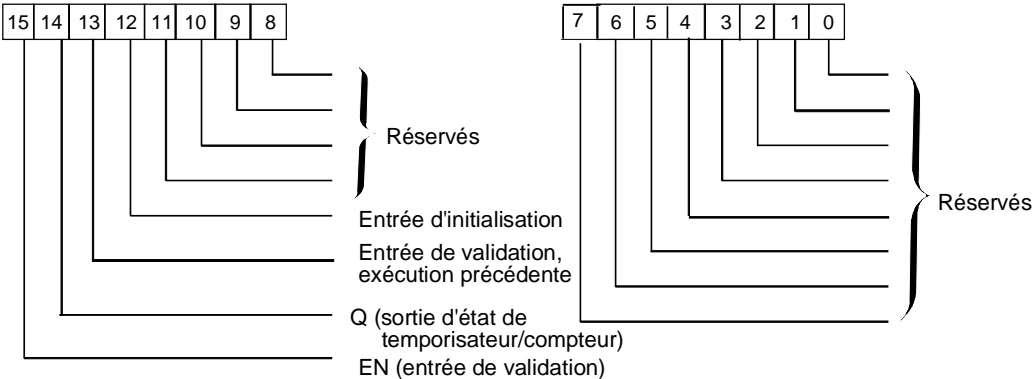
Lorsqu'on entre une instruction de temporisation ou de comptage, on doit entrer une adresse de début pour ces trois mots (registres) immédiatement sous le graphique représentant l'instruction. Par exemple :



Remarque

N'utilisez pas de registres consécutifs pour les 3 blocs de mot du temporisateur/compteur. Logicmaster ne vérifie *pas* et ne vous avertit pas en cas de chevauchement des blocs du registre. Les temporisateurs et les compteurs ne fonctionnent pas lorsque vous placez la valeur courante d'un bloc au-dessus de la valeur présélectionnée du bloc précédent.

Le mot de contrôle stocke l'état des E/S booléennes de son bloc fonctionnel associé, comme indiqué dans la présentation suivante :



Les bits 0 à 11 sont utilisés pour la précision du temporisateur ; les bits 0 à 11 ne sont pas utilisés pour les compteurs.

Remarque

Soyez extrêmement prudent lorsque vous utilisez la même adresse pour la valeur présélectionnée et le deuxième mot du bloc de trois mots. Quand la valeur présélectionnée n'est pas une constante, elle est généralement définie à un emplacement différent du deuxième mot. Certaines applications choisissent d'utiliser l'adresse du deuxième mot pour la valeur présélectionnée, notamment %R0102 lorsque le bloc de données inférieur commence à %R0101. Cela permet à une application de modifier la valeur présélectionnée pendant que le temporisateur ou le compteur tourne. Les applications peuvent lire la première valeur courante ou le troisième mot de contrôle, mais elles ne peuvent pas écrire de données dans ces valeurs, sinon l'instruction ne fonctionne pas.

ONDTR

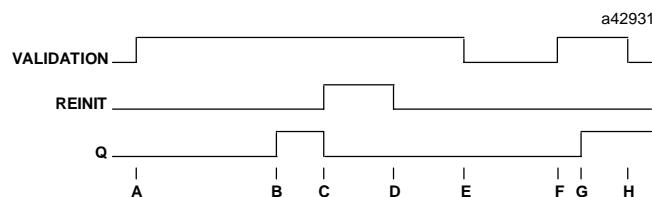
Un temporisateur rémanent suspensif (ONDTR) s'incrémente pendant qu'il reçoit le flux validant et conserve sa valeur lorsque le flux cesse. Le temps peut être compté en dixièmes ou centièmes de secondes. La plage est comprise entre 0 et +32767 unités de temps. L'état de ce temporisateur est rémanent en cas de coupure d'alimentation ; aucune initialisation automatique ne se produit à la mise sous tension.

Lorsque ONDTR reçoit le flux validant pour la première fois, il commence à incrémenter le temps (valeur courante). Lorsque ce temporisateur est rencontré dans le programme, sa valeur courante est rafraîchie.

Remarque

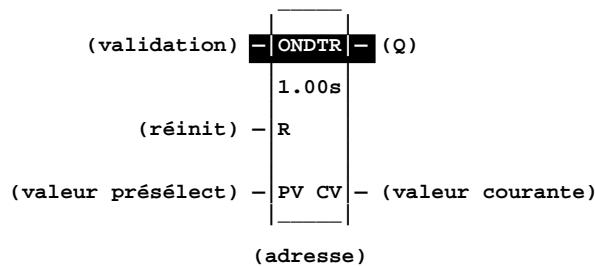
Si des occurrences multiples du même temporisateur avec la même adresse de référence sont validées au cours d'un cycle de l'UC, les valeurs courantes des temps seront les mêmes.

Lorsque la valeur courante égale ou excède la valeur présélectionnée (PV), la sortie Q est activée. Tant que le temporisateur continue de recevoir le flux validant, il continue de s'incrémenter jusqu'à ce que la valeur max. soit atteinte. Une fois la valeur max. atteinte, elle est conservée et la sortie Q reste activée quel que soit l'état de l'entrée de validation.



- A = VALIDATION est mis à "1" ; le temporisateur commence à s'incrémenter.
- B = La valeur courante atteint la valeur présélectionnée ; Q est mis à "1".
- C = REINIT est mis à "1" ; Q est mise à "0", le temps cumulé est réinitialisé.
- D = REINIT est mis à "0" ; le temporisateur s'incrémente à nouveau.
- E = VALIDATION est mis à "0" ; le temporisateur arrête de s'incrémenter. Le temps cumulé reste le même.
- F = VALIDATION est mis à "1" à nouveau ; le temporisateur continue à incrémenter le temps.
- G = La valeur courante est égale à la valeur présélectionnée ; Q est mis à "1". Le temporisateur continue à incrémenter le temps jusqu'à ce que VALIDATION soit mis à "0", que REINIT soit mis à "1" ou que la valeur courante devienne égale au temps maximum.
- H = VALIDATION est mis à "0" ; le temporisateur arrête de s'incrémenter.

Lorsque le flux validant vers le temporisateur s'arrête, la valeur courante cesse de s'incrémenter puis est conservée. La sortie Q reste activée si elle l'est déjà. Lorsque l'instruction reçoit à nouveau le flux validant, la valeur courante s'incrémente à nouveau, en démarrant à la valeur conservée. Lorsque REINIT R reçoit le flux validant, la valeur courante est remise à zéro et la sortie Q est désactivée. Sur les API 351 et 352, si la validation vers ONDTR est mise à "0", si la valeur présélectionnée est égale à zéro et si REINIT R reçoit le flux validant, la sortie est mise à "0". Sur les API 311–341, par contre, dans ces mêmes conditions, la sortie est mise à "1".



Paramètres :

Paramètre	Description
adresse	<p>ONDTR utilise trois mots consécutifs (registres) de la mémoire %R pour stocker les éléments suivant :</p> <ul style="list-style-type: none"> • Valeur courante (CV) = mot 1. • Valeur présélectionnée (PV) = mot 2. • Mot de contrôle = mot 3. <p>Lorsque vous entrez un ONDTR, vous devez spécifier l'adresse d'emplacement de ces trois mots consécutifs (registres) directement sous le graphique représentant la fonction.</p> <p>Remarque : N'utilisez pas cette adresse avec d'autres instructions.</p> <p>Attention : Les références qui se chevauchent entraînent un mauvais fonctionnement du temporisateur.</p>
validation	Lorsque Validation reçoit le flux validant, la valeur courante du temporisateur est incrémentée.
R	Lorsque R reçoit le flux validant, il réinitialise à zéro la valeur courante.
PV	PV est la valeur à copier dans la valeur courante du temporisateur lorsque le temporisateur est réinitialisé ou validé.
Q	La sortie Q est activée lorsque la valeur courante est supérieure ou égale à la valeur présélectionnée.
temps	Le temps augmente en dixièmes (0,1), centièmes (0,01) ou millièmes (0,001) de secondes pour le bit inférieur de la valeur présélectionnée et de la valeur courante.

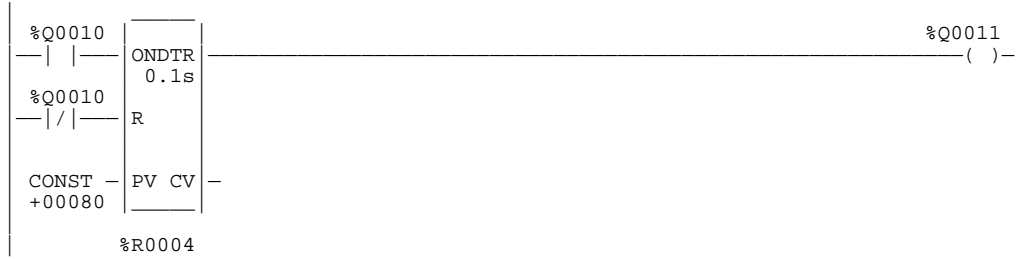
Types de mémoires valides :

Paramètre	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
adresse								•				
validation	•											
R	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, un temporisateur suspensif rémanent est utilisé pour créer un signal (%Q0011) s'activant 8 secondes après la mise à "1" de %Q0010, et se désactivant lorsque %Q0010 est mis à "0".



TMR

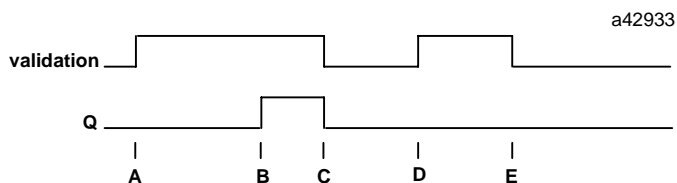
L'instruction temporisateur suspensif rémanent simplifié (TMR) s'incrémente pendant qu'il reçoit le flux validant et se réinitialise lorsque le flux cesse. Le temps peut être compté en dixièmes ou centièmes de seconde. La plage est comprise entre 0 et +32767 unités de temps. L'état de ce temporisateur est rémanent en cas de coupure d'alimentation ; aucune initialisation automatique ne se produit à la mise sous tension.

Lorsque TMR reçoit le flux validant, il commence à compter le temps (valeur courante). La valeur courante est rafraîchie lorsqu'elle est rencontrée dans la logique pour refléter le temps total écoulé durant lequel le temporisateur a été validé, depuis sa dernière réinitialisation.

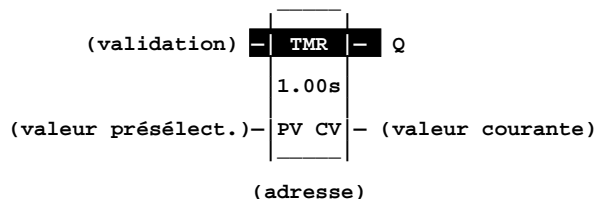
Remarque

Si des occurrences multiples du même temporisateur avec la même adresse de référence sont validées pendant un cycle d'UC, les valeurs courantes des temps sont les mêmes.

Ce rafraîchissement s'effectue tant que la logique de validation reste à "1". Lorsque la valeur courante égale ou excède la valeur présélectionnée (PV), l'instruction commence à diriger le flux validant vers la droite. Le temporisateur continue d'incrémenter le temps jusqu'à ce que la valeur max. soit atteinte. Lorsque le paramètre de validation passe de "1" à "0", le temporisateur arrête d'incrémenter le temps et la valeur courante est remise à zéro.



- A = VALIDATION est mis à "1" ; le temporisateur commence à s'incrémenter.
- B = La valeur courante atteint la valeur présélectionnée (PV) ; Q est mis à "1", et le temporisateur continue à incrémenter le temps.
- C = VALIDATION est mis à "0" ; Q est mis à "0" ; le temporisateur stoppe l'incrémentation et le temps courant est remis à zéro.
- D = VALIDATION est mis à "1" ; le temporisateur commence à incrémenter le temps.
- E = VALIDATION est mis à "0" avant que la valeur courante n'atteigne la valeur présélectionnée (PV) ; Q reste à "0" ; le temporisateur stoppe l'incrémentation puis est remis à zéro.



Paramètres :

Paramètre	Description
adresse	<p>TMR utilise trois mots consécutifs (registres) de la mémoire %R pour stocker les éléments suivants :</p> <ul style="list-style-type: none"> • Valeur courante (CV) = mot 1. • Valeur présélectionnée (PV) = mot 2. • Mot de contrôle = mot 3. <p>Lorsque vous entrez un TMR, vous devez spécifier l'adresse d'emplacement de ces trois mots consécutifs (registres) directement sous le graphique représentant la fonction.</p> <p>Remarque : N'utilisez pas cette adresse avec d'autres instructions.</p> <p>Attention : Les références qui se chevauchent entraînent un mauvais fonctionnement du temporisateur.</p>
validation	Lorsque Validation reçoit le flux validant, la valeur courante du temporisateur est incrémentée. Lorsque TMR n'est pas validé, la valeur courante est remise à zéro et Q est mise à "0".
PV	PV est la valeur à copier dans la valeur courante du temporisateur lorsque le temporisateur est réinitialisé ou validé.
Q	La sortie Q est activée lorsque TMR est validé et que la valeur courante est supérieure ou égale à la valeur présélectionnée.

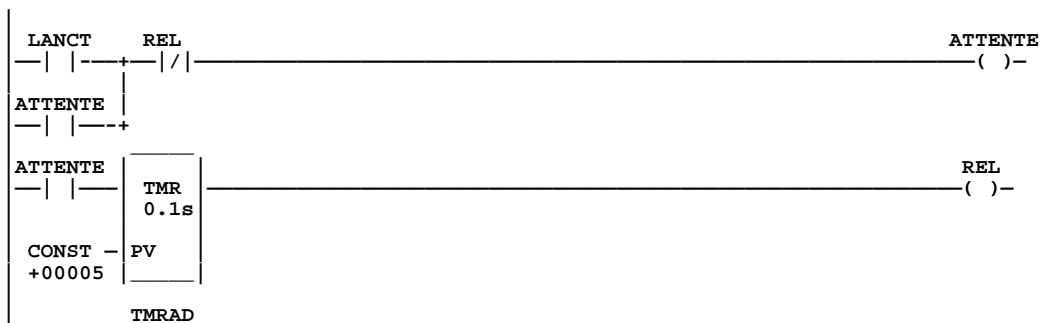
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
adresse								•				
validation	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, un temporisateur suspensif TMRAD est utilisé pour contrôler le temps pendant lequel cette bobine ATTENTE (maintien) est activée. Lorsque le contact (momentané) N.O. LANCT est à "1", la bobine ATTENTE est activée. Le contact de la bobine ATTENTE maintient activée la bobine ATTENTE (lorsque le contact LANCT est réouvert), et démarre également le temporisateur TMRAD. Lorsque TMRAD atteint sa valeur présélectionnée d'une demi-seconde, la bobine REL (réouverture) est activée, interrompant la condition de verrouillage à "1" de la bobine ATTENTE. Le contact ATTENTE interrompt le flux validant vers TMRAD, réinitialisant sa valeur courante et désactivant la bobine REL. Le circuit est ensuite prêt pour une autre activation momentanée du contact LANCT.



OFDT

Le temporisateur à l'ouverture (OFDT) augmente lorsque le flux validant est désactivé et est remis à zéro quand il est activé. Le temps peut être compté en dixièmes ou centièmes de seconde. La plage est comprise entre 0 et +32767 unités de temps. L'état de ce temporisateur est rémanent en cas de coupure d'alimentation ; aucune initialisation automatique ne se produit à la mise sous tension.

La première fois que OFDT reçoit le flux validant, il le passe vers la droite et la valeur courante (CV) est mise à zéro. (OFDT utilise le mot 1 [registre] comme emplacement de stockage de la valeur courante ; pour plus d'informations, voir la section "Paramètres" à la page suivante.) La sortie demeure à 1 aussi longtemps que l'instruction reçoit le flux validant. Dès qu'elle cesse de recevoir le flux validant par la gauche, elle continue à le passer vers la droite tandis que le temporisateur commence à comptabiliser le temps dans la valeur courante.

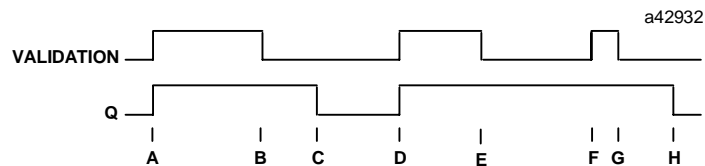
Remarque

Si des occurrences multiples du même temporisateur avec la même adresse de référence sont validées pendant un cycle d'UC, les valeurs courantes des temps sont les mêmes.

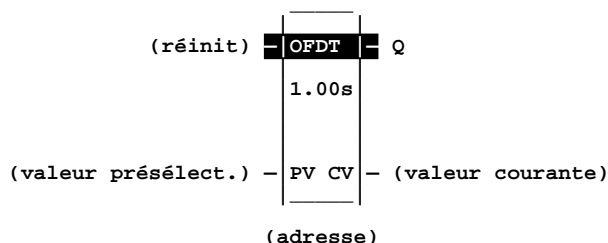
OFDT ne passe pas le flux validant si la valeur présélectionnée est négative ou égale à zéro.

Chaque fois que l'instruction est appelée en mettant à zéro la logique d'activation, la valeur courante est mise à jour pour refléter le temps écoulé depuis la remise à zéro du temporisateur. Quand la valeur courante (CV) est égale à la valeur présélectionnée (PV), l'instruction cesse de passer le flux validant vers la droite et le temporisateur s'arrête (voir la partie C ci-dessous).

Lorsque l'instruction reçoit à nouveau le flux validant, la valeur courante est remise à zéro.



- A = VALIDATION et Q sont mis à "1" ; le temporisateur est remis à zéro (CV = 0).
- B = VALIDATION est mis à "0" ; le temporisateur commence à s'incrémenter.
- C = La valeur courante est égale à la valeur présélectionnée ; Q est mis à "0" et le temporisateur arrête de s'incrémenter.
- D = VALIDATION est mis à "1" ; le temporisateur est remis à zéro (CV = 0).
- E = VALIDATION est mis à "0" ; le temporisateur commence à s'incrémenter.
- F = VALIDATION est mis à "1" ; le temporisateur est remis à zéro (CV = 0).
- G = VALIDATION est mis à "0" ; le temporisateur commence à s'incrémenter.
- H = La valeur courante est égale à la valeur présélectionnée ; Q est mis à "0" et le temporisateur arrête de s'incrémenter.



Quand OFDT est utilisé dans un bloc de programme qui n'est *pas* appelé à chaque cycle, le temporisateur accumule le temps entre les appels du bloc de programme jusqu'à sa remise à zéro. Cela signifie qu'il fonctionne comme le temporisateur d'un programme dont le cycle est nettement plus lent que celui du bloc de programme principal. Dans le cas des blocs de programme qui sont inactifs pendant une durée prolongée, le temporisateur doit être programmé pour supporter cette fonction. Si le temporisateur d'un bloc de programme est remis à zéro, par exemple, et si le bloc de programme n'est pas appelé (est inactif) pendant quatre minutes, au moment de l'appel du bloc, quatre minutes se sont déjà accumulées. Cette durée est appliqué au temporisateur lorsqu'il est activé, sauf s'il est préalablement remis à zéro.

Paramètres :

Paramètre	Description
adresse	<p>OFDT utilise trois mots consécutifs (registres) de la mémoire %R pour stocker les éléments suivants :</p> <ul style="list-style-type: none"> • Valeur courante (CV) = mot 1. • Valeur présélectionnée (PV) = mot 2. • Mot de contrôle = mot 3. <p>Lorsque vous entrez un OFDT, vous devez spécifier l'adresse d'emplacement de ces trois mots consécutifs (registres) directement sous le graphique représentant la fonction.</p> <p>Remarque : N'utilisez pas cette adresse avec d'autres instructions.</p> <p>Attention : Les références qui se chevauchent entraînent un mauvais fonctionnement du temporisateur.</p>
validation	Lorsque Validation reçoit le flux validant, la valeur courante du temporisateur est incrémentée.
temps	Le temps (P1) spécifie le type d'unités (millisecondes, etc.) utilisé par les registres.
PV	PV est la valeur à copier dans la valeur courante du temporisateur lorsque le temporisateur est réinitialisé ou validé. Pour une référence PV du registre (%R), le paramètre PV est spécifié comme deuxième mot du paramètre d'adresse. Le paramètre d'adresse %R0001, par exemple, utilise %R0002 comme paramètre PV.
Q	La sortie Q est activée lorsque la valeur courante est inférieure à la valeur présélectionnée. L'état de Q est rémanent en cas de panne de secteur et aucune initialisation automatique n'est effectuée à la mise sous tension.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
adresse								.				
validation	.											
PV
Q	.											.

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

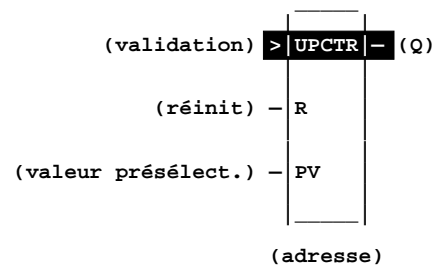
Dans l'exemple suivant, un temporisateur OFDT est utilisé pour désactiver une sortie (%Q0001) chaque fois qu'une entrée (%I0001) est activée. La sortie est à nouveau activée 0,3 seconde après la désactivation de l'entrée.



UPCTR

Le compteur (UPCTR) est une instruction utilisée pour compter jusqu'à une valeur présélectionnée. La plage est comprise entre 0 et +32767. Lorsque la réinitialisation du compteur est à "1", la valeur courante du compteur est réinitialisée à zéro. Chaque fois que l'entrée de validation passe de "0" à "1", la valeur courante est incrémentée de 1. La valeur courante peut être incrémentée au-delà de la valeur présélectionnée (PV). La sortie est mise à "1" chaque fois que la valeur courante est supérieure ou égale à la valeur présélectionnée.

L'état de UPCTR est rémanent en cas de coupure d'alimentation ; aucune initialisation automatique ne se produit à la mise sous tension.



Paramètres :

Paramètre	Description
adresse	<p>UPCTR utilise trois mots consécutifs (registres) de la mémoire %R pour stocker les éléments suivants :</p> <ul style="list-style-type: none"> • Valeur courante (CV) = mot 1. • Valeur présélectionnée (PV) = mot 2. • Mot de contrôle = mot 3. <p>Lorsque vous entrez un UPCTR, vous devez spécifier l'adresse d'emplacement de ces trois mots consécutifs (registres) directement sous le graphique représentant la fonction.</p> <p>Remarque : N'utilisez pas cette adresse avec un autre compteur, un autre décompteur ou toute autre instruction, car cela risquerait d'entraîner un mauvais fonctionnement.</p> <p>Attention : Les références qui se chevauchent entraînent un mauvais fonctionnement du compteur.</p>
validation	Lors d'une transition positive de Validation, le comptage actuel est incrémenté de 1.
R	Lorsque R reçoit le flux validant, il réinitialise la valeur courante à zéro.
PV	PV est la valeur à copier dans la valeur présélectionnée du compteur lorsque celui-ci est validé ou réinitialisé.
Q	La sortie Q est activée lorsque la valeur courante est supérieure ou égale à la valeur présélectionnée.

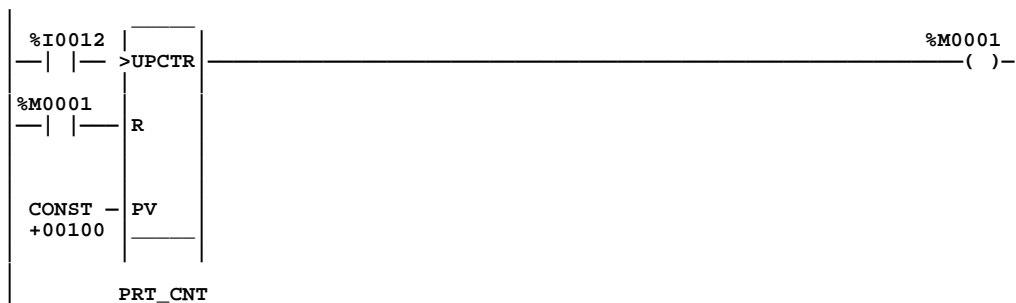
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
adresse								•				
validation	•											
R	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

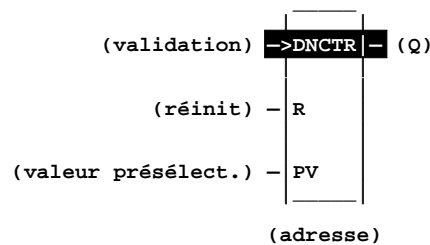
Dans l'exemple suivant, chaque entrée de temps %I0012 passe de "0" à "1", le compteur PRT_CNT est incrémenté de 1 ; la bobine interne %M0001 est activée tous les 100 comptages. Chaque fois que %M0001 est à "1", la valeur cumulée est remise à zéro.



DNCTR

L'instruction de décomptage (DNCTR) est utilisée pour décompter à partir d'une valeur présélectionnée. La valeur présélectionnée minimale est zéro ; la valeur courante maximale est de +32767. La valeur courante minimale est de -32768. Lorsqu'elle est réinitialisée, la valeur courante du compteur prend la valeur présélectionnée (PV). Lorsque l'entrée de validation passe de "0" à "1", la valeur courante est décrétementée de 1. La sortie est mise à "1" chaque fois que la valeur courante est inférieure ou égale à zéro.

La valeur courante de DNCTR est rémanente en cas de coupure d'alimentation ; aucune initialisation automatique ne se produit à la mise sous tension.



Paramètres :

Paramètre	Description
adresse	<p>DNCTR utilise trois mots consécutifs (registres) de la mémoire %R pour stocker les éléments suivants :</p> <ul style="list-style-type: none"> • Valeur courante (CV) = mot 1. • Valeur présélectionnée (PV) = mot 2. • Mot de contrôle = mot 3. <p>Lorsque vous entrez un DNCTR, vous devez spécifier l'adresse d'emplacement de ces trois mots consécutifs (registres) directement sous le graphique représentant la fonction.</p> <p>Remarque : N'utilisez pas cette adresse avec un autre compteur, un autre décompteur ou toute autre instruction, car cela risquerait d'entraîner un mauvais fonctionnement.</p> <p>Attention : Les références qui se chevauchent entraînent un mauvais fonctionnement du compteur.</p>
validation	Lors d'une transition positive de Validation, la valeur courante est décrétementée de un.
R	Lorsque R reçoit le flux validant, il réinitialise la valeur courante à la valeur présélectionnée.
PV	PV est la valeur à copier dans la valeur présélectionnée du compteur lorsque celui-ci est validé ou réinitialisé.
Q	La sortie Q est activée lorsque la valeur courante est inférieure à ou égale à zéro.

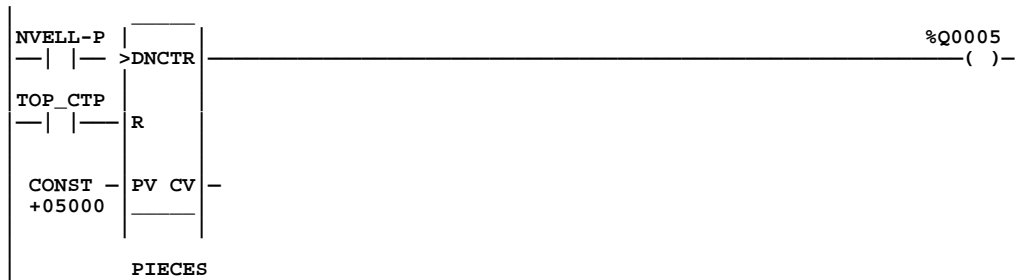
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
adresse								•				
validation	•											
R	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

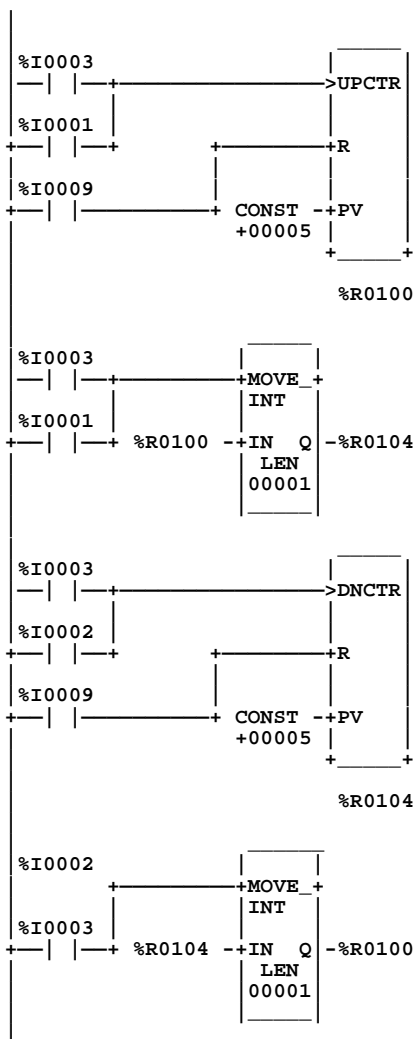
Dans l'exemple suivant, le décompteur appelé PIECES compte 500 nouvelles pièces avant d'activer la sortie %Q0005.



Exemple :

Dans l'exemple suivant, l'API est utilisée pour suivre le nombre de pièces contenues dans une zone d'entreposage temporaire. Le jeu d'instructions Série 90-30/20/Micro propose deux méthodes pour exécuter cette instruction.

La première méthode consiste à utiliser une paire compteur/décompteur avec un registre partagé pour la valeur cumulée ou courante. Lorsque les pièces entrent dans la zone de stockage, le compteur est incrémenté de 1, augmentant d'une valeur de 1 le nombre actuel de pièces stockées. Lorsqu'une pièce quitte la zone de stockage, le décompteur décrémente de 1, diminuant de 1 la valeur du stock. Pour éviter un conflit avec le registre partagé, le compteur et le décompteur utilisent des adresses de registre différentes. Lorsqu'un registre compte, sa valeur courante doit être calquée sur le registre de la valeur courante du décompteur.



Section 3 : Instructions arithmétiques

Cette section décrit les instructions arithmétiques du jeu d'instructions Série 90-30/20/Micro :

Abréviation	Fonction	Description	Page
ADD	Addition	Additionne deux nombres.	4-28
SUB	Soustraction	Soustrait un nombre d'un autre.	4-28
MUL	Multiplication	Multiplie deux nombres.	4-28
DIV	Division	Divise un nombre par un autre, et obtient un quotient.	4-28
MOD	Division modulo	Divise un nombre par un autre, et obtient un reste.	4-32
SQRT	Racine carrée	Trouve la racine carrée d'un nombre entier.	4-34
SIN, COS, TAN, ASIN, ACOS, ATAN	Instructions de trigonométrie †	Applique l'instruction appropriée à la valeur réelle de l'entrée IN.	4-40
LOG, LN EXP, EXPT	Instructions de logarithme/exposant †	Applique l'instruction appropriée à la valeur réelle de l'entrée IN.	4-153
RAD, DEG	Conversion radian †	Applique l'instruction appropriée à la valeur réelle de l'entrée IN.	4-40

† Les instructions de trigonométrie, de logarithme/exposant et de conversion radian sont disponibles uniquement sur l'UC modèle 352.

Remarque

La division et la division modulo sont des instructions similaires qui diffèrent par leur résultat. La division trouve un quotient, alors que la division modulo trouve un reste.

Paramètres:

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
I1	I1 contient une constante ou une référence pour la première valeur utilisée dans l'opération. (I1 est sur le côté gauche de l'équation arithmétique, comme dans I1_I2).
I2	I2 contient une constante ou une référence pour la seconde valeur utilisée dans l'opération (I2 est sur le côté droit de l'équation arithmétique, comme dans I1_I2).
ok	La sortie ok est activée lorsque l'instruction est exécutée sans dépassement, sauf si une opération invalide se produit.
Q	La sortie Q contient le résultat de l'opération.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		o	o	o	o		o	•	•	•	•†	
I2		o	o	o	o		o	•	•	•	•†	
ok	•											•
Q		o	o	o	o		o	•	•	•		

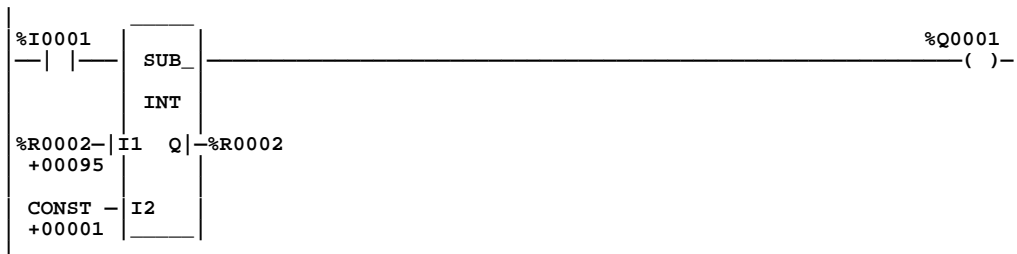
- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Référence valide pour les données de type INT seulement ; n'est pas valide pour DINT.
- † Les constantes sont limitées aux valeurs comprises entre -32768 et +32767 pour les opérations avec nombres entiers signés en double précision.

Remarque

Le type par défaut est INT pour les opérandes uniques ou 16 bits du registre. Appuyez sur **F10** pour sélectionner le type DINT, double mot à 32 bits ou REAL (pour l'UC 352 seulement). Les valeurs INT de l'API occupent un seul registre de 16 bits, %R, %AI ou %AQ. Les valeurs DINT, par contre, nécessitent deux registres consécutifs et placent les 16 bits inférieurs dans le premier mot et les 16 bits supérieurs dans le deuxième. Sur l'UC 352 seulement, les valeurs REAL occupent aussi un double registre de 32 bits et placent le signe dans le bit supérieur, suivi de l'exposant et de la mantisse.

Exemple :

Dans l'exemple suivant, chaque fois que l'entrée %I0001 est mise à "1", le contenu entier de %R0002 est décrémenté de 1 et la bobine %Q0001 est mise à "1", s'il n'y a pas de dépassement dans la soustraction.



Instructions arithmétiques et types de données

Instruction	Opération	Affichage
ADD INT	$Q(16 \text{ bit}) = I1(16 \text{ bit}) + I2(16 \text{ bit})$	Nombre de base 10 à 5 chiffres avec signe
ADD DINT	$Q(32 \text{ bit}) = I1(32 \text{ bit}) + I2(32 \text{ bit})$	Nombre de base 10 à 8 chiffres avec signe
ADD REAL*	$Q(32 \text{ bit}) = I1(32 \text{ bit}) + I2(32 \text{ bit})$	Nombre de base 10 à 7 chiffres, signe et décimale
SUB INT	$Q(16 \text{ bit}) = I1(16 \text{ bit}) - I2(16 \text{ bit})$	Nombre de base 10 à 5 chiffres avec signe
SUB DINT	$Q(32 \text{ bit}) = I1(32 \text{ bit}) - I2(32 \text{ bit})$	Nombre de base 10 à 8 chiffres avec signe
SUB REAL*	$Q(32 \text{ bit}) = I1(32 \text{ bit}) - I2(32 \text{ bit})$	Nombre de base 10 à 7 chiffres, signe et décimale
MUL INT	$Q(16 \text{ bit}) = I1(16 \text{ bit}) * I2(16 \text{ bit})$	Nombre de base 10 à 5 chiffres avec signe
MUL DINT	$Q(32 \text{ bit}) = I1(32 \text{ bit}) * I2(32 \text{ bit})$	Nombre de base 10 à 8 chiffres avec signe
MUL REAL*	$Q(32 \text{ bit}) = I1(32 \text{ bit}) * I2(32 \text{ bit})$	Nombre de base 10 à 7 chiffres, signe et décimale
DIV INT	$Q(16 \text{ bit}) = I1(16 \text{ bit}) / I2(16 \text{ bit})$	Nombre de base 10 à 5 chiffres avec signe
DIV DINT	$Q(32 \text{ bit}) = I1(32 \text{ bit}) / I2(32 \text{ bit})$	Nombre de base 10 à 8 chiffres avec signe
DIV REAL*	$Q(32 \text{ bit}) = I1(32 \text{ bit}) / I2(32 \text{ bit})$	Nombre de base 10 à 7 chiffres, signe et décimale

* UC 352 uniquement

Remarque

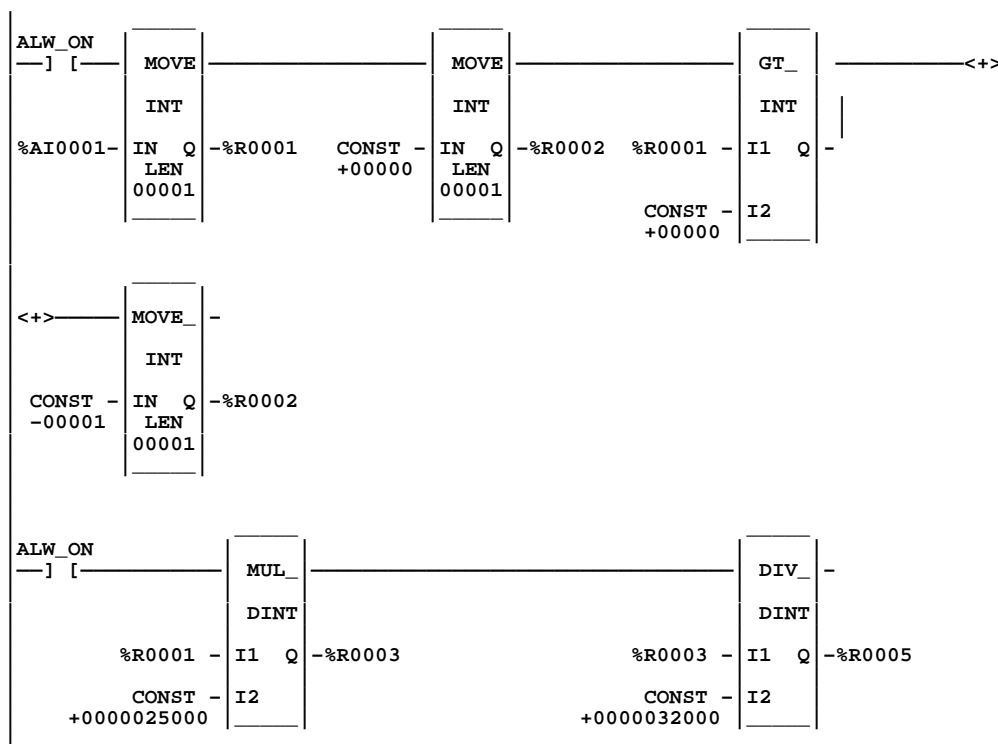
Les types de données d'entrée et de sortie doivent être identiques. Les instructions MUL et DIV ne supportent pas de mode mixte comme le fait l'API 90-70. L'instruction MUL INT de 2 entrées à 16 bits, par exemple, donne un produit de 16 bits et non de 32 bits. L'utilisation de MUL DINT pour un produit de 32 bits nécessite deux entrées de 32 bits. L'instruction DIV INT divise I2 à 16 bits pour un résultat de 16 bits tandis que DIV DINT divise I1 à 32 bits par I2 à 32 bits pour un résultat de 32 bits.

Ces instructions passent le flux en l'absence de dépassement mathématique. Dans le cas contraire, le résultat correspond à la valeur la plus élevée accompagnée du signe adéquat sans passage du flux.

Veillez à éviter les dépassements lorsque vous utilisez les instructions MUL et DIV. Si vous devez convertir des valeurs INT en valeurs DINT, n'oubliez pas que l'UC utilise le complément standard de 2 en étendant le signe vers le bit le plus élevé du deuxième mot. Vous devez vérifier le signe du mot de 16 bits inférieur et l'étendre au deuxième mot de 16 bits. Si le bit le plus significatif d'un mot INT de 16 bits est 0 (positif), déplacez un 0 vers le deuxième mot. Si le bit le plus significatif d'un mot de 16 bits est -1 (négatif), déplacez un -1 ou la valeur hexadécimale 0FFFFh vers le deuxième mot. La conversion de DINT vers INT est plus facile, car le mot de 16 bits inférieur (premier registre) correspond à la partie INT d'un mot DINT de 32 bits. Les 16 bits supérieurs ou le deuxième mot doivent avoir la valeur 0 (positif) ou -1 (négatif), sinon le nombre DINT est trop élevé pour effectuer la conversion en 16 bits.

La conversion des valeurs d'entrée analogique à l'aide d'une opération MUL suivie d'une opération DIV et, éventuellement, d'une opération ADD est une application mathématique courante. Avec une plage maximale de 32000, l'utilisation de MUL INT provoque un dépassement. L'utilisation d'une valeur %AI avec MUL DINT ne fonctionne pas non plus, car il à 32 bits combine simultanément 2 entrées analogiques. Vous devez déplacer l'entrée analogique vers le mot inférieur d'un registre double, puis tester le signe et affecter la valeur 0 au deuxième registre s'il est positif ou la valeur -1 s'il est négatif. Utilisez le registre double avec MUL DINT afin d'obtenir un produit 32 pour la fonction DIV ci-dessous.

La logique ci-dessous, par exemple, peut être utilisée pour convertir une entrée %AI1 de +/-10 volts en unités d'ingénierie de +/- 25000 dans %R5.



MOD (INT, DINT)

L'instruction Modulo (MOD) est utilisée pour diviser une valeur par une autre du même type, pour obtenir le reste. Le signe du résultat est toujours celui du paramètre d'entrée I1.

L'instruction MOD opère sur les types de données suivants :

Type de données	Description
INT	Nombre entier signé.
DINT	Nombre entier signé en double précision.

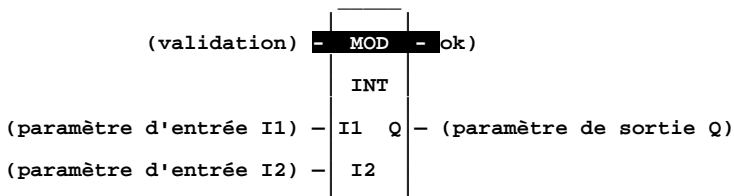
Le type de données par défaut est le nombre entier ; toutefois, il peut être changé après avoir choisi l'instruction. Pour plus d'informations sur ces types de données, voir le chapitre 2, § 2, "Organisation du programme et données/références utilisateur".

Lorsque l'instruction reçoit le flux validant, elle divise le paramètre d'entrée I1 par le paramètre d'entrée I2. Ces paramètres doivent être du même type de données. La sortie Q est calculée en utilisant la formule :

$$Q = I1 - ((I1 \text{ DIV } I2) * I2)$$

où DIV donne un nombre entier. Q est du même type de données que les paramètres d'entrée I1 et I2.

OK est toujours à "1" lorsque l'instruction reçoit le flux validant, sauf s'il y a une tentative de division par zéro. Dans ce cas, elle est mise à "0".



Paramètres :

Paramètres	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
I1	I1 contient une constante ou une référence pour la valeur à diviser par I2.
I2	I2 contient une constante ou une référence pour que I1 soit divisé par la valeur.
ok	La sortie ok est activée lorsque l'instruction est exécutée sans dépassement.
Q	La sortie Q contient le résultat de la division de I1 par I2 pour obtenir un reste.

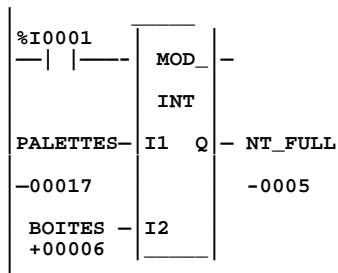
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		o	o	o	o		o	•	•	•	•†	
I2		o	o	o	o		o	•	•	•	•†	
ok	•											•
Q		o	o	o	o		o	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Référence validation pour les données de type INT seulement ; n'est pas valide pour DINT.
- † Les constantes sont limitées aux valeurs comprises entre -32768 et +32767 pour les opérations à nombres entiers en double précision.

Exemple :

Dans l'exemple suivant, le reste de la division entière, BOITES par PALETTES est placé dans NT_FULL chaque fois que %I0001 est à "1".



SQRT (INT, DINT, REAL)

L'instruction SQRT est utilisée pour trouver la racine carrée d'une valeur. Lorsque l'instruction reçoit le flux validant, la sortie Q a pour valeur la partie entière de la racine carrée de l'entrée IN. La sortie Q doit être du même type que IN.

L'instruction SQRT opère sur les types de données suivants :

Type de données	Description
INT	Nombre entier signé.
DINT	Nombre entier signé en double précision.
REAL	Virgule flottante.

Remarque

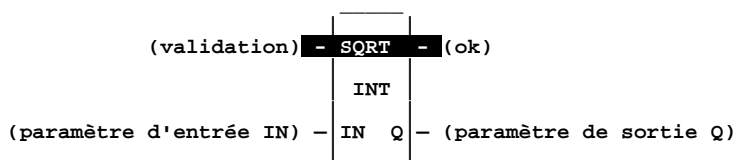
Le type de données REAL est disponible uniquement sur les UC 352.

Le type de données par défaut est le nombre entier signé ; toutefois, il peut être changé après avoir choisi l'instruction. Pour plus d'informations sur ces types de données, voir le chapitre 2, § 2, "Organisation du programme et données/références utilisateur".

OK est mis à "1" si l'instruction est exécutée sans dépassement, sauf en présence d'une des opérations REAL non valides ci-dessous :

- $IN < 0$.
- IN est NaN (Not a Number).

Sinon, ok est mis à "0".



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
IN	IN contient une constante ou une référence pour la valeur dont la racine carrée doit être calculée. Si IN est inférieur à zéro, l'instruction ne passe pas le flux validant.
ok	La sortie ok est activée lorsque l'instruction est exécutée sans dépassement, sauf si une opération invalide se produit.
Q	La sortie Q contient la racine carrée de IN.

Instructions de trigonométrie (SIN, COS, TAN, ASIN, ACOS, ATAN)

Les instructions SIN, COS et TAN permettent de calculer respectivement le sinus, le cosinus et la tangente trigonométriques de l'entrée. Lorsqu'une de ces instructions reçoit le flux validant, elle calcule le sinus (le cosinus ou la tangente) d'IN, dont les unités sont les radians, et stocke le résultat dans la sortie Q. IN et Q sont des valeurs à virgule flottante.

Les instructions ASIN, ACOS et ATAN permettent de calculer respectivement le sinus, le cosinus et la tangente inverse de l'entrée. Lorsqu'une de ces instructions reçoit le flux validant, elle calcule le sinus (le cosinus ou la tangente) inverse d'IN et stocke le résultat dans la sortie Q, dont les unités sont les radians. IN et Q sont des valeurs à virgule flottante.

Les instructions SIN, COS et TAN acceptent une vaste plage de valeurs d'entrée, où $-2^{63} < IN < +2^{63}$, ($2^{63} \approx 9,22 \times 10^{18}$).

Les instructions ASIN et ACOS, par contre, n'acceptent qu'une plage limitée de valeurs valides, où $-1 \leq IN \leq 1$. Si le paramètre IN possède une valeur valide, l'instruction ASIN_REAL produit un résultat Q semblable à ceci :

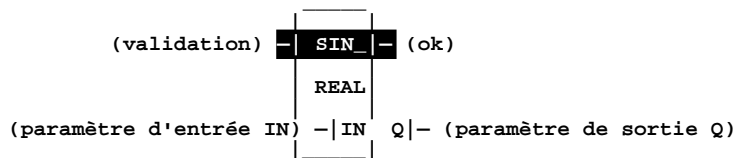
$$\text{ASIN (IN)} = -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$

L'instruction ACOS_REAL produit un résultat Q semblable à ceci :

$$\text{ACOS (IN)} = 0 \leq Q \leq \pi$$

L'instruction ATAN accepte la plus vaste plage de valeurs d'entrée, où $-\infty \leq IN \leq +\infty$. Si le paramètre IN possède une valeur valide, l'instruction ATAN_REAL produit un résultat Q semblable à ceci :

$$\text{ATAN (IN)} = -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$



Remarque

Les instructions TRIG sont disponibles uniquement sur l'UC modèle 352.

Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
IN	IN contient la constante ou la valeur réelle de référence qui doit être calculée.
ok	La sortie ok est activée lorsque l'instruction est exécutée sans dépassement, sauf si une opération invalide se produit et/ou si IN est NaN.
Q	La sortie Q contient la valeur trigonométrique d'IN.

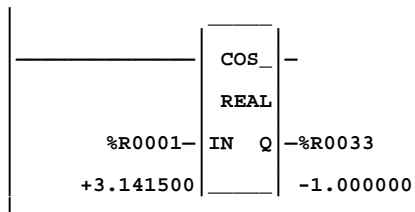
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN								•	•	•	•	
ok	•											•
Q								•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple ci-dessous, le COS de la valeur de %R0001 est placé dans %R0033.

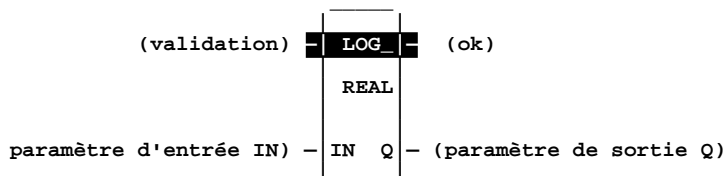


Instructions de logarithme/exposant (LOG, LN, EXP, EXPT)

Les instructions LOG, LN et EXP possèdent deux paramètres d'entrée et deux paramètres de sortie. Quand l'instruction reçoit le flux validant, elle applique l'opération de logarithme/exposant appropriée à la valeur réelle de l'entrée IN et place le résultat dans la sortie Q.

- Pour l'instruction LOG, le logarithme de base 10 d'IN est placé dans Q.
- Pour l'instruction LN, le logarithme naturel d'IN est placé dans Q.
- Pour l'instruction EXP, e est élevé à la puissance spécifiée par IN et le résultat est placé dans Q.
- Pour l'instruction EXPT, la valeur de l'entrée I1 est élevée à la puissance spécifiée par I2 et le résultat est placé dans Q. (L'instruction EXPT possède trois paramètres d'entrée et deux paramètres de sortie.)

La sortie ok reçoit le flux validant, sauf si IN est NaN (Not a Number) ou négatif.



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
IN	IN contient la valeur réelle à calculer.
ok	La sortie ok est activée lorsque l'instruction est exécutée sans dépassement, sauf si une opération invalide se produit et/ou si IN est NaN ou négatif.
Q	La sortie Q contient la valeur logarithmique/exponentielle d'IN.

Remarque

Les fonctions LOG, LN, EXP et EXPT sont disponibles uniquement sur l'UC modèle 352.

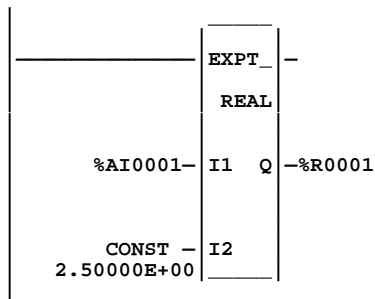
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN*								•	•	•	•	
ok	•											•
Q								•	•	•		

- * Pour l'instruction EXPT, l'entrée IN est remplacée par les paramètres d'entrée I1 et I2.
- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

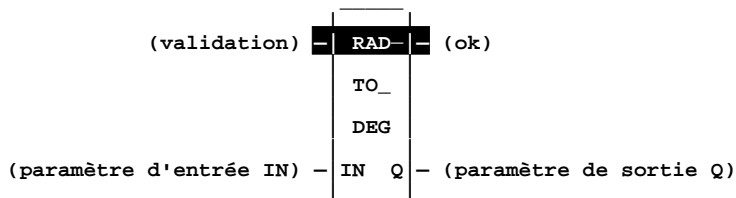
Dans l'exemple ci-dessous, la valeur de %AI001 est élevée à la puissance 2,5 et le résultat est placé dans %R001.



Conversion radian (RAD, DEG)

Quand l'instruction reçoit le flux validant, la conversion appropriée (RAD_TO_DEG ou DEG_TO_RAD, radians en degrés et inversement) est appliquée à la valeur réelle de l'entrée IN et le résultat est placé dans la sortie Q.

La sortie ok reçoit le flux validant, sauf si IN est NaN (Not a Number).



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
IN	IN contient la valeur réelle à calculer.
ok	La sortie ok est activée lorsque l'instruction est exécutée sans dépassement, sauf si IN est NaN.
Q	La sortie Q contient la valeur convertie d'IN.

Remarque

Les instructions de conversion radian sont disponibles uniquement sur l'UC 352.

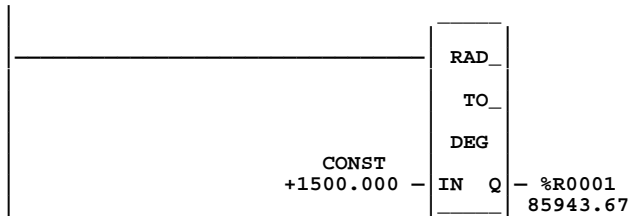
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN								•	•	•	•	
ok	•											•
Q								•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple ci-dessous, +1500 est converti en DEG et placé dans %R0001.



Section 4 : Instructions de comparaison

Les instructions de comparaison sont utilisées pour comparer deux nombres. Cette section décrit les instructions de comparaison suivantes :

Abréviation	Fonction	Description	Page
EQ	Egal à	Teste si deux nombres sont égaux.	4-42
NE	Différent de	Teste si deux nombres sont différents.	4-42
GT	Supérieur à	Teste si un nombre est supérieur à un autre.	4-42
GE	Supérieur ou égal à	Teste si un nombre est supérieur ou égal à un autre.	4-42
LT	Inférieur à	Teste si un nombre est inférieur à un autre.	4-42
LE	Inférieur ou égal à	Teste si un nombre est inférieur ou égal à un autre.	4-42
RANGE	Intervalle	Détermine si un nombre est compris dans un intervalle spécifié.	4-45

Les instructions de comparaison sont utilisées pour déterminer la relation entre deux valeurs. Lorsque l'instruction reçoit le flux validant, elle compare le paramètre d'entrée I1 au paramètre d'entrée I2. Ces paramètres doivent être du même type de données :

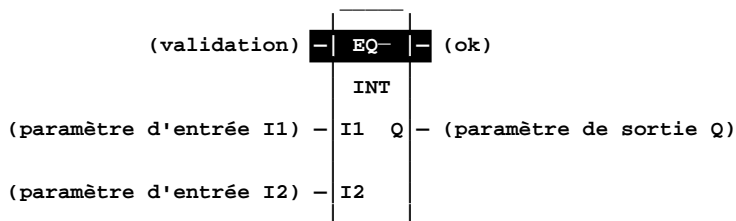
Type de données	Description
INT	Nombre entier signé.
DINT	Nombre entier signé en double précision.
REAL	Virgule flottante

Remarque

Le type de données REAL est disponible uniquement sur l'UC 352. En outre, le bloc d'instruction Range n'accepte pas le type REAL. Le bit %S0020 est mis à "1" quand une instruction de comparaison utilisant des données REAL est exécutée correctement. Il est effacé quand une entrée correspond à NaN (Not a Number).

Le type de données par défaut est le nombre entier signé. Pour comparer des entiers signés ou des entiers signés en double précision, choisir le nouveau type de données après avoir choisi l'instruction de comparaison. Pour comparer les données d'autres types ou de types différents, utiliser d'abord l'instruction de conversion appropriée (décrite dans le § 8, "Instructions de conversion") pour que les données passent à l'un des types entiers.

Si les paramètres d'entrée I1 et I2 correspondent à la comparaison spécifiée, la sortie Q reçoit le flux validant et est mise à "1" ; sinon, elle est mise à "0".



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
I1	I1 contient une constante ou une référence pour la première valeur à comparer. (I1 est sur le côté gauche de l'équation de comparaison, comme dans I1 < I2).
I2	I2 contient une constante ou une référence pour la seconde valeur à comparer. (I2 est sur le côté droit de l'équation de comparaison, comme dans I1 < I2).
Q	La sortie Q est activée lorsque I1 et I2 correspondent à la comparaison spécifiée.

Remarque

I1 et I2 doivent être des nombres valides, c'est-à-dire différents de NaN (Not a Number).

Types de mémoires valides

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		o	o	o	o		o	•	•	•	•†	
I2		o	o	o	o		o	•	•	•	•†	
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Référence validation pour les données de type INT seulement ; n'est pas valide pour DINT.
- † Les constantes sont limitées aux opérations avec nombres entiers en double précision.

Exemple :

Dans l'exemple suivant, deux nombres entiers en double précision PWR_MDE et BIN_FUL sont comparés chaque fois que %I0001 est mis à "1". Si PWR_MDE est inférieur ou égal à BIN_FUL, la bobine %Q0002 est mise à "1".



RANGE (INT, DINT, WORD)

L'instruction RANGE est utilisée pour déterminer si une valeur est comprise entre deux nombres.

Remarque

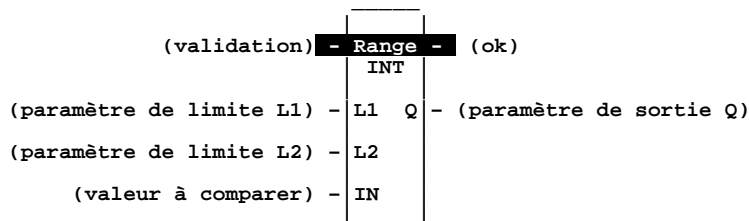
Cette instruction est disponible *uniquement* sur les UC version 4.41 ou ultérieures.

L'instruction RANGE opère sur les types de données suivants :

Type de données	Description
INT	Nombre entier signé.
DINT	Nombre entier signé en double précision.
WORD	Données de type "mot".

Le type de données par défaut est le nombre entier signé ; toutefois, il peut être changé après avoir choisi l'instruction. Pour plus d'informations sur les types de données, voir le chapitre 2, § 2, "Organisation du programme et données/références utilisateur".

Lorsque l'instruction est validée, le bloc fonctionnel RANGE compare la valeur dans le paramètre d'entrée IN par rapport à l'intervalle spécifié par les paramètres de limite L1 et L2. Lorsque la valeur est comprise dans l'intervalle spécifié par L1 et L2, inclus, le paramètre de sortie Q est mis à "1". Sinon, Q est mis à "0".



Remarque

Les paramètres de limite L1 et L2 représentent les extrémités (point de départ/d'arrivée) d'un intervalle. Il n'existe aucune connotation mini./max. ou haut/bas attribuée à l'un ou à l'autre de ces paramètres. Vous pouvez par exemple spécifier l'intervalle de 0 à 100 en attribuant 0 à L1 et 100 à L2 ou 0 à L2 et 100 à L1.

Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
L1	L1 contient le point de départ de l'intervalle.
L2	L2 contient le point d'arrivée de l'intervalle.
IN	IN contient la valeur à comparer dans l'intervalle spécifié par L1 et L2.
Q	La sortie Q est activée lorsque la valeur dans IN est comprise dans l'intervalle spécifié par L1 et L2, inclus.

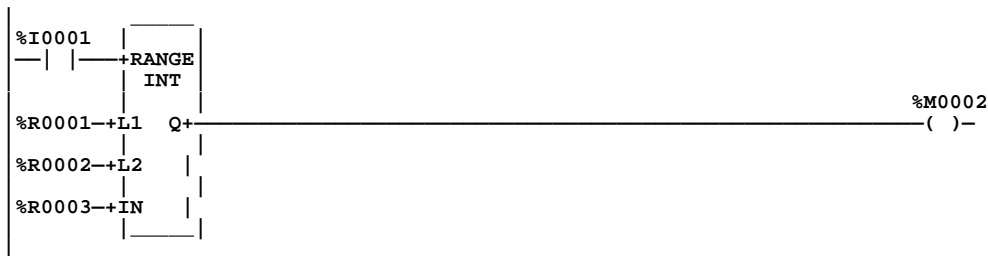
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
L1		o	o	o	o		o	•	•	•	‡	
L2		o	o	o	o		o	•	•	•	‡	
IN		o	o	o	o		o	•	•	•		
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Référence validation pour les données de type INT seulement ; n'est pas valide pour DINT.
- ‡ Les constantes sont limitées aux valeurs entières pour les opérations à nombres entiers en double précision.

Exemple 1:

Dans l'exemple suivant, on vérifie si %AI001 est compris dans un intervalle spécifié par deux constantes, 0 et 100.



Etat de validation %I0001	Valeur constante L1	Valeur constante L2	Valeur IN %AI001	Etat de Q %Q0001
"1"	100	0	< 0	"0"
"1"	100	0	0 — 100	"1"
"1"	100	0	> 100	"0"
"0"	100	0	Indifférent	"0"

Exemple 2 :

Dans cet exemple, on vérifie si %AI001 est compris dans un intervalle spécifié par deux valeurs de registre.

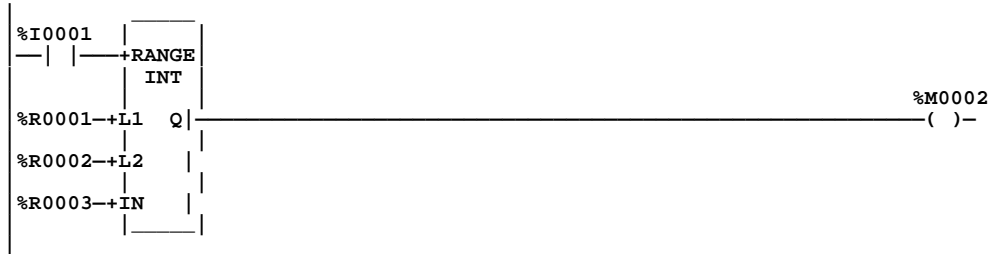


Table de vérité de RANGE				
Etat de validation %I001	Valeur L1 %R0001	Valeur L2 %R0002	Valeur IN %AI001	Etat de Q %Q0001
"1"	500	0	< 0	"0"
"1"	500	0	0 — 500	"1"
"1"	500	0	> 500	"0"
"0"	500	0	Indifférent	

Section 5 : Opérations Logiques

Les opérations logiques exécutent des opérations de comparaison logiques et de transfert sur des chaînes binaires. Les opérations AND, OR, XOR et NOT fonctionnent sur un seul mot. Les autres opérations logiques peuvent fonctionner sur plusieurs mots, avec une longueur de chaîne maximale de 256 mots. Toutes les opérations logiques exigent des données de type WORD (mot).

Même si les données doivent être spécifiées en incréments de 16 bits, ces instructions opèrent sur les données comme si elles étaient une chaîne binaire continue, le bit 1 du premier mot étant le bit de poids faible (BPf), et le dernier bit du dernier mot le bit de poids fort (BPF). Par exemple, si vous avez spécifié trois mots de données commençant à la référence %R0100, l'instruction opérerait sur une chaîne de 48 bits contigus.

%R0100	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	← bit 1 (LSB)
%R0101	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	
%R0102	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	
	↑																
	(MSB)																

Remarque

Le chevauchement des intervalles d'adresses de références des E/S, dans les instructions multi-mots, peut produire des résultats imprévisibles.

Les opérations logiques suivantes sont décrites dans cette section :

Abréviation	Fonction	Description	Page
AND	Et logique	Si un bit dans la chaîne binaire I1 et le bit correspondant dans la chaîne binaire I2 sont tous deux à 1, place un 1 dans la position correspondante dans la chaîne de sortie Q.	4-50
OR	Ou logique	Si un bit dans la chaîne binaire I1 et le bit correspondant dans la chaîne binaire I2 sont tous deux à 0, place un 0 dans la position correspondante dans la chaîne de sortie Q.	4-50
XOR	Ou exclusif logique	Si un bit dans la chaîne binaire I1 et le bit correspondant dans la chaîne I2 sont différents, place un 1 dans la position correspondante dans la chaîne binaire de sortie.	4-52
NOT	Inversion logique	Met chaque bit dans la chaîne binaire de sortie Q à l'état opposé du bit correspondant dans la chaîne binaire I1.	4-54
SHL	Décalage à gauche	Décale vers la gauche tous les bits d'un mot ou d'une chaîne de mots, d'un nombre spécifié de positions.	4-56
SHR	Décalage à droite	Décale vers la droite tous les bits d'un mot ou d'une chaîne de mots, d'un nombre spécifié de positions.	4-56
ROL	Rotation à gauche	Permute à gauche tous les bits d'une chaîne, d'un nombre spécifié de positions.	4-59
ROR	Rotation à droite	Permute à droite tous les bits d'une chaîne, d'un nombre spécifié de positions.	4-57
BTST	Tester un bit	Teste un bit d'une chaîne binaire pour déterminer s'il est actuellement à 1 ou 0.	4-61
BSET	Mise à "1"	Met à "1" un bit d'une chaîne binaire.	4-63
BCLR	Mise à "0"	Met à "0" un bit d'une chaîne binaire.	4-63
BPOS	Position binaire	Recherche, dans une chaîne binaire, un bit mis à "1".	4-65
MSKCMP	Comparaison masquée	Compare le contenu de deux chaînes de bit séparées avec possibilité de masquer des bits sélectionnés (disponible sur les UC version 4.5 ou ultérieures).	4-67

AND et OR (WORD)

A chaque cycle où le flux validant est reçu, l'instruction AND (ou OR) examine les deux chaînes binaires bit à bit en commençant, dans chacune, par le bit de poids faible.

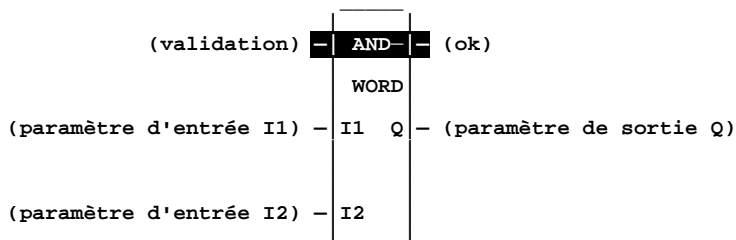
L'instruction AND, si les deux bits comparés sont à "1", place un "1" dans la position correspondante dans la chaîne de sortie Q. Si l'un des deux bits ou les deux sont à "0", elle place un "0" dans la chaîne Q dans cette position.

L'instruction AND est utile pour créer des masques, où seuls certains bits sont filtrés (ceux en face d'un 1 dans le masque), et tous les autres bits sont mis à "0". Cette instruction peut être également utilisée pour mettre à "0" la zone choisie de la référence de mots en exécutant une opération Et avec une constante égale à zéro "0". Les chaînes binaires I1 et I2 spécifiées peuvent se chevaucher.

L'instruction OR, si l'un ou l'autre bit comparés ou les deux sont à "1", place un "1" dans la position correspondante dans la chaîne de sortie Q. Si les deux bits sont à "0", elle place un "0" dans la chaîne Q dans cette position.

L'instruction OR est utile pour combiner des chaînes binaires et contrôler plusieurs entrées en utilisant une fonction logique simple. Cette instruction équivaut à deux contacts de relais en parallèle, multipliés par le nombre de bits dans la chaîne. Elle peut être utilisée pour commander des voyants directement depuis l'état des entrées, ou pour superposer des conditions de clignotement sur des voyants d'état.

L'instruction passe le flux validant vers la droite chaque fois qu'elle le reçoit.



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
I1	I1 contient une constante ou une référence pour le premier mot de la première chaîne.
I2	I2 contient une constante ou une référence pour le premier mot de la seconde chaîne.
ok	La sortie ok est activée chaque fois que Validation est activé.
Q	La sortie Q contient le résultat de l'opération.

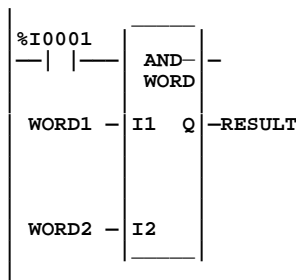
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		•	•	•	•	•	•	•	•	•	•	
I2		•	•	•	•	•	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.
- † %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple :

Dans l'exemple suivant, chaque fois que l'entrée %I0001 est mise à "1", les chaînes de 16 bits représentées par les symboles WORD1 et WORD2 sont examinées. Les résultats de Et logique sont placés dans la chaîne de sortie RESULT.



WORD1	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
WORD2	1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1
RESULT	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0

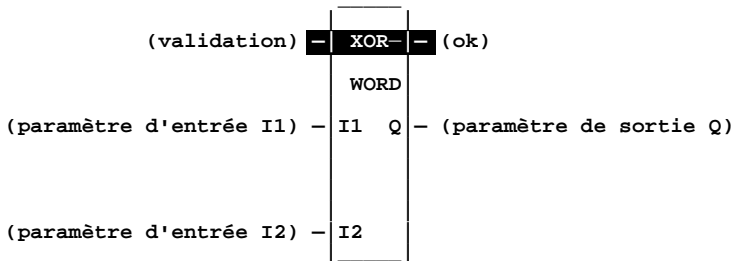
XOR (WORD)

L'instruction Ou exclusif (XOR) est utilisée pour comparer chaque bit de la chaîne binaire I1 avec le bit correspondant dans la chaîne I2. Si les bits sont différents, un 1 est placé dans la position correspondante dans la chaîne binaire de sortie.

A chaque cycle, si le flux validant est reçu, l'instruction examine les deux chaînes binaires bit à bit, en commençant par le bit de poids faible. Pour les bits examinés, si un seul est à "1", la chaîne binaire Q place un "1" dans la position correspondante. La sortie OK est active chaque fois que Validation est activé.

Si la chaîne I2 et la chaîne de sortie Q commencent à la même référence, un "1" placé dans la chaîne I1 force le bit correspondant dans la chaîne I2 à alterner entre 0 et 1, changeant d'état à chaque scrutation tant que le flux validant est reçu. Des cycles plus longs peuvent être programmés en pulsant le flux validant vers l'instruction à deux fois la fréquence désirée du clignotement ; l'impulsion du flux validant devrait être longue d'une scrutation (bobine de type à impulsion ou temporisateur à réinitialisation automatique).

L'instruction XOR est utile pour comparer rapidement deux chaînes binaires, ou pour faire alterner un groupe de bits à une fréquence d'un état "1" toutes les deux scrutations.



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
I1	I1 contient une constante ou une référence pour le premier mot devant subir une opération XOR.
I2	I2 contient une constante ou une référence pour le second mot devant subir une opération XOR.
ok	La sortie ok est activée chaque fois que Validation est activé.
Q	La sortie Q contient le résultat de I1 ayant subi une opération XOR avec I2.

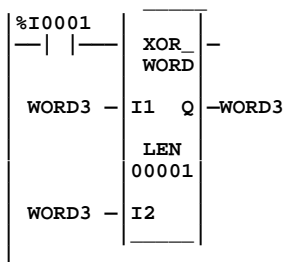
Types de mémoires valides :

Paramètre	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		•	•	•	•	•	•	•	•	•	•	
I2		•	•	•	•	•	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.
- † %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple :

Dans l'exemple suivant, chaque fois que %I0001 est mise à "1", la chaîne binaire représentée par le symbole WORD3 est mise entièrement à zéro.

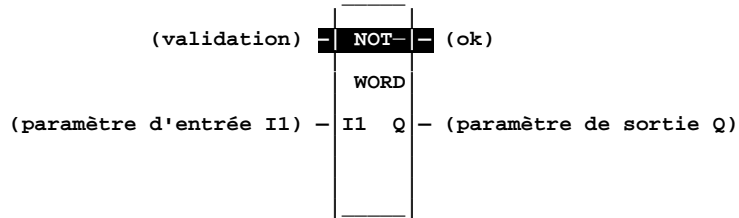


I1 (WORD3)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
I2 (WORD3)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
Q (WORD3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOT (WORD)

L'instruction NOT (Pas) est utilisée pour donner à chaque bit de la chaîne binaire de sortie Q l'état opposé à celui du bit correspondant dans la chaîne binaire I1.

Tous les bits changent d'état à chaque cycle si le flux validant est reçu, faisant de la chaîne de sortie Q le complément logique de I1. La sortie OK est activée chaque fois que Validation est activé.



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
I1	I1 contient la constante ou la référence du mot à inverser.
ok	La sortie ok est activée chaque fois que Validation est activé.
Q	La sortie Q contient la négation (NOT) de I1.

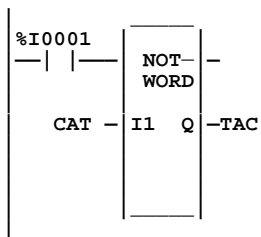
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		•	•	•	•	•	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.
- † %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

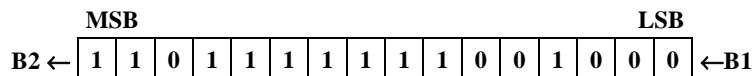
Exemple :

Dans l'exemple suivant, chaque fois que l'entrée %I0001 est mise à "1", la chaîne binaire représentée par le symbole TAC est inversée par rapport à la chaîne binaire CAT.

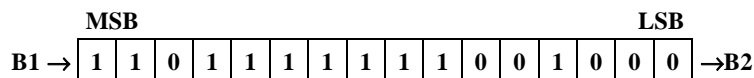


SHL et SHR (WORD)

L'instruction de Décalage à gauche (SHL) est utilisée pour décaler à gauche tous les bits d'un mot ou d'un groupe de mots, d'un nombre déterminé de positions. Au moment du décalage, le nombre déterminé de bits est décalé hors de la chaîne de sortie vers la gauche. A mesure que les bits sont décalés hors de la chaîne par son extrémité haute, le même nombre de bits est entré par décalage à l'extrémité basse.



L'instruction de Décalage à droite (SHR) est utilisée pour décaler vers la droite tous les bits d'un mot ou groupe de mots, d'un nombre déterminé de positions. Lorsque le décalage se produit, le nombre déterminé de bits est décalé hors de la chaîne de sortie. A mesure que les bits sont décalés hors de la chaîne par son extrémité basse, le même nombre de bits est entré par décalage à l'extrémité haute.



Une chaîne de 1 à 256 mots de longueur peut être choisie pour l'une ou l'autre instruction.

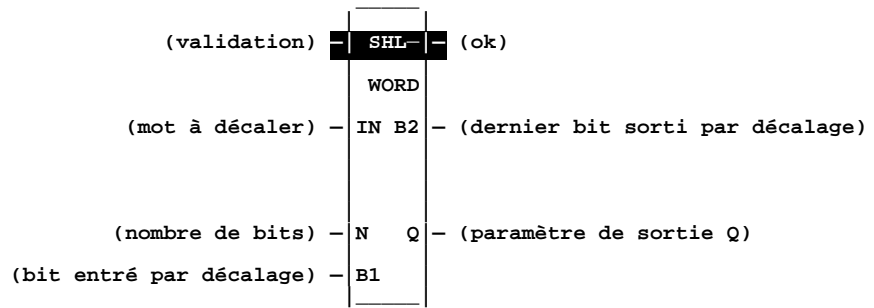
Si le nombre de bits à décaler (N) est supérieur au nombre de bits du tableau (LEN) * 16 ou égal à zéro, le tableau (Q) est complété à l'aide de copies du bit d'entrée (B1) tandis que le bit d'entrée est copié vers le flux validant de la sortie (B2). Si le nombre de bits à décaler est égal à zéro, aucun décalage n'est effectué, le tableau d'entrée est copié dans le tableau de sortie et le bit d'entrée (B1) est copié vers le flux validant.

Les bits entrés par décalage au début de la chaîne sont déterminés par le paramètre d'entrée B1. Si une longueur supérieure à 1 a été spécifiée comme nombre de bits à décaler, chacun des bits reçoit la même valeur (0 ou 1). Ce peut être :

- La sortie booléenne d'une autre instruction de programme.
- Tous à 1. Pour cela, utiliser le symbole spécial de référence ALW_ON comme permissif vers l'entrée B1.
- Tous à 0. Pour cela, utiliser le symbole spécial de référence ALW_OFF comme permissif vers l'entrée B1.

L'instruction SHL ou SHR passe le flux validant vers la droite, sauf si le nombre de bits à décaler est égal à zéro.

La sortie Q correspond à la copie décalée de la chaîne d'entrée. Si vous souhaitez décaler la chaîne d'entrée, le paramètre de sortie Q doit utiliser le même emplacement mémoire que le paramètre d'entrée IN. La totalité de la chaîne décalée est écrite à chaque scrutation où le flux validant est reçu. La sortie B2 correspond au dernier bit décalé. Si quatre bits ont été décalés, par exemple, B2 correspond au quatrième bit.



Paramètres :

Paramètre	Description
validation	Le décalage est exécuté lorsque l'instruction est validée.
IN	IN contient le premier mot à décaler.
N	N contient le nombre de positions (bits) décalant le tableau.
B1	B1 contient la valeur binaire à entrer par décalage dans le tableau.
B2	B2 contient la valeur binaire du dernier bit décalé hors du tableau.
Q	La sortie Q contient le premier mot du tableau décalé.
LEN	LEN est le nombre de mots décalés dans le tableau.

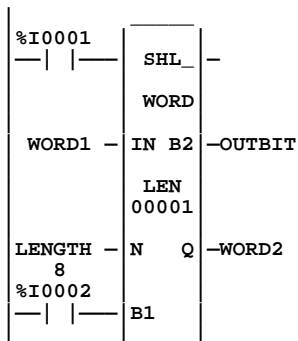
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	•	•	•	•	•		
N		•	•	•	•		•	•	•	•	•	
B1	•											
B2	•											•
Q		•	•	•	•	•†	•	•	•	•		

• Référence ou position validation où le flux validant peut traverser l'instruction.
 † %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple :

Dans l'exemple suivant, chaque fois que l'entrée %I0001 est mise à "1", la chaîne binaire de sortie représentée par le symbole WORD2 devient une copie de WORD1, décalée à gauche du nombre de bits représentés par le symbole LENGTH (longueur). Les bits d'ouverture résultant, au début de la chaîne de sortie, prennent la valeur de %I0002.



ROL et ROR (WORD)

L'instruction de Rotation à gauche (ROL) est utilisée pour permuter vers la gauche tous les bits d'une chaîne d'un nombre spécifié de positions. Au moment de la rotation, le nombre spécifié de bits est sorti de la chaîne d'entrée par permutation vers la gauche, puis entré à nouveau dans la chaîne par la droite.

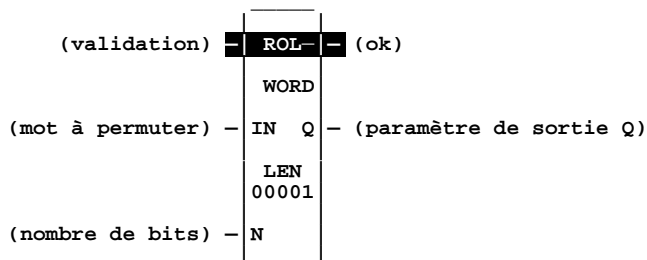
L'instruction de Rotation à droite (ROR) est utilisée pour permuter les bits d'une chaîne vers la droite. Au moment de la rotation, le nombre spécifié de bits est sorti de la chaîne d'entrée par permutation vers la droite, puis entré à nouveau dans la chaîne par la gauche.

Une chaîne de 1 à 256 mots de longueur peut être choisie pour l'une ou l'autre instruction.

Le nombre de positions spécifiées pour la rotation doit être supérieur à zéro et inférieur au nombre de bits dans la chaîne. Sinon, aucune rotation ne se produit et aucun flux validant n'est généré.

L'instruction ROL ou ROR dirige le flux validant vers la droite, sauf si le nombre spécifié de bits à permuter, est supérieur à la longueur totale de la chaîne ou inférieur à zéro.

Le résultat est placé dans la chaîne de sortie Q. Si vous souhaitez faire pivoter la chaîne d'entrée, le paramètre de sortie Q doit utiliser le même emplacement mémoire que le paramètre d'entrée IN. L'ensemble de la chaîne permutée est écrite à chaque scrutation où le flux validant est reçu.



Paramètres :

Paramètre	Description
validation	La rotation est exécutée lorsque l'instruction est validée.
IN	IN contient le premier mot à permuter.
N	N contient le nombre de positions permutées dans le tableau.
ok	La sortie ok est activée lorsque la rotation est activée et si la longueur de la rotation n'est pas supérieure à la taille du tableau.
Q	La sortie Q contient le premier mot du tableau permuté.
LEN	LEN est le nombre de mots permutés dans le tableau.

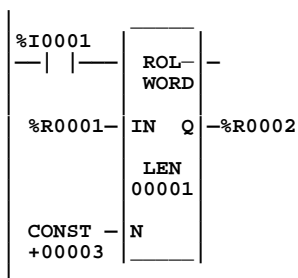
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	•	•	•	•	•		
N		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

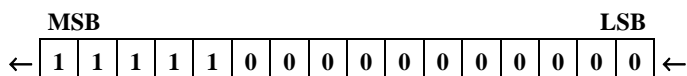
- Référence ou position validation où le flux validant peut traverser l'instruction.
- † %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple :

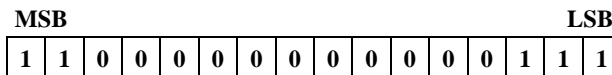
Dans l'exemple suivant, chaque fois que l'entrée %I0001 est mise à "1", la chaîne binaire d'entrée %R0001 est permutée de trois bits et le résultat est placé dans %R0002. Une fois cette instruction exécutée, la chaîne binaire d'entrée %R0001 reste inchangée. Si la même référence est utilisée pour IN et Q, une rotation se produit à la place.



%R0001:



%R0002 (après la mise à "1" de %I0001) :

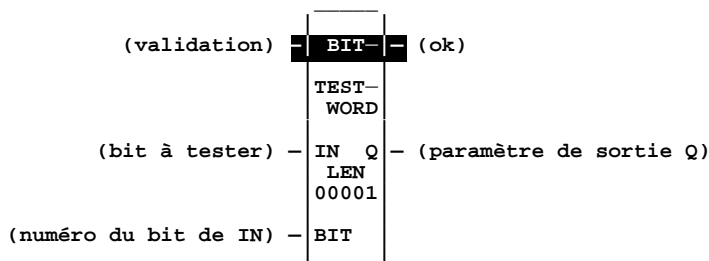


BTST (WORD)

L'instruction Tester un bit (BTST) est utilisée pour déterminer si l'état d'un bit dans une chaîne binaire est actuellement à "1" ou à "0". Le résultat du test est placé dans la sortie Q.

A chaque cycle, si le flux validant est reçu, l'instruction BTST met sa sortie Q au même état que le bit spécifié. Si un registre, plutôt qu'une constante, est utilisé pour spécifier le numéro du bit, le même bloc fonctionnel peut tester différents bits lors de cycles successifs. Si la valeur de BIT est hors de l'intervalle ($1 \leq \text{BIT} \leq (16 * \text{LEN})$), Q est mise à "0".

Une chaîne de 1 à 256 mots peut être choisie.



Paramètres :

Paramètre	Description
validation	L'opération Tester un bit est exécutée lorsque l'instruction est validée.
IN	IN contient le premier mot de données sur lequel porte l'opération.
BIT	BIT contient le numéro de bit de IN qui doit être testé. L'intervalle valide est ($1 \leq \text{BIT} \leq (16 * \text{LEN})$).
ok	La sortie ok est activée chaque fois que Validation est activé et que BIT est supérieur à la longueur de la chaîne ou égal à zéro.
Q	La sortie Q est activée si le bit testé était un 1.
LEN	LEN est le nombre de mots à tester dans la chaîne.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	•	•	•	•	•		
BIT		•	•	•	•		•	•	•	•	•	
Q	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, chaque fois que l'entrée %I0001 est mise à "1", le bit à la position contenue dans la référence PICKBIT est testé. Ce bit fait partie de la chaîne PRD_CDE. S'il est à 1, la sortie Q laisse passer le flux validant et la bobine %Q0001 est mise à "1".

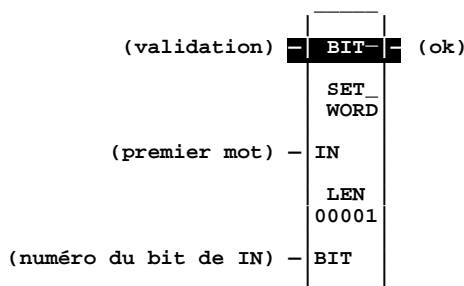


BSET et BCLR (WORD)

L'instruction de mise à "1" (BSET) est utilisée pour mettre à "1" un bit d'une chaîne binaire.
L'instruction de mise à "0" (BCLR) est utilisée pour mettre à "0" un bit d'une chaîne.

A chaque cycle, si le flux validant est reçu, l'instruction met le bit spécifié à "1" pour l'instruction BSET, ou à "0" pour l'instruction BCLR. Si une variable (registre), plutôt qu'une constante, est utilisée pour spécifier le numéro du bit, le même bloc fonctionnel peut mettre à "1" différents bits lors de cycles successifs.

Une chaîne de 1 à 256 mots de longueur peut être sélectionnée. L'instruction dirige le flux validant vers la droite, sauf si la valeur de BIT est hors de l'intervalle ($1 \leq \text{BIT} \leq (16 * \text{LEN})$). A ce moment-là, ok est mise à "0".



Paramètres :

Paramètre	Description
validation	L'opération au niveau du bit est exécutée lorsque l'instruction est validée.
IN	IN contient le premier mot de données sur lequel porte l'opération.
BIT	BIT contient le numéro du bit de IN devant être mis à "1" ou à "0". L'intervalle valide est ($1 \leq \text{BIT} \leq (16 * \text{LEN})$).
ok	La sortie ok est activée chaque fois que Validation est activé.
LEN	LEN est le nombre de mots dans la chaîne binaire.

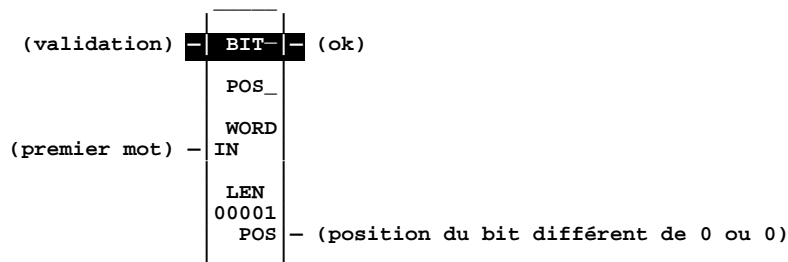
BPOS (WORD)

L'instruction Position binaire (BPOS) est utilisée pour rechercher un bit mis à "1" dans une chaîne binaire.

A chaque cycle, si le flux validant est reçu, l'instruction scrute la chaîne binaire en commençant par le premier bit du premier mot. L'instruction stoppe la scrutation lorsqu'un bit égal à "1" a été trouvé ou que toute la longueur de la chaîne a été scrutée.

POS est mis sur la position, dans la chaîne binaire, du premier bit différent de zéro ; POS est mis à "0" si aucun bit différent de zéro n'est trouvé.

Une longueur de chaîne de 1 à 256 mots peut être sélectionnée. La sortie ok est activée chaque fois que Validation est activé.



Paramètres :

Paramètre	Description
validation	Une recherche de bit est exécutée lorsque l'instruction est validée.
IN	IN contient le premier mot de données sur lequel porte l'opération.
ok	La sortie ok est activée chaque fois que Validation est activé.
POS	Position où le premier bit différent de zéro est trouvé, ou zéro si un bit différent de zéro n'est pas trouvé.
LEN	LEN est le nombre de mots dans la chaîne binaire.

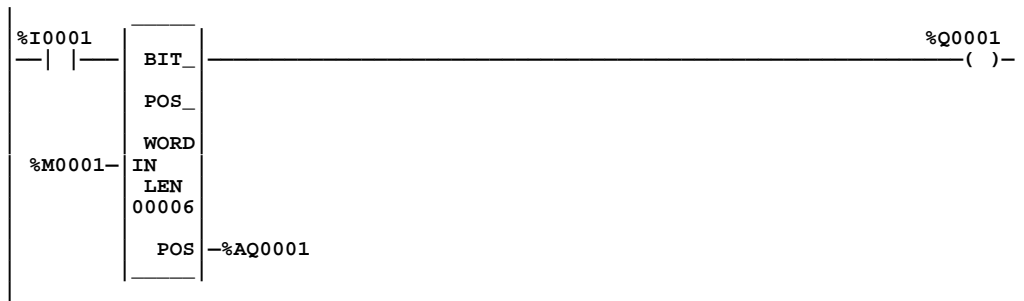
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	•	•	•	•	•		
POS		•	•	•	•		•	•	•	•		
ok	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, si %I0001 est mise à "1", la chaîne binaire commençant à %M0001 est recherchée jusqu'à ce qu'un bit égal à "1" soit trouvé, ou que 6 mots aient été recherchés. La bobine %Q0001 est mise à "1". Si un bit égal à 1 est trouvé, sa position dans la chaîne binaire est écrite dans %A0001. Si %I0001 est mise à "1", le bit %M0001 est 0, et le bit %M0002 est 1, puis la valeur écrite dans %A0001 est 2.



MSKCMP (WORD, DWORD)

L'instruction de comparaison masquée (MSKCMP) (*disponible avec les UC version 4.41 ou ultérieures*) permet de comparer le contenu de deux chaînes de bits séparées et de masquer les bits sélectionnés. La longueur des chaînes de bits à comparer est spécifiée par le paramètre LEN (où la valeur de LEN détermine le nombre de mots de 16 bits de l'instruction MSKCMPW et le nombre de mots de 32 bits de l'instruction MSKCMPD).

Quand la logique qui contrôle l'entrée de validation vers l'instruction passe le flux validant à l'entrée de validation (EN), l'instruction commence à comparer les bits de la première chaîne avec les bits correspondants de la deuxième chaîne. La comparaison se poursuit jusqu'à ce qu'une incohérence soit trouvée ou jusqu'à la fin de la chaîne.

L'entrée BIT permet de stocker le numéro du bit auquel la comparaison suivante doit commencer (où 0 désigne le premier bit de la chaîne). La sortie BN permet de stocker le numéro du bit dans lequel a eu lieu la dernière comparaison (où 1 désigne le premier bit de la chaîne). L'attribution de la même référence à BIT et BN permet de commencer la comparaison à la position de bit qui suit une incohérence ou au début, si tous les bits ont été comparés avec succès dès le prochain appel du bloc fonctionnel.

Si vous souhaitez commencer la comparaison suivante à un autre endroit dans la chaîne, vous pouvez spécifier des références différentes pour BIT et BN. Si la valeur de BIT correspond à un emplacement situé au-delà de la fin de la chaîne, BIT est remis à "0" avant le début de la comparaison suivante.

Si tous les bits d'I1 et d'I2 sont identiques

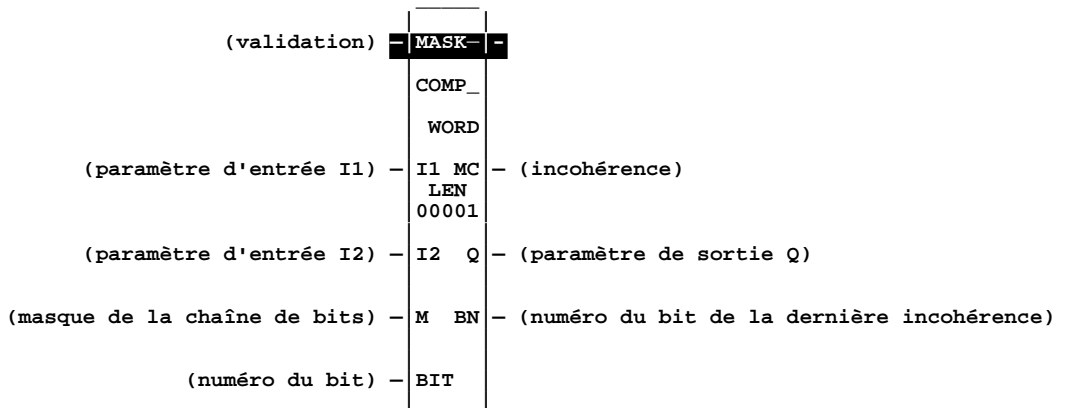
Si tous les bits correspondants des chaînes I1 et I2 concordent, l'instruction met à "0" la sortie d'incohérence MC et attribue à BN le nombre de bits le plus élevé dans les chaînes d'entrée. Ensuite, la comparaison s'arrête. Au prochain appel de MSKCMPW, elle est remise à "0".

En présence d'une incohérence

Lorsque deux bits concurrents ne sont pas identiques, l'instruction vérifie le numéro de bit correspondant dans la chaîne M (masque). Si le bit du masque a la valeur 1, la comparaison se poursuit jusqu'à la prochaine incohérence ou jusqu'à la fin des chaînes d'entrée.

Si une incohérence est trouvée et si le bit du masque correspondant a la valeur 0, l'instruction exécute les opérations suivantes :

1. Elle met à "1" le bit du masque correspondant dans M.
2. Elle met à "1" la sortie de l'incohérence (MC).
3. Elle met à jour la chaîne des bits de sortie Q de façon à ce qu'elle concorde avec le nouveau contenu de la chaîne du masque M.
4. Elle attribue à la sortie du numéro de bit (BN) le numéro du bit de l'incohérence.
5. Elle arrête la comparaison.



Paramètres :

Paramètre	Description
validation	Valide la logique pour activer l'instruction.
I1	Référence de la première chaîne de bits à comparer.
I2	Référence de la deuxième chaîne de bits à comparer.
M	Référence du masque de la chaîne de bits.
BIT	Référence du numéro de bit auquel la prochaine comparaison doit commencer.
MC	Logique utilisateur déterminant si une incohérence est présente.
Q	Copie de sortie de la chaîne de bits du masque (M).
BN	Numéro du bit auquel a eu lieu la dernière comparaison.
LEN	Nombre de mots de la chaîne de bits.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
I1		o	o	o	o	o	o	•	•	•		
I2		o	o	o	o	o	o	•	•	•		
M		o	o	o	o	o†	o	•	•	•		
BIT		•	•	•	•	•	•	•	•	•	•	
LEN											•‡	
MC	•											•
Q		o	o	o	o	o†	o	•	•	•		
BN		•	•	•	•	•	•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Référence valide pour les données WORD uniquement et non pour DWORD.
- † %SA, %SB, %SC uniquement ; %S ne peut pas être utilisé.
- ‡ La valeur maximale de la constante est 4095 pour WORD et 2047 pour DWORD.

Exemple :

L'exemple ci-dessous exécute le bloc fonctionnel MSKCMPW après la première scrutation. %M0001 à %M0016 sont comparés avec %M0017 à %M0032. %M0033 à %M0048 contiennent la valeur du masque. La valeur de %R0001 détermine à quel emplacement de bit commence la comparaison entre les deux chaînes d'entrée. Avant l'exécution du bloc fonctionnel, les références ci-dessus contiennent les éléments suivants :

(I1) – %M0001 = 6C6Ch =

0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(I2) – %M0017 = 606Fh =

0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(M/Q) – %M0033 = 000Fh =

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(BIT/BN) – %R0001 = 0

(MC) – %Q0001 = OFF

Après l'exécution du bloc fonctionnel, ces références contiennent les éléments suivants :

(I1) – %M0001 =

0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(I2) – %M0017 =

0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(M/Q) – %M0033

0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(BIT/BN) – %R0001 = 8

(MC) – %Q0001 = ON

Représentation du diagramme en échelle



Notez que dans l'exemple ci-dessus, nous avons utilisé le contact FST_SCN pour forcer une seule et unique exécution. Dans le cas contraire, la comparaison masquée se répète, sans pour autant produire les résultats souhaités.

Section 6 : Instructions de transfert de données

Ces instructions offrent des fonctions de base permettant de transférer des données. Cette section décrit les instructions de transfert de données suivantes :

Abréviation	Fonction	Description	Page
MOVE	Transfert	Copie les données sous forme binaire. La longueur max. autorisée est de 256 mots, à l'exception de MOVE_BIT de 256 bits. Les données peuvent être transférées dans un type de données différent sans conversion préalable.	4-72
BLKMOV	Transfert de bloc	Copie un bloc de sept constantes dans un emplacement mémoire spécifié. Les constantes sont entrées comme éléments de l'instruction.	4-75
BLKCLR	Mise à "0" de bloc	Remplace le contenu d'un bloc de données par des zéros. Cette instruction peut être utilisée pour mettre à "0" une zone de mémoire de bits (%I, %Q, %M, %G ou %T) ou de mots (%R, %AI ou %AQ). La longueur max. autorisée est de 256 mots.	4-77
SHFR	Registre de décalage	Décalle un ou plusieurs mots de données dans un tableau. La longueur max. autorisée est 256 mots.	4-79
BITSEQ	Séquenceur binaire	Exécute une séquence de décalage binaire dans un tableau binaire. La longueur max. autorisée est de 256 mots.	4-82
COMMREQ	Demande de communication	Permet au programme de communiquer avec un coprocesseur, tel qu'un module de communication Genius ou un module coprocesseur programmable.	4-85

MOVE (BIT, INT, WORD, REAL)

L'instruction de transfert (MOVE) est utilisée pour copier des données d'une position à une autre. Comme les données sont copiées en format binaire, il n'est pas nécessaire que la nouvelle position soit du même type de données que la position d'origine.

L'instruction MOVE a 2 paramètres d'entrée et 2 paramètres de sortie. Lorsque l'instruction reçoit le flux validant, elle copie les données du paramètre d'entrée IN vers le paramètre de sortie Q sous forme binaire. Si les données sont transférées d'une position dans la mémoire logique à une autre (par exemple, de la mémoire %I à la mémoire %T), les informations de transition associées aux éléments de la mémoire logique sont également copiées dans la nouvelle position. Les données du paramètre d'entrée restent inchangées sauf s'il existe un chevauchement dans la destination d'origine.

Pour le type de données BIT, d'autres éléments doivent être pris en compte. Si un tableau de type BIT spécifié dans le paramètre Q n'englobe pas la totalité des bits d'un octet, les bits de transition associés à cet octet (qui ne figurent pas dans le tableau) sont effacés quand l'instruction MOVE_BIT reçoit le flux validant.

L'entrée IN peut être une référence pour les données à transférer ou une constante. Si une constante est spécifiée, sa valeur est placée dans la position spécifiée par la référence de sortie. Par exemple, si une valeur constante de 4 est spécifiée pour IN, 4 est placé dans l'emplacement mémoire spécifié par Q. Si la longueur est supérieure à 1 et qu'une constante est spécifiée, cette dernière est placée dans l'emplacement mémoire spécifié par Q et les emplacements suivants, jusqu'à la longueur spécifiée. Par exemple, si la valeur constante 9 est spécifiée pour IN et que la longueur est de 4, 9 est placé dans l'emplacement mémoire spécifié par Q et les trois emplacements suivants.

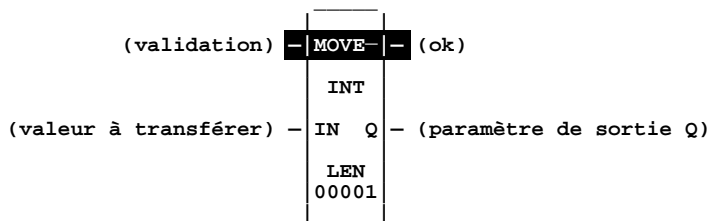
L'opérande LEN spécifie le nombre de

- mots à transférer pour MOVE_INT et MOVE_WORD,
- bits à transférer pour MOVE_BIT,
- données de type Real à transférer pour MOVE_REAL.

Remarque

Le type de données REAL est disponible uniquement sur les UC modèle 352.

La sortie ok est activée chaque fois que l'instruction est validée.



Paramètres :

Paramètre	Description
validation	Le transfert est exécuté lorsque l'instruction est validée.
IN	IN contient la valeur à transférer. Pour MOVE_BIT, toute référence logique peut être utilisée ; le rang du bit de départ peut être quelconque. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
ok	La sortie ok est activée chaque fois que l'instruction est validée.
Q	Lorsque le transfert est exécuté, la valeur à IN est écrite dans Q. Pour MOVE_BIT, toute référence logique peut être utilisée ; il n'est pas nécessaire d'avoir un alignement de 8 bits. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
LEN	LEN spécifie le nombre de mots ou de bits à transférer. Pour MOVE_WORD et MOVE_INT, LEN doit être compris entre 1 et 256 mots. Pour MOVE_BIT, lorsque IN est une constante, LEN doit être compris entre 1 et 16 bits ; sinon, LEN doit être compris entre 1 et 256.

Remarque

Sur les UC 351 et 352, les instructions MOVE_INT et MOVE_WORD ne fonctionnent pas correctement lorsque les paramètres IN et Q se chevauchent.

Notez aussi que l'UC 352 est la seule UC à virgule flottante de la Série 90-30 qui est actuellement disponible. Elle est donc la seule à pouvoir exécuter l'instruction MOVE_REAL.

Types de mémoires valides :

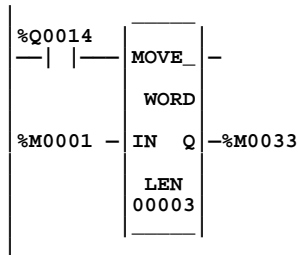
Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	o	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	o†	•	•	•	•		

Remarque : Pour les données REAL, les seuls types valides sont %R, %AI et %AQ.

- Référence validation pour données de type BIT, INT, WORD, ou position où le flux validant peut traverser l'instruction. Pour MOVE_BIT, le bit de départ des références utilisateur logiques %I, %Q, %M et %T peut être quelconque.
- o Référence validation pour données de type BIT ou WORD seulement ; n'est pas valide pour INT.
- † %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Exemple 1:

Lorsque l'entrée de validation %Q0014 est à "1", 48 bits sont transférés de l'emplacement mémoire %M0001 à l'emplacement mémoire %M0033. Même si la destination chevauche la source sur 16 bits, le transfert s'effectue correctement (sauf sur les UC 351 et 352, comme décrit plus haut).



Avant d'utiliser l'instruction de Transfert:

ENTREE (%M0001 à %M0048)

	1															
%M0016	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
%M0032	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
%M0048	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Après avoir utilisé l'instruction de Transfert:

ENTREE (%M0033 à %M0080)

	33															
%M0048	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
%M0064	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
%M0080	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Exemple 2:

Dans cet exemple, chaque fois que %I0001 est mise à "1", les trois bits %M0001, %M0002, et %M0003 sont transférés à %M0100, %M0101 et %M0102 respectivement. La bobine %Q0001 est mise à "1".



BLKMOV (INT, WORD, REAL)

L'instruction de Transfert de bloc (BLKMOV) est utilisée pour copier un bloc de sept constantes vers un emplacement spécifié.

Remarque

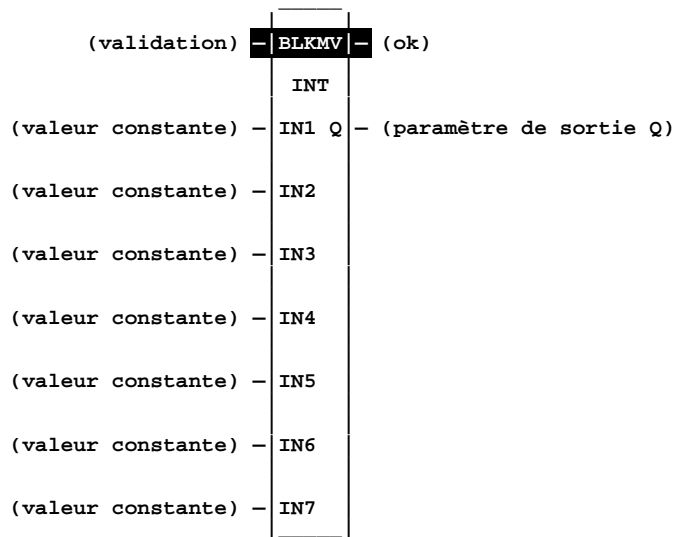
Le type de données REAL est disponible uniquement sur les UC 352.

L'instruction BLKMOV possède 8 paramètres d'entrée et 2 paramètres de sortie. Lorsque l'instruction reçoit le flux validant, elle copie les valeurs des constantes dans les positions consécutives, en commençant à la destination spécifiée dans la sortie Q. La sortie Q ne peut être l'entrée d'une autre instruction de programme.

Remarque

Pour BLKMOV_INT, les valeurs de IN1 à IN7 sont affichées en décimal signées. Pour BLKMOV_WORD, IN1 à IN7 sont affichées en hexadécimal. Pour BLKMOV_REAL, IN1 à IN7 sont affichées au format Real.

La sortie ok est activée chaque fois que l'instruction est validée.



Paramètres

Paramètre	Description
validation	Le transfert de bloc est exécuté lorsque l'instruction est validée.
IN1— IN7	IN1 à IN7 contiennent sept valeurs de constantes.
ok	La sortie ok est activée chaque fois que l'instruction est validée.
Q	La sortie Q contient le premier nombre entier du tableau transféré. IN1 est transféré dans Q.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN1 — IN7											•	
ok	•											•
Q		•	•	•	•	o†	•	•	•	•		

Remarque : Pour les données REAL, les seuls types valides sont %R, %AI et %AQ.

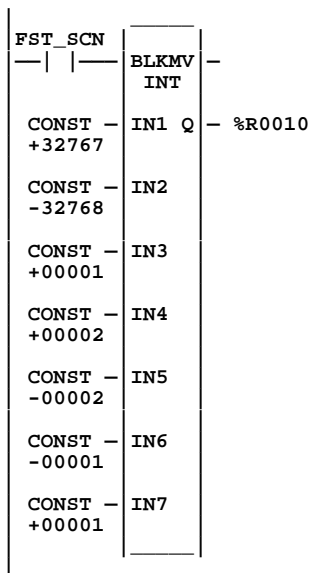
- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Référence validation pour les données de type WORD seulement ; n'est pas valide pour INT ou REAL.
- † %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Remarque

L'UC 352 est la seule UC à virgule flottante de la Série 90-30 qui est actuellement disponible. Elle est donc la seule à pouvoir exécuter l'instruction BLKMOV_REAL.

Exemple :

Dans l'exemple suivant, lorsque l'entrée de validation représentée par le symbole FST_SCN est mise à "1", l'instruction BLKMOV copie les sept constantes d'entrée dans les emplacements mémoire %R0010 à %R0016.

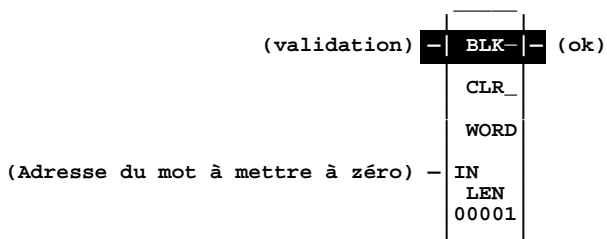


BLKCLR (WORD)

L'instruction Mise à "0" d'un bloc (BLKCLR) est utilisée pour mettre à zéro un bloc de données spécifié.

L'instruction BLKCLR possède 2 paramètres d'entrée et 1 paramètre de sortie. Lorsque l'instruction reçoit le flux validant, elle écrit des zéros dans l'emplacement mémoire, en commençant à la référence spécifiée par IN. Lorsque la zone à mettre à "0" pointe sur la mémoire logique (%I, %Q, %M, %G ou %T), les informations de transition associées aux références sont également mises à "0".

La sortie ok est activée chaque fois que l'instruction est validée.



Paramètres :

Paramètre	Description
validation	Le tableau est mis à "0" lorsque l'instruction est validée.
IN	IN contient le premier mot du tableau à mettre à "0".
ok	La sortie ok est activée chaque fois que l'instruction est validée.
LEN	LEN doit être compris entre 1 et 256 mots.

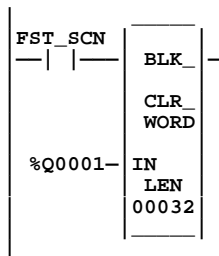
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	•†	•	•	•	•		
ok	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.
- † %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Exemple :

Dans l'exemple suivant, à la mise sous tension, 32 mots de mémoire %Q (512 points), commençant à %Q0001, sont remplis de zéros.



SHFR (BIT, WORD)

L'instruction Registre à décalage (SHFR) est utilisée pour décaler un ou plusieurs mots de données d'une adresse de référence dans un emplacement mémoire spécifié. Par exemple, un mot peut être entré par décalage dans un emplacement mémoire. Comme conséquence de ce décalage, un autre mot de données est sorti par décalage de la fin de l'emplacement mémoire.

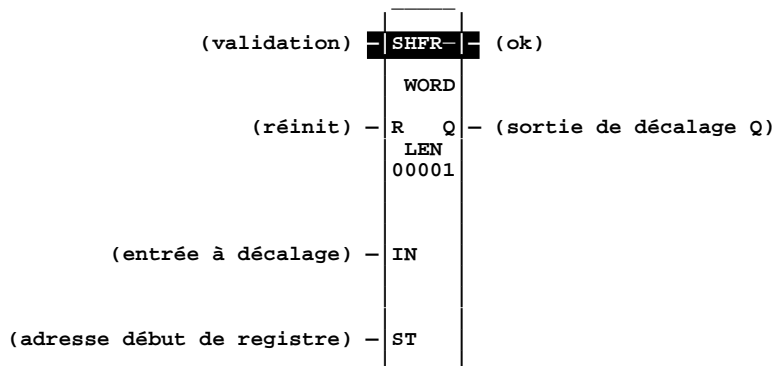
Remarque

Lorsqu'on attribue des adresses de références, le chevauchement des intervalles d'adresses de références d'entrée et de sortie dans les instructions multi-mots peut produire des résultats imprévisibles.

L'instruction SHFR possède 4 paramètres d'entrée et 2 paramètres de sortie. L'entrée de réinitialisation (R) a priorité sur l'entrée de validation de l'instruction. Lorsque la réinitialisation est active, toutes les références commençant au registre à décalage (ST) jusqu'à la longueur spécifiée (LEN) sont remplies de zéros.

Si l'instruction reçoit le flux validant et que l'initialisation n'est pas active, chaque bit ou mot du registre à décalage est transféré à la référence immédiatement supérieure. Le dernier élément dans le registre à décalage est décalé dans Q. La référence la plus élevée de l'élément IN du registre à décalage est entré par décalage dans l'élément rendu libre commençant à ST. Le contenu du registre à décalage est accessible pendant tout le programme car ce contenu est recouvert sur les positions absolues dans la mémoire logique adressable.

L'instruction dirige le flux validant vers la droite chaque fois qu'il est reçu via la logique de validation.



Paramètres :

Paramètre	Description
validation	Lorsque Validation est activé et R ne l'est pas, le décalage est exécuté.
R	Lorsque R est activé, le registre à décalage situé à ST est rempli de zéros.
IN	IN contient la valeur à décaler dans le premier bit ou mot du registre à décalage. Pour SHFR_BIT, toute référence logique peut être utilisée ; il n'est pas nécessaire d'avoir un alignement de 8 bits. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
ST	ST contient le premier bit ou mot du registre à décalage. Pour SHFR_BIT, toute référence logique peut être utilisée ; il n'est pas nécessaire d'avoir un alignement de 8 bits. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
ok	La sortie ok est activée chaque fois que l'instruction est validée et que R n'est pas validé.
Q	La sortie Q contient le bit ou mot décalé hors du registre à décalage. Pour SHFR_BIT, toute référence logique peut être utilisée ; il n'est pas nécessaire d'avoir un alignement de 8 bits. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
LEN	LEN détermine la longueur du registre à décalage. Pour SHFR_WORD, LEN doit être compris entre 1 et 256 mots. Pour SHFR_BIT, LEN doit être compris entre 1 et 256 bits.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
R	•											
IN		•	•	•	•	•	•	•	•	•	•	
ST		•	•	•	•	•†	•	•	•	•		
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

- Référence validation pour données de type BIT ou WORD, ou position où le flux validant peut traverser l'instruction.. Pour SHFR_BIT, il n'est pas nécessaire que les références logiques utilisateur %I, %Q, %M et %T aient un alignement de 8 bits..

† %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

BITSEQ (BIT)

L'instruction Séquenceur binaire (BITSEQ) exécute une séquence de décalage binaire dans un tableau de bits. L'instruction BITSEQ possède 5 paramètres d'entrée et 1 paramètre de sortie. L'exécution de l'instruction dépend de la valeur précédente du paramètre EN, comme indiqué dans le tableau suivant.

Exécution actuelle de R	Exécution précédente de EN	Exécution actuelle de EN	Exécution du séquenceur binaire
DESACT	DESACT	DESACT	Le séquenceur binaire n'est pas exécuté.
DESACT	DESACT	ACTIVE	Le séquenceur binaire est incrémenté/décémenté de 1.
DESACT	ACTIVE	DESACT	Le séquenceur binaire n'est pas exécuté.
DESACT	ACTIVE	ACTIVE	Le séquenceur binaire n'est pas exécuté.
ACTIVE	INDIFFERENT	INDIFFERENT	Le séquenceur binaire est réinitialisé.

L'entrée de réinitialisation (R) force la validation (EN) et réinitialise toujours le séquenceur. Lorsque R est activé, le numéro de la phase (STEP) courante prend la valeur transmise via le paramètre de numéro de phase. Si aucun numéro de phase n'est transmis, la phase est mise à "1". Tous les bits du séquenceur sont mis à "0", à l'exception du bit sur lequel pointe la phase courante, qui est mis à "1".

Lorsque EN est activé et R ne l'est pas, le bit sur lequel pointe le numéro de la phase courante est mis à "0". Le numéro de la phase courante est incrémenté ou décrementé, selon le paramètre de direction. Ensuite, le bit sur lequel pointe le nouveau numéro de phase est mis à "1".

- Lorsque le numéro de phase est incrémenté et qu'il sort de l'intervalle de ($1 \leq \text{numéro de phase} \leq \text{LEN}$), il est remis à "1".
- Lorsque le numéro de phase est décrementé et qu'il sort de l'intervalle de ($1 \leq \text{numéro de phase} \leq \text{LEN}$), il prend la valeur de LEN.

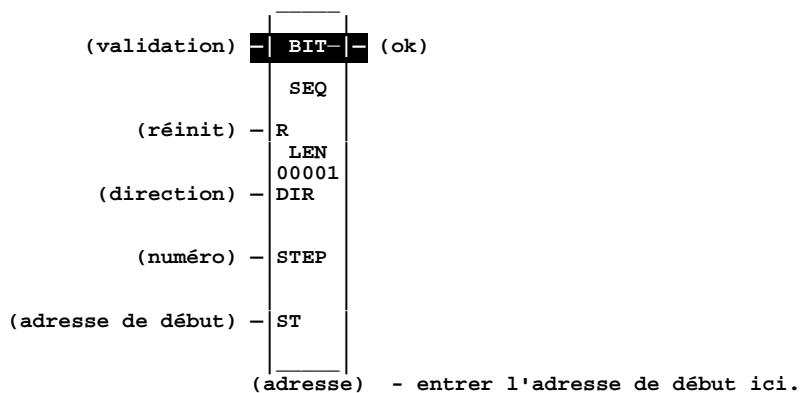
Le paramètre ST est optionnel. S'il n'est pas utilisé, l'instruction BITSEQ opère comme indiqué ci-dessus, excepté qu'aucun bit n'est mis à "1" ou à "0". Ensuite, BITSEQ itère simplement le numéro de la phase actuelle dans toute son intervalle autorisé.

Occupation mémoire d'un séquenceur binaire

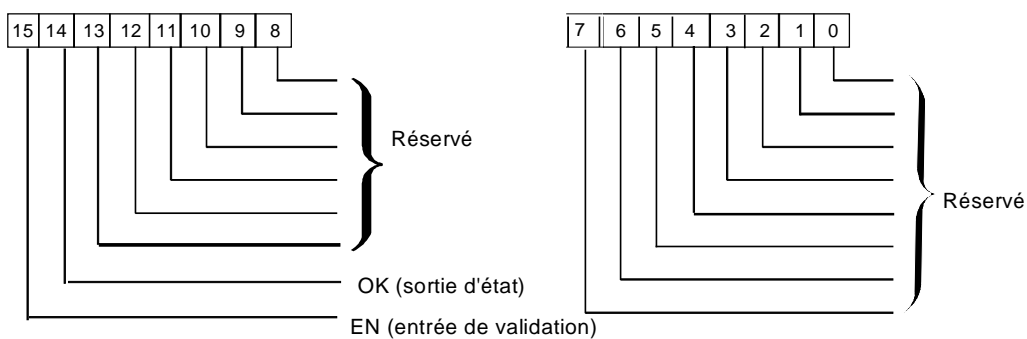
Chaque séquenceur binaire utilise 3 mots (registres) de mémoire %R pour stocker les informations suivantes :

numéro de la phase actuelle	mot 1
longueur de la séquence (en bits)	mot 2
mot de contrôle	mot 3

Lorsque vous entrez un séquenceur binaire, vous devez entrer une adresse de début pour ces trois mots (registres) immédiatement sous le graphique représentant l'instruction (voir exemple page suivante).



Le mot de contrôle stocke l'état des E/S booléennes de son bloc fonctionnel associé, comme indiqué dans le format suivant :



Remarque

Les bits 0 à 13 ne sont pas utilisés.

Paramètres :

Paramètre	Description
adresse	Adresse est la position de la phase actuelle du séquenceur binaire, sa longueur, et les derniers états ok et de validation.
validation	Lorsque l'instruction est validée, si elle n'était pas validée lors du cycle précédent et si R n'est pas activé, le décalage de séquence binaire est exécuté.
R	Lorsque R est activé, le numéro de phase du séquenceur binaire prend la valeur de STEP (par défaut = 1), et le séquenceur binaire est rempli de zéros, à l'exception du bit de numéro de phase actuelle.
DIR	Lorsque DIR est activé, le numéro de phase du séquenceur binaire est incrémenté avant le décalage. Sinon, il est décrémenté.
STEP	Lorsque R est activé, le numéro de phase prend cette valeur.
ST	ST contient le premier mot du séquenceur binaire.
ok	La sortie ok est activée chaque fois que l'instruction est validée.
LEN	LEN doit être compris entre 1 et 256 bits.

Remarque

Dans le cas de l'instruction BITSEQ, la vérification de la bobine vérifie 16 bits à partir du paramètre ST, même si la valeur de LEN est inférieure à 16.

Types de mémoires valides :

Paramètre	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
adresse								•				
validation	•											
R	•											
DIR	•											
STEP		•	•	•	•		•	•	•	•	•	•
ST		•	•	•	•	•†	•	•	•	•		•
ok	•											•

• Référence ou position validation où le flux validant peut traverser l'instruction.

† SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

COMMREQ

L'instruction Demande de communication (COMMREQ) est utilisée si le programme doit communiquer avec un coprocesseur, tel qu'un module de communication Genius ou un module coprocesseur programmable (PCM).

Remarque

Les informations indiquées dans les pages suivantes présentent le format de l'instruction COMMREQ. Des informations supplémentaires sont nécessaires pour programmer COMMREQ pour chaque type de module. Les spécifications de programmation pour chaque module utilisant l'instruction COMMREQ sont décrites dans la documentation du module.

L'instruction COMMREQ possède 3 paramètres d'entrée et un paramètre de sortie. Lorsque l'instruction COMMREQ reçoit le flux validant, un bloc de commande de données est envoyé à la TACHE de communication. Le bloc de commande commence à la référence spécifiée dans le paramètre IN. Le module destinataire est indiqué en entrant dans SYSID son bac et son numéro d'emplacement.

COMMREQ peut envoyer un message et attendre une réponse, ou envoyer un message et continuer sans attendre de réponse. Si le bloc de commande spécifie que le programme ne doit pas attendre de réponse, le contenu du bloc de commande est envoyé au module destinataire et l'exécution du programme reprend immédiatement. (La valeur du délai d'exécution est ignorée.) Ce mode est appelé mode **NOWAIT** (pas d'attente).

Si le bloc de commande spécifie que le programme doit attendre une réponse, le contenu du bloc de commande est envoyé au module destinataire et l'UC attend une réponse. Le temps d'attente maximum de l'API est spécifié dans le bloc de commande. Si le module ne répond pas dans les limites de ce délai, l'exécution du programme reprend. Ce mode s'appelle mode **WAIT** (attente).

La sortie Instruction en défaut (FT) peut être mise à "1" si :

1. l'adresse destinataire spécifiée n'est pas présente (SYSID),
2. la tâche spécifiée n'est pas valide pour ce module (TASK),
3. la longueur des données est de 0,
4. l'adresse du pointeur d'état du module (incluse dans le bloc de commande) n'existe pas. Ce peut être dû à une sélection incorrecte du type de mémoire, ou à une adresse hors limites dans ce type de mémoire.

Bloc de commande

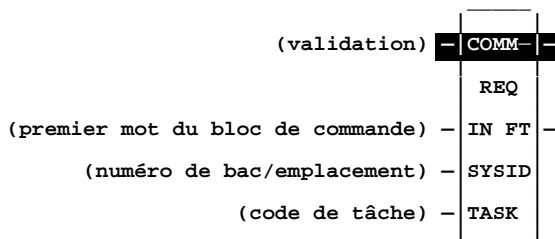
Le bloc de commande fournit au module intelligent des informations supplémentaires sur la commande à exécuter.

L'adresse du bloc de commande est spécifiée pour l'entrée IN de l'instruction COMMREQ. Cette adresse peut être n'importe quelle zone de mémoire de type mot (%R, %AI, ou %AQ). La longueur du bloc de commande dépend du volume de données transmis au module.

Le bloc de commande a la structure suivante :

Longueur (en mots)	adresse
Indicateur Attente/Pas d'attente	adresse + 1
Mémoire du pointeur d'état	adresse + 2
Décalage du pointeur d'état	adresse + 3
Valeur du délai d'attente	adresse + 4
Temps max. de communication	adresse + 5
Bloc de données	adresse + 6
	à
	adresse + 133

Les informations nécessaires au bloc de commande peuvent être placées dans la zone de mémoire désignée en utilisant une instruction de programmation appropriée.



Paramètres :

Paramètre	Description
validation	La demande de communication est exécutée lorsque l'instruction est validée.
IN	IN contient le premier mot du bloc de commande.
SYSID	SYSID contient le numéro de bac (octet de poids fort) et le numéro d'emplacement (octet de poids faible) du module destinataire.
TASK	TASK contient le code de tâche du procédé sur le module destinataire.
FT	FT est activé lorsque la demande de communication échoue.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN								•	•	•		
SYSID		•	•	•	•		•	•	•	•	•	
TASK								•	•	•	•	
FT	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, lorsque l'entrée de validation %M0020 est à "1", un bloc de commande implanté commençant à %R0016 est envoyé à la tâche de communication 1 dans le module situé dans le bac 1, emplacement 2 de l'API. Si une erreur se produit, %Q0100 est mis à "1".



Remarque

Sur les systèmes qui ne possèdent pas de bac d'extension, SYSID doit avoir la valeur 0 pour le bac principal.

Section 7 : Instructions sur tableau

Les instructions sur tableau sont utilisées pour exécuter les opérations suivantes :

Abréviation	Fonction	Description	Page
ARRAY_MOVE	Transfert de tableau	Copie un nombre spécifié d'éléments de données d'un tableau source vers un tableau de destination.	4-90
SRCH_EQ	Cherche équivalent	Cherche, dans un tableau, toutes les valeurs égales à une valeur spécifiée.	4-94
SRCH_NE	Cherche non équivalent	Cherche, dans un tableau, toutes les valeurs différentes d'une valeur spécifiée.	4-94
SRCH_GT	Cherche supérieur à	Cherche, dans un tableau, toutes les valeurs supérieures à une valeur spécifiée.	4-94
SRCH_GE	Cherche supérieur ou égal à	Cherche, dans un tableau, toutes les valeurs supérieures ou égales à une valeur spécifiée.	4-94
SRCH_LT	Cherche inférieur à	Cherche, dans un tableau, toutes les valeurs inférieures à une valeur spécifiée.	4-94
SRCH_LE	Cherche inférieur ou égal à	Cherche, dans un tableau, toutes les valeurs inférieures ou égales à une valeur spécifiée.	4-94

La longueur maximale autorisée pour ces instructions est de 32 767 octets ou mots, ou bien 262 136 bits (les bits n'étant disponibles qu'avec l'instruction ARRAY_MOVE).

Les instructions sur tableau opèrent sur les types de données suivants :

Type de données	Description
INT	Nombre entier signé.
DINT	Nombre entier signé en double précision.
BIT *	Données de type bit.
BYTE	Données de type octet.
WORD	Données de type mot.

* Disponible uniquement pour ARRAY_MOVE.

Le type de données par défaut est l'entier signé. Il est possible de choisir un autre type de données après avoir choisi l'instruction de tableau désirée. Pour comparer les données d'autres types ou de deux types différents, utilisez d'abord l'instruction de conversion appropriée (décrite dans le § 8, *Instructions de conversion*) pour choisir l'un des autres types de données, listés ci-dessus.

ARRAY_MOVE (INT, DINT, BIT, BYTE, WORD)

L'instruction Transfert de tableau (ARRAY_MOVE) est utilisée pour copier un nombre déterminé d'éléments de données d'un tableau source vers un tableau de destination.

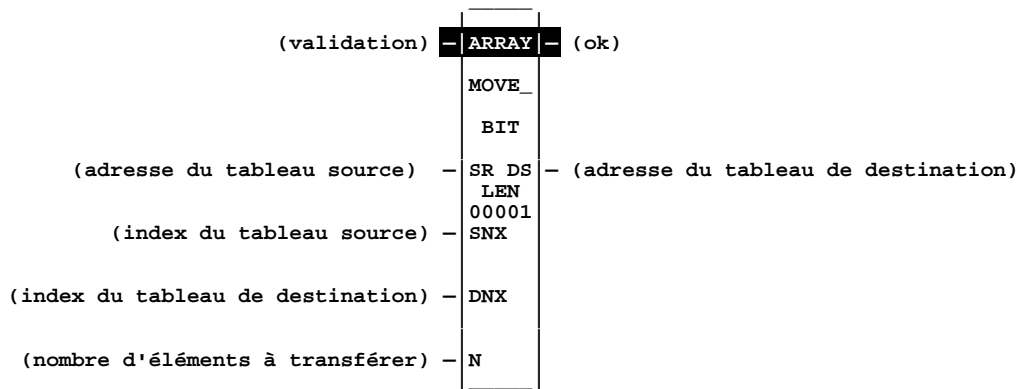
L'instruction ARRAY_MOVE possède 6 paramètres d'entrée et 2 paramètres de sortie. Lorsque l'instruction reçoit le flux validant, le nombre d'éléments (N) est extrait du tableau d'entrée en commençant par la position indexée (SR + SNX — 1). Les éléments de données sont écrits dans le tableau de sortie en commençant par la position indexée (DS + DNX — 1). L'opérande LEN spécifie le nombre d'éléments constituant chaque tableau.

Pour ARRAY_MOVE_BIT, lorsqu'une adresse "mot" est choisie pour les paramètres de l'adresse de début du tableau source et/ou de destination, le bit de poids faible du mot spécifié est le premier bit du tableau. La valeur affichée reste de 16 bits, quelle que soit la longueur du tableau.

Les index d'une instruction ARRAY_MOVE sont à base de 1. Avec ARRAY_MOVE, aucun élément hors du tableau source ou destination (déterminés par l'adresse de début et la longueur) ne peut être désigné.

La sortie ok reçoit le flux validant, sauf si l'une des conditions suivantes se produit :

- Validation est désactivé.
- (N + SNX — 1) est supérieur à LEN.
- (N + DNX — 1) est supérieur à LEN.



Paramètres

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
SR	SR contient l'adresse de début du tableau source. Pour ARRAY_MOVE_BIT, toute référence peut être utilisée ; il n'est pas nécessaire d'avoir un alignement de 8 bits. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
SNX	SNX contient l'index du tableau source.
DNX	DNX contient l'index du tableau destination.
N	N offre un indicateur de comptage.
ok	La sortie ok est activée chaque fois que Validation est activé.
DS	DS contient l'adresse de début du tableau de destination. Pour ARRAY_MOVE_BIT, toute référence peut être utilisée ; il n'est pas nécessaire d'avoir un alignement de 8 bits. Toutefois, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés alignés.
LEN	LEN spécifie le nombre d'éléments commençant à SR et DS, constituant chaque tableau.

Types de mémoires valides

Paramètre	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
SR		o	o	o	o	Δ†	o	•	•	•		
SNX		•	•	•	•		•	•	•	•	•	
DNX		•	•	•	•		•	•	•	•	•	
N		•	•	•	•		•	•	•	•	•	
ok	•											•
DS		o	o	o	o	†	o	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction. Pour ARRAY_MOVE_BIT, il n'est pas nécessaire que les références utilisateurs logiques %I, %Q, %M, et %T aient un alignement de 8 bits.
- o Référence validation pour les données de type INT, BIT, BYTE, ou WORD seulement ; n'est pas valide pour DINT.
- Δ Type de données validation pour données BIT, BYTE, ou WORD seulement ; n'est pas valide pour INT ou DINT.
- † %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Exemple 1 :

Dans cet exemple, si %R100=3, %R003 à %R007 du tableau %R001 à %R016 sont écrits dans %R0104 à %R0109 du tableau %R0100 à %R0115.

```

%I0001  |-----|
--| |---| ARRAY |-----|
      |-----| MOVE_ |-----|
      |-----| WORD |-----|
%R0001- | SG93R DS |-----| %R0100
      |-----| LEN |-----|
CONST - | 00016 |-----|
00003   | SNX |-----|
CONST - | DNX |-----|
00005   |-----|
CONST - | N |-----|
00005   |-----|
  
```

Exemple 2 :

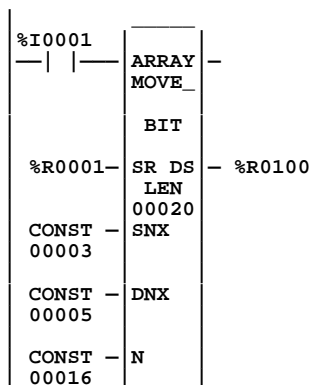
En utilisant la mémoire de bits pour SR et DS, %M0011 à %M0017 du tableau %M0009 à %M0024 sont écrits dans %Q0026 à %Q0032 du tableau %Q0022 à %Q0037.

```

%I0001  |-----|
--| |---| ARRAY |-----|
      |-----| MOVE_ |-----|
      |-----| _BIT |-----|
%M0009- | SR DS |-----| %Q0022
      |-----| LEN |-----|
CONST - | 00016 |-----|
00003   | SNX |-----|
CONST - | DNX |-----|
00005   |-----|
CONST - | N |-----|
00007   |-----|
  
```


Exemple 3 :

En utilisant la mémoire de mots, pour SR et DS, le 3e bit de poids faible de %R0001 jusqu'au 2e bit de poids faible de %R0002 du tableau contenant les 16 bits de %R0001 et 4 bits de %R0002 sont lus puis écrits dans le 5e bit de poids faible de %R0100 jusqu'au 4e bit de poids faible de %R0101 du tableau contenant les 16 bits de %R0100 et 4 bits de %R0101.



SRCH_EQ et SRCH_NE (INT, DINT, BYTE, WORD)
SRCH_GT et SRCH_LT
SRCH_GE et SRCH_LE

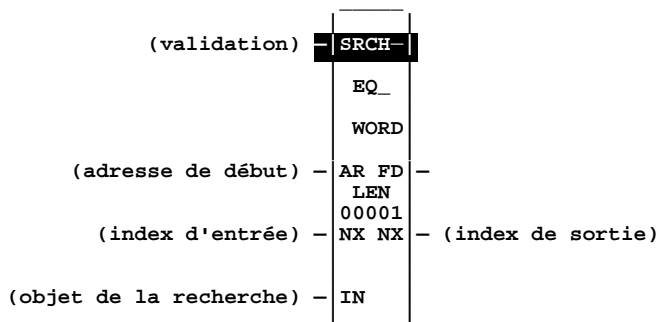
Utiliser l'instruction Search appropriée, listée ci-dessous, pour rechercher dans un tableau toutes les valeurs correspondantes à cette fonction particulière.

Abréviation	Fonction	Description
SRCH_EQ	Cherche équivalent	Cherche, dans un tableau, la première valeur égale à une valeur spécifiée.
SRCH_NE	Cherche non équivalent	Cherche, dans un tableau, la première valeur différente d'une valeur spécifiée.
SRCH_GT	Cherche supérieur à	Cherche, dans un tableau, la première valeur supérieure à une valeur spécifiée.
SRCH_GE	Cherche supérieur ou égal à	Cherche, dans un tableau, la première valeur supérieure ou égale à une valeur spécifiée.
SRCH_LT	Cherche inférieur à	Cherche, dans un tableau, la première valeur inférieure à une valeur spécifiée.
SRCH_LE	Cherche inférieur ou égal à	Cherche, dans un tableau, le première valeur inférieure ou égale à une valeur spécifiée.

Chaque instruction possède 4 paramètres d'entrée et 2 paramètres de sortie. Lorsque l'instruction reçoit le flux validant, le tableau est exploré en commençant par (AR + entrée NX). C'est l'adresse de début du tableau (AR) plus l'index dans ce tableau (entrée NX).

La recherche continue jusqu'à ce que l'élément du tableau, faisant l'objet de la recherche (IN), soit trouvé, ou jusqu'à ce que la fin du tableau soit atteinte. Si un élément de tableau est trouvé, le paramètre de sortie (FD) est mis à "1" et le paramètre de sortie (sortie NX) prend la position relative de cet élément dans le tableau. Si aucun élément de tableau n'est trouvé avant la fin du tableau, le paramètre de sortie (FD) est mis à "0" et le paramètre de sortie (sortie NX) est mis à "0".

Les valeurs valides de l'entrée NX sont comprises entre 0 et LEN — 1. NX doit être mis à zéro pour commencer la recherche au premier élément. Cette valeur s'incrémente de 1 au moment de l'exécution. Par conséquent, les valeurs de la sortie NX sont comprises entre 1 et LEN. Si la valeur de l'entrée NX est hors limites, (< 0 ou ≥ LEN), sa valeur prend celle par défaut (zéro).



Paramètres :

Paramètre	Description
validation	L'opération est exécutée lorsque l'instruction est validée.
AR	AR contient l'adresse de début du tableau sur lequel porte la recherche.
entrée NX	L'entrée NX contient l'index dans le tableau, à partir duquel la recherche doit commencer.
IN	IN contient l'objet de la recherche.
sortie NX	La sortie NX contient la position, dans le tableau, de l'objet de la recherche.
FD	FD indique qu'un élément du tableau a été trouvé et que l'instruction a été menée à bien.
LEN	LEN spécifie le nombre d'éléments commençant à AR constituant le tableau où doit porter la recherche. Ce peut être de 1 à 32 767 octets ou mots.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
AR		o	o	o	o	Δ	o	•	•	•		
NX in		•	•	•	•		•	•	•	•	•	
IN		o	o	o	o	Δ	o	•	•	•	•	
NX out		•	•	•	•		•	•	•	•		
FD	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction
- o Référence validation pour les données de type INT, BYTE, ou WORD seulement ; n'est pas valide pour DINT.
- Δ Référence validation pour les données de type BYTE ou WORD seulement ; n'est pas valide pour INT ou DINT.

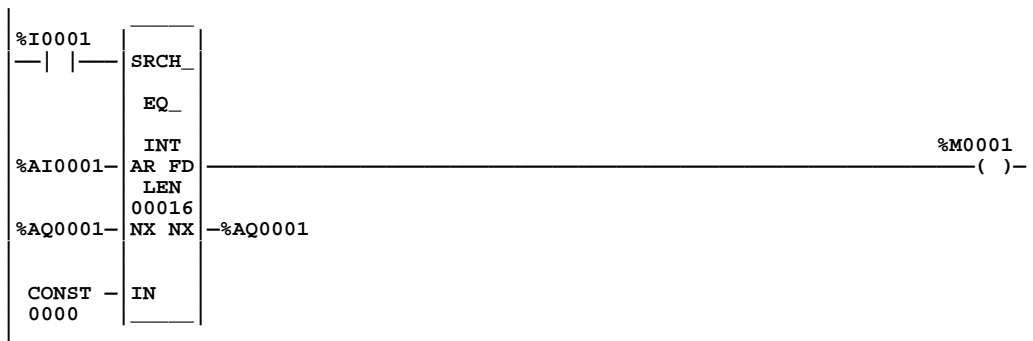
Exemple 1 :

Le tableau AR est défini comme étant les adresses de mémoire %R0001 à %R0005. Lorsque EN est à "1", la partie du tableau entre %R0004 et %R0005 est explorée pour rechercher un élément dont la valeur est égale à IN. Si %R0001 = 7, %R0002 = 9, %R0003 = 6, %R0004 = 7, %R0005 = 7, et %R0100 = 7, la recherche commence à %R0004 et se termine à %R0004 lorsque FD est mis à "1" et un 4 est écrit dans %R0101.



Exemple 2 :

Le tableau AR est défini comme étant les adresses de mémoire %AI001 à %AI016. Les valeurs des éléments du tableau sont 100, 20, 0, 5, 90, 200, 0, 79, 102, 80, 24, 34, 987, 8, 0, et 500. Au départ, %AQ001 est 5. Lorsque EN est à "1", chaque cycle explore le tableau pour rechercher une correspondance à la valeur 0 d'IN. Le premier cycle commence la recherche à %AI006 et trouve une correspondance à %AI007, si bien que FD est à "1" et %AQ001 est 7. Le deuxième cycle commence la recherche à %AI008 et trouve une correspondance à %AI015, si bien que FD reste à "1" et %AQ001 est 15. Le cycle suivant commence à %AI016. Comme la fin du tableau est atteinte sans correspondance, FD est mis à "0" et %AQ001 est mis à zéro. Le cycle suivant commence la recherche au début du tableau.



Section 8 : *Instructions de conversion*

Les instructions de conversion sont utilisées pour convertir un type de nombre en un autre. Plusieurs instructions de programmation, notamment les instructions arithmétiques, doivent utiliser des données du même type. Cette section décrit les instructions de conversion suivantes :

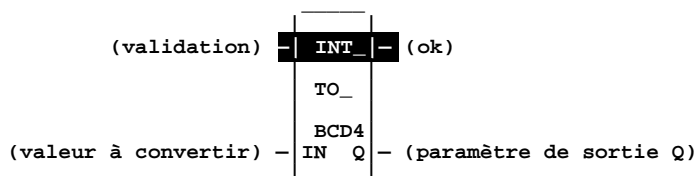
Abréviation	Fonction	Description	Page
BCD-4	Conversion en DCB à 4 chiffres	Convertit un nombre entier signé en DCB à 4 chiffres.	4-98
INT	Conversion en entier signé	Convertit un nombre DCB à 4 chiffres en nombre entier signé.	4-100
DINT	Conversion en entier signé à double précision	Convertit des données REAL en entier signé à double précision.	4-102
REAL	Conversion en REAL	Convertit INT, DINT, DCB-4 ou WORD au format REAL.	4-104
WORD	Conversion en WORD	Convertit le format REAL au format WORD.	4-106
TRUN	Tronquage	Arrondit le nombre réel vers zéro.	4-108

—>BCD-4 (INT)

L'instruction BCD-4 est utilisée pour obtenir en sortie l'équivalent DCB à 4 chiffres d'un nombre entier signé. Les données de départ ne sont pas modifiées par cette instruction. Les données de sortie peuvent être utilisées directement comme entrée pour une autre instruction de programme. Si le nombre est négatif, les sorties ok et Q sont à "0".

Les données peuvent être converties en format DCB pour piloter les voyants codés en DCB ou les valeurs présélectionnées de modules externes tels que les compteurs rapides (HSC).

Lorsque l'instruction est validée, elle exécute la conversion, mettant le résultat à disposition à la sortie Q. La sortie ok est validée, sauf si la conversion spécifiée donne une valeur en dehors de l'intervalle 0 à 9999.



Paramètres :

Paramètre	Description
validation	La conversion est exécutée lorsque l'instruction est validée.
IN	IN contient une référence pour le nombre entier à convertir en DCB à 4 chiffres.
ok	La sortie ok est activée lorsque l'instruction est exécutée sans erreur.
Q	La sortie Q contient le format DCB à 4 chiffres de la valeur de départ dans IN.

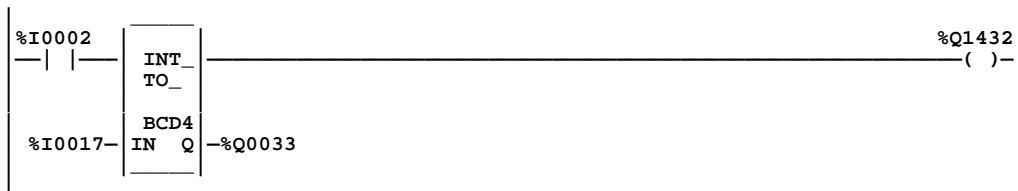
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, chaque fois que l'entrée %I0002 est mise à "1" et qu'aucune erreur n'existe, le nombre entier à la position d'entrée %I0017 à %I0032 est converti en DCB à 4 chiffres, et le résultat est stocké dans les emplacements mémoire %Q0033 à %Q0048. La bobine %Q1432 est utilisée pour contrôler si la conversion a réussi.



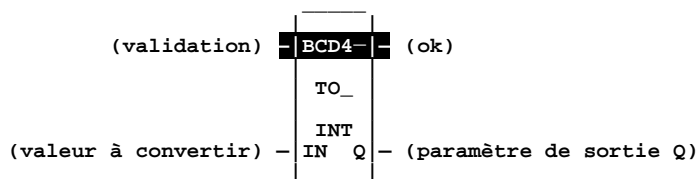
—>INT (BCD-4, REAL)

L'instruction Convertir en entier signé est utilisée pour obtenir en sortie l'équivalent d'un nombre entier DCB à 4 chiffres ou de données REAL. Les données de départ ne sont pas modifiées par cette instruction. Les données de sortie peuvent être utilisées directement comme entrée pour une autre instruction de programme.

Remarque

Le type de données REAL est disponible uniquement sur les UC 352.

Lorsque l'instruction est validée, elle exécute la conversion, mettant le résultat à disposition à la sortie Q. L'instruction transmet la validation, sauf si les données sont hors limites.



Paramètres :

Paramètre	Description
validation	La conversion est exécutée lorsque l'instruction est validée.
IN	IN contient une référence pour la valeurDCB-4, REAL ou Constante à convertir en entier.
ok	La sortie ok est activée lorsque la validation est activée, sauf si les données ne sont pas comprises dans la plage ou si elles correspondent à NaN (Not a Number).
Q	La sortie Q contient le format d'entier de la valeur d'origine de IN.

Types de mémoires valides :

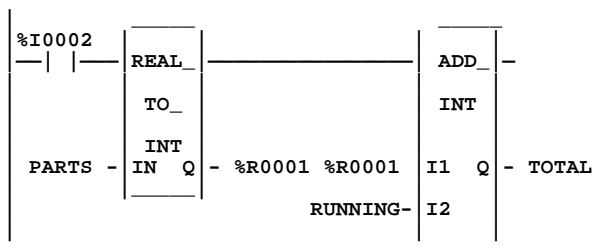
Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

Remarque : Pour les données REAL, les seuls types valides sont %R, %AI et %AQ.

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

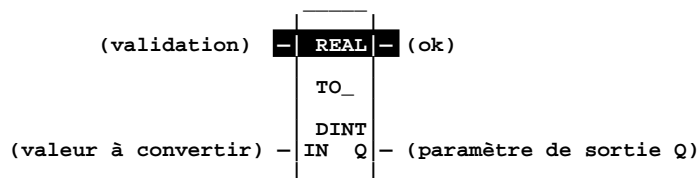
Dans l'exemple suivant, chaque fois que l'entrée %I0002 est mise à "1", la valeur DCB à 4 chiffres dans PARTS est convertie en nombre entier signé et transmise à l'instruction ADD, où elle est ajoutée au nombre entier signé représenté par la référence RUNNING. Le total est envoyé en sortie par l'instruction ADD à la référence TOTAL.



—>DINT (REAL)

L'instruction Convertir en entier signé à double précision permet de produire en sortie l'équivalent en données Real d'un entier signé à double précision. Les données de départ ne sont pas modifiées par cette instruction. Les données de sortie peuvent être utilisées directement comme entrée pour une autre instruction de programme.

Lorsque l'instruction reçoit le flux validant, elle exécute la conversion et met le résultat à disposition à la sortie Q. L'instruction passe toujours le flux validant lorsqu'elle le reçoit, sauf si la valeur réelle n'est pas comprise dans la plage.



Paramètres :

Paramètre	Description
validation	La conversion est exécutée quand l'instruction est validée.
IN	IN contient la référence de la valeur à convertir en entier à double précision.
ok	La sortie ok est activée lorsque la validation est activée, sauf si la valeur réelle n'est pas comprise dans la plage.
Q	Q contient l'entier signé à double précision équivalent à la valeur de départ dans IN.

Remarque

Dans la mesure où le format REAL possède 24 bits significatifs, la conversion du format REAL au format DINT peut entraîner une perte de précision.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		o	o	o	o		o	•	•	•	•	
ok	•											•
Q								•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple ci-dessous, chaque fois que l'entrée %I0002 est mise à "1", le nombre entier à la position d'entrée %I0017 est converti en entier signé à double précision, et le résultat est stocké dans l'emplacement mémoire %R0001. La sortie %Q1001 est mise à "1" chaque fois que l'instruction est exécutée correctement.



—>REAL (INT, DINT, BCD-4, WORD)

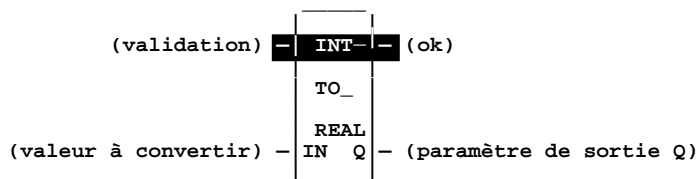
L'instruction Convertir au format REAL permet de produire en sortie la valeur réelle des données de l'entrée. Les données de départ ne sont pas modifiées par cette instruction. Les données de sortie peuvent être utilisées directement comme entrée pour une autre instruction de programme.

Lorsque l'instruction reçoit le flux validant, elle exécute la conversion et met le résultat à disposition à la sortie Q. L'instruction passe toujours le flux validant lorsqu'elle le reçoit, sauf si le résultat de la conversion n'est pas compris dans la plage.

Dans la mesure où le nombre de bits significatifs est réduit à 24, la conversion de DINT au format REAL peut entraîner une perte de précision.

Remarque

Cette fonction est disponible uniquement sur les UC 352.



Paramètres :

Paramètre	Description
validation	La conversion est exécutée quand l'instruction est validée.
IN	IN contient la référence de la valeur d'entier à convertir au format REAL.
ok	La sortie ok est activée quand l'instruction est exécutée sans erreur.
Q	Q contient le format REAL de la valeur de départ dans IN.

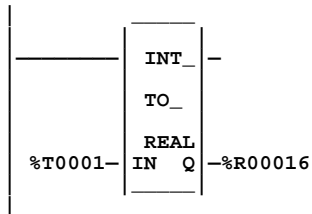
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		o	o	o	o		o	•	•	•	•	
ok	•											•
Q								•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.
- o Non valide pour DINT_TO_REAL.

Exemple :

Dans l'exemple ci-dessous, la valeur d'entier de l'entrée IN est 678. La valeur de résultat qui est stockée dans l'emplacement mémoire %T0016 est 678,000.



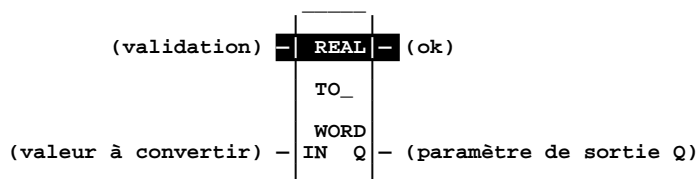
—>WORD (REAL)

L'instruction Convertir au format WORD permet de produire en sortie l'équivalent WORD des données réelles. Les données de départ ne sont pas modifiées par cette instruction.

Remarque

Cette fonction est disponible uniquement sur les UC 352.

Lorsque l'instruction reçoit le flux validant, elle exécute la conversion et met le résultat à disposition à la sortie Q. L'instruction passe toujours le flux validant lorsqu'elle le reçoit, sauf si le résultat de la conversion produit une valeur qui n'est pas comprise dans la plage 0 à FFFFh.



Paramètres :

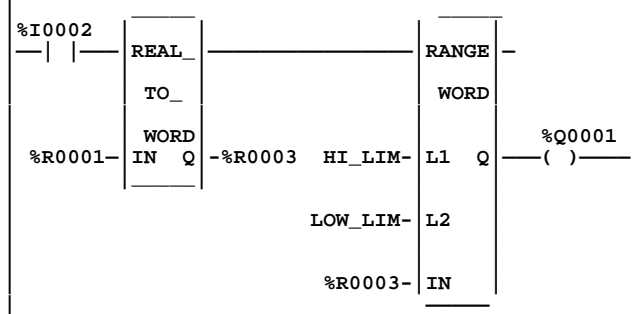
Paramètre	Description
validation	La conversion est exécutée quand l'instruction est validée.
IN	IN contient la référence de la valeur à convertir au format WORD.
ok	La sortie ok est activée quand l'instruction est exécutée sans erreur.
Q	Q contient l'entier non signé de la valeur de départ dans IN.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN								•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :



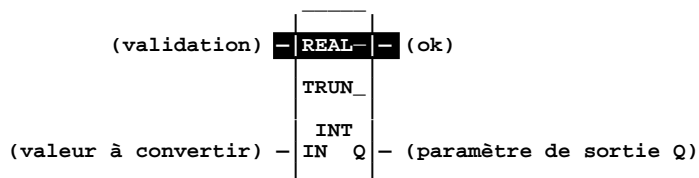
TRUN (INT, DINT)

L'instruction Tronquer permet d'arrondir le nombre réel vers zéro. Les données de départ ne sont pas modifiées par cette instruction. Les données de sortie peuvent être utilisées directement comme entrée pour une autre instruction de programme.

Remarque

Dans la mesure où l'UC 352 est la seule UC Série 90-30 à virgule flottante, l'instruction TRUN ne s'applique pas aux autres UC 90-30.

Lorsque l'instruction reçoit le flux validant, elle exécute la conversion et met le résultat à disposition à la sortie Q. L'instruction passe toujours le flux validant lorsqu'elle le reçoit, sauf si le résultat de la conversion produit une valeur qui n'est pas comprise dans la plage ou si IN correspond à NaN (Not a Number).



Paramètres :

Paramètre	Description
validation	La conversion est exécutée quand l'instruction est validée.
IN	IN contient la référence de la valeur réelle à tronquer.
ok	La sortie ok est activée quand l'instruction est exécutée sans erreur, sauf si la valeur n'est pas comprise dans la plage ou si IN correspond à NaN.
Q	Q contient la valeur INT ou DINT tronquée de la valeur de départ dans IN.

Remarque

Dans la mesure où le format REAL possède 24 bits significatifs, la conversion de REAL au format DINT peut entraîner une perte de précision.

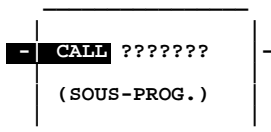
Section 9 : Instructions de commande

Cette section décrit les instructions de commande pouvant être utilisées pour limiter l'exécution du programme et modifier la façon dont l'UC exécute le programme d'application. (voir le chapitre 2, § 1, "Séquences de cycle d'API", pour plus d'informations sur le cycle d'API.

Fonction	Description	Page
CALL	Commande à l'exécution du programme d'aller à un bloc de sous-programme spécifié.	4-111
DOIO	Rafraîchissement immédiat d'un champ d'entrée ou de sortie pendant un cycle. Les rafraîchissements partiels des modules d'E/S ne sont pas exécutés. Optionnellement, une copie des E/S scrutées peut être placée dans la mémoire interne plutôt que dans les mémoires des E/S.	4-112
END	Offre une fin temporaire de la logique. Le programme est exécuté depuis le premier segment jusqu'au dernier ou jusqu'à l'instruction END, si elle est présente. Cette instruction est utile pour la mise au point, mais n'est pas autorisée dans la programmation SFC (voir la remarque à la page 4-118).	4-118
MCR et MCRN	Programme un relais de contrôle maître. Une instruction MCR force l'exécution de toutes les lignes entre MCR et son instruction ENDMCR (fin de MCR) consécutive, sans flux validant. Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction MCR : une forme non imbriquée (MCR) et une forme imbriquée (MCRN).	4-119
ENDMCR et ENDMCRN	Indique que la logique suivante doit être exécutée avec le flux validant normal. Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction ENDMCR : une forme non imbriquée (ENDMCR) et une forme imbriquée (ENDMCRN).	4-122
JUMP et JUMPN	Commande à l'exécution du programme de sauter à une position déterminée (indiquée par une ETIQUETTE (LABEL)). Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction JUMP : une forme non imbriquée (JUMP) et une forme imbriquée (JUMPN).	4-123
LABEL et LABELN	Spécifie la position destinataire d'une instruction JUMP. Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction LABEL : une forme non imbriquée (LABEL) et une forme imbriquée (LABELN).	4-125
COMMENT	Place un commentaire dans le programme. Une fois l'instruction programmée, le type peut être tapé en effectuant un zoom" dans l'instruction.	4-126
SVCREQ	Demande l'un des services spéciaux d'API suivants : <ul style="list-style-type: none"> • Modifier/Lire l'état de la tâche et le nombre de mots que doit vérifier le total de contrôle. • Modifier/Lire l'horloge interne. • Mettre l'API hors fonction. • Vider les Tables des défauts. • Lire la dernière entrée de Table des défauts enregistrée. • Lire le compteur de temps de fonctionnement. • Lire l'état de forçage des E/S. • Lire Checksum maître • Interroger les E/S • Lire le compteur de mise hors tension 	4-127
PID	Offre deux algorithmes de commande en boucle fermée PID (proportionnelle, intégrale et dérivée) : <ul style="list-style-type: none"> • Algorithme PID standard ISA à termes dépendants (PIDISA). • Algorithme à termes indépendants (PIDIND). 	4-146

CALL

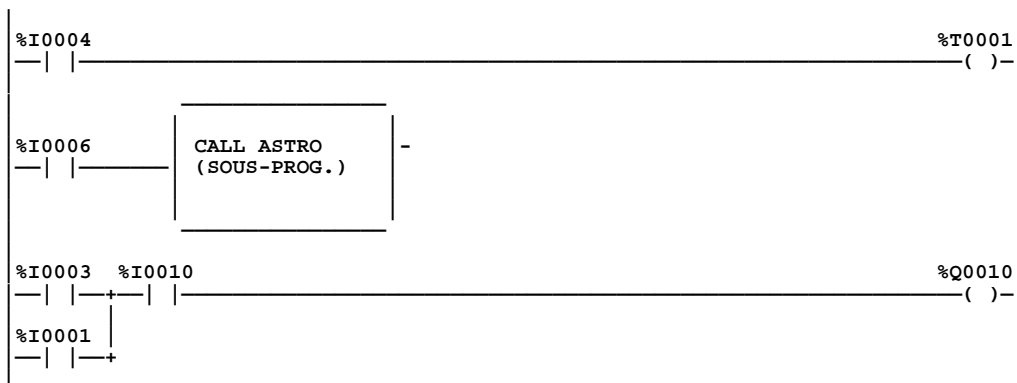
L'instruction CALL est utilisée pour commander à l'exécution du programme d'aller à un bloc de sous-programme spécifié.



Lorsque l'instruction CALL est validée, la scrutation va immédiatement au bloc de sous-programme désigné, qui est ensuite exécuté. Une fois ce bloc exécuté, le contrôle retourne au point de la logique qui suit immédiatement l'instruction CALL.

Exemple :

L'exemple dans l'écran suivant présente l'instruction CALL telle qu'elle apparaît dans le bloc appelant. Pour effectuer un zoom dans le sous-programme, placer le curseur dans l'instruction et appuyer sur **F10**.



Remarque

Comme les UC Micro ne supportent pas les sous-programmes, l'instruction CALL ne peut pas être utilisée avec une UC Micro.

DOIO

L'instruction DO I/O (DOIO) est utilisée pour rafraîchir les entrées ou les sorties pendant une scrutation au cours de l'exécution du programme. DOIO peut être également utilisée pour rafraîchir les E/S choisies durant le programme en plus de la scrutation normale des E/S.

Si des références d'entrée sont spécifiées, l'instruction va chercher les valeurs les plus récentes de ces entrées pour utilisation par la logique du programme. Si des références de sortie sont spécifiées, DO I/O rafraîchit les sorties d'après les valeurs les plus récentes, stockées dans la mémoire des E/S. Les E/S sont desservies par incréments de modules d'E/S entiers ; si nécessaire, l'API ajuste les références pendant l'exécution de l'instruction.

L'instruction DOIO possède 4 paramètres d'entrée et 1 paramètre de sortie. Lorsque l'instruction est validée et que des références d'entrée sont spécifiées, les bits d'entrée depuis la référence de début (ST) jusqu'au END sont scrutés. Si une référence est spécifiée pour ALT, une copie des nouvelles valeurs d'entrée est stockée en mémoire, commençant à cette référence, et les bits d'entrée réels ne sont pas rafraîchis. ALT doit être de même taille que le type de référence scruté. Si une référence logique est utilisée pour ST et END, ALT doit être également logique. Si aucune référence n'est spécifiée pour ALT, les bits réels sont rafraîchis.

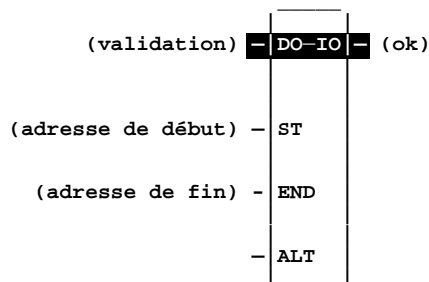
Lorsque l'instruction DOIO est validée et que des références de sortie sont spécifiées, les bits de sortie depuis la référence de début (ST) jusqu'au END sont écrits dans les modules de sortie. Si les sorties doivent être écrites dans les modules de sortie depuis la mémoire interne, autre que %Q ou %AQ, la référence de début peut être spécifiée pour ALT. L'intervalle des sorties écrites dans les modules de sortie est spécifié par la référence de début (ST) et la référence de fin (END).

L'exécution de l'instruction continue jusqu'à ce que toutes les entrées dans le champ choisi aient été lues, ou jusqu'à ce que toutes les sorties aient été écrites dans les modules d'E/S. Ensuite, l'exécution du programme retourne à l'instruction qui suit DO I/O.

Si la plage de références comprend un module en option (HSC, APM, etc.), la totalité des données d'entrée (%I et %AI) ou de sortie (%Q et %AQ) de ce module est scrutée. Le paramètre ALT est ignoré pendant le scrutage des modules en option. En outre, la plage de références ne peut pas comprendre de module GCM amélioré.

La validation est transmise vers la droite chaque fois que le flux validant est reçu, sauf si :

- certaines références du type spécifié ne font pas partie de l'intervalle choisi.
- l'UC n'est pas en mesure de gérer correctement la liste temporaire des E/S, créée par l'instruction.
- l'intervalle spécifié inclut les modules d'E/S associés à un défaut de module d'E/S perdu".



Paramètres :

Paramètre	Description
validation	Lorsque l'instruction est validée, une scrutation limitée des entrées ou des sorties est exécutée.
ST	ST est l'adresse de début de l'ensemble de points d'entrée ou de sortie ou de mots à desservir.
END	END est l'adresse de fin ou l'ensemble de points d'entrée ou de sortie ou de mots à desservir.
ALT	Pour la scrutation des entrées, ALT spécifie l'adresse où stocker les valeurs des mots/points d'entrée scrutés. Pour la scrutation des sorties, ALT spécifie l'adresse où obtenir les valeurs des mots/bits de sortie, à envoyer aux modules d'E/S. Pour les UC modèles 331 et supérieurs, le paramètre ALT peut influencer la vitesse d'exécution du bloc fonctionnel DOIO (voir la remarque ci-dessous ainsi que la section relative à l'instruction DO I/O améliorée pour les UC modèles 331 et supérieurs, à la page 4-112).
ok	La sortie ok est activée lorsque la scrutation des entrées ou des sorties est exécutée normalement.

Remarque

Sur les UC modèles 331 et supérieurs, le paramètre ALT du bloc fonctionnel DOIO peut être utilisé pour entrer l'emplacement d'un module unique dans le bac principal. Dans ce cas, le bloc fonctionnel DOIO s'exécute en 80 microsecondes en lieu en place des 236 microsecondes requises pour l'exécuter sans le paramètre ALT. Aucune vérification d'erreur n'est effectuée pour empêcher le chevauchement des adresses de référence ou les incohérences entre les types de module.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
ST		•	•						•	•		
END		•	•						•	•		
ALT		•	•	•	•		•	•	•	•		•
ok	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple d'entrée 1 :

Dans l'exemple suivant, lorsque l'entrée de validation %I0001 est à "1", les références %I0001 à %I0064 sont scrutées, et %Q0001 est mise à "1". Une copie des entrées scrutées est placée dans la mémoire interne, entre la référence %M0001 et %M0064. Les points d'entrée réels ne sont pas rafraîchis. Cette forme d'instruction peut être utilisée pour comparer les valeurs actuelles des points d'entrée aux valeurs des points d'entrée au début de la scutation.



Exemple d'entrée 2 :

Dans l'exemple suivant, lorsque l'entrée de validation %I0001 est à "1", les références %I0001 à %I0064 sont scrutées, et %Q0001 est mise à "1". Les entrées scrutées sont placées dans la mémoire d'état des entrées, entre les références %I0001 et %I0064. Cette forme de l'instruction permet de scruter les points d'entrée une ou plusieurs fois au cours de la partie "exécution de programme" du cycle d'UC.



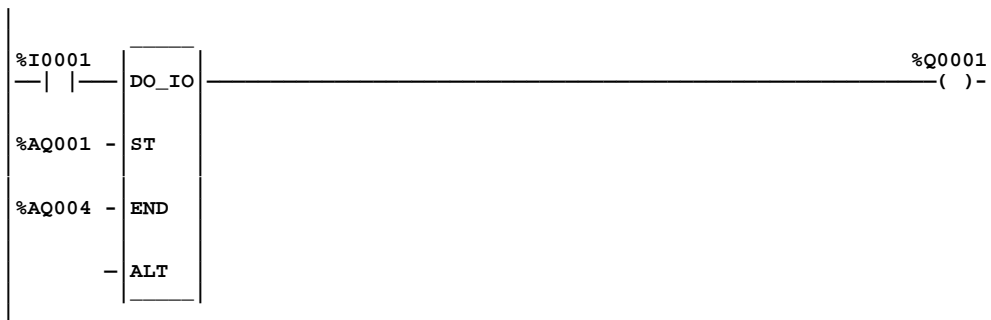
Exemple de sortie 1 :

Dans l'exemple suivant, lorsque l'entrée de validation %I0001 est à "1", les valeurs aux références %R0001 à %R0004 sont écrites dans les voies des sorties analogiques %AQ001 à %AQ004, et %Q0001 est mise à "1". Les valeurs de %AQ001 à %AQ004 ne sont pas écrites dans les modules de sorties analogiques.



Exemple de sortie 2 :

Dans l'exemple suivant, lorsque l'entrée de validation %I0001 est à "1", les valeurs dans %AQ001 à %AQ004 sont écrites dans les voies des modules de sorties analogiques %AQ001 à %AQ004, et %Q0001 est mise à "1".



Instruction DO I/O évoluée pour les UC modèles 331 et 341

Attention

Si l'instruction DO I/O évoluée est utilisée dans un programme, celui-ci ne doit pas être chargé par une version du logiciel Logimaster 90-30/20 antérieure à 4.01.

Une version évoluée de l'instruction DO I/O (DOIO) est disponible pour la version 4.20, ou une version ultérieure des UC modèles 331 et supérieurs. Cette version évoluée de l'instruction DOIO ne peut être utilisée que sur un module d'entrées à 8, 16 ou 32 points, ou sur un module de sorties analogiques.

Le paramètre ALT identifie l'emplacement, dans le bac principal, où se trouve le module. Par exemple, une valeur constante de 2 dans ce paramètre indique à l'UC qu'elle doit exécuter la version évoluée du bloc fonctionnel DOIO pour le module dans l'emplacement 2.

Remarque

Le seul contrôle effectué par le bloc fonctionnel DOIO est celui de l'état du module se trouvant dans l'emplacement spécifié afin de vérifier s'il est correct..

L'instruction DOIO évoluée s'applique uniquement aux modules situés dans le bac principal. C'est pourquoi le paramètre ALT doit être compris entre 2 et 5 pour un bac à 5 emplacements, ou entre 2 et 10 pour un bac à 10 emplacements.

Les références de début et de fin doivent être %I ou %Q. Ces références spécifient la première et la dernière référence pour lesquelles le module est configuré. Par exemple, si un module d'entrée à 16 points est configuré de %I0001 à %I0016 dans l'emplacement 10 d'un bac principal à 10 emplacements, le paramètre ST doit être %I0001, le paramètre END doit être %I0016, et le paramètre ALT doit être 10, comme indiqué ci-dessous :



Le tableau suivant compare les temps d'exécution d'un bloc fonctionnel DOIO normal pour un module d'E/S logiques à 8, 16, ou 32 points à ceux d'un bloc fonctionnel DOIO évolué.

Module	Temps d'exécution DOIO normal	Temps d'exécution DOIO évolué
Module d'entrée logique à 8 points	224 microsecondes	67 microsecondes
Module de sortie logique à 8 points	208 microsecondes	48 microsecondes
Module d'entrée logique à 16 points	224 microsecondes	68 microsecondes
Module de sortie logique à 16 points	211 microsecondes	47 microsecondes
Module d'entrée logique à 32 points	247 microsecondes	91 microsecondes
Module de sortie logique à 32 points	226 microsecondes	50 microsecondes

END

L'instruction END permet de mettre fin temporairement à la logique. Le programme est exécuté depuis la première ligne jusqu'à la dernière ou jusqu'à l'instruction END, si elle est présente.

L'instruction END termine de manière inconditionnelle l'exécution du programme. Il ne peut rien y avoir après END dans une ligne. Aucune logique n'est exécutée au-delà de END, et le contrôle est retourné au début du programme pour le cycle suivant.

L'instruction END est utile à la mise au point car elle évite l'exécution de toute logique venant après.

Le logiciel de programmation Logicmaster offre un marqueur [END OF PROGRAM LOGIC] indiquant la fin du programme exécuté. Ce marqueur est utilisé si aucune instruction END n'est programmée dans la logique.

- [END]

Exemple :

Dans l'exemple suivant, une instruction END est programmée pour terminer le cycle actuel.

```
STOP  
-[ END ]
```

Remarque

Le placement d'une instruction END dans la logique SFC ou la logique appelée par SFC produit un défaut "Exécution de l'instruction END à partir d'une action SFC" sur les UC version 7 ou ultérieures. (Cette instruction ne fonctionnait pas bien non plus sur les UC antérieures à la version 7, mais ne produisait cependant pas de défaut.) Pour plus d'informations sur ce défaut, voir le paragraphe "Incohérence dans la configuration système" du chapitre 3, Section 2.

MCR

Toutes les lignes entre un relais de contrôle maître (MCR) et sa Fin de relais de contrôle maître (ENDMCR) correspondante sont exécutées sans flux validant vers les bobines. La condition de validation de l'instruction MCR est appliquée en NAND (NON ET) à l'ensemble des lignes de programme comprises entre MCR et ENDMCR. Une instruction ENDMCR associée à la MCR est utilisée pour reprendre normalement l'exécution du programme. A la différence de l'instruction de Saut JUMP, les MCR ne peuvent avoir lieu que vers l'avant. Dans le programme, l'instruction ENDMCR doit apparaître après son instruction MCR correspondante.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction MCR : une forme non imbriquée et une forme imbriquée. La forme non imbriquée est appelée MCR.

Remarque

Les UC modèles 351 et supérieurs ne possèdent pas la forme non imbriquée MCR. N'utilisez donc que la forme imbriquée (MCRN) avec les UC 351 et 352.

Il ne peut y avoir qu'une seule instruction MCR pour chaque instruction ENDMCR. L'intervalle des MCR et ENDMCR non imbriquées ne peut pas chevaucher l'intervalle d'une autre paire MCR/ENDMCR ou d'une paire d'instructions JUMP/LABEL. Les MCR non imbriquées ne peuvent pas entrer dans l'intervalle d'une autre paire MCR/ENDMCR ou d'une paire JUMP/LABEL. De plus, une paire JUMP/LABEL ou une paire MCR/ENDMCR ne peut pas entrer dans l'intervalle d'une paire MCR/ENDMCR.

Remarque

L'instruction MCR non imbriquée est la seule instruction de relais de commande maître qui peut être utilisée sur une UC Série 90-30 Version 1. La fonction MCR imbriquée doit être utilisée avec toutes les nouvelles applications.

La forme imbriquée d'une instruction MCR est appelée MCRN ; elle doit être utilisée dans toutes les nouvelles applications. Une instruction MCRN peut être imbriquée avec d'autres instructions MCRN, à la condition d'être imbriquées correctement. L'instruction MCRN et son instruction ENDMCRN correspondante doivent être incluses entièrement dans une autre paire MCRN/ENDMCRN.

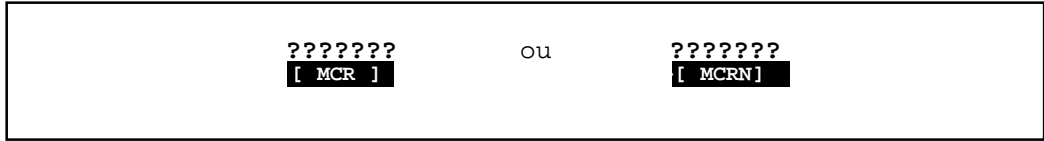
Une instruction MCRN peut être placée n'importe où dans un programme, à la condition d'être imbriquée correctement par rapport aux autres MCRN, et de ne pas intervenir dans l'intervalle d'une instruction MCR non imbriquée ou JUMP non imbriquée.

Remarque

N'utilisez qu'une seule instruction MCRN pour chaque ENDMCRN sur les UC 351.

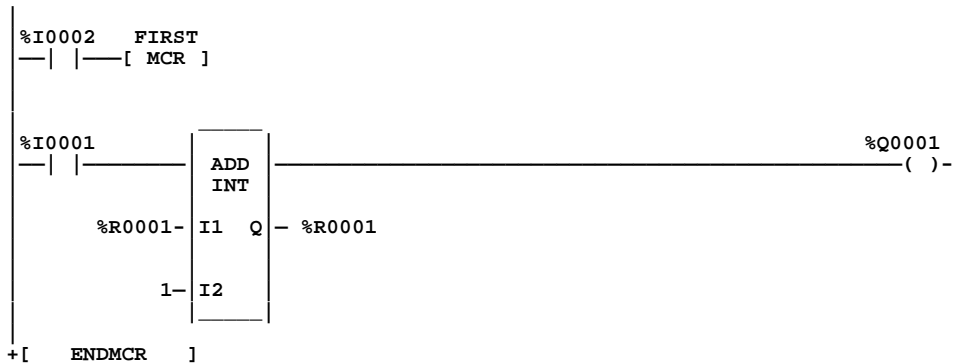
Il peut y avoir plusieurs instructions MCRN correspondant à une même ENDMCRN (sauf pour les UC 351 et 352, comme décrit plus haut). Cela est comparable à l'instruction JUMP imbriquée, où vous pouvez avoir plusieurs sauts (JUMP) vers la même étiquette (LABEL). Pour plus d'informations sur les différences entre les instructions JUMP et MCR, voir la section "Différences entre les instructions MCR et JUMP" à la page 4- 120.

Les deux formes de l'instruction MCR ont les mêmes paramètres. Elles ont toutes deux une entrée booléenne de validation (EN) et également un nom indiquant qu'il s'agit d'une MCR. Ce nom se retrouve dans une instruction ENDMCR ; La MCR, comme la MCRN, n'a pas de sortie ; il ne peut rien y avoir dans un segment après une MCR.

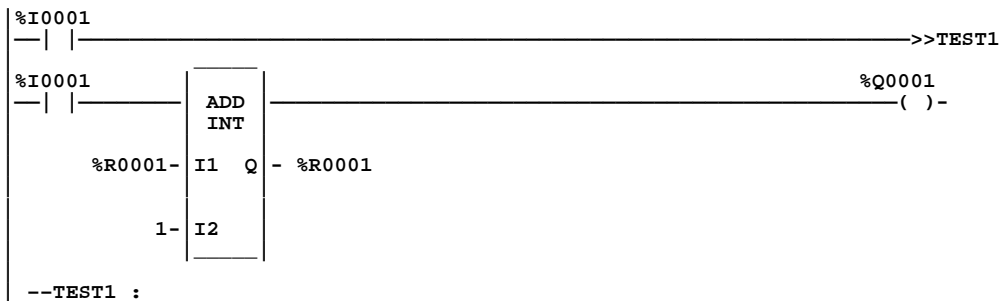


Différences entre les instructions MCR et JUMP

Dans le cas d'une instruction MCR, les blocs fonctionnels compris dans la portée de MCR sont exécutés *sans flux validant* et les bobines *sont désactivées*. Dans l'exemple ci-dessous, quand %I0002 est à "1", MCR est activé. Quand MCR est activé, même si %I0001 est à "1", le bloc fonctionnel ADD est exécuté *sans* flux validant (c'est-à-dire qu'il n'ajoute pas 1 à %R0001) et %Q0001 est à "0".

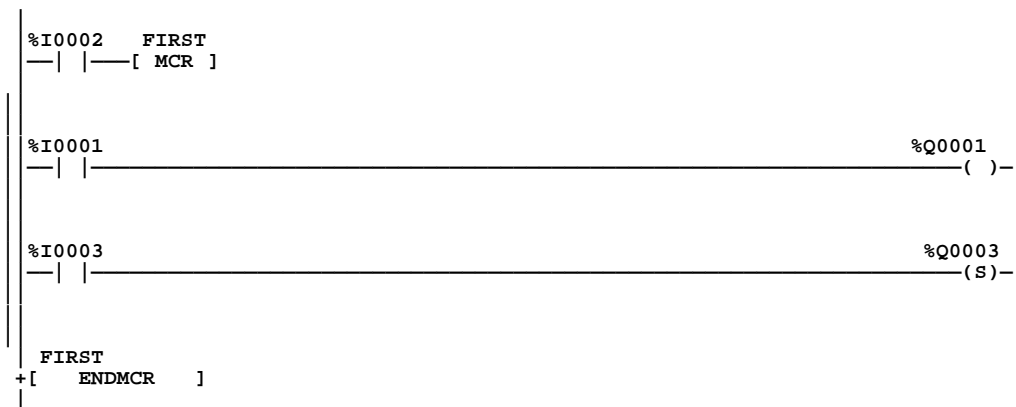


Dans le cas d'une instruction JUMP, tous les blocs fonctionnels compris entre JUMP et LABEL *ne sont pas* exécutés et les bobines *ne sont pas affectées*. Dans l'exemple ci-dessous, quand %I0002 est à "1", JUMP est pris. Comme la logique comprise entre JUMP et LABEL est sautée, %Q0001 n'est pas affecté (c'est-à-dire qu'il demeure à "0" s'il était à "0" ou à "1" s'il était à "1").



Exemple :

Dans l'exemple ci-dessous, chaque fois que %I0002 autorise le flux validant dans l'instruction MCR, l'exécution du programme se poursuit sans passer le flux validant aux bobines, jusqu'à ce que ENDMCR associé soit atteint. Si %I0001 et %I0003 sont à "1", %Q0001 est mis à "0" et %Q0003 demeure à "1".

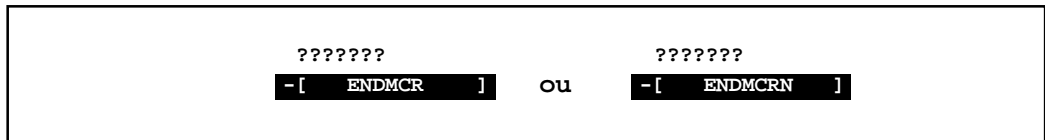


ENDMCR

L'instruction Fin de relais de contrôle maître (ENDMCR) est utilisée pour reprendre l'exécution normale du programme après une instruction MCR. Lorsque la MCR associée à la ENDMCR est activée, la ENDMCR force l'exécution du programme à reprendre avec le flux validant normal. Lorsque la MCR associée à la ENDMCR n'est pas activée, la ENDMCR est sans effet.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction ENDMCR : une forme non imbriquée et une forme imbriquée. La forme non imbriquée (ENDMCR) doit être utilisée avec l'instruction MCR non imbriquée (MCR). La forme imbriquée (ENDMCRN) doit être utilisée avec l'instruction MCR imbriquée (MCRN).

La validation de l'instruction doit être fournie par la barre d'alimentation dépendante d'une instruction MCR. L'exécution n'est pas conditionnelle. L'instruction ENDMCR a également un nom, qui identifie la ENDMCR et l'associe à la/aux MCR correspondante(s). L'instruction ENDMCR n'a pas de sorties ; il ne peut rien y avoir avant ou après une ENDMCR dans un segment.



Exemple :

Dans les exemples suivants, une instruction ENDMCR est programmée pour terminer l'intervalle MCR "clear."

Exemple de ENDMCR non imbriquée :

```

CLEAR
- [ ENDMCR ]

```

Exemple de ENDMCR imbriquée :

```

CLEAR
- [ ENDMCRN ]

```

JUMP

L'instruction JUMP est utilisée pour sauter une partie de la logique de programme. L'exécution du programme continue à l'étiquette (LABEL) spécifiée. Lorsque JUMP est activée, toutes les bobines dans son intervalle de validité sont laissées à leur état précédent. Cela inclut les bobines associées aux temporisateurs, compteurs, verrous, et relais.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction JUMP : une forme non imbriquée et une forme imbriquée. La forme non imbriquée a la forme _____
>>LABEL01, où LABEL01 est le nom de l'instruction LABEL non imbriquée.

Pour les JUMP non imbriqués, il ne peut y avoir qu'une seule instruction JUMP pour chaque instruction LABEL. Le saut (JUMP) peut être effectué vers l'avant ou vers l'arrière.

L'intervalle des JUMP et LABEL non imbriqués ne peut pas chevaucher l'intervalle d'une autre paire d'instructions JUMP/LABEL ou MCR/ENDMCR. Les JUMP non imbriqués et leurs LABEL correspondants ne peuvent pas être dans l'intervalle de validité d'une autre paire JUMP/LABEL ou MCR/ENDMCR. De plus, une paire MCR/ENDMCR ou une autre paire JUMP/LABEL ne peut pas être dans l'intervalle de validité d'une paire JUMP/LABEL non imbriqués.

Remarque

La forme non imbriquée de l'instruction JUMP est la seule instruction JUMP qui peut être utilisée sur une UC Série 90-30 Version 1. L'instruction JUMP imbriquée peut être utilisée (et est même recommandée) avec toutes les nouvelles applications.

Veillez noter aussi que les UC 351 et supérieures supportent uniquement les instructions JUMP imbriquées et non la forme non imbriquée.

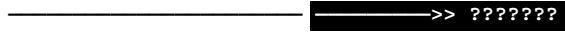
La forme imbriquée de l'instruction JUMP a la forme _____N_____>>LABEL01, où LABEL01 est le nom de l'instruction LABEL imbriquée correspondante. Elle est disponible dans les Versions 2 et ultérieures du logiciel Logicmaster 90-30/20/Micro et le micrologiciel de l'API.

Une instruction JUMP imbriquée peut être placée n'importe où dans un programme, à la condition de ne pas intervenir dans l'intervalle d'une MCR non imbriquée ou d'une JUMP non imbriquée.

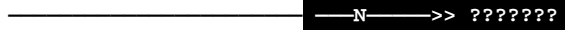
Il peut y avoir plusieurs instructions JUMP imbriquées correspondant à un même LABEL imbriqué. Les JUMP imbriqués peuvent être des sauts vers l'avant ou vers l'arrière.

Les deux formes de l'instruction JUMP sont toujours placées dans les colonnes 9 et 10 du segment en cours ; il ne peut rien y avoir après l'instruction JUMP dans le segment. Le flux validant saute directement de l'instruction vers le segment avec l'étiquette (LABEL) désignée.

JUMP non imbriqué :



JUMP imbriqué :



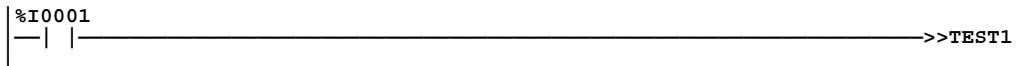
Attention

Pour éviter de créer une boucle sans fin avec des instructions JUMP vers l'avant et vers l'arrière, une instruction JUMP vers l'arrière doit contenir un argument rendant le saut conditionnel.

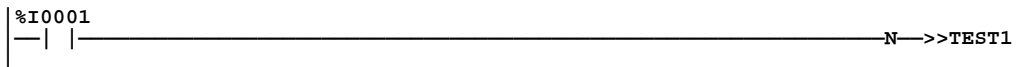
Exemple :

Dans les exemples suivants, chaque fois que JUMP TEST1 est activé, le flux validant est dirigé vers LABEL TEST1.

Exemple de JUMP non imbriqué :



Exemple de JUMP imbriqué :



LABEL

L'instruction LABEL fonctionne comme cible d'une instruction JUMP. Utiliser l'instruction LABEL pour reprendre l'exécution du programme après une instruction JUMP.

Il ne peut y avoir, dans un programme, qu'une seule instruction LABEL avec un nom d'étiquette particulier. Les programmes sans paire JUMP/LABEL associée peuvent être créés et stockés dans l'API mais ils ne peuvent pas être exécutés.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes d'instruction LABEL : une forme non imbriquée et une forme imbriquée. La forme non imbriquée (LABEL01:) doit être utilisée avec l'instruction non imbriquée JUMP, —————>>LABEL01. La forme imbriquée, LABEL01:(imbriquée), doit être utilisée avec l'instruction JUMP imbriquée, ———N——>>LABEL01.

L'instruction LABEL ne possède ni entrée ni sortie ; il ne peut rien y avoir avant ou après un LABEL dans un segment.

LABEL non imbriqué :

```
???????:
```

LABEL imbriqué :

```
???????:(imbriqué)
```

Exemple :

Dans les exemples suivants, le flux validant provenant de JUMP TEST1 est repris, démarrant à LABEL TEST1.

Exemple de LABEL non imbriqué :

```
| TEST1 :
```

Exemple de LABEL imbriqué :

```
| TEST1 :(imbriqué)
```

COMMENT

L'instruction COMMENT est utilisée pour entrer un commentaire (une explication de segment) dans le programme. Un commentaire peut comporter jusqu'à 2048 caractères de texte. Il est représenté dans le diagramme en échelle comme suit :



Le texte peut être lu ou modifié en déplaçant le curseur sur `(* COMMENT *)` après avoir accepté le segment et choisi Zoom (F10). Le texte du commentaire peut être également imprimé.

Un texte plus long peut être inclus dans les éditions en utilisant un fichier-texte d'annotation, comme indiqué ci-dessous :

1. Créer le commentaire :
 - A. Entrer le texte jusqu'au point où le texte de l'autre fichier doit commencer.
 - B. Déplacer le curseur au début d'une nouvelle ligne et entrer `\I` ou `\i`, l'unité de disque suivie de deux points (:), le répertoire, le nom du fichier, comme indiqué dans l'exemple suivant :

```
\I d:\text\commnt1
```

L'indication de l'unité de disque n'est pas utile si le fichier est situé dans celle du fichier de programme.

- C. Continuer l'édition du programme, ou passer au MS-DOS.
2. Après avoir quitté la console de programmation, créer un fichier-texte à l'aide de l'utilitaire compatible MS-DOS. Donner au fichier le nom de fichier entré dans le commentaire et placer le dans l'unité spécifiée dans le commentaire.

SVCREQ

Utiliser l'instruction Demande de service (SVCREQ) pour demander l'un des services spéciaux d'API suivants :

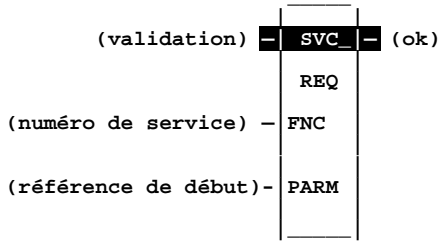
Tableau 4-3. Instructions de demande de service

Fonction	Description
6	Modifier/lire l'état de la tâche de CHECKSUM et le nombre de mots qu'elle doit vérifier.
7	Modifier/lire l'horloge interne.
13	Mettre l'API hors service.
14	Vider les Tables des défauts.
15	Lire la dernière entrée de Table des défaut enregistrée.
16	Lire le compteur de temps de fonctionnement.
18	Lire l'état de forçage des E/S.
23	Lire Checksum maître
26/30	Interroger les E/S
29	Lire le compteur de mise hors tension

L'instruction SVCREQ possède 3 paramètres d'entrée et 1 paramètre de sortie. Lorsque SVCREQ est validé, l'API reçoit l'instruction d'exécuter l'instruction FNC indiquée. Les paramètres de cette instruction commencent à la référence donnée pour PARM. L'instruction SVCREQ valide la sortie ok, à moins qu'il ne soit spécifié un numéro d'instruction incorrect, des paramètres incorrects, ou des références hors limites. D'autres causes de défaut sont décrites dans les pages suivantes.

La référence donnée pour PARM peut représenter n'importe quel type de mémoire de mots (%R, %AI, ou %AQ). Cette référence est la première d'un groupe constituant le "bloc de paramètres" de l'instruction. Les positions à 16 bits successives stockent des paramètres additionnels. Le nombre total de références demandées dépend du type d'instruction SVCREQ utilisée.

Les blocs de paramètres peuvent être utilisés comme entrées pour l'instruction et comme position où les données peuvent être sorties une fois l'instruction exécutée. De ce fait, les données retournées par l'instruction sont accessibles à une position identique à celle spécifiée pour PARM.



Paramètres :

Paramètre	Description
validation	La demande de service est exécutée lorsque Validation est activé.
FNC	FNC contient la constante ou la référence du service demandé.
PARM	PARM contient la référence de début du bloc de paramètres du service demandé.
ok	La sortie ok est activée si l'instruction est exécutée sans erreur.

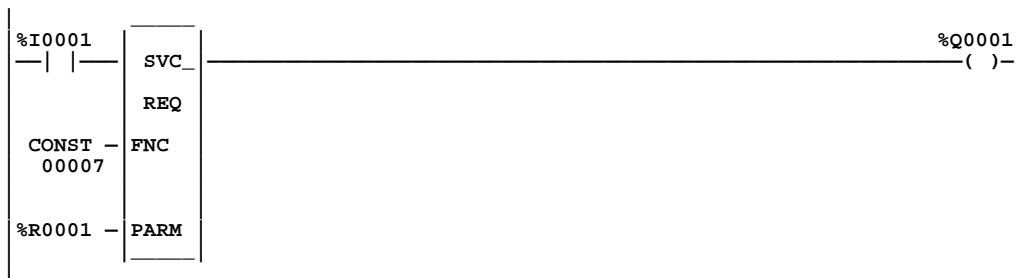
Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
FNC		•	•	•	•		•	•	•	•	•	
PARM		•	•	•	•		•	•	•	•		
ok	•											•

- Référence ou position validation où le flux validant peut traverser l'instruction.

Exemple :

Dans l'exemple suivant, lorsque l'entrée de validation %I0001 est à "1", l'instruction SVCREQ numéro 7 est appelée, avec le bloc de paramètres positionné commençant à %R0001. La bobine de sortie %Q0001 est mise à "1" si l'instruction est menée à bien.



SVCREQ #6: Modifier/lire l'état de la tâche de CHECKSUM et le nombre de mots qu'elle doit vérifier

Utiliser l'instruction SVCREQ numéro 6 pour :

- lire la valeur courante du nombre de mots pour le calcul de CHECKSUM
- écrire un nouveau nombre de mots pour le calcul de CHECKSUM

L'instruction est menée à bien, à moins qu'un nombre autre que 0 ou 1 ne soit entré comme opération demandée (voir ci-dessous).

Pour les instructions de CHECKSUM, le bloc de paramètres a une longueur de 2 mots.

Pour lire la valeur courante du nombre de bits :

Entrer l'instruction SVCREQ numéro 6 avec ce bloc de paramètres :

0	adresse
ignoré	adresse + 1

Une fois exécutée, l'instruction retourne la CHECKSUM courante dans le second mot du bloc de paramètres. Aucun intervalle n'est spécifié pour la fonction de lecture ; la valeur retournée est le nombre de mots sur lequel porte actuellement la CHECKSUM.

0	adresse
nombre actuel	adresse + 1

Pour écrire un nouveau nombre de mots :

Entrer l'instruction SVCREQ numéro 6 avec ce bloc de paramètres :

1	adresse
nouveau nombre	adresse + 1

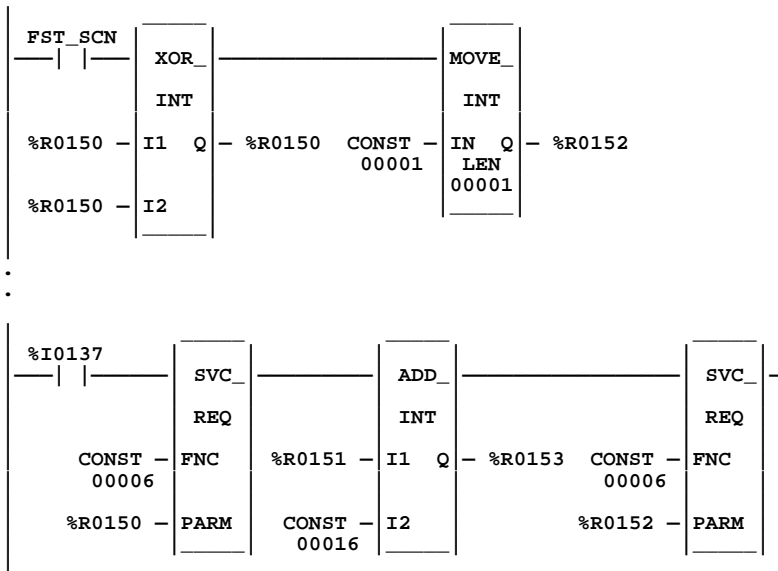
En entrant 1, l'API ajuste le nombre de mots sur lequel doit porter la CHECKSUM à la valeur donnée dans le second mot du bloc de paramètres. Pour les UC 331 et 311, cette valeur peut être 0 ou 8. Pour l'UC 211, elle peut être 0 ou 4.

Remarque

Cette demande de service n'est pas disponible sur les API Micro.

Exemple :

Dans l'exemple suivant, lorsque le contact de validation FST_SCN est mis à "1", les blocs de paramètres de la fonction de CHECKSUM sont créés. Plus loin dans le programme, lorsque l'entrée %I0137 est mise à "1", le nombre de mots sur lesquels porte la CHECKSUM est lu par le système d'exploitation de l'API. Ce nombre est augmenté de 16, les résultats de l'instruction ADD_INT étant placés dans le paramètre "hold new count for set" (mémoire nouvelle valeur). Le second bloc de demande de service demande à l'API la nouvelle valeur en service.



Dans cet exemple, les blocs de paramètres sont situés à l'adresse %R0150. Ils ont le contenu suivant :

0 = lire valeur courante	%R0150
mémoire valeur courante	%R0151
1 = mise à jour du compteur	%R0152
mémoire nouvelle valeur	%R0153

SVCREQ #7: Modifier/lire l'horloge interne

Utiliser l'instruction SVCREQ numéro 7 pour lire et régler l'horloge interne.

Remarque

Cette instruction est disponible uniquement sur les UC Série 90-30 331 ou supérieures ainsi que sur les UC des API à 28 points Série 90 Micro (c'est-à-dire IC693UDR005, IC693UAA007 et IC693UDR010) et les UC des API à 23 points Série 90 Micro (IC693UAL006).

L'instruction est exécutée correctement sauf si :

1. un nombre autre que 0 ou 1 est entré comme opération demandée (voir ci-dessous).
2. des données au format invalide sont spécifiées.
3. le format des données n'est pas celui attendu.

Pour les fonctions date/heure, la longueur du bloc de paramètres dépend du format de données. Le format DCB exige 6 mots et le format ASCII compressé 12 mots.

0 = lire heure/date	adresse
1 = régler heure/date	
1 = format DCBt	adresse + 1
3 = format ASCII compressé	
données	adresse + 2 jusqu'à la fin

Dans le mot 1, spécifier si l'instruction doit lire ou modifier les valeurs.

0	=	lire
1	=	modifier

Dans le mot 2, spécifier un format de données :

1	=	DCB
3	=	ASCII compressé avec espaces et ":" intercalés

Les mots 3 jusqu'à la fin du bloc de paramètres contiennent les données de sortie retournées par une lecture, ou de nouvelles données fournies par une modification. Dans les deux cas, le format de ces mots de données est le même. Lors de la lecture de la date et de l'heure, les mots (adresse + 2) à (adresse + 8) du bloc de paramètres sont ignorés à l'entrée.

Contenu du bloc de paramètres

Le contenu du bloc de paramètres, pour les différents formats de données, est présenté dans les pages suivantes. Pour les deux formats de données :

- les heures sont stockées en format de 24 heures.
- le jour de la semaine est une valeur numérique :

Valeur	Jour de la semaine
1	Dimanche
2	Lundi
3	Mardi
4	Mercredi
5	Jeudi
6	Vendredi
7	Samedi

Pour modifier/lire la date et l'heure en utilisant le format DCB :

En format DCB, chaque élément de l'heure et de la date occupe un seul octet. Ce format exige 6 mots. Le dernier octet du 6e mot n'est pas utilisé. Lorsque vous réglez la date et l'heure, cet octet est ignoré ; lorsque vous lisez la date et l'heure, l'instruction retourne un caractère nul (00).

Octet à "1"		Octet à "0"		
0 = modifier		ou		0 = lire
1				adresse
mois				adresse + 1
		année		adresse + 2
heures		quantième du mois		adresse + 3
secondes		minutes		adresse + 4
(vide)		jour de la semaine		adresse + 5

Exemple de bloc de paramètres de sortie :
Lire la date et l'heure en format DCB
(Lun 3 juillet 1988, 14:45:30)

0	
1	
07	88
14	03
30	45
00	01

Pour modifier/lire la date et l'heure en utilisant le format ASCII compressé avec (:) intercalé

En format ASCII compressé, chaque chiffre de l'heure et de la date occupe un octet en format ASCII. De plus, les espaces et les deux points (:) sont intercalés dans les données pour les sortir tels quels sur une imprimante ou une console d'affichage. Ce format exige 12 mots.

Octet à "1"		Octet à "0"		
1 = modifier	ou	0 = lire		adresse
3				adresse + 1
année		année		adresse + 2
mois		(espace)		adresse + 3
(espace)		mois		adresse + 4
quantième du mois		quantième du mois		adresse + 5
heures		(espace)		adresse + 6
:		heures		adresse + 7
minutes		minutes		adresse + 8
secondes		:		adresse + 9
(espace)		secondes		adresse + 10
jour de la semaine		jour de la semaine		adresse + 11

Exemple de bloc de paramètres de sortie :
Lire la date et l'heure en format ASCII compressé
(Jeu 2 oct. 1989, 23:13:00)

0	
3	
39	38
31	20
20	30
32	30
32	20
3A	33
33	31
30	3A
20	30
32	30

SVCREQ #13: Mettre hors service (arrêter) l'API

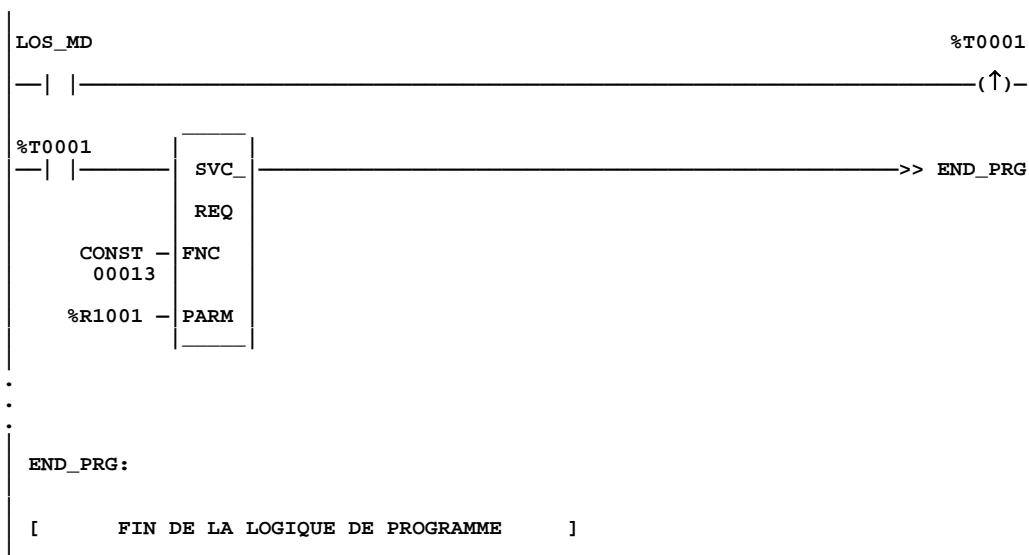
L'instruction SVCREQ numéro 13 est utilisée pour arrêter l'API à la fin du cycle courant. Toutes les sorties prennent leur état par défaut, désigné, au début du cycle d'API suivant. Un défaut informationnel est placé dans la Table des défauts automate, indiquant qu'un bloc fonctionnel "SHUT DOWN PLC" a été exécuté. La scrutation des E/S continue telle qu'elle a été configurée.

Cette instruction n'a pas de bloc de paramètres.

Exemple :

Dans l'exemple suivant, lorsqu'un défaut "module d'E/S perdu" se produit, l'instruction SVCREQ numéro 13 est exécutée. Comme aucun bloc de paramètres n'est nécessaire, l'entrée PARM n'est pas utilisée ; toutefois, le logiciel de programmation exige que PARM ait une entrée.

Cet exemple utilise une instruction JUMP jusqu'à la fin du programme pour forcer la mise hors service de l'API si l'instruction Shutdown PLC est menée à bien. Ce saut (JUMP) et cette étiquette (LABEL) sont nécessaires car la transition vers le mode **STOP** n'intervient qu'à la fin du cycle dans lequel l'instruction est exécutée.



Remarque

Pour s'assurer que le contact %S0002 LST_SCN fonctionne correctement, l'API exécute un cycle supplémentaire après celui dans lequel l'instruction SVCREQ #13 a été exécutée.

SVCREQ #15: Lire la dernière entrée enregistrée dans la table des défauts

L'instruction SVCREQ numéro 15 est utilisée pour lire la dernière entrée enregistrée dans la Table des défauts automate ou la Table des défauts d'E/S. La sortie SVCREQ est mise à "1" sauf si un autre nombre autre que 0 ou 1 est entré comme opération demandée (voir ci-dessous), ou si la table des défauts est vide. (Pour plus d'informations sur les entrées des tables des défauts, voir le chapitre 3, "Description et correction des défauts".)

Pour cette instruction, le bloc de paramètres a une longueur de 22 mots. Le bloc de paramètres d'entrée a le format suivant :

0 = Lire la Table des défauts automate.	adresse
1 = Lire la Table des défauts d'E/S.	

Le format du bloc de paramètres de sortie dépend de la table des défauts qui sera lue par l'instruction.

Format de sortie de la Table des défauts automate

Octet à "0"	Octet à "1"
0	
long/court	adresse + 1
de remplissage	adresse + 2
adresse du défaut automate	adresse + 3 adresse + 4
groupe de défauts et intervention défaut	adresse + 5
code d'erreur	adresse + 6 adresse + 7 adresse + 8 adresse + 9 adresse + 10 adresse + 11
données spécifiques au défaut	adresse + 12 adresse + 13 adresse + 14 adresse + 15 adresse + 16 adresse + 17 adresse + 18
marquage date/heure	adresse + 19 adresse + 20 adresse + 21

Format de sortie de la Table des défauts d'E/S

Octet à "0"	Octet à "1"
	1
long/court	adresse de référence
	adresse du défaut d'E/S
	groupe de défauts et intervention défaut
	catégorie de défaut type de défaut
	description du défaut
	données spécifiques au défaut
	marquage date/heure

Exemple 2 :

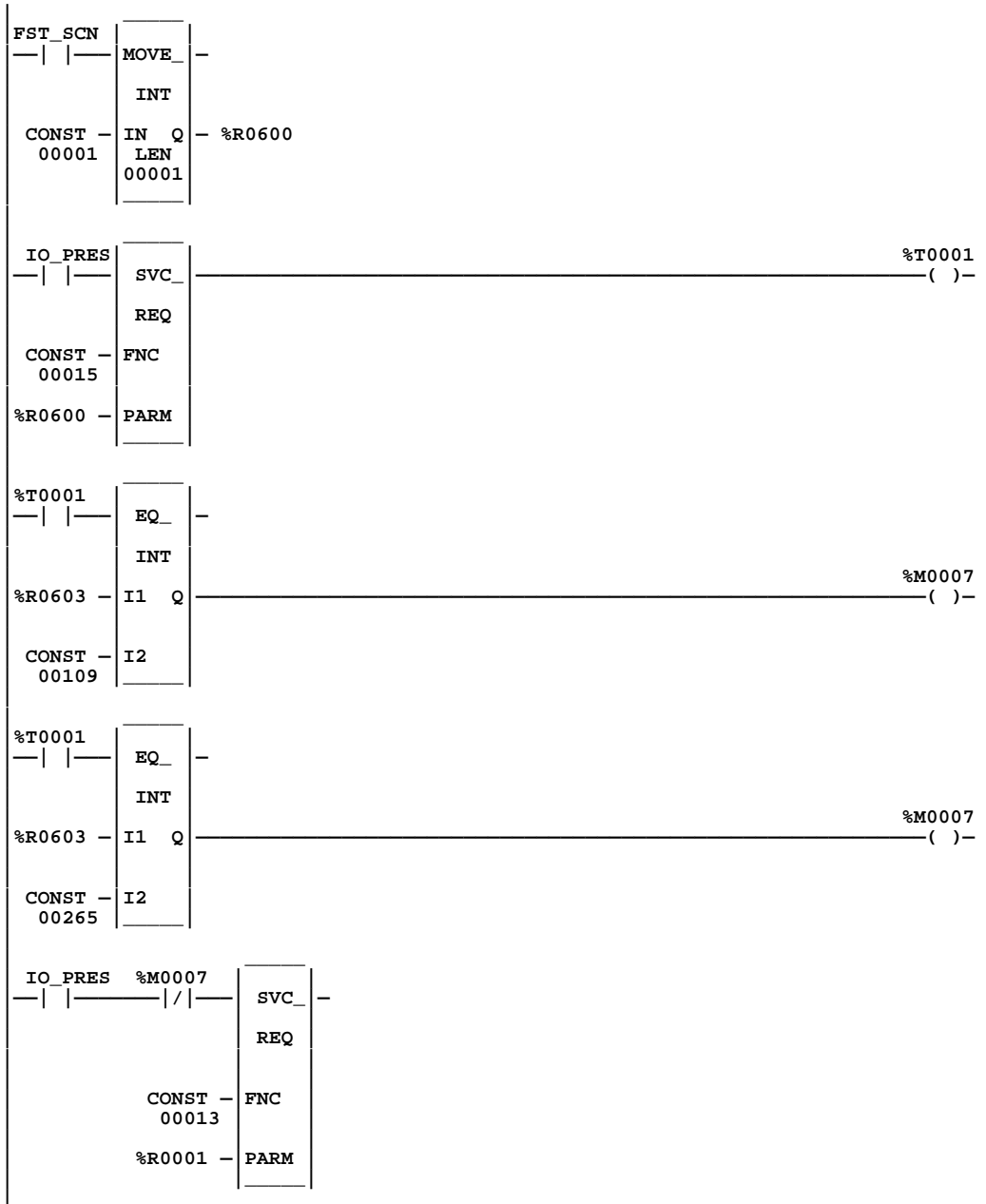
Dans l'exemple suivant, l'API est mis hors service lorsqu'un défaut se produit dans un module d'E/S, sauf si le défaut se produit dans les modules du bac 0, emplacement 9, et du bac 1, emplacement 9. Si des défauts se produisent dans ces deux modules, le système continue à fonctionner. Le paramètre pour "type de table" est configuré lors du premier cycle. Le contact IO_PRES, lorsque mis à "1", indique la Table des défauts d'E/S contient une entrée. L'UC de l'API met à "1" le contact N.O. dès le premier cycle après que la logique de défaut a placé un défaut dans la table. Si des défauts sont placés dans la table pendant deux cycles consécutifs, le contact N.O. est mis à "1" pendant deux cycles consécutifs.

Cet exemple utilise un bloc de paramètres situé à %R0600. Une fois l'instruction SVCREQ exécutée, les 4e, 5e et 6e mots du bloc de paramètres contiennent l'adresse du module d'E/S mis en défaut :

1		%R0600
long/court		%R0601
adresse de référence		%R0602
numéro de bac	numéro d'emplacement	%R0603
numéro du bus d'E/S	adresse du bus	%R0604
adresse du bit		%R0605

données du défaut

Dans le programme, les blocs EQ_INT comparent l'adresse du bac/emplacement dans la table aux constantes hexadécimales. La bobine interne %M00007 est mise à "1" lorsque le bac/emplacement où le défaut s'est produit rencontre le critère spécifié ci-dessus. Si %M00007 est mise à "1", son contact N.F. est mis à "0", évitant la mise hors service. A l'inverse, si %M00007 est mise à "0" du fait que le défaut s'est produit dans un module différent, le contact N.F. est mis à "1" et la mise hors service se produit.



SVCREQ #16: Lire le compteur de temps de fonctionnement

L'instruction SVCREQ numéro 16 est utilisée pour lire le compteur de temps de fonctionnement, indiquant le temps écoulé (en secondes) depuis la mise sous tension de l'API. Le temporisateur retourne à zéro environ une fois tous les 100 ans.

Cette instruction possède uniquement un bloc de paramètres. Le bloc de paramètres a une longueur de 3 mots.

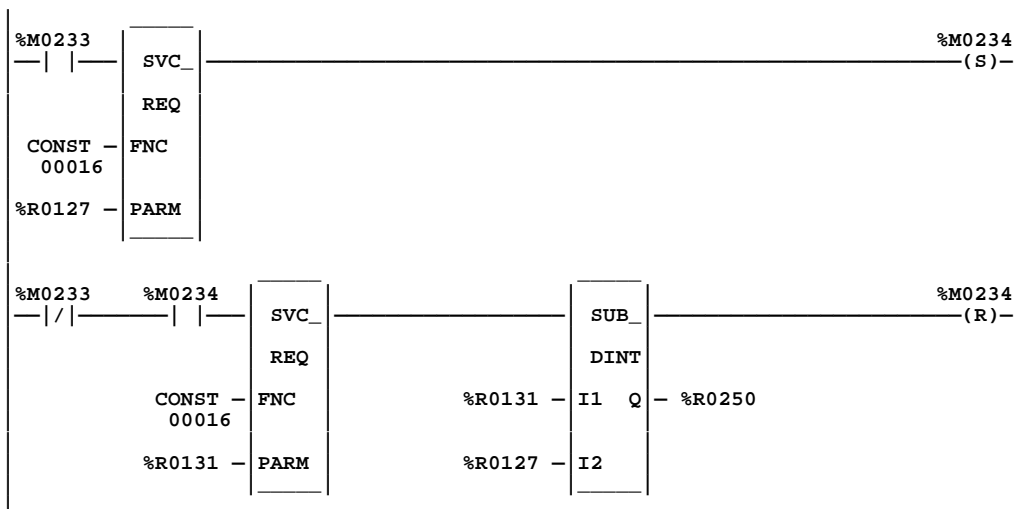
secondes depuis mise sous tension (poids faible)	adresse
secondes depuis mise sous tension (poids fort)	adresse + 1
impulsions de 100 microsecondes	adresse + 2

Les deux premiers mots sont le temps écoulé (en secondes). Le dernier mot est le nombre d'impulsions de 100 microsecondes dans la seconde actuelle.

Exemple :

Dans l'exemple suivant, lorsque la bobine interne %M0233 est à "1", la valeur du compteur de temps de fonctionnement est lue et la bobine %M0234 est mise à "1". Lorsqu'elle est à "0", la valeur est lue à nouveau. La différence entre les valeurs est ensuite calculée, puis le résultat stocké dans la mémoire des registres à l'emplacement %R0250.

Le bloc de paramètres pour la première lecture est à %R0127, et pour la seconde lecture à %R0131. Le calcul ignore le nombre d'impulsions de 100 microsecondes et que le type DINT est en réalité une valeur signée. Le calcul est correct jusqu'à ce que le temps écoulé depuis la mise sous tension soit d'environ 50 ans.



SVCREQ #18: Lire l'état des forçages des E/S

L'instruction SVCREQ numéro 18 est utilisée pour lire l'état courant des forçages dans l'UC.

Remarque

Cette instruction est disponible *uniquement* pour les UC 331 ou supérieures.

Pour cette instruction, le bloc de paramètres a une longueur de 1 mot. C'est uniquement un bloc de paramètres de sortie.

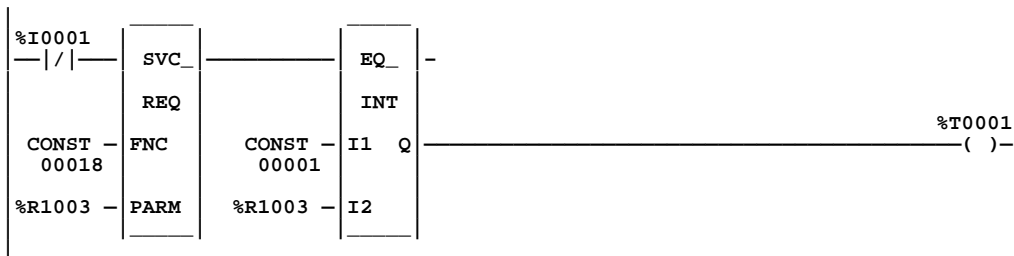
0 = Pas de forçage activé.	adresse
1 = Les forçages sont mis.	

Remarque

SVCREQ #18 signale uniquement les forçages des références %I et %Q.

Exemple :

Dans l'exemple suivant, l'état des forçages des E/S est toujours écrit dans l'emplacement %R1003. Si aucun forçage n'est présent, la sortie %T0001 est mise à "1".



SVCREQ #23: Lire Checksum maître

Utilisez l'instruction SVCREQ numéro 23 pour lire les checksums maître du programme utilisateur et de la configuration. La sortie SVCREQ est toujours mise à "1" si l'instruction est activée et le bloc de sortie d'information (voir ci-dessous) commence à l'adresse donnée dans le paramètre 3 (PARM) de l'instruction SVCREQ.

Quand un **RUN MODE STORE** est actif, les checksums du programme peuvent ne pas être valides aussi longtemps que le stockage n'est pas terminé. Deux drapeaux sont donc fournis au début du bloc de paramètres de sortie pour indiquer à quel moment les checksums du programme et de la configuration sont valides.

Pour cette instruction, le bloc de paramètres de sortie a une longueur de 12 mots avec le format suivant :

Checksum du programme maître valide (0 = non valide, 1 = valide)	adresse
Checksum de la configuration maître valide (0 = non valide, 1 = valide)	adresse + 1
Nombre de blocs de programme (y compris _MAIN)	adresse + 2
Taille en octets du programme utilisateur (type de données DWORD)	adresse + 3
Checksum de programme supplémentaire	adresse + 5
Checksum CRC du programme (type de données DWORD)	adresse + 6
Taille en octets des données de configuration	adresse + 8
Checksum de configuration supplémentaire	adresse + 9
Checksum CRC de la configuration (type de données DWORD)	adresse + 10

Exemple :

Dans l'exemple ci-dessous, quand l'entrée %I0251 est à "1", les informations de checksum maître sont stockées dans le bloc de paramètres et la bobine de sortie (%Q0001) est mise à "1". Le bloc de paramètres se trouve à l'emplacement mémoire %R0050.



SVCREQ #26/30: Interroger les E/S

Utilisez l'instruction SVCREQ numéro 26 (ou numéro 30 ; ces instructions étant identiques, vous pouvez utiliser l'un ou l'autre numéro pour exécuter la même tâche) pour interroger les modules actuellement présents et les comparer avec la configuration de bac/emplacement, en générant des alarmes d'ajout, de suppression ou d'incohérence, comme si vous aviez procédé à un stockage de la configuration. Cette instruction SVCREQ génère des défauts dans les tables des défauts de l'API et des E/S, selon le défaut.

Cette instruction est dépourvue de bloc de paramètres et produit toujours le flux validant.

Remarque

La durée d'exécution de cette instruction SVCREQ varie en fonction du nombre de défauts. L'exécution de cette instruction SVCREQ peut donc durer plus longtemps lorsque plusieurs modules sont défectueux.

Exemple :

Dans l'exemple ci-dessous, quand l'entrée %I0251 est à "1", les modules présents sont interrogés et comparés avec la configuration de bac/emplacement . La sortie %Q0001 est mise à "1" une fois que l'exécution de SVCREQ est terminée.



Remarque

Cette demande de service n'est pas disponible sur les API Micro.

SVCREQ #29 : Lire le compteur de mise hors tension

Utilisez l'instruction SVCREQ numéro 29 pour lire la durée écoulée entre la dernière mise hors tension et la mise sous tension la plus récente. La sortie SVCREQ est toujours mise à "1" et le bloc de sortie d'informations (voir ci-dessous) commence à l'adresse spécifiée dans le paramètre 3 (PARM) de l'instruction SVCREQ.

Remarque

Cette instruction est disponible uniquement sur les UC 331 et supérieures.

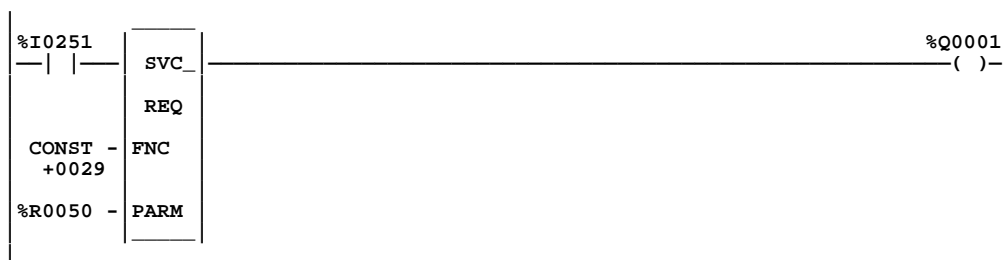
Cette instruction ne possède qu'un bloc de paramètres de sortie, d'une longueur de 3 mots.

Durée en secondes de la mise hors tension (ordre inférieur)	adresse
Durée en secondes de la mise hors tension (ordre supérieur)	adresse + 1
Intervalles de 100 microsecondes	adresse + 2

Les deux premiers mots correspondent à la durée en secondes de la mise hors tension et le dernier, à la durée de mise hors tension restante, par intervalles de 100 microsecondes (qui est toujours égal à 0). Chaque fois que l'API ne peut pas calculer correctement la durée de mise hors tension, la durée est mise à "0". Cela se produit notamment lorsque l'API est mise sous tension en appuyant sur la touche CLR M/T de la miniconsole de programmation ou lorsque le temporisateur chien de garde expire avant la mise hors tension.

Exemple :

Dans l'exemple ci-dessous, quand l'entrée %I0251 est mise à "1", le compteur de mise hors tension est placé dans le bloc de paramètres et la bobine de sortie (%Q0001) est mise à "1". Le bloc de paramètres figure à l'emplacement mémoire %R0050.



sera exécutée à raison d'un cycle sur deux avec un intervalle de 18 millisecondes entre chaque exécution.

Paramètres :

Paramètre	Description
validation	L'instruction PID est exécutée lorsqu'elle est validée par l'intermédiaire d'un contact.
SP	SP est le point de consigne du procédé ou de la boucle de commande. Il est défini à l'aide des compteurs PV tandis que PID ajuste la sortie CV afin que PV concorde avec SP (erreur zéro).
PV	Variable de procédé entrée à partir du procédé qui est commandé, souvent une entrée %AI.
MAN	Lorsqu'il est mis à "1" (via un contact), le bloc PID est en mode MANUAL . Si le bloc PID est mis manuellement à "0", il est en mode automatique.
UP	Lorsqu'il est activé avec MAN, il augmente CV à raison d'un CV par solution.*
DN	Lorsqu'il est activé avec MAN, il diminue CV à raison d'un CV par solution.*
RefArray Address	Adresse est l'emplacement des informations du bloc de commande PID (paramètres internes et paramètres utilisateur). Il utilise 40 mots %R qui ne peuvent pas être partagés.
ok	La sortie ok est activée quand l'instruction est exécutée sans erreur. En présence d'une ou plusieurs erreurs, elle est mise à "0".
CV	CV est la sortie de la variable de commande vers le procédé, souvent une sortie analogique %AQ.

*Incrémenté (paramètre UP) ou décrémenté (paramètre DN) d'une unité (1) par accès de l'instruction PID.

Types de mémoires valides :

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
SP		•	•	•	•		•	•	•	•	•	
PV		•	•	•	•		•	•	•	•		
MAN	•											
UP	•											
DN	•											
adresse								•				
ok	•											•
CV		•	•	•	•		•	•	•	•		

- Référence ou position validation où le flux validant peut traverser l'instruction.

Bloc de paramètres PID :

Outre les 2 mots d'entrée et les 3 contacts de commande manuelle, le bloc PID utilise 13 paramètres de RefArray, qui doivent être définis avant d'appeler le bloc. Les autres paramètres sont utilisés par l'API et ne sont pas configurables. %Ref qui figure dans le tableau ci-dessous correspond à la même adresse RefArray à la fin du bloc PID. Le chiffre qui suit le signe plus (+) désigne le décalage dans le tableau. Si RefArray commence à l'emplacement %R100, par exemple, %R113 contiendra la commande manuelle utilisée pour définir CV et l'intégrateur au mode manuel.

Tableau 4-4. Présentation des paramètres PID

Registre	Paramètre	Unités du bit inférieur	Plage de valeurs
%Ref+0000	Numéro de boucle	Entier	0 à 255 (pour l'affichage de l'utilisateur uniquement)
%Ref+0001	Algorithme	N/A; défini et géré par l'API	Non configurable
%Ref+0002	Période d'échantillonnage	10 millisecondes	0 (chaque cycle) à 65535 (10,9 Min). Utilisez au moins 10 pour les API 90-30 (voir Remarque à la page 4-146).
%Ref+0003	Bande morte +	Comptages PV	0 à 32000 (jamais négatif)
%Ref+0004	Bande morte —	Comptages PV	-32000 à 0 (jamais positif)
%Ref+0005	Gain proportionnel -Kp	0,01 CV%/PV%	0 à 327,67 %/%
%Ref+0006	Dérivé -Kd	0,01 seconde	0 à 327,67 sec.
%Ref+0007	Coefficient intégral -Ki	Répétition/1000 Sec	0 à 32,767 répétitions/sec.
%Ref+0008	Décalage en sortie	Comptages CV	-32000 à 32000 (ajouté à la sortie de l'intégrateur)
%Ref+0009	Limite haute	Comptages CV	-32000 à 32000 (>%Ref+10) limite de sortie
%Ref+0010	Limite basse	Comptages CV	-32000 à 32000(<%Ref+09) limite de sortie
%Ref+0011	Temps minimum de parcours de l'échelle totale	Seconde/Parcours total	0 (aucun) à 32000 sec. pour transférer 32000 CV
%Ref+0012	Mot de configuration	Utilisation des 5 bits inférieurs	Bits 0 à 2 pour Erreur+/-, polarité de sortie, action dérivée.
%Ref+0013	Commande manuelle	Comptages CV	Suit CV en mode auto ou définit CV en mode manuel
%Ref+0014	Mot de commande	Géré par l'API, <i>sauf si</i> le bit 1 est mis à "1".	Géré par l'API sauf s'il est défini autrement : remplace la définition des bits inférieurs s'il est mis à "1" (voir la description dans le tableau "Détails des paramètres PID", à la page 4-153)
%Ref+0015	Consigne (SP)	N/A; défini et géré par l'API	Non configurable
%Ref+0016	Variable de commande (CV)	N/A; défini et géré par l'API	Non configurable
%Ref+0017	Variable procédé (PV)	N/A; défini et géré par l'API	Non configurable
%Ref+0018	Sortie	N/A; défini et géré par l'API	Non configurable

Tableau 4-4. Présentation des paramètres PID (suite)

Registre	Paramètre	Unités des bits inférieurs	Plage des valeurs
%Ref+0019	Stockage du terme différentiel	N/A; défini et géré par l'API	Non configurable
%Ref+0020 and %Ref+0021	Stockage du terme intégral	N/A; défini et géré par l'API	Non configurable
%Ref+0022	Stockage du terme de parcours de l'échelle totale	N/A; défini et géré par l'API	Non configurable
%Ref+0023	Horloge	N/A; défini et géré par l'API	Non configurable
%Ref+0024	(temps écoulé de la dernière exécution)		
%Ref+0025			
%Ref+0026	Stockage du reste Y	N/A; défini et géré par l'API	Non configurable
%Ref+0027	Plage basse pour SP, PV	Comptages PV	-32000 à 32000 (>%Ref+28) pour l'affichage
%Ref+0028	Plage haute pour SP, PV	Comptages PV	-32000 à 32000 (<%Ref+27) pour l'affichage
%Ref+0029 • %Ref+0034	Réservé pour usage interne	N/A	Non configurable
%Ref+0035 • %Ref+0039	Réservé pour usage interne	N/A	Non configurable

Le tableau RefArray doit comporter %R registres sur l'API 90-30. Notez que chaque appel de bloc PID doit utiliser un tableau de 40 mots différent, même si les 13 paramètres utilisateur sont identiques, car d'autres mots du tableau sont réservés au stockage des données PID internes. Assurez-vous que le tableau ne s'étend pas au-delà de la mémoire.

Pour configurer les paramètres utilisateur, sélectionnez l'instruction PID et appuyez sur la touche **F10** pour effectuer un zoom avant de l'écran des paramètres utilisateur ; utilisez ensuite les touches fléchées pour sélectionner des champs, puis tapez les valeurs souhaitées. Vous pouvez utiliser 0 pour la plupart des valeurs par défaut, à l'exception de la limite haute CV, qui doit être supérieure à la limite basse pour que le bloc PID fonctionne. Notez que le bloc PID ne passe **pas** le flux en présence d'une erreur dans les paramètres utilisateur. Surveillez donc la modification des données à l'aide d'une bobine temporaire.

Une fois que vous avez sélectionné les valeurs PID appropriées, vous devez les définir comme constantes dans BLKMOV afin de pouvoir éventuellement les utiliser pour recharger les paramètres utilisateur PID par défaut.

Fonctionnement de l'instruction

En mode de fonctionnement automatique normal, le bloc PID est appelé à chaque cycle en passant le flux validant au contact de validation, mais pas aux contacts d'entrée manuelle. Le bloc compare l'horloge de l'API en cours avec l'heure de la dernière solution PID stockée dans le RefArray interne. Si la différence de temps est supérieure à la période d'échantillonnage définie dans le troisième mot (%Ref+2) de RefArray, l'algorithme PID est résolu à l'aide de la différence temporelle tandis que l'heure de la dernière solution et la sortie CV sont mises à jour. En mode automatique, la sortie CV est placée dans le paramètre de commande manuelle %Ref+13.

Si le flux validant est fourni aux contacts de validation et aux contacts d'entrée manuelle, le bloc PID est placé en mode manuel et la sortie CV est définie à partir du paramètre de commande manuelle %Ref+13. Si les entrées UP ou DN sont activées, le mot de la commande manuelle est incrémenté ou décrémenté d'un comptage CV pour chaque solution PID. Afin d'accélérer les modifications manuelles de la sortie CV, il est possible d'ajouter ou de soustraire directement n'importe quelle valeur de comptage CV au/du mot de la commande manuelle.

Le bloc PID utilise les paramètres de limites haute et basse pour limiter la sortie CV. Si un temps minimum de parcours de l'échelle totale est défini, il est utilisé pour limiter le coefficient de modification de la sortie CV. En cas de dépassement de la limite de coefficient ou d'amplitude CV, la valeur stockée dans l'intégrateur est ajustée de façon à adapter CV à la limite. Grâce à la fonction d'anti-saturation (définie à la page 4-40), même si l'erreur a tenté d'entraîner CV au-delà (ou en-deçà) des limites pendant une période prolongée, la sortie CV est extraite de la limite dès que le terme de l'erreur change de signe.

Le suivi de CV par la commande manuelle en mode automatique ainsi que le passage de CV au mode manuel assure un transfert sans heurt entre les modes automatique et manuel. Les limites haute et basse de CV ainsi que le temps minimum de parcours de l'échelle totale sont toujours appliqués à la sortie CV en mode manuel tandis que la valeur interne stockée dans l'intégrateur est mise à jour. Cela signifie que si vous tentez de modifier la commande manuelle en mode manuel, la sortie CV ne changera pas aussi rapidement que la limite du temps minimum de parcours de l'échelle totale et ne dépassera pas les limites haute et basse de CV.

Remarque

Une instruction PID spécifique ne doit pas être appelée plus d'une fois par cycle.

Le tableau ci-dessous fournit davantage de détails sur les paramètres énumérés dans le tableau 4-4. Le numéro entre parenthèses qui suit le nom de chaque paramètre correspond au décalage dans le RefArray.

Tableau 4-5. Détails des paramètres PID

Elément de données	Description
Numéro de boucle (00)	Ce paramètre optionnel permet d'identifier un bloc PID. Il s'agit d'un entier sans signe offrant une identification commune dans l'API avec le numéro de boucle défini par une interface opérateur. Le numéro de boucle s'affiche sous l'adresse du bloc lorsque la logique est contrôlée à partir du logiciel Logicmaster 90-30/20/Micro.
Algorithme (01)	Nombre entier sans signe réglé par l'API pour identifier l'algorithme utilisé par le bloc fonctionnel. L'algorithme ISA est défini comme algorithme 1 et l'algorithme indépendant, comme algorithme 2.
Période d'échantillonnage (02)	Temps le plus court, en incréments de 10 millisecondes, entre les solutions de l'algorithme PID. Utilisez 10, par exemple, pour une période d'échantillonnage de 100 millisecondes. UINT peut prendre la valeur maximale 65535 pour une période d'échantillonnage de 10,9 minutes. Si elle est mise à "0", l'algorithme est résolu chaque fois que le bloc est appelé (voir la section ci-dessous relative à la planification du bloc PID). L'algorithme PID n'est résolu que si l'horloge de l'API courante est égale ou supérieure à l'heure de la dernière solution PID + cette période d'échantillonnage. N'oubliez pas que les API 90-30 n'utilisent pas les temps de solution inférieurs à 10 millisecondes (voir la Remarque à la page 4-146) ; des temps de cycle plus courts entraînent donc le saut de certains cycles. Cette instruction compense le temps réel écoulé depuis la dernière exécution, à 100 microsecondes près. Si cette valeur est mise à "0", l'instruction est exécutée chaque fois qu'elle est validée ; elle est toutefois limitée à 10 millisecondes minimum, comme décrit plus haut.
Bande morte (+/–) (03/04)	Valeur INT définissant les limites supérieure (+) et inférieure (–) de bande morte des comptages PV. Si aucune bande morte n'est nécessaire, ces valeurs doivent être mises à "0". Si l'erreur PID (SP – PV) ou (PV – SP) est supérieure à la valeur (–) et inférieure à la valeur (+), les calculs PID sont résolus en mettant l'erreur à "0". Si celle-ci n'est pas à "0", la valeur (+) doit être supérieure à zéro et la valeur (–) inférieure à 0, sinon le bloc PID ne fonctionnera pas. <i>Vous devez laisser ces valeurs à "0" jusqu'à ce que les gains en boucle PID soient définis ou ajustés.</i> Ensuite, vous souhaitez peut-être ajouter une bande morte pour éviter toute modification mineure de la sortie CV due à des légères variations de l'erreur, visant notamment à réduire l'usure mécanique.
Gain proportionnel–Kp (05)	Ce nombre INT, appelé gain de contrôleur dans la version ISA, détermine le changement de CV dans les comptages CV pour une modification de comptage PV 100 dans le terme d'erreur. Il est affiché sous la forme 0,00 %/% avec deux chiffres décimaux par défaut. Un Kp de 450, par exemple, est affiché sous la forme 4,50 et donne pour résultat une contribution de $Kp \cdot \text{Erreur} / 100$ ou $450 \cdot \text{Erreur} / 100$ à la sortie PID. Kp est généralement le premier gain qui est défini pendant le réglage d'une boucle PID.
Dérivé–Kd (06)	Ce nombre INT détermine le changement de CV dans les comptages CV si Erreur ou PV change à raison d'un comptage PV toutes les 10 millisecondes. Entré sous la forme d'une heure dont le bit inférieur indique 10 millisecondes, ce nombre est affiché à l'aide du format 0,00 secondes avec deux chiffres décimaux par défaut. Un Kd de 120, par exemple, est affiché sous la forme 1,20 Sec et donne pour résultat une contribution de $Kd \cdot \text{Erreur} \cdot \Delta t$ ou $120 \cdot \Delta t$ à la sortie PID si l'erreur a été modifiée par 4 comptages PV toutes les 30 millisecondes. Kd peut être utilisé pour accélérer une réponse en boucle lente, mais est extrêmement sensible aux parasites de l'entrée PV.
Coefficient intégral–Ki (07)	Ce nombre INT détermine le changement de CV dans les comptages CV si Erreur correspond à un comptage PV constant. Il s'affiche sous la forme de 0,000 répétitions/sec. avec 3 chiffres décimaux par défaut. Un Ki de 1400, par exemple, est affiché sous la forme 1,400 répétitions/sec et donne pour résultat une contribution de $Ki \cdot \text{Erreur} \cdot dt$ ou $1400 \cdot 20 \cdot 50 / 1000$ à la sortie PD pour une erreur de 20 comptages PV et un temps de cycle de l'API de 50 millisecondes (avec une période d'échantillonnage de 0). Ki est généralement le deuxième gain qui est défini après Kp.
Décalage en sortie (08)	Valeur INT dans les comptages CV qui est ajoutée à la sortie PID avant les limites de coefficient et d'amplitude. Il peut servir à définir des valeurs CV différentes de 0 lorsque seuls des gains proportionnels Kp sont utilisés ou pour faire avancer la commande de cette sortie en boucle PID depuis une autre boucle de commande.

Tableau 4-5. Détails des paramètres PID - suite

Élément de données	Description
Limites CV haute et basse (09/10)	Valeurs INT des comptages CV qui définissent les valeurs supérieure et inférieure de CV. Ces valeurs sont obligatoires et la limite haute doit posséder une valeur supérieure à la limite basse pour que le bloc PID puisse fonctionner. Elles sont généralement utilisées pour définir des limites basées sur des limites physiques d'une sortie CV. Elles permettent aussi de convertir le graphique à barres de CV de l'affichage PID LM90 ou ADS. Le bloc possède une fonction d'anti-saturation qui permet de modifier la valeur de l'intégrateur quand une limite du CV est atteinte.
Temps minimum de parcours de l'échelle totale (11)	Valeur UINT positive qui définit le nombre minimal de secondes requis par la sortie CV pour passer de 0 à l'échelle totale de 100% ou à 32000 comptages CV. Cet élément limite la rapidité à laquelle la sortie CV peut être modifiée. Quand la valeur est positive, CV ne peut pas changer de plus de 32000 comptages CV pendant la durée Delta (secondes) divisée par le temps minimum de parcours de l'échelle totale. Si la période d'échantillonnage est de 2,5 secondes, par exemple, et si le temps minimum de parcours de l'échelle totale est de 500 secondes, CV ne peut pas changer de plus de $32000 * 2,5 / 500$ ou de 160 comptages CV par solution PID. De même que les limites CV, une fonction anti-saturation ajuste la valeur de l'intégrateur en cas de dépassement de la limite CV. Si le temps minimum de parcours de l'échelle totale est égal à 0, il n'y a pas de limite de coefficient CV. N'oubliez pas d'affecter la valeur 0 au temps minimum de parcours de l'échelle totale pendant que vous définissez ou ajustez des gains en boucle PID.
Mot de configuration	<p>Les 5 premiers bits de ce mot sont utilisés pour modifier trois paramètres PID standard. Les autres bits doivent être mis à "0". Mettez le premier bit à "1" pour remplacer le terme d'erreur PID standard normal (SP – PV) par (PV – SP), en inversant le signe du terme de retour. Cela permet d'inverser l'action des commandes où CV doit descendre quand PV monte. Mettez le deuxième bit à "1" pour inverser la polarité de sortie afin que CV corresponde à la valeur négative de la sortie PID plutôt qu'à la valeur positive normale. Mettez le quatrième bit à "1" pour modifier l'action dérivée de façon à ce qu'elle n'utilise pas la modification normale du terme d'erreur mais bien celle du terme de retour de PV.</p> <p>Les cinq premiers bits du mot de configuration sont définis en détail ci-dessous :</p> <p>Bit 0 = Terme d'erreur. Lorsque ce bit est mis à "0", le terme d'erreur est SP – PV. Lorsque ce bit est mis à "1", le terme d'erreur est PV – SP.</p> <p>Bit 1 = Polarité de sortie. Lorsque ce bit est mis à "0", la sortie CV représente la sortie du calcul PID. Lorsque ce bit est mis à "1", la sortie CV représente la négative de la sortie du calcul PID.</p> <p>Bit 2 = Action dérivée sur PV. Lorsque ce bit est mis à "0", l'action dérivée est appliqué au terme d'erreur. Lorsqu'il est mis à "1", l'action dérivée est appliquée à PV. Tous les autres bits doivent être mis à "0".</p> <p>Bit 3 = Action de bande morte. Quand elle est mise à "0", aucune action de bande morte n'est sélectionnée. Si l'erreur se trouve dans les limites de la bande morte, sa mise à "0" est forcée. Sinon, l'erreur n'est pas affectée par les limites de la bande morte. Si le bit de l'action de bande morte est mis à "1", l'action de bande morte est sélectionnée. Si l'erreur se situe dans les limites de la bande morte, sa mise à "0" est forcée. Si l'erreur se trouve en dehors des limites de la bande morte, elle est réduite selon la limite de bande morte (erreur = erreur – limite de bande morte).</p> <p>Bit 4 = Action d'anti-saturation. Quand ce bit est mis à "0", l'action d'anti-saturation utilise un calcul de réinitialisation. Quand la sortie est limitée, elle remplace la valeur du reste Y accumulé (défini à la page 4-40) par la valeur requise pour produire la sortie limitée exacte. Quand le bit est mis à "1", il remplace le terme Y accumulé par la valeur du terme Y au début du calcul. De cette façon, la valeur Y de prélimite est conservée aussi longtemps que la sortie est limitée.</p> <p>REMARQUE : L'action d'anti-saturation est disponible uniquement sur les UC 90-30 version 6.50 et ultérieures.</p> <p>N'oubliez pas que les bits sont définis en puissances de 2. Pour mettre à "0" le mot de configuration pour la configuration PID par défaut, par exemple, vous devez ajouter 1 pour remplacer le terme d'erreur SP-PV par PV-SP, ajouter 2 pour remplacer la polarité de sortie de CV = sortie PID par CV = –sortie PID ou ajouter 4 pour ne plus appliquer l'action dérivée au terme d'erreur, mais bien à PV, etc.</p>

Tableau 4-5. Détails des paramètres PID - suite

Elément de données	Description																								
Commande manuelle (13)	Valeur INT définie à l'aide de la sortie CV courante quand le bloc PID est en mode automatique. Lorsque le bloc passe en mode Manuel , cette valeur est utilisée pour définir la sortie CV et la valeur interne de l'intégrateur au sein des limites haute et basse et du temps minimum de parcours de l'échelle totale.																								
Mot de commande (14)	<p>Paramètre interne qui demeure en principe à "0".</p> <p>Si le bit Forçage est mis à "1", ce mot ainsi que d'autres paramètres CV, PV et SP internes doivent être utilisés pour commander le fonctionnement à distance de ce bloc PID (voir plus bas). Cela permet à un périphérique d'interface opérateur distant tel qu'un ordinateur d'empêcher le programme de l'API de commander. Attention : Si vous ne souhaitez pas que cela se produise, assurez-vous que le mot de commande est mis à "0". Si le premier bit est à "0", les 4 bits suivants peuvent être lus pour suivre l'état des contacts d'entrée PID aussi longtemps que le contact de validation PID possède le flux validant. Ce mot correspond à une structure de données discrètes dont les positions des cinq premiers bits ont le format suivant :</p> <table border="1"> <thead> <tr> <th>Bit:</th> <th>Valeur du mot:</th> <th>Fonction:</th> <th>Etat ou action externe si le bit de forçage est mis à 1 :</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>Forçage</td> <td>S'il est mis à "0", il surveille les contacts de bloc ci- dessous. S'il est mis à "1", les contacts doivent être définis de manière externe.</td> </tr> <tr> <td>1</td> <td>2</td> <td>Manuel/ Auto</td> <td>S'il est mis à "1", le bloc est en mode Manuel; S'il possède un autre nombre, il est en mode Automatique.</td> </tr> <tr> <td>2</td> <td>4</td> <td>Validation</td> <td>Doit en principe être à "1", sinon le bloc n'est jamais appelé.</td> </tr> <tr> <td>3</td> <td>8</td> <td>UP/Aug menter</td> <td>S'il est à "1" et si Manuel (bit 1) est aussi à "1", CV est incrémenté à chaque solution.</td> </tr> <tr> <td>4</td> <td>16</td> <td>DN/Diminuer</td> <td>S'il est à "1" et si Manuel (bit 1) est aussi à "1", CV est incrémenté à chaque solution.</td> </tr> </tbody> </table>	Bit:	Valeur du mot:	Fonction:	Etat ou action externe si le bit de forçage est mis à 1 :	0	1	Forçage	S'il est mis à "0", il surveille les contacts de bloc ci- dessous. S'il est mis à "1", les contacts doivent être définis de manière externe.	1	2	Manuel/ Auto	S'il est mis à "1", le bloc est en mode Manuel ; S'il possède un autre nombre, il est en mode Automatique .	2	4	Validation	Doit en principe être à "1", sinon le bloc n'est jamais appelé.	3	8	UP/Aug menter	S'il est à "1" et si Manuel (bit 1) est aussi à "1", CV est incrémenté à chaque solution.	4	16	DN/Diminuer	S'il est à "1" et si Manuel (bit 1) est aussi à "1", CV est incrémenté à chaque solution.
Bit:	Valeur du mot:	Fonction:	Etat ou action externe si le bit de forçage est mis à 1 :																						
0	1	Forçage	S'il est mis à "0", il surveille les contacts de bloc ci- dessous. S'il est mis à "1", les contacts doivent être définis de manière externe.																						
1	2	Manuel/ Auto	S'il est mis à "1", le bloc est en mode Manuel ; S'il possède un autre nombre, il est en mode Automatique .																						
2	4	Validation	Doit en principe être à "1", sinon le bloc n'est jamais appelé.																						
3	8	UP/Aug menter	S'il est à "1" et si Manuel (bit 1) est aussi à "1", CV est incrémenté à chaque solution.																						
4	16	DN/Diminuer	S'il est à "1" et si Manuel (bit 1) est aussi à "1", CV est incrémenté à chaque solution.																						
SP (15)	(Non configurable ; défini et géré par l'API) Suit l'entrée SP et doit être défini de manière externe si Forçage est mis à "1".																								
CV (16)	(Non configurable ; défini et géré par l'API) Suit la sortie CV.																								
PV (17)	(Non configurable ; défini et géré par l'API) Suit l'entrée PV et doit être défini de manière externe si Forçage est mis à "1".																								
Sortie (18)	(Non configurable ; défini et géré par l'API) Valeur de mot signée représentant la sortie du bloc fonctionnel avant l'application de l'inversion optionnelle. Si aucune inversion de sortie n'est configurée et si le bit de polarité de sortie dans le mot de commande est mis à "0", cette valeur est égale à la sortie CV. Si l'inversion est choisie et si le bit de polarité de sortie est mis à "1", cette valeur est égale à l'inverse de la sortie CV.																								
Stockage du terme différentiel (19)	Utilisé de manière interne pour stocker les valeurs intermédiaires. <i>Ne pas écrire dans cette position.</i>																								
Stockage du terme intégral (20/21)	Utilisé de manière interne pour stocker les valeurs intermédiaires. <i>Ne pas écrire dans cette position.</i>																								
Stockage du terme de parcours de l'échelle totale (22)	Utilisé de manière interne pour stocker les valeurs intermédiaires. <i>Ne pas écrire dans cette position.</i>																								
Horloge (23–25)	Compteur de temps de fonctionnement (temps écoulé depuis la dernière exécution). <i>Ne pas écrire dans cette position.</i>																								
Stockage du reste Y (26)	Conserve le reste de la mise à l'échelle de la division de l'intégrateur en l'absence d'erreur d'état stable.																								
Plage basse et haute (27/28)	Valeurs INT optionnelles des comptages PV qui définissent les valeurs d'affichage supérieure et inférieure du graphique à barre horizontal de SP et PV de la touche Zoom du logiciel Logicmaster et de l'affichage de la plaque PID ADS.																								
Réservés (29–34 et 35–39)	29 à 34 sont réservés à l'usage interne et 35 à 39 à l'usage externe. Ces mots sont destinés à être utilisés par GE Fanuc et ne peuvent pas servir à d'autres fins.																								

Paramètres internes de RefArray

Comme décrit dans le Tableau 4-6 des pages précédentes, le bloc PID lit 13 paramètres utilisateur et utilise les autres éléments du tableau RefArray de 40 mots pour le stockage PID interne. En principe, vous ne devez modifier aucune de ces valeurs. Si vous appelez le bloc PID en mode automatique après un délai prolongé, vous souhaitez peut-être utiliser SVC_REQ #16 pour charger l'horloge de l'API courante dans %Ref+23 afin de mettre à jour l'heure de la dernière solution PID et éviter ainsi une modification d'échelon dans l'intégrateur. Si vous avez mis à "1" le bit de forçage du mot de commande (%Ref+14), vous devez définir les quatre bits suivants du mot de commande pour commander les contacts d'entrée du bloc PID (comme décrit dans le Tableau 4-5 des pages précédentes). Comme le bloc PID n'est plus commandé par le logiciel de programmation, vous devez aussi définir SP et PV. Les mots de paramètre interne sont les suivants :

Sélection de l'algorithme PID (PIDISA ou PIDIND) et gains

Le bloc PID peut être programmé en sélectionnant le terme indépendant (PID_IND) ou les versions ISA standard (PID_ISA) de l'algorithme PID. La manière dont les gains intégral et dérivé sont définis constitue la seule différence entre ces algorithmes. Pour comprendre cette différence, vous devez maîtriser les notions suivantes :

Les deux types de PID calculent le terme d'erreur à l'aide de $SP - PV$, qui peut être remplacé par $PV - SP$ si le terme d'erreur (premier bit 0 du mot de configuration %Ref+12) est mis à "1". L'action dérivée peut être utilisée pour déplacer la sortie CV dans la direction opposée aux changements d'entrée PV (CV descend pendant que PV monte) plutôt que dans le sens normal de la montée de CV pendant la montée de PV.

Erreur = $(SP - PV)$ ou $(PV - SP)$ si le premier bit du mot de configuration est mis à "1"

L'action dérivée est en principe basée sur la modification du terme d'erreur depuis la dernière solution PID, qui peut entraîner une importante modification de la sortie en cas de changement de la valeur de SP. Si vous ne souhaitez pas que cela se produise, vous pouvez mettre à "1" le troisième bit du mot de configuration pour calculer l'action dérivée sur base de la modification de PV. dt (heure delta) est déterminée en soustrayant l'heure de la dernière solution PID pour ce bloc de l'heure de l'API courante.

dt = Heure de l'API courante – Heure API de la dernière solution PID

Action dérivée = $(Erreur - erreur précédente)/dt$ ou $(PV - PV précédent)/dt$ si le 3ème bit du mot de configuration est mis à "1"

L'algorithme PID à termes indépendants (PID_IND) calcule la sortie de la manière suivante :

Sortie PID = $K_p * Erreur + K_i * Erreur * dt + K_d * Action dérivée + Décalage en sortie$

L'algorithme ISA standard (PID_ISA) utilise un format différent :

Sortie PID = $K_c * (Erreur + Erreur * dt/T_i + T_d * Action dérivée) + Décalage en sortie$

où K_c est le gain en commande, T_i l'heure intégrale et T_d l'heure dérivée. L'algorithme ISA permet d'ajuster K_c pour modifier la contribution des termes intégral et dérivé ainsi que du terme proportionnel, en vue de faire tourner la boucle plus rapidement. Si vous possédez des gains PID à termes ou T_i et T_d , utilisez

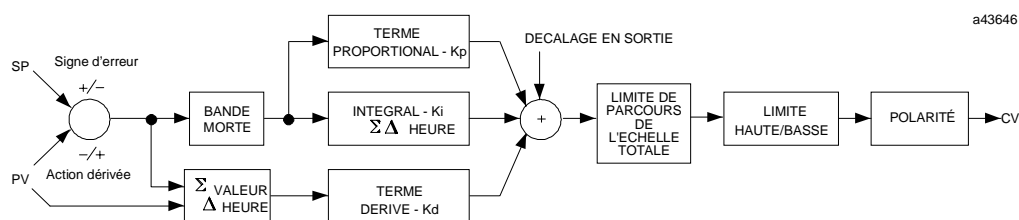
$K_p = K_c$ $K_i = K_c/T_i$ et $K_d = K_c/T_d$

pour les convertir et les utiliser comme entrées des paramètres utilisateur PID.

Le décalage en sortie ci-dessus est un terme supplémentaire, distinct des composants PID, qui peut être requis si vous utilisez uniquement le gain proportionnel K_p et si vous souhaitez que CV possède une valeur différente de zéro lorsque PV est égal à SP et que Erreur est mis à "0". Dans ce cas, attribuez au décalage en sortie la valeur CV souhaitée quand PV est à SP. Le décalage en sortie peut aussi être utilisé pour faire avancer la commande là où une autre boucle PID ou un algorithme de commande est utilisé pour ajuster la sortie CV de cette boucle PID.

Si un gain K_i intégral est utilisé, le décalage en sortie est en principe à "0", car l'intégrateur fait office de décalage en sortie automatique. Il suffit de démarrer en mode manuel et d'utiliser le mot Commande manuelle (%Ref+13) pour attribuer à l'intégrateur le CV souhaité, puis de passer en mode automatique. Cette procédure fonctionne aussi lorsque K_i est à "0", à l'exception du fait que l'intégrateur n'est pas ajusté en fonction de l'erreur après le passage en mode automatique.

Le diagramme ci-dessous illustre le fonctionnement des algorithmes PID :



Algorithme à termes indépendants (PIDIND)

L'algorithme ISA (PIDISA) est similaire, à l'exception du fait que le gain K_p est appliqué à K_i et K_d de sorte que le gain intégral soit égal à $K_p * K_i$ et le gain dérivé, à $K_p * K_d$. Le signe d'erreur, l'action dérivée et la polarité sont définis par des bits du paramètre utilisateur Mot de configuration.

Limites d'amplitude et de coefficient de CV

Le bloc n'envoie pas directement la sortie PID calculée à CV. Les deux algorithmes PID peuvent imposer l'amplitude et le coefficient des limites de changement de la variable de commande de sortie. Le coefficient maximal de changement est déterminé en divisant la valeur CV maximale 100% (32000) par le temps minimum de parcours de l'échelle totale, si elle est supérieure à 0. Par exemple, si le temps minimum de parcours de l'échelle totale est de 100 secondes, la limite de coefficient sera de 320 comptages CV par seconde. Si le délai de solution dt était de 50 milli-secondes, la nouvelle sortie CV ne pourrait pas changer de plus de $320 * 50 / 1000$ ou 16 comptages CV par rapport à la sortie CV précédente.

La sortie CV est ensuite comparée aux limites CV haute et basse. En cas de dépassement de l'une ou l'autre limite, la sortie CV est ajustée à la valeur de la limite. Si la modification de CV entraîne un dépassement de la limite d'amplitude ou de coefficient, la valeur interne de l'intégrateur est ajustée à la valeur limite de façon à éviter l'anti-saturation.

Pour terminer, le bloc vérifie la polarité de sortie (deuxième bit du mot de configuration %Ref+12) et modifie le signe de la sortie si ce bit est mis à "1".

$CV =$ Sortie PID limitée ou $-$ Sortie PID limitée si le bit de la polarité de sortie est mis à "1"

Si le bloc est en mode automatique, la variable de commande finale est placée dans la commande manuelle %Ref+13. S'il est en mode manuel, l'équation PID est sautée puisque CV est définie par la commande manuelle. Les limites d'amplitude et de coefficient sont cependant vérifiées. Cela signifie que la commande manuelle ne peut pas modifier la sortie d'une valeur supérieure à la

limite haute ou inférieure à la limite basse de CV ou à une vitesse supérieure au temps minimum de parcours de l'échelle totale.

Période d'échantillonnage et planification du bloc PID

Comme le bloc PID est une mise en œuvre numérique d'une instruction de commande analogique, la période d'échantillonnage de l'équation de sortie PID n'est pas aussi infiniment petite que celle des commandes analogiques. La moyenne de la plupart des procédés peut être calculée sous forme de gain à l'aide d'un retard de premier ou de deuxième ordre, voire même une temporisation pure. Le bloc PID affecte une sortie CV au procédé et utilise le PV de retour du procédé pour déterminer l'erreur à l'aide de laquelle la sortie CV suivante doit être ajustée. La constante de temps total est un paramètre clé du procédé, car elle mesure la rapidité de réaction de PV quand CV subit une modification. Comme décrit à la section Définition des gains en boucle ci-dessous, la constante de temps total, T_p+T_c , d'un système de premier ordre, correspond à la durée requise par PV pour atteindre 63% de sa valeur finale quand CV est mis à l'échelle. Le bloc PID ne peut contrôler un procédé que si sa période d'échantillonnage est nettement inférieure à la moitié de la constante de temps total. Des périodes d'échantillonnage plus longues peuvent rendre le procédé instable.

La période d'échantillonnage ne doit pas être supérieure à la constante de temps total divisée par 10 (ou, au pire, par 5). Si PV semble atteindre les 2/3 environ de sa valeur finale en 2 secondes, par exemple, la période d'échantillonnage doit être inférieure à 0,2 seconde ou 0,4 seconde dans le pire des cas. Il ne faut pas non plus que la période d'échantillonnage soit extrêmement réduite, notamment 1000 fois plus petite que la constante de temps total, sinon le terme $K_i * \text{Erreur} * dt$ de l'intégrateur PID l'arrondira vers 0. Un procédé extrêmement lent qui nécessite 10 heures ou 36000 secondes pour atteindre le niveau 63%, par exemple, doit avoir une période d'échantillonnage de 40 secondes ou davantage.

A moins que le procédé soit extrêmement rapide, il n'est généralement pas nécessaire d'utiliser une période d'échantillonnage 0 pour résoudre l'algorithme PID à chaque cycle PID. Si de nombreuses boucles PID sont utilisées avec une période d'échantillonnage supérieure à la durée du cycle, le temps de cycle de l'API peut varier considérablement si plusieurs boucles résolvent l'algorithme en même temps. La solution la plus simple consiste à séquencer un ou plusieurs bits par l'intermédiaire d'un tableau de bits mis à "0" pour permettre au flux validant d'atteindre des blocs PID individuels.

Détermination des caractéristiques du procédé

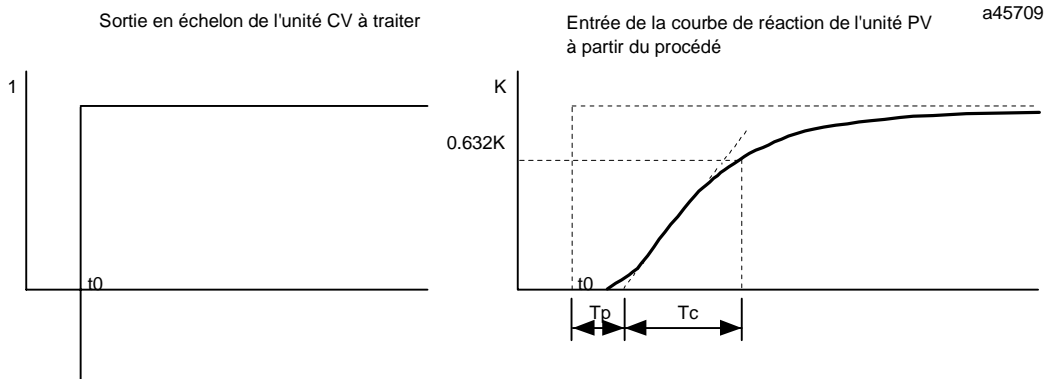
Les gains en boucle PID, K_p , K_i et K_d , sont déterminés par les caractéristiques du procédé qui est commandé. Les deux principales questions à poser au sujet de la définition d'une boucle PID sont les suivantes :

1. Quelle est l'importance de la modification de PV quand une modification fixe est appliquée à CV, ou quel est le gain en boucle ouverte ?
2. Quelle est la rapidité de réaction du système, ou à quelle vitesse PV change-t-il après une mise à l'échelle de la sortie CV ?

La moyenne de la plupart des procédés peut être calculée à l'aide d'un gain de procédé, un retard de premier ou de deuxième ordre et une temporisation pure. Dans le domaine de fréquence, l'instruction de transfert d'un système avec un retard de premier ordre et une temporisation pure est :

$$PV(s)/CV(s) = G(s) = K * e^{**}(-Tp s)/(1 + Tc s)$$

Le traçage d'une réponse en échelon au temps t_0 dans le domaine du temps donne une courbe de réaction d'unité à boucle ouverte :



Les paramètres du modèle de procédé ci-dessous peuvent être déterminés à partir de la courbe de réaction de l'unité PV :

K	Gain en boucle ouverte du procédé = modification finale dans PV/modification dans CV au temps t_0 (Notez l'absence d'indice avec K)
T_p	Temporisation du pipeline ou du procédé ou temps mort après t_0 préalable au début du déplacement de la PV de sortie du procédé
T_c	Constante de temps du procédé de premier ordre, durée requise après T_p pour que PV atteigne 63,2% de sa course finale.

Le moyen le plus rapide de mesurer ces paramètres consiste généralement à placer le bloc PID en mode manuel et à créer un petit échelon dans la sortie CV, en modifiant la commande manuelle %Ref+13 et en traçant la réponse de PV sur le temps. Pour les procédés plus lents, ces actions peuvent être exécutées manuellement, mais pour les procédés plus rapides, il est préférable de consigner les données graphiques sur un ordinateur ou un enregistreur de graphiques. La taille de l'échelon CV doit être suffisamment élevée pour que la modification de PV puisse être observée, mais aussi suffisamment basse pour ne pas altérer le procédé qui est mesuré. Une taille optimale est comprise entre 2 et 10% de la différence entre les limites haute et basse de CV.

Définition des paramètres utilisateur et réglage des gains en boucle

Dans la mesure où tous les paramètres PID dépendent totalement du procédé qui est commandé, aucune valeur prédéterminée ne peut fonctionner. La procédure de recherche d'un gain en boucle acceptable est toutefois généralement simple et itérative.

1. Mettez tous les paramètres utilisateur à "0", puis attribuez aux limites haute et basse les valeurs CV les plus élevée et les plus basse que vous êtes susceptible de rencontrer. Attribuez à la période d'échantillonnage la valeur de la constante de temps estimée du procédé et divisez-la par un nombre compris entre 10 et 100.
2. Placez le bloc en mode manuel et attribuez des valeurs différentes à la commande manuelle (%Ref+13) afin de vérifier si CV peut être déplacé jusqu'aux limites haute et basse. Enregistrez la valeur PV à un point CV déterminé et chargez-la dans SP.
3. Attribuez un petit gain, notamment $100 * CV \text{ maximum} / PV \text{ maximum}$ à Kp et désactivez le mode manuel. Echelonnez SP entre 2 et 10% de la PV maximum et observez la réponse de PV. Augmentez Kp si la réponse en échelon de PV est trop lente ou réduisez-le si PV augmente et oscille sans atteindre une valeur stable.
4. Dès que vous avez trouvé Kp, commencez à augmenter Ki pour obtenir un dépassement qui amortit une valeur stable en 2 ou 3 cycles. Pour cela, il se peut que vous deviez réduire Kp. Testez aussi différentes tailles d'échelon et différents points d'exploitation de CV.
5. Une fois que vous avez trouvé les gains Kp et Ki acceptables, essayez d'ajouter Kd pour accélérer la réponse aux modifications de l'entrée, sans pour autant provoquer des oscillations. Kd est souvent superflu et ne fonctionne pas avec une variable de procédé qui engendre des parasites.
6. Vérifiez les gains par rapport à différents points d'exploitation de SP et ajoutez éventuellement la bande morte ainsi que le temps minimum de parcours de l'échelle totale. Certains procédés d'action inverse peuvent nécessiter une définition du signe d'erreur du mot de configuration ou des bits de polarité.

Définition des gains en boucle - approche de réglage de Ziegler et Nichols

Une fois que vous avez déterminé les trois paramètres de modèle du procédé, à savoir K, Tp et Tc, vous pouvez les utiliser pour estimer les gains en boucle PID de départ. L'approche ci-dessous, développée par Ziegler et Nichols dans les années 40, est spécialement conçue pour assurer une bonne réponse aux interférences du système avec des gains produisant un rapport d'amplitude de 1/4. Le rapport d'amplitude correspond au rapport du deuxième sommet comparé au premier dans la réponse en boucle fermée.

1. Calculez le temps de réaction :

$$R = K / Tc$$

2. Pour la commande proportionnelle uniquement, calculez Kp à l'aide de la formule suivante :

$$Kp = 1 / (R * Tp) = Tc / (K * Tp)$$

3. Pour les commandes proportionnelle et intégrale, utilisez la formule suivante :

$$K_p = 0,9 / (R * T_p) = 0,9 * T_c / (K * T_p)$$

$$K_i = 0,3 * K_p / T_p$$

4. Pour les commandes proportionnelle, intégrale et dérivée, utilisez la formule suivante :

$$K_p = G / (R * T_p) \quad \text{où } G \text{ est compris entre } 1,2 \text{ et } 2,0$$

$$K_i = 0,5 * K_p / T_p$$

$$K_d = 0,5 * K_p * T_p$$

5. Vérifiez si la période d'échantillonnage est comprise entre $(T_p + T_c)/10$ et $(T_p + T_c)/1000$

La procédure de "réglage optimal" est une autre approche spécialement conçue pour assurer une réponse optimale aux modifications de SP, temporisées uniquement par le délai du procédé T_p ou le temps mort.

$$K_p = 2 * T_c / (3 * K * T_p)$$

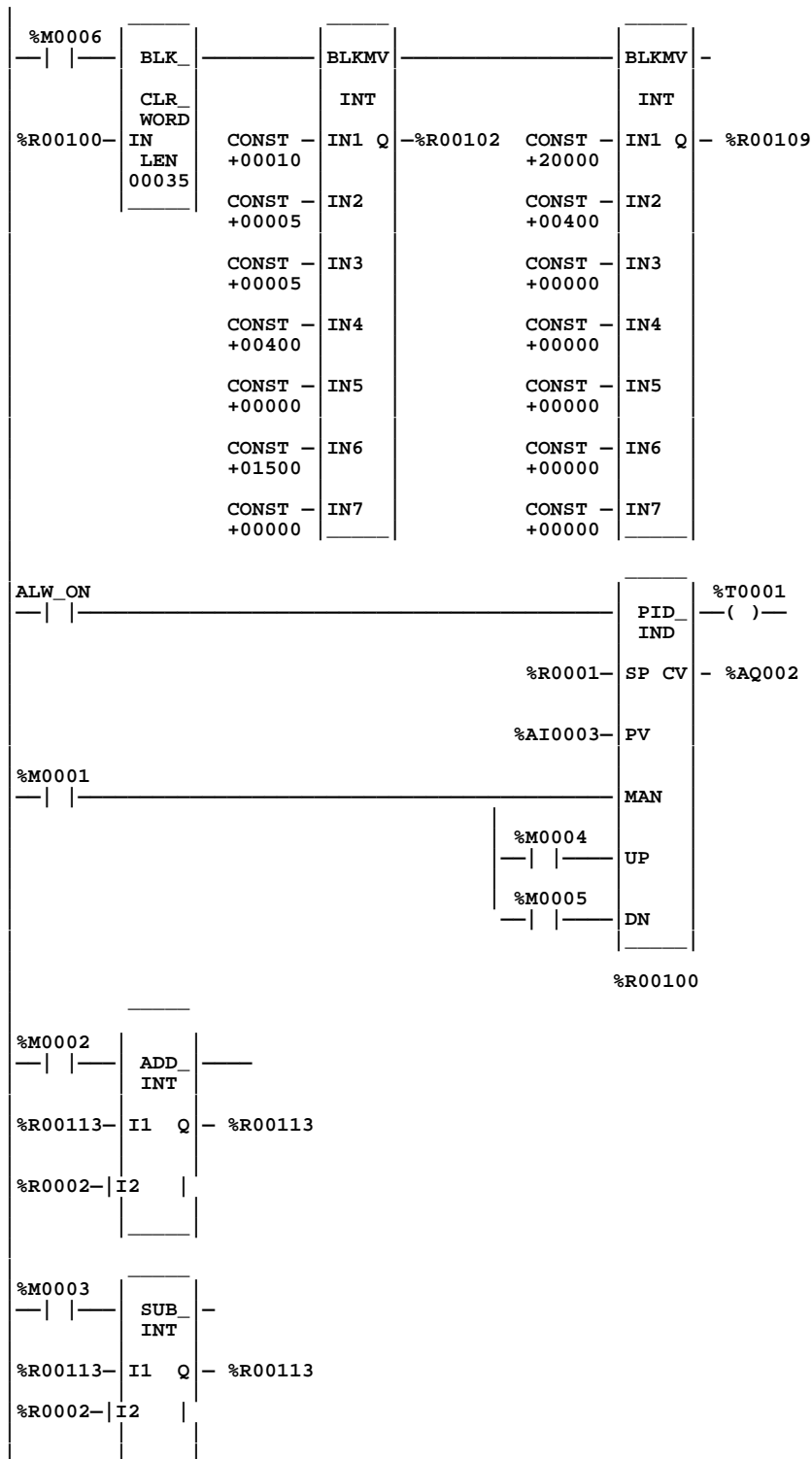
$$K_i = T_c$$

$$K_d = K_i / 4 \quad \text{Si le terme dérivé est utilisé}$$

Une fois que les gains de départ sont déterminés, ils doivent être convertis en paramètres utilisateur au format de nombre entier. Pour éviter les problèmes de conversion, vous devez calculer le gain du procédé, K , sous la forme d'une modification des comptages PV de l'entrée divisée par la modification de l'échelon de sortie dans les comptages CV, et non en unités d'ingénierie CV ou PV du procédé. Vous devez aussi spécifier toutes les durées en secondes. Une fois que K_p , K_i et K_d sont déterminés, K_p et K_d peuvent être multipliés par 100 et entrés sous forme d'entiers tandis que K_i peut être multiplié par 1000 et entré dans le paramètre utilisateur %RefArray.

Exemple d'appel PID

L'exemple ci-dessous utilise une période d'échantillonnage de 100 millisecondes, un gain K_p de 4,00 et un gain K_i de 1,500. Le point de consigne (SP) est stocké dans %R1 et la sortie de la variable de commande dans %AQ2. La variable de procédé est renvoyée dans %AI3. Comme les limites haute et basse de CV doivent être définies, leurs valeurs respectives sont 20000 et 400 dans ce cas-ci. Une petite bande morte optionnelle comprise entre +5 et -5 est également spécifiée. Le tableau RefArray de 40 mots commence à l'emplacement %R100. En principe, les paramètres utilisateur sont définis dans le tableau RefArray à l'aide de la touche de zoom PID **F10**, mais %M6 peut être défini pour remettre à "0" les 14 mots qui commencent à l'emplacement %R102 (%Ref+2) à partir des constantes stockées dans la logique.



Le bloc peut être converti au mode manuel à l'aide de %M1 afin de pouvoir régler la commande manuelle %R113. Les bits de %M4 ou %M5 peuvent être utilisés pour augmenter ou diminuer %R113, CV PID et l'intégrateur d'une unité pour chaque solution de 100 millisecondes. Afin d'accélérer le fonctionnement en mode manuel, vous pouvez utiliser les bits %M2 et %M3 pour ajouter ou soustraire la valeur de %R2 à/de %R113 à chaque cycle de l'API. La sortie %T1 est mise à "1" quand PID est OK.

Temps d'exécution des instructions

L'API Série 90-30, 90-20 et Micro supporte plusieurs instructions et blocs fonctionnels différents. Cette annexe contient des tableaux indiquant, pour chaque instruction, leur taille (en octets) et leur temps d'exécution. La taille est le nombre d'octets occupés par l'instruction dans un programme d'application par diagramme en échelle.

Deux temps d'exécution sont indiqués pour chaque instruction :

Temps d'exécution	Description
Validé	Temps nécessaire pour exécuter l'instruction ou le bloc fonctionnel lorsque le flux validant entre ou sort de l'instruction. Généralement, les temps les plus favorables sont ceux lorsque les données utilisées par le bloc se trouvent dans la RAM utilisateur (mémoire de mots) et non dans la mémoire cache du coprocesseur ISCP (mémoire logique).
Invalidé	Temps nécessaire pour exécuter l'instruction lorsque le flux validant entre dans l'instruction ou le bloc ; c'est toutefois un état inactif, comme lorsque un temporisateur est maintenu dans l'état de réinitialisation.

Remarque

Les temporisateurs et compteurs sont rafraîchis chaque fois qu'ils sont rencontrés dans la logique ; les temporisateurs par la durée du dernier cycle et les compteurs d'impulsions.

Tableau A-1. Temps d'exécution des instructions

Groupe d'instructions	Fonction	Validé				Invalidé				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
Temporisateur	Temporisateur suspensif	146	81	101	42	105	39	46	21	-	-	-	-	15
	Compteur temps de fonc.	98	47	54	23	116	63	73	32	-	-	-	-	9
	Temporisateur	122	76	95	40	103	54	66	30	-	-	-	-	15
Compteurs	Compteur	137	70	87	36	130	63	78	33	-	-	-	-	11
	Décompteur	136	70	86	37	127	61	75	31	-	-	-	-	11
Arithmétique	Addition (INT)	76	47	56	24	41	0	0	0	-	-	-	-	13
	Addition (DINT)	90	60	76	34	41	1	0	0	-	-	-	-	13
	Soustraction (INT)	75	46	57	25	41	0	0	0	-	-	-	-	13
	Soustraction (DINT)	92	62	78	34	41	1	0	0	-	-	-	-	13
	Multiplication (INT)	79	49	62	28	41	0	0	0	-	-	-	-	13
	Multiplication (DINT)	108	80	100	43	41	1	0	0	-	-	-	-	13
	Division (INT)	79	51	61	27	41	0	0	0	-	-	-	-	13
	Division (DINT)	375	346	434	175	41	1	0	0	-	-	-	-	13
	Division modulo (INT)	78	51	61	27	41	0	0	0	-	-	-	-	13
	Division modulo (DINT)	134	103	130	54	41	1	0	0	-	-	-	-	13
	Racine carrée (INT)	153	124	155	65	42	0	0	0	-	-	-	-	9
	Racine carrée (DINT)	268	239	299	120	42	0	1	1	-	-	-	-	9
Compara- raison	Egal à (INT)	66	35	44	19	41	1	1	0	-	-	-	-	9
	Egal à (DINT)	86	56	68	29	41	1	1	0	-	-	-	-	9
	Différent de (INT)	67	39	48	22	41	1	1	0	-	-	-	-	9
	Différent de (DINT)	81	51	65	28	41	1	1	0	-	-	-	-	9
	Supérieur à (INT)	64	33	42	20	41	1	1	0	-	-	-	-	9
	Supérieur à (DINT)	89	59	73	32	41	1	1	0	-	-	-	-	9
	Supérieur/égal à (INT)	64	36	42	19	41	1	1	0	-	-	-	-	9
	Supérieur/égal à (DINT)	87	58	73	30	41	1	1	0	-	-	-	-	9
	Inférieur à (INT)	66	35	44	19	41	1	1	0	-	-	-	-	9
	Inférieur à (DINT)	87	57	70	30	41	1	1	0	-	-	-	-	9
	Inférieur/égal à (INT)	66	36	44	21	41	1	1	0	-	-	-	-	9
	Inférieur/égal à (DINT)	86	57	70	31	41	1	1	0	-	-	-	-	9
	Compris entre (INT)	92	58	66	29	46	1	0	1	-	-	-	-	15
	Compris entre (DINT)	106	75	84	37	45	0	0	0	-	-	-	-	15
	Compris entre (WORD)	93	60	67	29	0	0	0	0	-	-	-	-	15

Remarques :

1. Le temps (en microsecondes) est basé sur la version 5.01 du logiciel Logicmaster 90-30/20 pour les UC modèles 311, 313, 331, 340 et 341.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.

Informations relatives aux temps d'instruction de l'API Micro : Voir le *Series 90™ Micro Programmable Logic Controller User's Manual* (GFK-1065B ou ultérieur).

Informations relatives aux temps d'instruction des API 351 et 352 : Voir pages A-6 et suivantes .

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé				Invalidé				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
Opérations au niveau du bit	Et logique	67	37	48	22	42	0	0	1	-	-	-	-	13
	Ou logique	68	38	48	21	42	0	0	1	-	-	-	-	13
	Ou exclusif logique	66	38	47	20	42	0	0	1	-	-	-	-	13
	Inversion logique, Non	62	32	40	17	42	0	0	1	-	-	-	-	9
	Décalage bit à gauche	139	89	111	47	74	26	30	13	11.61	11.61	15.05	6.29	15
	Décalage bit à droite	135	87	107	45	75	26	30	13	11.63	11.62	15.07	6.33	15
	Rotation bit à gauche	156	127	158	65	42	1	0	0	11.70	11.78	15.18	6.33	15
	Rotation bit à droite	146	116	147	62	42	1	0	0	11.74	11.74	15.23	6.27	15
	Position bit	102	72	126	38	42	1	153	0	-	-	-	-	13
	Mise à "0"	68	38	34	21	42	1	33	1	-	-	-	-	13
	Tester un bit	79	49	132	28	41	0	126	1	-	-	-	-	13
	Mise à "1"	67	37	0	20	42	0	36	0	-	-	-	-	13
	Comparaison masquée (WORD)	217	154	177	74	107	44	50	21	-	-	-	-	25
	Comparaison masquée (DWORD)	232	169	195	83	108	44	49	22	-	-	-	-	25
Transfert de données	Transfert (INT)	68	37	49	20	43	0	1	0	1.62	1.62	6.60	1.31	13
	Transfert (BIT)	94	62	77	35	42	0	0	0	12.61	12.64	15.78	6.33	13
	Transfert (WORD)	67	37	49	20	41	0	1	0	1.62	1.63	6.60	1.31	13
	Transfert de bloc (INT)	76	48	61	28	59	30	34	16	-	-	-	-	27
	Transf. de bloc (WORD)	76	48	62	29	59	29	35	15	-	-	-	-	27
	Mise à "0" d'un bloc	56	28	33	14	43	0	0	0	1.35	1.29	1.78	0.78	9
	Registre à décalage (BIT)	201	153	192	79	85	36	43	18	0.69	0.68	0.88	0.37	15
	Registre à déc. (WORD)	103	53	67	29	73	25	29	12	1.62	1.62	2.54	1.31	15
	Séquenceur binaire	165	101	127	53	96	31	37	16	0.07	0.07	0.10	0.05	15
	COMM_REQ	1317	1272	1577	884	41	2		0	-	-	-	-	13
Tableau	Transfert de tableau													
	INT	230	201	254	104	72	41	49	20	1.29	1.15	7.16	2.06	21
	DINT	231	202	260	105	74	44	53	23	3.24	3.24	13.20	2.61	21
	BIT	290	261	329	135	74	43	51	23	-0.3	-0.3	0.39	0.79	21
	BYTE	228	198	252	104	74	42	51	23	0.81	0.82	5.58	1.25	21
	WORD	230	201	254	104	72	41	49	20	1.29	1.15	7.16	2.06	21
	Cherche égal à													
	INT	197	158	199	82	78	39	46	20	1.93	1.97	3.17	1.55	19
	DINT	206	166	209	87	79	38	47	21	4.33	4.34	5.72	2.44	19
	BYTE	179	141	177	74	78	38	45	21	1.53	1.49	2.29	1.03	19
WORD	197	158	199	82	78	39	46	20	1.93	1.97	3.17	1.55	19	

Remarques :

1. Le temps (en microsecondes) est basé sur la version 5.01 du logiciel Logicmaster 90-30/20 pour les UC modèles 311, 313, 331, 340 et 341.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.
7. Pour les instructions qui possèdent une valeur d'incrément, multipliez l'incrément par (Longueur -1) et ajoutez cette valeur au temps de base.

Informations relatives aux temps d'instruction de l'API Micro : Voir le *Series 90™ Micro Programmable Logic Controller User's Manual* (GFK-1065B ou ultérieur).

Informations relatives aux temps d'instruction des API 351 et 352 : Voir pages A-6 et suivantes .

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé				Invalidé				Incrément				Taille	
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41		
	Cherche différent de														
	INT	198	159	200	83	79	39	46	21	1.93	1.93	3.17	1.52	19	
	DINT	201	163	204	84	79	37	46	21	6.49	6.47	8.63	3.82	19	
	BYTE	179	141	178	73	79	38	47	19	1.54	1.51	2.29	1.05	19	
	WORD	198	159	200	83	79	39	46	21	1.93	1.93	3.17	1.52	19	
	Cherche supérieur à														
	INT	198	160	200	82	79	37	47	19	3.83	3.83	5.62	2.59	19	
	DINT	206	167	210	88	78	38	46	20	8.61	8.61	11.29	4.88	19	
	BYTE	181	143	178	73	79	37	45	19	3.44	3.44	4.69	2.03	19	
	WORD	198	160	200	82	79	37	47	19	3.83	3.83	5.62	2.59	19	
	Cherche supérieur/égal à														
	INT	197	160	200	83	77	38	46	20	3.86	3.83	5.62	2.52	19	
	DINT	205	167	210	87	80	39	46	21	8.62	8.61	11.30	4.87	19	
	BYTE	180	142	178	75	79	37	46	20	3.47	3.44	4.69	2.00	19	
	WORD	197	160	200	83	77	38	46	20	3.86	3.83	5.62	2.52	19	
	Cherche inférieur à														
	INT	199	159	201	84	78	38	46	20	3.83	3.86	5.59	2.48	19	
	DINT	206	168	210	87	79	38	45	19	8.62	8.60	11.29	4.88	19	
	BYTE	181	143	178	75	80	38	46	20	3.44	3.44	4.69	2.00	19	
	WORD	199	159	201	84	78	38	46	20	3.83	3.86	5.55	2.48	19	
Cherche inférieur/égal à															
INT	200	158	200	82	79	38	46	21	3.79	3.90	5.59	2.55	19		
DINT	207	167	209	88	78	39	46	19	8.60	8.61	11.30	4.86	19		
BYTE	180	143	178	74	78	40	46	19	3.46	3.44	4.69	2.02	19		
WORD	200	158	200	82	79	38	46	21	3.79	3.90	5.59	2.55	19		
Con- version	Convertir en entier	74	46	57	25	42	1	0	1	–	–	–	–	9	
	Convertir en DCB-4	77	50	60	25	42	1	0	1	–	–	–	–	9	

Remarques :

1. Le temps (en microsecondes) est basé sur la version 5.1 du logiciel Logicmaster 90-30/20 pour les UC modèles 311, 313, 331, 340 et 341.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.
7. Pour les instructions qui possèdent une valeur d'incrément, multipliez l'incrément par (Longueur – 1) et ajoutez cette valeur au temps de base.

Informations relatives aux temps d'instruction de l'API Micro : Voir le *Series 90™ Micro Programmable Logic Controller User's Manual* (GFK-1065B ou ultérieur).

Informations relatives aux temps d'instruction des API 351 et 352 : Voir pages A-6 et suivantes .

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé				Invalidé				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
Com- mande	Appeler un sous-progr.	155	93	116	85	41	0	0	0	-	-	-	-	7
	Rafraî. imm. des E/S	309	278	355	177	38	1	0	0	-	-	-	-	12
	Algor. PID dépendant (ISA)	1870	1827	2311	929	91	56	71	30	-	-	-	-	15
	Algorithme PID indépend.	2047	2007	2529	1017	91	56	71	30	-	-	-	-	15
	Instruction de fin (END)	-	-	-	-	-	-	-	-	-	-	-	-	-
	Demande de service													
	# 6	93	54	68	45	41	2	0	0	-	-	-	-	9
	# 7 (lire)	-	37	363	161	-	2	0	0	-	-	-	-	9
	# 7 (mettre à "1")	-	37	363	161	-	2	0	0	-	-	-	-	9
	#14	447	418	599	244	41	2	0	0	-	-	-	-	9
	#15	281	243	305	139	41	2	0	0	-	-	-	-	9
	#16	131	104	131	69	41	2	0	0	-	-	-	-	9
	#18	-	56	365	180	-	2	0	0	-	-	-	-	9
	#23	1689	1663	2110	939	43	1	0	0	-	-	-	-	9
	#26//30*	1268	1354	8774	3538	42	0	0	0	-	-	-	-	9
#29	-	-	58	41	-	-	1	0	-	-	-	-	9	
Combinaison de MCR/ENDMCR imbriqué	135	73	88	39	75	25	28	12	-	-	-	-	8	

*La demande de service #26/30 a été mesurée à l'aide d'un compteur rapide et d'une sortie à 16 points dans un bac à 5 emplacements.

Remarques:

1. Le temps (en microsecondes) est basé sur la version 4.5 du logiciel Logicmaster 90-30/20 pour les UC modèles 311, 313, 331, 340 et 341.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.
7. Pour les instructions qui possèdent une valeur d'incrément, multipliez l'incrément par (Longueur -1) et ajoutez cette valeur au temps de base.

Informations relatives aux temps d'instruction de l'API Micro : Voir le *Series 90™ Micro Programmable Logic Controller User's Manual* (GFK-1065B ou ultérieur).

Informations relatives aux temps d'instruction des API 351 et 352 : Voir pages A-6 et suivantes .

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé	Invalidé	Incrément	Taille
		351/352	351/352	351/352	
Temporisateurs	Temporisateur suspensif	4	4	–	15
	Temporisateur	2	3	–	15
	Compteur temps de fonc.	2	2	–	15
Compteurs	Compteur	2	2	–	13
	Décompteur	2	2	–	13
Arithmétique	Addition (INT)	1	0	–	13
	Addition (DINT)	2	0	–	19
	Addition (REAL), 352 seulement	33	0	–	17
	Soustraction (INT)	1	0	–	13
	Soustraction (DINT)	2	0	–	19
	Soustraction (REAL), 352 seulement	34	0	–	17
	Multiplication (INT)	21	0	–	13
	Multiplication (DINT)	24	0	–	19
	Multiplication (REAL), 352 seulement	38	1	–	17
	Division (INT)	22	0	–	13
	Division (DINT),	25	0	–	19
	Division (REAL), 352 seulement	36	2	–	17
	Division modulo (INT)	21	0	–	13
	Div modulo (DINT)	25	0	–	19
	Racine carrée (INT)	41	1	–	10
	Racine carrée (DINT)	76	0	–	13
	Racine carrée (REAL), 352 seulement	35	0	–	11
	Trigonométrie	SIN (REAL), 352 seulement	32	0	–
COS (REAL), 352 seulement		29	0	–	11
TAN (REAL), 352 seulement		32	1	–	11
ASIN (REAL), 352 seulement		45	0	–	11
ACOS (REAL), 352 seulement		63	0	–	11
ATAN (REAL), 352 seulement		33	1	–	11
Logarithme	LOG (REAL), 352 seulement	32	0	–	11
	LN (REAL), 352 seulement	32	0	–	11
Exposant	EXP, 352 seulement	42	0	–	11
	EXPT, 352 seulement	54	1	–	17
Conversion radian	Convertir RAD en DEG, 352 seulement	32	1	–	11
	Convertir DEG en RAD, 352 seulement	32	0	–	11

Remarques:

1. Le temps (en microsecondes) est basé sur la version 7 du logiciel Logicmaster 90-30/20/Micro pour les UC modèles 351 et 352.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé	Invalidé	Incrément	Taille
		351/352	351/352	351/352	
Comparaison	Egal à (INT)	1	0	-	10
	Egal à (DINT)	2	0	-	16
	Egal à (REAL), 352 seulement	33	0	-	14
	Différent de (INT)	1	0	-	10
	Différent de (DINT)	1	0	-	16
	Différent de (REAL), 352 seulement	31	0	-	14
	Supérieur à (INT)	1	0	-	10
	Supérieur à (DINT)	1	0	-	16
	Supérieur à (REAL), 352 seulement	32	0	-	14
	Supérieur/Egal à (INT)	1	0	-	10
	Supérieur/Egal à (DINT)	1	0	-	10
	Supérieur/Egal à (REAL), 352 seulement	36	1	-	14
	Inférieur à (INT)	1	0	-	10
	Inférieur à (DINT)	1	0	-	16
	Inférieur à (REAL), 352 seulement	36	1	-	14
	Inférieur/Egal à (INT)	1	0	-	10
	Inférieur/Egal à (DINT)	3	0	-	16
	Inférieur/Egal à (REAL), 352 seulement	37	0	-	14
	Compris entre (INT)	2	1	-	13
	Compris entre (DINT)	2	1	-	22
Compris entre (WORD)	1	0	-	13	
Opérations au niveau du bit	Et logique	2	0	-	13
	Ou logique	2	0	-	13
	Ou exclusif logique	1	0	-	13
	Inversion logique, Non	1	0	-	10
	Décalage bit à gauche	31	1	1.37	16
	Décalage bit à droite	28	0	3.03	16
	Rotation bit à gauche	25	0	3.12	16
	Rotation bit à droite	25	0	4.14	16
	Position bit	20	1	-	13
	Mise à "0"	20	0	-	13
	Tester un bit	20	0	-	13
	Mise à "1"	19	1	-	13
	Comparaison masquée (WORD)	46	0	-	25
	Comparaison masquée (DWORD)	48	0	-	25

Remarques:

1. Le temps (en microsecondes) est basé sur la version 7 du logiciel Logicmaster 90-30/20/Micro pour les UC modèles 351 et 352.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.
7. Pour les instructions qui possèdent une valeur d'incrément, multipliez l'incrément par (Longueur - 1) et ajoutez cette valeur au temps de base.

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé	Invalidé	Incrément	Taille
		351/352	351/352	351/352	
Transfert de données	Transfert (INT)	0	0	0.41	10
	Transfert (BIT)	28	0	4.98	13
	Transfert (WORD)	1	1	0.41	10
	Transfert (REAL), 352 <i>seulement</i>	24	1	0.82	13
	Transfert de bloc (INT)	3	0	–	28
	Transfert de bloc (WORD)	3	0	–	28
	Transfert de bloc (REAL)	36	0	–	13
	Mise à "0" d'un bloc	1	0	0.24	11
	Registre à décalage (BIT)	46	0	0.23	16
	Registre à décalage (WORD)	27	0	0.41	16
	Séquenceur binaire	38	22	0.02	16
Tableau	Transfert de tableau				
	INT	54	0	0.97	22
	DINT	54	0	0.81	22
	BIT	69	0	0.36	22
	BYTE	54	1	0.64	22
	WORD	54	0	0.97	22
	Cherche égal à				
	INT	37	0	0.62	19
	DINT	41	1	1.38	22
	BYTE	35	0	0.46	19
	WORD	37	0	0.62	19
	Cherche différent de				
	INT	37	0	0.62	19
	DINT	38	0	2.14	22
	BYTE	37	0	0.47	19
	WORD	37	0	0.62	19
	Cherche supérieur à				
	INT	37	0	1.52	19
	DINT	39	0	2.26	22
	BYTE	36	1	1.24	19
	WORD	37	0	1.52	19
	Cherche supérieur/égal à				
	INT	37	0	1.48	19
	DINT	39	0	2.33	22
	BYTE	37	1	1.34	19
	WORD	37	0	1.48	19

Remarques:

1. Le temps (en microsecondes) est basé sur la version 7 du logiciel Logicmaster 90-30/20/Micro pour les UC modèles 351 et 352.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.
7. Pour les instructions qui possèdent une valeur d'incrément, multipliez l'incrément par (Longueur – 1) et ajoutez cette valeur au temps de base.

Tableau A-1. Temps d'exécution des instructions - suite

Groupe d'instructions	Fonction	Validé	Invalidé	Incrément	Taille
		351/352	351/352	351/352	
	Cherche inférieur à				
	INT	37	0	1.52	19
	DINT	41	1	2.27	22
	BYTE	37	0	1.41	19
	WORD	37	0	1.52	19
	Cherche inférieur/égal à				
	INT	38	0	1.48	19
	DINT	40	1	2.30	22
	BYTE	37	0	1.24	19
	WORD	38	0	1.48	19
Conversion	Convertir en entier	19	1	–	10
	Convertir en DCB-4	21	1	–	10
	Convertir au format REAL, 352 <i>seulement</i>	21	0	–	8
	Convertir au format WORD, 352 <i>seulement</i>	30	1	–	11
	Tronquer au format entier, 352 <i>seulement</i>	32	0	–	11
	Tronquer au format entier à double position, 352 <i>seulement</i>	31	0	–	11
Commande	Appeler un sous-programme	40	1	–	7
	Rafraî. imm. des E/S	123	1	–	13
	Algorithme PID – ISA *	162	34	–	16
	Algorithme PID – IND *	146	34	–	16
	Instruction de fin	–	–	–	–
	Demande de service				
	#6	22	1	–	10
	#7 (lire)	75	1	–	10
	#7 (mettre à "1")	75	1	–	10
	#14	121	1	–	10
	#15	46	1	–	10
	#16	36	1	–	10
	#18	261	1	–	10
	#23	426	0	–	10
	#26//30**	2910	1	–	10
	#29	20	0	–	10
	Combinaison MCR/ENDMCR imbriqué	1	1	–	4
COMM_REQ	732	0	–	13	

*Les temps PID ci-dessus sont basés sur la version 6.5 de l'UC 351.

**La demande de service #26/30 a été mesurée à l'aide d'un compteur rapide et d'une sortie à 16 points dans un bac à 5 emplacements.

Remarques:

1. Le temps (en microsecondes) est basé sur la version 7 du logiciel Logicmaster 90-30/20/Micro pour les UC modèles 351 et 352.
2. Pour les instructions de tableau, l'incrément est spécifié en unités de longueur. Pour les opérations binaires, il est spécifié en microsecondes/bit et pour les instructions de transfert de données, en microsecondes/nombre de bits ou de mots.
3. Temps validé pour les unités de longueur simples de type %R, %AI et %AQ.
4. La durée COMMREQ a été mesurée entre l'UC et HSC.
5. DOIO correspond à la durée requise pour sortir des valeurs sur le module de sortie discrète.
6. Lorsqu'il existe plusieurs cas possibles, la durée indiquée ci-dessus s'applique au pire des cas.
7. Pour les instructions qui possèdent une valeur d'incrément, multipliez l'incrément par (Longueur –1) et ajoutez cette valeur autempsde base.

Taille des instructions pour les UC 351 et 352

La taille de la mémoire correspond au nombre d'octets requis par l'instruction d'un programme d'application à diagramme en échelle. Les UC modèles 351 et 352 nécessitent trois (3) octets pour la plupart des instructions booléennes standard, voir le tableau A-2.

Tableau A-2. Taille des instructions pour les UC 351 et 352

Instruction	Taille
Pas d'opération	1
Monter la pile et ET vers le haut	1
Monter la pile et OU vers le haut	1
Dupliquer le haut de la pile	1
Monter la pile	1
Pile initiale	1
Etiquette	5
Saut	5
Toutes les autres instructions	3
Blocs fonctionnels - voir Tableau A-1	–

Les API Série 90-30, 90-20 et 90 Micro mettent à jour deux tables de défauts : la Table des défauts générés par les périphériques d'E/S (y compris les contrôleurs d'E/S) et la Table des défauts internes de l'API. Les informations contenues dans cette annexe permettront d'interpréter le format de structure des messages en lisant ces tables. Ces deux tables contiennent des informations similaires.

- La Table des défauts automate contient :
 - la localisation du défaut.
 - la description du défaut.
 - la date et l'heure du défaut.
- La Table des défauts d'E/S contient :
 - la localisation du défaut.
 - l'adresse de référence.
 - la catégorie de défaut.
 - le type de défaut.
 - la date et l'heure du défaut.

Table des défauts automate

Vous accédez à la table des défauts automate par l'intermédiaire du logiciel de programmation. Si vous utilisez le logiciel Logicmaster, voir le chapitre 5 du *Logicmaster™ 90 Series 90™-30/20/ Micro Programming Software User's Manual* (GFK-0466) pour obtenir des informations sur la manière d'accéder aux tables des défauts.

L'entrée "Incohérence dans la configuration système" est décrite ci-dessous. (Toutes les données sont en hexadécimal.)

Champ	Valeur	Description
Long/court	00	Ce défaut contient 8 octets d'informations additionnelles.
Bac	00	Bac principal (bac 0).
Emplacement	03	Emplacement 3.
Tâche	44	
Groupe de défauts	0B	Défaut "Incohérence dans la configuration système".
Intervention défaut	03	Défaut de type FATAL.
Code d'erreur	01	

Les sections suivantes décrivent chaque champ de l'entrée de défaut. Des tableaux sont inclus, indiquant l'intervalle de valeurs que peut avoir chaque champ.

Indicateur Long/Court

Cet octet indique si le défaut contient 8 octets ou 24 octets d'informations additionnelles sur le défaut.

Type	Code	Données add. sur défaut
Court	00	8 octets
Long	01	24 octets

Remplissage

Ces six octets sont des octets de remplissage utilisés pour que l'entrée de la Table des défauts automate ait exactement la même longueur que l'entrée de la Table des défauts d'E/S.

Bac

Le numéro de bac est compris entre 0 et 7. Zéro est le bac principal contenant l'API. Les bacs 1 à 7 sont les bacs d'extension, connectés à l'API par l'intermédiaire d'un câble d'extension.

Emplacement

Le numéro d'emplacement est compris entre 0 et 9. L'UC de l'API occupe toujours l'emplacement 1 dans le bac principal (bac 0).

Tâche

Le numéro de tâche est compris entre 0 et +65 535. Parfois, le numéro de tâche fournit des informations additionnelles aux techniciens d'API ; généralement, la tâche peut être ignorée.

Groupe de défauts automate

Le groupe de défaut est la classification la plus haute d'un défaut ; il identifie sa catégorie générale. Le texte descriptif du défaut, affiché par le logiciel Logicmaster 90-30/20/Micro, est fonction du groupe de défauts et des codes d'erreur.

Le Tableau B-1 liste les groupes de défauts possibles dans la table des défauts automate.

Le dernier groupe de défauts non masquables, les *Codes de défauts automate additionnels*, est signalé dans le cas du traitement de nouvelles conditions de défaut dans le système dont l'API ne connaît pas expressément les codes d'alarme. Tous les codes d'alarme non reconnus, de type automate, appartiennent à ce groupe.

Tableau B-1. Groupe de défauts automate

Numéro de groupe		Nom du groupe	Intervention défaut
Décimal	Hexadécimal		
1	1	Bac perdu ou manquant.	Fatal
4	4	Coprocasseur perdu ou manquant.	Diagnostic
5	5	Bac additionnel ou non configuré.	Diagnostic
8	8	Coprocasseur additionnel ou non configuré	Diagnostic
11	B	Incohérence dans la configuration système.	Fatal
12	C	Erreur du bus système.	Diagnostic
13	D	Défaut matériel de l'UC de l'API.	Fatal
14	E	Défaut matériel de module, non fatal.	Diagnostic
16	10	Défaut logiciel de coprocasseur.	Diagnostic
17	11	Défaut de CHECKSUM de bloc de programme.	Fatal
18	12	Signal "seuil pile bas".	Diagnostic
19	13	Dépassement du temps de cycle constant.	Diagnostic
20	14	Table des défauts automate pleine.	Diagnostic
21	15	Table des défauts d'E/S pleine.	Diagnostic
22	16	Défaut de l'application utilisateur.	Diagnostic
-	-	Codes de défaut automate additionnels.	Tels que spécifiés
128	80	Défaut du bus système.	Fatal
129	81	Pas de programme utilisateur à la mise sous tension.	Informationnel
130	82	Détection de RAM utilisateur altérée.	Fatal
132	84	Echec de l'accès par mot de passe.	Informationnel
135	87	Défaut logiciel de l'UC de l'API.	Fatal
137	89	Défaut de stockage de séquence d'API.	Fatal

Action sur défaut

Chaque défaut est associé à une ou plusieurs actions. Ces actions sont fixes pour l'API Série 90-30 et ne peuvent pas être modifiées par l'utilisateur.

Tableau B-2. Interventions défaut automate

Action sur défaut	Intervention de l'UC	Code
Informationnel	Enregistre le défaut dans la table des défauts.	1
Diagnostic	Enregistre le défaut dans la table des défauts. Met à "1" les références de défaut.	2
Fatal	Enregistre le défaut dans la table des défauts. Met à "1" les références des défauts. Passe en mode STOP.	3

Code d'erreur

Le code d'erreur apporte un supplément d'informations à la description du défaut. Chaque groupe de défauts a sa propre série de codes. Le Tableau B-3 présente les codes d'erreur du Groupe d'erreurs des défauts logiciels d'API (Groupe 87H).

Tableau B-3. Codes d'erreur des défauts logiciels de l'UC de l'API

Décimal	Hexadécimal	Nom
20	14	Mémoire programme d'API altérée.
39	27	Mémoire programme d'API altérée.
82	52	Echec de la communication fond de bac.
90	5A	Mise hors service utilisateur demandée.
Tous les autres		Erreur système interne de l'UC de l'API.

Le Tableau B-4 présente les codes d'erreur pour tous les autres groupes de défauts.

Tableau B-4. Codes d'erreur des défauts de l'UC de l'API

Décimal	Hexadécimal	Nom
<i>Codes d'erreur d'API pour groupe Coprocesseur perdu</i>		
44	2C	Echec de la réinitialisation utilisateur du coprocesseur.
45	2D	Echec de la réinitialisation utilisateur du coprocesseur.
255	FF	Echec de la communication du coprocesseur.
<i>Codes d'erreur du groupe Coprocesseur réinitialisé, additionnel, ou non configuré</i>		
2	2	Redémarrage module terminé.
	Tous les autres	Coprocesseur réinitialisé, additionnel, ou non configuré.
<i>Codes d'erreur du groupe Défaut logiciel de coprocesseur</i>		
1	1	Type de carte non supporté.
2	2	COMREQ – boîte à lettres pleine lors du message sortant, démarrant la demande de communication.
3	3	COMREQ – boîte à lettres pleine lors de la réponse.
5	5	Communication de fond de bac avec API ; demande perdue.
11	B	Erreur de ressource (affectation, dépassement de table, etc.).
13	D	Erreur du programme utilisateur.
401	191	Logiciel de module altéré ; demande rechargement.
<i>Codes d'erreur du groupe Incohérence dans la configuration système</i>		
8	8	Incohérence des extensions analogiques.
10	A	Caractéristique non supportée.
23	17	La taille du programme excède la capacité de la mémoire.
<i>Codes d'erreur du groupe Erreur du bus système</i>		
	Tous les autres	Erreur du bus système.
<i>Codes d'erreur du groupe de contrôle de CHECKSUM de bloc de programme.</i>		
3	3	Défaut de CHECKSUM du programme ou du bloc de programme.
<i>Codes d'erreur pour signal Seuil pile bas</i>		
0	0	Pile déchargée sur l'UC de l'API ou autre module.
1	1	Seuil pile bas sur l'UC de l'API ou autre module.
<i>Codes d'erreur du groupe Défauts de l'application utilisateur</i>		
2	2	Dépassement du temporisateur chien de garde d'API.
5	5	COMREQ – le mode ATTENTE (WAIT) n'est pas disponible pour cette instruction.
6	6	COMREQ – code de tâche incorrect.
7	7	Dépassement de la pile d'application.
<i>Codes d'erreur du groupe Défaut de bus système</i>		
1	1	Système d'exploitation.
<i>Codes d'erreur du groupe RAM utilisateur altérée à la mise sous tension</i>		
1	1	RAM utilisateur altérée à la mise sous tension.
2	2	Détection d'un code opération booléen interdit.
3	3	DEBORDEMENT_API_ISCP_PC.
4	4	ERREUR_SYNTAXE_PRG.
<i>Codes d'erreur pour Défauts matériels de l'UC de l'API</i>		
	Tous les codes	Défaut matériel de l'UC de l'API

Informations supplémentaires sur le défaut

Ce champ contient des informations additionnelles sur l'entrée de défaut. Exemple de données pouvant être présentées :

Quatre des codes d'erreur dans le groupe Incohérence dans la configuration système fournissent des informations supplémentaires :

Groupe RAM
utilisateur altéré : **interdit**

Tableau B-5. Informations défaut automate - Détection d'un code opération booléen

Infos add. sur défaut	Incohérence des numéros de modèle
[0]	Contenu du registre de défauts d'ISCP.
[1]	Code opération incorrect.
[2,3]	Compteur d'instructions de l'ISCP.
[4,5]	Numéro d'instruction.

Pour un défaut de RAM dans l'UC de l'API (un des défauts signalés comme défauts matériels de l'UC de l'API), l'adresse du défaut est stockée dans les quatre premiers octets de ce champ.

Défaut matériel de l'UC de l'API (défaut RAM) :

Horodatage de défaut automate

L'horodatage à six octets est la valeur de l'horloge système lorsque le défaut a été enregistré par l'UC de l'API. (Les valeurs sont codées en format DCB.)

Tableau B-6. Horodatage de défaut automate

Numéro d'octet	Description
1	Secondes.
2	Minutes.
3	Heures.
4	Quantième du mois.
5	Mois.
6	Année.

Les parties suivantes décrivent chaque champ de la Table des défauts d'E/S, et incluent des tableaux indiquant l'intervalle des valeurs que chaque champ peut avoir.

Indicateur long/court

Cet octet indique si le défaut contient 5 octets ou 21 octets d'informations spécifiques sur le défaut.

Tableau B-7. Octet indicateur de format de la Table des défauts d'E/S

Type	Code	Infos spécifiques sur défaut
Short	02	5 octets
Long	03	21 octets

Adresse de référence

L'adresse de référence est une adresse à 3 octets contenant le type de mémoire d'E/S et la position (ou décalage) dans cette mémoire, qui correspond au point en défaut. Ou bien, lorsqu'un défaut de bloc Genius ou de module analogique intégré se produit, l'adresse de référence désigne le premier point du bloc où le défaut s'est produit.

Tableau B-8. Adresse de référence des E/S

Octet	Description	Intervalle
0	Type de mémoire	0 – FF
1–2	Décalage	0 – 12K (décimal)

L'octet de type de mémoire est l'une des valeurs suivantes :

Tableau B-9. Type de mémoire des adresses de références

Nom	Valeur (Hexadécimal)
Entrée analogique	0A
Sortie analogique	0C
Groupée logique	0D
Entrée logique	10 ou 46
Sortie logique	12 ou 48
Groupée logique	1F

Adresse de défaut d'E/S

L'adresse de défaut d'E/S est une adresse à 6 octets contenant l'adresse du bac, de l'emplacement, du bloc et du point d'E/S à l'origine du défaut. L'adresse du point est un mot ; toutes les autres adresses ont un octet chacune. Parfois, les cinq adresses ne sont pas toutes présentes dans le défaut.

Lorsqu'une adresse de défaut d'E/S ne contient pas les cinq adresses ensemble, un code 7F hex apparaît dans l'adresse pour indiquer où le poids stoppe. Par exemple, si 7F apparaît dans l'octet de bus, il s'agit d'un défaut de module. Seules les valeurs du bac et de l'emplacement sont significatives.

Bac

Le numéro de bac est compris entre 0 et 7. Zéro correspond au bac principal, c'est-à-dire celui contenant l'API. Les bac 1 à 7 sont des bacs d'extension.

Emplacement

Le numéro d'emplacement est compris entre 0 et 9. L'UC de l'API occupe toujours l'emplacement 1 dans le bac principal (bac 0).

Point

Le point est compris entre 1 et 1024 (décimal). Il indique le point en défaut dans le bloc lorsque le défaut est de type point.

Groupe de défauts d'E/S

Le groupe de défaut est la classification la plus haute d'un défaut ; il identifie sa catégorie générale. Le texte descriptif du défaut, affiché par le logiciel Logicmaster 90-30/20/Micro, est fonction du groupe de défauts et des codes d'erreur.

Le Tableau B-10 liste les groupes de défauts possibles dans la Table des défauts d'E/S. Les numéros de groupe inférieurs à 80 (Hex) correspondent à des défauts masquables.

Le dernier groupe de défauts non masquables, *Codes des défauts d'E/S*, est signalé dans le cas du traitement de nouvelles conditions de défaut dans le système dont l'API ne connaît pas expressément les codes d'alarme. Tous les codes d'alarme non reconnus, de type E/S, appartiennent à ce groupe.

Tableau B-10. Groupes de défauts d'E/S

Numéro de groupe	Nom du groupe	Intervention défaut
3	Module d'E/S perdu ou manquant.	Diagnostic
7	Module d'E/S additionnel ou non configuré.	Diagnostic
9	Défaut de bus d'E/S ou de contrôleur d'E/S.	Diagnostic
A	Défaut de module d'E/S.	Diagnostic
–	Codes additionnels de défauts d'E/S.	Tels que spécifiés

Intervention défaut d'E/S

L'intervention défaut spécifie quelle doit être l'intervention de l'UC de l'API lorsqu'un défaut se produit. Le Tableau B-11 liste les interventions défaut possibles.

Tableau B-11. Intervention défaut d'E/S

Intervention défaut	Intervention de l'UC	Code
Informationnel	Enregistre le défaut dans la table des défauts.	1
Diagnostic	Enregistre le défaut dans la table des défauts. Met à "1" les références de défaut.	2
Fatal	Enregistre le défaut dans la table des défauts. Met à "1" les références de défaut. Passe en mode STOP.	3

Informations spécifiques de défaut d'E/S

Une entrée de la Table des défauts d'E/S peut contenir jusqu'à 5 octets d'informations spécifiques de défaut d'E/S.

Informations spécifiques de défaut symbolique

Le Tableau B-12 liste les données nécessaires pour la configuration du circuit de bloc.

Tableau B-12. Informations spécifiques de défaut d'E/S

Numéro décimal	Code hex.	Description
<i>Configuration du circuit</i>		
	1	Le circuit est une entrée - trois états
	2	Le circuit est une entrée.
	3	Le circuit est une sortie.

Interventions pour défauts spécifiques

Les défauts des circuits forcés/non forcés sont signalés comme défauts de type informationnel. Tous les autres sont des défauts de type diagnostic et fatal.

L'incohérence des numéros de modèle, l'incohérence du type d'E/S et les défauts de module d'E/S non existant sont signalés dans la Table des défauts automate sous le groupe Incohérence dans la configuration système. Ils ne sont pas supportés dans la Table des défauts d'E/S.

Horodatage de défaut d'E/S

L'horodatage de 6 octets est la valeur de l'horloge système lorsque le défaut a été enregistré par l'UC de l'API. Les valeurs sont codées en format DCB.

Tableau B-13. Horodatage de défaut d'E/S

Numéro d'octet	Description
1	Secondes.
2	Minutes.
3	Heures.
4	Quantième du mois.
5	Mois.
6	Année.

Annexe

C

Mnémoniques des instructions

Le mode Afficher/Modifier programme permet d'entrer et de rechercher rapidement une instruction de programmation en tapant le signe "&" suivi du mnémonique de l'instruction. Pour certaines instructions, on peut également spécifier une adresse de référence ou un symbole, une étiquette ou une adresse de référence de position.

Cette annexe liste les mnémoniques des instructions de programmation du logiciel de programmation Logimaster 90-30/20/Micro. Le mnémonique complet est indiqué dans la colonne 3 de ce tableau, et l'entrée la plus courte que vous pouvez effectuer pour chaque instruction est listée dans la colonne 4.

Vous pouvez afficher, à tout moment de la programmation, un écran d'aide présentant ces mnémoniques en appuyant sur les touches ALT et I.

Groupe d'instruc.	Instruction	Mnémonique						
		Toutes	INT	DINT	BIT	BYTE	WORD	REAL
Contacts	Tout contact	&CON	&CON					
	Contact N.O.	&NOCON	&NOCON					
	Contact N.F.	&NCCON	&NCCON					
	Contact "suite"	&CONC	&CONC					
Bobines	Toute bobine	&COI	&COI					
	Bobine N.O.	&NOCOI	&NOCOI					
	Bobine inversée	&NCCOI	&NCCOI					
	Bobine de transition positive	&PCOI	&PCOI					
	Bobine de transition négative	&NCOI	&NCOI					
	Bobine de mise à "1"	&SL	&SL					
	Bobine de remise à "0"	&RL	&RL					
	Bobine rémanente de mise à "1"	&SM	&SM					
	Bobine rémanente de remise à "0"	&RM	&RM					
	Bobine rémanente	&NOM	&NOM					
	Bobine rémanente inversée	&NCM	&NCM					
	Bobine "suite"	&COILC	&COILC					
Liens	Lien horizontal	&HO	&HO					
	Lien vertical	&VE	&VE					
Temporiseurs	Temporisateur suspensif	&ON	&ON					
	Compteur de temps de fonctionnement	&TM &OF	&TM &OF					
	Compteurs	&UP	&UP					
Compteurs	Compteur	&UP	&UP					
	Décompteur	&DN	&DN					

Groupe d'instruc.	Instruction	Mnémonique							
		Toutes	BCD-4	INT	DINT	BIT	BYTE	WORD	REAL
Arith-métique	Addition	&AD		&AD_I	&AD_DI				&AD_R
	Soustraction	&SUB		&SUB_I	&SUB_DI				&SUB_R
	Multiplication	&MUL		&MUL_I	&MUL_DI				&MUL_R
	Division	&DIV		&DIV_I	&DIV_DI				&DIV_R
	Modulo	&MOD		&MOD_I	&MOD_DI				&MOD_R&&SQ_R
	Racine carrée	&SQ		&SQ_I	&SQ_DI				
	Sinus	&SIN							
	Cosinus	&COS							
	Tangente	&TAN							
	Sinus inverse	&ASIN							
	Cosinus inverse	&ACOS							
	Tangente inverse	&ATAN							
	Logarithme de base 10	&LOG							
	Logarithme naturel	&LN							
	Puissance de e	&EXP							
Puissance de x	&EXPT								
Com-paraison	Egal à	&EQ		&EQ_I	&EQ_DI				&EQ_R
	Différent de	&NE		&NE_I	&NE_DI				&NE_R
	Supérieur à	>		>_I	>_DI				>_R
	Supérieur ou égal à	&GE		&GE_I	&GE_DI				&GE_R
	Inférieur à	<		<_I	<_DI				<_R
Opération logique	Et logique	&AN						&AN_W	
	Ou logique	&OR						&OR_W	
	Ou exclusif logique	&XO						&XO_W	
	Non	&NOT						&NOT_W	
	Décalage binaire à gauche	&SHL						&SHL_W	
	Décalage binaire à droite	&SHR						&SHR_W	
	Rotation binaire à gauche	&ROL						&ROL_W	
	Rotation binaire à droite	&ROR						&ROR_W	
	Tester un bit	&BT						&BT_W	
	Mise à "1"	&BS						&BS_W	
	Mise à "0"	&BCL						&BCL_W	
	Position binaire	&BP						&BP_W	
	Comparaison masquée	&MCM						&MCM_W	
Conversion	Convertir en entier	&TO_INT	&TO_INT_BCD4	&MOV					
	Convertir en entier double	&TO_DINT		&BLKM					
	Convertir en DCB-4	&BCD4		&BLKC	&TO_REAL_DI				&BCD4_R
	Convertir au format REAL	&TO_REAL		&SHF				&TO_REAL_W	
	Convertir au format WORD	&TO_W		&BI					
	Tronquer au format entier	&TRINT		&COMMR					
Tronquer au format entier double	&TRDINT								

Cette annexe liste les commandes-clavier actives dans l'environnement logiciel. Ces informations peuvent être également affichées sur la console de programmation en appuyant sur ALT-K pour accéder à l'écran d'aide.

Séquence de touches	Description	Séquence de touches	Description
<i>Touches disponibles dans l'environnement logiciel</i>			
ALT-A	Annuler.	CTRL-Break	Quitter le logiciel.
ALT-C	Effacer le champ.	Esc	Quitter zoom.
ALT-M	Changer le mode de la console de prog.	CTRL-Home	Contenu de la ligne de commande préc.
ALT-R	Changer l'état de l'API (RUN/STOP).	CTRL-End	Contenu de la ligne de commande suiv.
ALT-E	Valider la zone d'état.	CTRL- ←	Curseur à gauche dans le champ.
ALT-J	Valider la ligne de commande.	CTRL- →	Curseur à droite dans le champ.
ALT-L	Lister les fichiers répertoires.	CTRL-D	Décrémenter l'adresse de référence.
ALT-P	Imprimer l'écran.	CTRL-U	Incrémenter l'adresse de référence.
ALT-H	Aide.	Tab	Modifier/incrémenter le contenu du champ.
ALT-K	Aide aux touches.	Shift-Tab	Modifier/décrémenter le contenu du champ.
ALT-I	Aide aux mnémoniques d'instructions.	Entrée	Accepter le contenu du champ.
ALT-N	Valider les options d'affichage.	CTRL-E	Afficher la dernière erreur système.
ALT-T	Démarrer le mode Apprentissage.	F12 ou clavier -	Valider la référence logique.
ALT-Q	Stopper le mode Apprentissage.	F11 ou clavier*	Invalider la référence logique.
ALT-n	Rejouer le fichier n (n = 0 à 9).		
<i>Touches disponibles dans l'éditeur de programme seulement</i>			
ALT-B	Valider la sonnerie de l'éditeur de texte.	Clavier +	Accepter le segment.
ALT-D	Effacer l'élément de ligne/Effacer la ligne.	Entrée	Accepter le segment.
ALT-S	Stocker le bloc dans l'API et le disque.	CTRL-PgUp	Segment précédent.
ALT-X	Afficher le niveau de zoom.	CTRL-PgDn	Segment suivant.
ALT-U	Mettre à jour le disque.	~	Lien horizontal.
ALT-V	Fenêtre de la table de variables		Lien vertical.
ALT-F2	Aller à la table de références des opérandes.	Tab	Aller au champ d'opérandes suivant.
<i>Touches spéciales</i>			
ALT-O	Invalider le mot de passe. Disponible uniquement dans l'écran Mot de passe du logiciel de configuration.		

Le tableau d'aide de la page suivante présente la liste d'aide pour les touches et également le texte d'aide pour les mnémoniques d'instructions utilisées dans le logiciel Logicmaster 90-30/20/Micro. Ce tableau est imprimé en trois exemplaires et est perforé afin de faciliter son retrait de ce manuel.

Lorsque vous utilisez des nombres à virgule flottante, vous devez vous familiariser avec certains éléments. La première section de cette annexe traite de ces considérations générales. Pour obtenir des instructions sur la manière de saisir et d'afficher des nombres à virgule flottante, voir les pages A-5 et suivantes.

Nombres à virgule flottante

Le logiciel Logimaster 90 permet de modifier, d'afficher, de stocker et d'extraire des nombres avec des valeurs réelles. Certaines instructions s'appliquent à des nombres à virgule flottante. Vous ne pouvez toutefois utiliser des nombres à virgule flottante avec le logiciel Logimaster 90-30/20/Micro que si vous possédez une UC 352. Les nombres à virgule flottante sont représentés dans la notation scientifique décimale à l'aide de six chiffres significatifs.

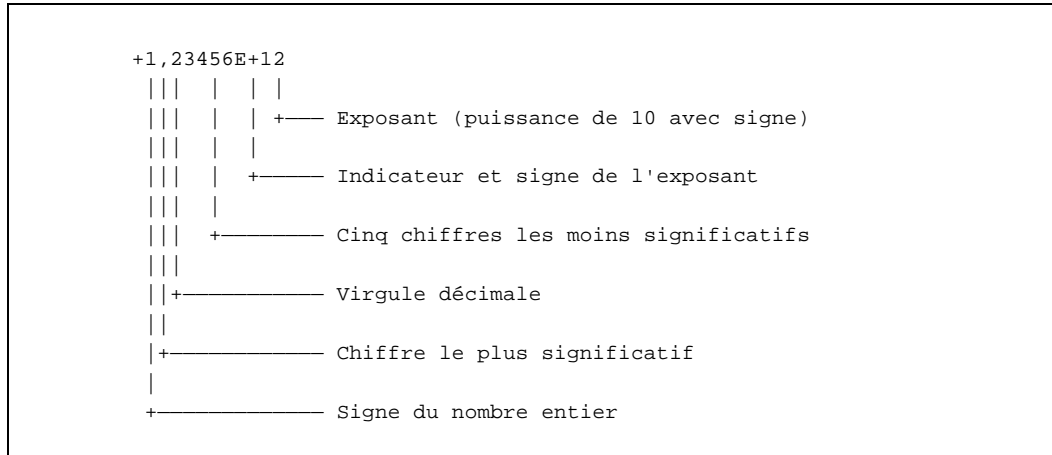
Remarque

Ce manuel utilise indifféremment les termes "virgule flottante" et "REAL" pour décrire la fonction d'entrée/affichage des nombres à virgule flottante du logiciel Logimaster.

Le logiciel Logimaster 90 utilise le format ci-dessous. Pour les nombres compris entre 999999999 et 0,0001, l'affichage est dépourvu d'exposant mais utilise entre six et sept chiffres significatifs. Par exemple :

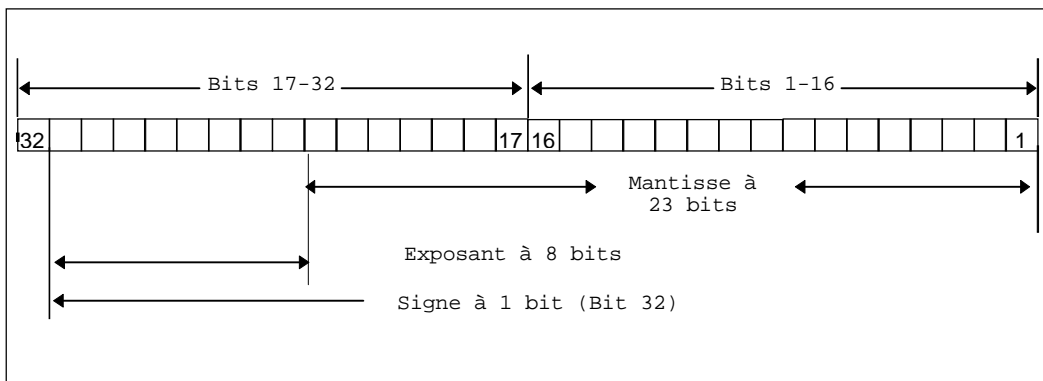
Saisie	Affichage	Description
0,000123456789	+0,0001234567	Dix chiffres, six ou sept significatifs.
-12,345e-2	-0,1234500	Sept chiffres, six ou sept significatifs.
1234	+1234,000	Sept chiffres, six ou sept significatifs.

En dehors de la plage ci-dessus, six chiffres significatifs seulement sont affichés sous le format ci-dessous :

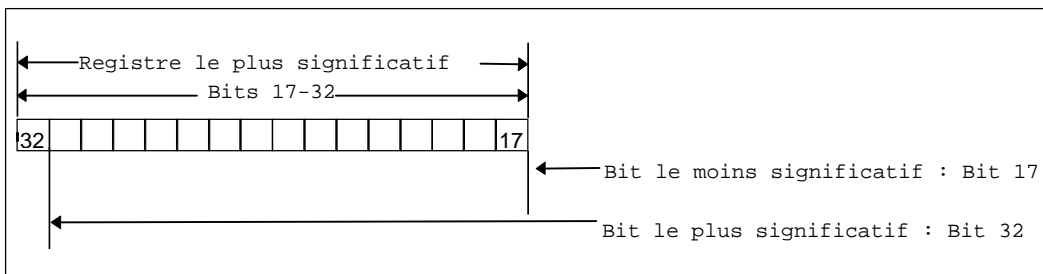
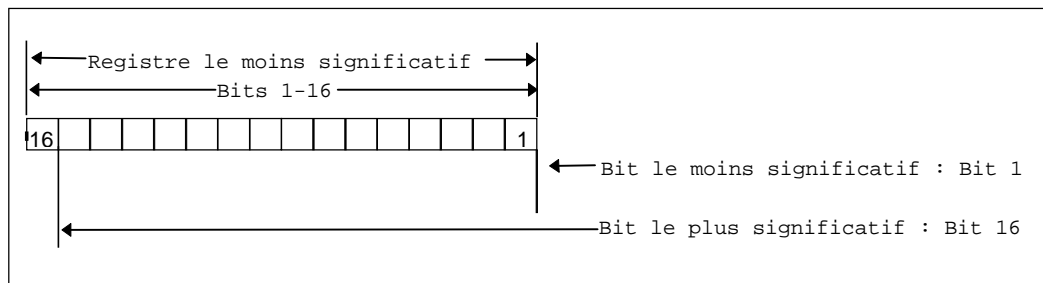


Format interne des nombres à virgule flottante

Les nombres à virgule flottante sont stockés au format IEEE standard à précision simple. Ce format nécessite 32 bits pour convertir les deux registres (adjacents) de 16 bits de l'API . Le diagramme ci-dessous illustre l'encodage des bits.



Le registre utilisé par un nombre à virgule flottante simple est illustré ci-dessous. Dans ce diagramme, si le nombre à virgule flottante utilise les registres R5 et R6, par exemple, R5 correspond au registre le moins significatif et R6, au registre le plus significatif.



Valeurs des nombres à virgule flottante

Utilisez le tableau ci-dessous pour calculer la valeur d'un nombre à virgule flottante à partir du nombre binaire stocké dans les deux registres.

Exposant (e)	Mantisse (f)	Valeur du nombre à virgule flottante
255	Différente de zéro	NaN, Not a valid number (n'est pas un nombre valide).
255	0	$-1^s * \infty$
$0 < e < 255$	Valeur quelconque	$-1^s * 2^{e-127} * 1, f$
0	Différente de zéro	$-1^s * 2^{-126} * 0, f$
0	0	0

- f = la mantisse. La mantisse est une fraction binaire.
- e = l'exposant. L'exposant est un entier E de sorte que $E+127$ est la puissance de 2 par laquelle la mantisse doit être multipliée pour produire la valeur à virgule flottante.
- s = le bit du signe.
- * = l'opérateur de multiplication.

Prenons l'exemple du nombre à virgule flottante 12,5. La représentation binaire à virgule flottante IEEE de ce nombre est :

01000001 01001000 00000000 00000000

ou 41480000 hex au format hexadécimal. Le bit le plus significatif (bit du signe) est zéro (s=0). Les huit bits suivants les plus significatifs sont 10000010 ou la décimale 130 (e=130).

La mantisse est stockée sous la forme d'un nombre binaire décimal dans lequel la virgule décimale précède le bit le plus significatif parmi les 23 bits. Le bit le plus significatif de la mantisse est donc un multiple de 2^{-1} , le bit suivant le plus significatif est un multiple de 2^{-2} , etc. jusqu'au bit le moins significatif, qui est un multiple de 2^{-23} . Les 23 bits finals (la mantisse) sont :

1001000 00000000 00000000

La valeur de la mantisse est donc 0,5625 (c'est-à-dire $2^{-1} + 2^{-4}$).

Comme $e > 0$ et que $e < 255$, nous utilisons la troisième formule du tableau ci-dessus :

$$\begin{aligned}
 \text{nombre} &= -1^s * 2^{e-127} * 1, f \\
 &= -1^0 * 2^{130-127} * 1, 5625 \\
 &= 1 * 2^3 * 1, 5625 \\
 &= 8 * 1, 5625 \\
 &= 12, 5
 \end{aligned}$$

Vous constatez donc que la représentation binaire ci-dessus est exacte.

La plage des nombres susceptibles d'être stockés dans ce format est comprise entre $\pm 1,401298E-45$ et $\pm 3,402823E+38$ ainsi que le nombre zéro.

Saisie et affichage des nombres à virgule flottante

Dans la mantisse, six ou sept chiffres significatifs de précision maximum peuvent être saisis et stockés. Le logiciel Logicmaster 90 n'affiche néanmoins que les six premiers chiffres. La mantisse peut être précédée d'un signe positif ou négatif. Si aucun signe n'est saisi, le nombre à virgule flottante est supposé être positif.

Si un exposant est saisi, il doit être précédé de la lettre **E** ou **e**, et la mantisse doit contenir une virgule décimale pour éviter de la confondre avec un nombre hexadécimal. L'exposant peut être précédé d'un signe, mais s'il n'en possède aucun, il est supposé être positif. Si aucun exposant n'est saisi, il est supposé être égal à zéro. Les espaces ne sont pas autorisés dans les nombres à virgule flottante.

Afin de vous faciliter la tâche, la ligne de commande et les champs d'entrée de données acceptent plusieurs formats, y compris un entier, un nombre décimal ou un nombre décimal suivi d'un exposant. Ces nombres sont convertis dans un format d'affichage standard dès que l'utilisateur a saisi les données et appuyé sur la touche **Entrée**.

Voici des exemples de saisie de nombre à virgule flottante valide et l'affichage normalisé correspondant :

Saisie	Affichage
250	+250 0000
+4	+4,000000
-2383019	-2383019,
34,	+34,00000
-0,0036209	-0,003620900
12,E+9	+1,20000E+10
-0,0004E-11	-4,00000E-15
731,0388	+731,0388
99,20003e-29	+9,92000E-28

Voici à présent des exemples de saisie de nombre à virgule flottante non valide :

Saisie non valide	Explication
-433E23	Absence de la virgule décimale.
10e-19	Absence de la virgule décimale.
10,e19	La mantisse ne peut pas contenir d'espace entre des chiffres ou des caractères. Cette saisie est acceptée sous la forme 10,e0, mais un message d'erreur est affiché.
4,1e19	L'exposant ne peut pas contenir d'espace entre des chiffres ou des caractères. Cette saisie est acceptée sous la forme 4,1e0, mais un message d'erreur est affiché.

Erreurs dans les opérations et les nombres à virgule flottante

Un dépassement survient lorsqu'un nombre supérieur à 3,402823E+38 ou inférieur à -3,402823E+38 est généré par une instruction REAL. Lorsque cela se produit, la sortie ok de l'instruction est mise à "0" et le résultat correspond à l'infini positif (pour un nombre supérieur à 3,402823E+38) ou à l'infini négatif (pour un nombre inférieur à -3,402823E+38). Vous pouvez déterminer l'endroit où cette erreur est survenue en testant sens de la sortie ok.

POS_INF	= 7F800000h	– Représentation de l'infini positif IEEE au format hexadécimal.
NEG_INF	= FF800000h	– Représentation de l'infini négatif IEEE au format hexadécimal.

Si les infinis produits par le dépassement sont utilisés comme opérandes pour d'autres instructions REAL, le résultat risque d'être indéfini. Ce résultat indéfini est appelé NaN (Not a Number, n'est pas un nombre). Le résultat de l'ajout de l'infini positif à l'infini négatif, par exemple, est indéfini. Quand l'instruction ADD_REAL est appelée en utilisant l'infini positif et l'infini négatif comme opérandes, elle produit NaN comme résultat.

Chaque instruction REAL capable de donner un résultat NaN produit un NaN particulier qui identifie l'instruction.

NaN_ADD.	= 7F81FFFFh	– Valeur d'erreur de l'addition au format Real en hex.
NaN_SUB	= 7F81FFFFh	– Valeur d'erreur de la soustraction au format Real en hex.
NaN_MUL	= 7F82FFFFh	– Valeur d'erreur de la multiplication au format Real en hex.
NaN_DIV	= 7F83FFFFh	– Valeur d'erreur de la division au format Real en hex.
NaN_SQRT	= 7F84FFFFh	– Valeur d'erreur de la racine carrée au format Real en hex.
NaN_LOG	= 7F85FFFFh	– Valeur d'erreur de l'algorithme au format Real en hex.
NaN_POW0	= 7F86FFFFh	– Valeur d'erreur de l'exposant au format Real en hex.
NaN_SIN	= 7F87FFFFh	– Valeur d'erreur du sinus au format Real en hex.
NaN_COS	= 7F88FFFFh	– Valeur d'erreur du cosinus au format Real en hex.
NaN_TAN	= 7F89FFFFh	– Valeur d'erreur de la tangente au format Real en hex.
NaN_ASIN	= 7F8AFFFFh	– Valeur d'erreur du sinus inverse au format Real en hex.
NaN_ACOS	= 7F8BFFFFh	– Valeur d'erreur du cosinus inverse au format Real en hex.
NaN_BCD	= 7F8CFFFFh	– Erreur de conversion DCB-4 au format Real.
REAL_INDEF	= FFC00000h	– Erreur de format Real indéfinie, division de 0 par 0.

Quand un résultat NaN est transféré à une autre instruction, il atteint aussi le résultat. Si un NaN_ADD est la première opérande de l'instruction SUB_REAL, par exemple, le résultat de SUB_REAL est NaN_ADD. Si les deux opérandes d'une instruction sont des NaN, la première opérande est transférée. Cette fonction de transfert des NaN aux instructions permet d'identifier l'instruction qui a généré le premier NaN.

Remarque

Pour NaN, la sortie ok est mise à "0" (elle n'est pas validée).

A

Accès
 privilèges 2-38
 demandes de changement 2-39
ACOS 4-36
Acquisition des entrées 2-8
Action sur défaut
 action sur défaut automate B-5
ADD 4-28
ADD_IOM 2-27
ADD_SIO 2-27
Addition 4-28
Alarme 3-2
Altération de la mémoire 3-7
ALW_OFF 2-26
ALW_ON 2-26
AND 4-50
ANY_FLT 2-28
API
 cycle 2-2
 calcul de CHECKSUM du programme
 utilisateur 2-9
 calcul du temps de cycle 2-7
 communication du PCM avec l'API 2-12
 contribution au temps de cycle 2-4
 exécution du programme d'application 2-8
 fenêtre de communication système 2-10
 gestion des entrées 2-8
 gestion interne 2-8
 mode temps de cycle constant 2-13
 restitution des sorties 2-8
 temporisateur de cycle constant 2-37
 variantes du cycle de programme 2-13
 cycle de programme normal 2-2
 fenêtre
 communication avec la console de
 programmation 2-9
 fonctionnement du système 2-1
API Série 90-20
 modules d'E/S Modèle 20 2-46
API Série 90-30
 E/S
 conditions par défaut des modules de
 sortie Modèle 30 2-44
 données de diagnostic 2-45
 données globales 2-45
 formats des données d'E/S 2-44
 modules d'E/S Modèle 30 2-42
APL_FLT 2-27
ARRAY_MOVE 4-90

ASIN 4-36
ATAN 4-36

B

BAD_PWD 2-28
BAD_RAM 2-27
BCD-4 2-25; 4-98
BCLR 4-63
BIT 2-25
BITSEQ 4-82
 occupation mémoire 4-83
BLKCLR 4-77
BLKMOV 4-75
Bloc de programme
 appel des sous-programmes 2-20
 bloc de sous-programme 2-19
Blocs de programme
 défaut de CHECKSUM 3-10
Blocs de sous-programme 2-19
Bobine
 avec vérification unique et multiple de la
 bobine 4-7
Bobine "suite" 4-9
Bobine de mise à "1" 4-6
Bobine de remise à "0" 4-6
Bobine de transition négative 4-6
Bobine de transition positive 4-5
Bobine inversée 4-5
Bobine rémanente 4-5
Bobine rémanente de mise à "1" 4-7
Bobine rémanente de remise à "0" 4-8
Bobine rémanente inversée 4-5
Bobines 4-3; 4-4
 bobine "suite" 4-9
 bobine de mise à "1" 4-6
 bobine de remise à "0" 4-6
 bobine de transition négative 4-6
 bobine de transition positive 4-5
 bobine inversée 4-5
 bobine rémanente 4-5
 bobine rémanente de mise à "1" 4-7
 bobine rémanente de remise à "0" 4-8
 bobine rémanente inversée 4-5
BPOS 4-65
BSET 4-63
BTST 4-61

C

Calcul de CHECKSUM 2-9
Calcul du temps de cycle 2-7
CALL 4-111
Catégorie de défaut 3-17

- CFG_MM 2-27
 - CHECKSUM
 - défaut de CHECKSUM des blocs de programme 3-10
 - Codes d'erreur B-5
 - Codes d'erreur d'alarme B-5
 - COMMENT 4-126
 - COMMREQ 4-86
 - Communication avec la console de programmation 2-9
 - Communication du PCM avec l'API 2-12
 - Compteur de temps de fonctionnement 2-36
 - Compteurs 2-36; 4-10
 - DNCTR 4-23
 - données de bloc fonctionnel 4-10
 - UPCTR 4-21
 - Conditions par défaut des modules de sortie
 - Modèle 30 2-44
 - Configuration 4-1
 - Configuration système
 - incohérence dans la 3-9
 - Console de programmation
 - fenêtre de communication 2-9
 - Contact "suite" 4-9
 - Contact normalement fermé 4-4
 - Contact normalement ouvert 4-4
 - Contacts 4-2
 - contact "suite" 4-9
 - contact normalement fermé 4-4
 - contact normalement ouvert 4-4
 - Contacts d'impulsions d'horloge 2-37
 - Contributions au temps de cycle des UC 351/352 2-5
 - Contributions au temps de scrutation des UC 351/352 2-6
 - COS 4-36
 - Cycle API 2-2
 - acquisition des entrées 2-8
 - calcul de CHECKSUM du programme utilisateur 2-9
 - calcul du temps de cycle 2-7
 - communication avec la console de programmation 2-9
 - communication du PCM avec l'API 2-12
 - contribution au temps de cycle 2-4
 - contributions au temps de scrutation des UC 351/352 2-5
 - contributions au temps de scrutation pour les UC 351/352 2-6
 - cycle de programme normal 2-2
 - exécution du programme d'application 2-8
 - fenêtre de communication système 2-10
 - gestion interne 2-8
 - mode temps de cycle constant 2-13
 - programme d'application 2-8
 - restitution des sorties 2-8
 - temporisateur de cycle constant 2-37
 - variantes du cycle de programme 2-13
 - Cycle API
 - mode temps de cycle constant configuré 2-13
 - Cycle de programme normal 2-2
 - cycle du processeur 2-2
 - Cycle, API
 - contributions au temps de scrutation des UC351/352 2-5
 - contributions au temps de scrutation pour les UC 351/352 2-6
- ## D
- Décompteur 4-23
 - Défauts
 - défauts externes des E/S 3-2
 - Défaut
 - défaut d'application 3-12
 - Défaut d'application 3-12
 - Défaut de communication pendant le stockage 3-16
 - Défaut de diagnostic 3-4
 - Défaut du système d'exploitation de l'UC 3-14
 - Défaut fatal 3-4
 - défaut de CHECKSUM des blocs de programme 3-10
 - défaut de communication pendant le stockage 3-16
 - défaut du système d'exploitation de l'UC 3-14
 - défaut logiciel du module optionnel 3-10
 - incohérence dans la configuration système 3-9
 - Défaut informationnel 3-4
 - Défaut logiciel du module optionnel 3-10
 - Défauts 3-2
 - accéder aux informations additionnelles concernant les défauts 3-6
 - action sur défaut automate B-5
 - catégories de défauts 3-2
 - codes d'erreur B-5
 - défaut de CHECKSUM des blocs de programme 3-10
 - défaut de communication pendant le stockage 3-16
 - défaut du système d'exploitation de l'UC 3-14
 - défaut logiciel du module optionnel 3-10
 - défauts de fonctionnement 3-2
 - dépassement du temps de cycle constant 3-11
 - description du défaut dans la table des défauts automate 3-7
 - description et correction 3-1
 - descriptions de la table des défauts d'E/S 3-17
 - échec de l'accès par mot de passe 3-13

- effets secondaires 3-5
 - groupe de défauts automate B-4
 - groupe de défauts d'E/S B-10
 - incohérence dans la configuration système 3-9
 - internes 3-2
 - interprétation B-1
 - interprétation d'un défaut B-1
 - intervention 3-8
 - intervention défaut 3-4
 - intervention défaut d'E/S B-11
 - module d'E/S additionnel 3-18
 - module d'E/S perdu 3-17
 - module optionnel perdu ou manquant 3-8
 - module optionnel réinitialisé, additionnel ou non configuré 3-8
 - pas de programme utilisateur présent 3-12
 - programme utilisateur altéré à la mise sous tension 3-13
 - références 3-4
 - réponse du système aux défauts 3-3
 - signal seuil pile bas 3-11
 - table des défauts automate 3-3; 3-5; 3-6
 - table des défauts d'E/S 3-3
 - Défauts de diagnostic**
 - défaut d'application 3-12
 - dépassement du temps de cycle constant 3-11
 - module d'E/S additionnel 3-18
 - module d'E/S perdu 3-17
 - module optionnel perdu ou manquant 3-8
 - module optionnel réinitialisé, additionnel ou non configuré 3-8
 - signal seuil pile bas 3-11
 - Défauts de fonctionnement** 3-2
 - Défauts de type informationnel**
 - échec de l'accès par mot de passe 3-13
 - Défauts externes des E/S** 3-2
 - DEG** 4-40
 - Demande de service**
 - interroger les E/S 4-144
 - lire checksum maître 4-143
 - lire le compteur de mise hors tension 4-145
 - Demandes de changements de niveau d'autorisation** 2-39
 - Dépannage** 3-1
 - accéder aux informations additionnelles concernant les défauts 3-6
 - défauts non configurables 3-8
 - description du défaut dans la table des défauts automate 3-7
 - descriptions de la table des défauts d'E/S 3-17
 - interprétation d'un défaut B-1
 - table des défauts automate 3-5; 3-6
 - Dépassement du temps de cycle constant** 3-11
 - Description du défaut** 3-17
 - Description et correction des défauts**
 - catégorie de défaut 3-17
 - Description et correction des défauts 3-1
 - accéder aux informations additionnelles concernant les défauts 3-6
 - défaut d'application 3-12
 - défaut de CHECKSUM des blocs de programme 3-10
 - défaut de communication pendant le stockage 3-16
 - défaut du système d'exploitation de l'UC 3-14
 - défaut logiciel du module optionnel 3-10
 - défauts non configurables 3-8
 - dépassement du temps de cycle constant 3-11
 - description du défaut 3-17
 - description du défaut dans la table des défauts automate 3-7
 - descriptions de la tables des défauts d'E/S 3-17
 - échec de l'accès par mot de passe 3-13
 - gestion des défauts 3-2
 - groupe de défauts automate B-4
 - groupe de défauts d'E/S B-10
 - incohérence dans la configuration système 3-9
 - interprétation d'un défaut B-1
 - module d'E/S additionnel 3-18
 - module d'E/S perdu 3-17
 - module optionnel perdu ou manquant 3-8
 - module optionnel réinitialisé, additionnel ou non configuré 3-8
 - pas de programme utilisateur présent 3-12
 - programme utilisateur altéré à la mise sous tension 3-13
 - signal seuil pile bas 3-11
 - table des défauts automate 3-5; 3-6
 - type de défaut 3-17
 - Deviation intégrale proportionnelle (PID)** 4-146
 - DINT** 2-25; 4-102
 - DIV** 4-28
 - Division** 4-28
 - DNCTR** 4-23
 - DOIO** 4-112
 - instruction DO I/O évoluée pour les UC modèles 331 et 341 4-116
 - Données**
 - rémanence 2-24
 - Données de diagnostic** 2-45
 - Données d'utilisateur et d'organisation du programme**
 - nombres à virgule flottante E-1
 - Données globales** 2-45
- ## E
- E/S de l'API Série 90- Micro UC Micro et E/S** 2-46
 - E/S de l'API Série 90-20** 2-41

E/S de l'API Série 90-30 2-41
 structure d'E/S 2-41
E/S, API Série 90-20 2-41
E/S, API Série 90-30 2-41
E/SI, API Série 90- Micro
 E/S Micro 2-46
Echec de l'accès par mot de passe 3-13
EDITLOCK 2-39
Effets secondaires de défauts 3-5
END 4-118
ENDMCR 4-122
Entier signé 2-25
Entier signé en double précision 2-25
Entrées
 acquisition 2-8
EQ 4-42
Etat système
 références 2-23; 2-26
Exécution du programme d'application 2-8
EXP 4-38
EXPT 4-38

F

Faute fatale
 programme utilisateur altéré à la mise sous
 tension 3-13
Fautes de type informationnel
 pas de programme utilisateur présent 3-12
Fenêtre de communication
 communication système 2-10
Flux validant 2-31
Fonctionnement du système 2-1
 compteurs et temporisateurs 2-36
 description d'un cycle API 2-2
 E/S de l'API Série 90-20 2-41
 E/S de l'API Série 90-30 2-41
 organisation du programme et
 données/références utilisateur 2-18
 sécurité du système 2-38
 séquences de mise sous/hors tension 2-32
Forçages 2-23
Formats des données d'E/S 2-44
FST_SCN 2-26

G

GE 4-42
Gestion des défauts 3-2
 processeur d'alarmes 3-2
Gestion interne 2-8
Groupe de défauts B-4; B-10
GT 4-42

H

Horloge interne 2-36
Horloges
 compteur de temps de fonctionnement 2-36
 horloge interne 2-36
HRD_CPU 2-27
HRD_FLT 2-28
HRD_SIO 2-27

I

Incohérence dans la configuration système 3-9
Instruction BCD-4 4-98
Instruction CALL 4-111
Instruction Cherche équivalent 4-94
Instruction Cherche inférieur à 4-94
Instruction Cherche inférieur ou égal à 4-94
Instruction Cherche non équivalent 4-94
Instruction Cherche supérieur à 4-94
Instruction Cherche supérieur ou égal à 4-94
Instruction COMMENT 4-126
Instruction Convertir au format REAL 4-104
Instruction Convertir au format WORD 4-106
Instruction Convertir en entier signé 4-100
Instruction Convertir en entier signé à double
 précision 4-102
Instruction de comparaison masquée 4-67
Instruction de cosinus inversé 4-36
Instruction de conversion 4-97
Instruction de conversion radian 4-40
Instruction de cosinus 4-36
Instruction de Décalage à droite 4-56
Instruction de Décalage à gauche 4-56
Instruction de logarithme de base 10 4-38
Instruction de logarithme naturel 4-38
Instruction de mise à "0" 4-63
Instruction de Mise à "0" d'un bloc 4-77
Instruction de mise à "1" 4-63
Instruction de Rotation à droite 4-59
Instruction de Rotation à gauche 4-59
Instruction de sinus 4-36
Instruction de sinus inversé 4-36
Instruction de tangente 4-36
Instruction de tangente inversée 4-36
Instruction de transfert 4-72
Instruction de Transfert de bloc 4-75
Instruction Demande de communication 4-86
Instruction Demande de service 4-127
Instruction Différent de 4-42
Instruction DO I/O 4-112

- instruction DO I/O évoluée pour les UC
modèles 331 et 341 4-116
- Instruction DO I/O évoluée pour les UC
modèles 331 et 341 4-116
- Instruction Egal à 4-42
- Instruction END 4-118
- Instruction Fin de relais de contrôle maître 4-122
- Instruction Inférieur à 4-42
- Instruction Inférieur ou égal à 4-42
- Instruction JUMP 4-123
- Instruction LABEL 4-125
- Instruction Modulo 4-32
- Instruction Position binaire 4-65
- Instruction RANGE 4-45
- Instruction Registre à décalage 4-79
- Instruction Relais de contrôle maître 4-119
- Instruction Séquenceur binaire 4-82
- Instruction Supérieur à 4-42
- Instruction Supérieur ou égal à 4-42
- Instruction Tester un bit 4-61
- Instruction Transfert de tableau 4-90
- Instruction Tronquer 4-108
- Instructions arithmétiques 4-27
 - ADD 4-28
 - DIV 4-28
 - MOD 4-32
 - MUL 4-28
 - SQRT 4-34
 - SUB 4-28
- Instructions de commande 4-110
 - CALL 4-111
 - COMMENT 4-126
 - DOIO 4-112
 - instruction DO I/O évoluée pour les UC
modèles 331 et 341 4-116
 - END 4-118
 - ENDMCR 4-122
 - JUMP 4-123
 - LABEL 4-125
 - MCR 4-119
 - PID 4-146
 - SVCREQ 4-127
 - temps d'exécution des instructions A-1
- Instructions de comparaison 4-42
 - EQ 4-42
 - GE 4-42
 - GT 4-42
 - LE 4-42
 - LT 4-42
 - NE 4-42
 - RANGE 4-45
- Instructions de conversion
 - BCD-4 4-98
 - DINT 4-102
 - INT 4-100
 - REAL 4-104
 - TRUN 4-108
 - WORD 4-106
- Instructions de logarithme 4-38
 - logarithme de base 10 4-38
 - logarithme naturel 4-38
- Instructions de programmation 4-1
 - instructions arithmétiques 4-27
 - instructions de commande 4-110
 - instructions de comparaison 4-42
 - instructions de conversion 4-97
 - instructions de transfert de données 4-71
 - instructions sur tableau 4-89
 - mnémoniques des instructions C-1
 - opérations logiques 4-48
 - par diagramme en échelle 4-2
 - temporisateurs et compteurs 4-10
- Instructions de tableau
 - instruction Recherche inférieur ou égal à 4-94
 - SRCH_EQ 4-94
 - SRCH_GE 4-94
 - SRCH_GT 4-94
 - SRCH_LT 4-94
 - SRCH_NE 4-94
- Instructions de transfert de données 4-71
 - BITSEQ 4-82
 - BLKCLR 4-77
 - BLKMOV 4-75
 - COMMREQ 4-86
 - MOVE 4-72
 - SHFR 4-79
- Instructions d'exposant 4-38
 - puissance de e 4-38
 - puissance de X 4-38
- Instructions mathématiques
 - ACOS 4-36
 - ASIN 4-36
 - ATAN 4-36
 - COS 4-36
 - DEG 4-40
 - EXP 4-38
 - EXPT 4-38
 - LN 4-38
 - LOG 4-38
 - RAD 4-40
 - SIN 4-36
 - TAN 4-36
- Instructions par diagramme en échelle 4-2
 - bobine "suite" 4-9
 - bobine de mise à "1" 4-6
 - bobine de remise à "0" 4-6
 - bobine de transition négative 4-6
 - bobine de transition positive 4-5
 - bobine inversée 4-5

- bobine rémanente 4-5
- bobine rémanente de mise à "1" 4-7
- bobine rémanente de remise à "0" 4-8
- bobine rémanente inversée 4-5
- bobines 4-3; 4-4
- contact "suite" 4-9
- contact normalement fermé 4-4
- contact normalement ouvert 4-4
- contacts 4-2
- liens horizontaux et verticaux 4-8
- Instructions sur tableau 4-89
 - ARRAY_MOVE 4-90
- INT 2-25; 4-100
- Interrupteur à clé des UC 351 352 2-15
- Intervention défaut 3-4; 3-8
 - défaut de diagnostic 3-4
 - défaut fatal 3-4
 - défaut informationnel 3-4
 - intervention défaut d'E/S B-11
- IO_FLT 2-28
- IO_FULL 2-26
- IO_PRES 2-28

J

- Jeu d'instructions 4-1
 - instructions arithmétiques 4-27
 - instructions de commande 4-110
 - instructions de comparaison 4-42
 - instructions de conversion 4-97
 - instructions de transfert de données 4-71
 - instructions par diagramme en échelle 4-2
 - instructions sur tableau 4-89
 - opérations logiques 4-48
 - temporisateurs et compteurs 4-10
- JUMP 4-123

L

- LABEL 4-125
- LE 4-42
- Lien horizontal 4-8
- Lien vertical 4-8
- Liens
 - horizontaux et verticaux 4-8
- LN 4-38
- LOG 4-38
- LOS_IOM 2-27
- LOS_SIO 2-27
- LOW_BAT 2-27
- LST_SCN 2-26
- LT 4-42

M

- Maintenance 3-1
- MCR 4-119
- Mémoire
 - altération 3-7
- Mise hors tension 2-34
- Mise sous tension 2-32
- Mnémoniques des instructions C-1
- MOD 4-32
- Mode temps de cycle constant 2-13
- Modèles Micro 2-46
- Modes de la fenêtre de communication 2-14
- Module d'E/S additionnel 3-18
- Module d'E/S perdu 3-17
- Module optionnel
 - défaut logiciel 3-10
- Module optionnel perdu ou manquant 3-8
- Module optionnel réinitialisé, additionnel ou non configuré 3-8
- Modules d'E/S des API Série 90-20
 - modules d'E/S Modèle 20 2-46
- Modules d'E/S des API Série 90-30
 - conditions par défaut des modules de sortie
 - Modèle 30 2-44
 - données de diagnostic 2-45
 - données globales 2-45
 - formats des données d'E/S 2-44
 - modules d'E/S Modèle 30 2-42
- Modules d'E/S Modèle 20 2-46
- Modules d'E/S Modèle 30 2-42
- MOT 2-25
- Mots de passe 2-38
- MOVE 4-72
- MSKCMP 4-67
- MUL 4-28
- Multiplication 4-28

N

- NE 4-42
- Nombres à virgule flottante E-1
 - erreurs dans les opérations et les nombres à virgule flottante E-6
 - format interne des nombres à virgule flottante E-3
 - saisie et affichage des nombres à virgule flottante E-5
 - valeurs des nombres à virgule flottante E-4
- NOT 4-54

O

OCTET 2-25
 OFDT 4-18
 ONDTR 4-12
 Opération logique AND 4-50
 Opération logique NOT 4-54
 Opération logique OR 4-50
 Opération logique XOR 4-52
 Opérations logiques 4-48
 AND 4-50
 BCLR 4-63
 BPOS 4-65
 BSET 4-63
 BTST 4-61
 MCMP 4-67
 NOT 4-54
 OR 4-50
 ROL 4-59
 ROR 4-59
 SHL 4-56
 SHR 4-56
 XOR 4-52
 OR 4-50
 Organisation du programme et
 données/références utilisateur 2-18
 état système 2-26
 références utilisateur 2-21
 rémanence des données 2-24
 structure des blocs fonctionnels 2-28
 transitions et forçages 2-23
 types de données 2-25
 OV_SWP 2-27
 OVR_PRE 2-26

P

Paramètres des blocs fonctionnels 2-30
 Pas de programme utilisateur présent 3-12
 PB_SUM 2-27
 PCM
 communication avec l'API 2-12
 PID 4-146
 PLC fault table
 interpreting a fault B-1
 PLC_BAT 2-26
 PRG_CHK 2-26
 Privilèges d'accès 2-38
 demandes de changement 2-39
 Processeur d'alarmes 3-2
 Programmation
 instructions 4-1
 instructions arithmétiques 4-27

instructions de commande 4-110
 instructions de comparaison 4-42
 instructions de conversion 4-97
 instructions de transfert de données 4-71
 instructions sur tableau 4-89
 mnémoniques des instructions C-1
 opérations logiques 4-48
 par diagramme en échelle 4-2
 temporisateurs et compteurs 4-10
 Programme d'application 2-8
 exécution 2-8
 Programme utilisateur
 calcul de CHECKSUM 2-9
 Programme utilisateur altéré à la mise sous
 tension 3-13
 Protection de la mémoire flash des UC 351 et
 352 2-15
 Puissance de l'instruction e 4-38
 Puissance de l'instruction X 4-38

R

Racine carrée 4-34
 RAD 4-40
 RANGE 4-45
REAL
 conversion au format REAL 4-104
 structre de type de données 2-25
 utilisation des nombres à virgule flottante E-1
 utilisation des nombres réels E-1
 Référence de sortie
 logique 2-22
 Référence d'entrée
 logique 2-22
 Références de défauts 3-4
 définition 3-5
 Références des données globales 2-23
 Références des registres 2-21
 entrées analogiques 2-21
 registres système 2-21
 sorties analogiques 2-21
 Références d'état
 système 2-23
 Références d'état système 2-26
 ADD_IOM 2-27
 ADD_SIO 2-27
 ALW_OFF 2-26
 ALW_ON 2-26
 ANY_FLT 2-28
 APL_FLT 2-27
 BAD_PWD 2-28
 BAD_RAM 2-27
 CFG_MM 2-27

- FST_SCN 2-26
 - HRD_CPU 2-27
 - HRD_FLT 2-28
 - HRD_SIO 2-27
 - IO_FLT 2-28
 - IO_FULL 2-26
 - IO PRES 2-28
 - LOS_IOM 2-27
 - LOS_SIO 2-27
 - LOW_BAT 2-27
 - LST_SCN 2-26
 - OV_SWP 2-27
 - OVR_PRE 2-26
 - PB_SUM 2-27
 - PLC_BAT 2-26
 - PRG_CHK 2-26
 - SFT_CPU 2-28
 - SFT_FLT 2-28
 - SFT_SIO 2-27
 - STOR_ER 2-28
 - SY_FLT 2-28
 - SY_FULL 2-26
 - SY PRES 2-28
 - T_100MS 2-26
 - T_10MS 2-26
 - T_MIN 2-26
 - T_SEC 2-26
 - Références internes
 - logiques 2-22
 - Références logiques 2-22
 - données globales 2-23
 - entrées logiques 2-22
 - état système 2-23; 2-26
 - logiques internes 2-22
 - logiques temporaires 2-22
 - références système 3-5
 - sorties logiques 2-22
 - Références registre système 2-21
 - Références système 3-5
 - Références temporaires
 - logiques 2-22
 - Références utilisateur 2-21
 - données globales 2-23
 - entrées analogiques 2-21
 - entrées logiques 2-22
 - état système 2-23; 2-26
 - logiques internes 2-22
 - logiques temporaires 2-22
 - références des registres 2-21
 - références logiques 2-22
 - références système 3-5
 - registres système 2-21
 - sorties analogiques 2-21
 - sorties logiques 2-22
 - Registre de sortie
 - analogique 2-21
 - Registre d'entrée
 - analogique 2-21
 - Rémanence des données 2-24
 - Restitution des sorties 2-8
 - ROL 4-59
 - ROR 4-59
- ## S
- Sécurité du système 2-38
 - demandes de changement de niveau d'autorisation 2-39
 - mots de passe 2-38
 - niveaux d'accès 2-38
 - verrouillage/déverrouillage des sous-programmes 2-39
 - Séquences de mise sous/hors tension 2-32
 - mise hors tension 2-34
 - mise sous tension 2-32
 - Seuil pile bas
 - signal 3-11
 - SFT_CPU 2-28
 - SFT_FLT 2-28
 - SFT_SIO 2-27
 - SHFR 4-79
 - SHL 4-56
 - SHR 4-56
 - Signal "seuil pile bas" 3-11
 - SIN 4-36
 - SNPX_RD 2-27
 - SNPX_WT 2-27
 - SNPXA CT 2-27
 - Sorties
 - restitution 2-8
 - Sous-programmes
 - verrouillage/déverrouillage 2-39
 - Sous-programmes périodiques 2-21
 - Soustraction 4-28
 - SQRT 4-34
 - SRCH_EQ 4-94
 - SRCH_GE 4-94
 - SRCH_GT 4-94
 - SRCH_LE 4-94
 - SRCH_LT 4-94
 - SRCH_NE 4-94
 - STOR_ER 2-28
 - Structure d'E/S, API Série 90-30 2-41
 - Structure des blocs fonctionnels 2-28
 - flux validant 2-31
 - format des blocs fonctionnels de programme 2-29
 - format des instructions à diagramme en échelle 2-28

- paramètres des blocs fonctionnels 2-30
 - Structure du programme
 - appel des sous-programmes 2-20
 - bloc de sous-programme 2-19
 - SUB 4-28
 - SVCREQ 4-127
 - interroger les E/S 4-144
 - lire checksum maître 4-143
 - lire la dernière entrée enregistrée dans la table des défauts 4-137
 - lire le compteur de mise hors tension 4-145
 - lire le compteur de temps de fonctionnement 4-141
 - lire l'état des forçages des E/S 4-142
 - mettre hors service (arrêter) l'API 4-135
 - modifier/lire l'état de la tâche de CHECKSUM et le nombre de mots qu'elle doit vérifier 4-129
 - modifier/lire l'horloge interne 4-131
 - vider les tables des défauts 4-136
 - SY_FLT 2-28
 - SY_FULL 2-26
 - SY_PRES 2-28
 - System status references
 - SNPX_RD 2-27
 - SNPX_WT 2-27
 - SNPXACT 2-27
 - Système
 - fonctionnement 2-1
- ## T
- T_100MS 2-26
 - T_10MS 2-26
 - T_MIN 2-26
 - T_SEC 2-26
 - Table des défauts automate 3-3; 3-5; B-1
 - action sur défaut B-5
 - bac B-3
 - codes d'erreur B-5
 - description 3-7
 - emplacement B-3
 - groupe de défauts B-4
 - horodatage de défaut automate B-7
 - indicateur long/court B-3
 - informations supplémentaires sur le défaut B-7
 - remplissage B-3
 - tâche B-4
 - Table des défauts d'E/S 3-3; 3-6; B-8
 - adresse de défaut B-10
 - adresse de référence B-9
 - bac B-10
 - descriptions 3-17
 - emplacement B-10
 - groupe de défauts B-10
 - horodatage de défauts d'E/S B-12
 - indicateur long/court B-9
 - informations spécifiques de défaut d'E/S B-11
 - informations spécifiques de défaut symbolique B-11
 - interprétation d'un défaut B-1
 - intervention défaut B-11
 - interventions pour défauts spécifiques B-11
 - point B-10
 - TAN 4-36
 - Temporisateur à l'ouverture 4-18
 - Temporisateur chien de garde 2-37
 - Temporisateur de cycle constant 2-37
 - Temporisateur suspensif 4-12; 4-15
 - Temporisateurs 2-36; 4-10
 - contacts d'impulsions d'horloge 2-37
 - données de bloc fonctionnel 4-10
 - OFDT 4-18
 - ONDTR 4-12
 - temporisateur chien de garde 2-37
 - temporisateur de cycle constant 2-37
 - TMR 4-15
 - Temps d'exécution des instructions A-1
 - TMR 4-15
 - Touches ALT D-1
 - Touches CTRL D-1
 - Transitions 2-23
 - TRUN 4-108
 - Type de défaut 3-17
 - Types de données 2-25
 - BCD-4 2-25
 - BIT 2-25
 - DINT 2-25
 - INT 2-25
 - MOT 2-25
 - OCTET 2-25
 - REAL 2-25
- ## U
- UC 351 et 352: changement de mode à l'aide l'interrupteur à clé 2-15
 - UC 351 et 352: interrupteur à clé 2-15
- ## V
- Variantes du cycle de programme 2-13
 - Verrouillage de blocs 2-39
 - EDITLOCK 2-39
 - VIEWLOCK 2-39
 - Verrouillage des blocs
 - verrouillage permanent d'un sous-programme 2-40
 - Verrouillage/déverrouillage des sous-programmes 2-39

VIEWLOCK 2-39

W

WORD 4-106

X

XOR 4-52