GE Fanuc Automation

Automates Programmables Industriels

API Série 90™-30/20/Micro
Jeu d'instructions de l'UC

Manuel de référence

GFK-0467L-FR *Juin 1999*

Avertissements, Précautions et Remarques utilisés dans cette publication

Avertissement

Les Avertissements sont utilisés dans cet ouvrage pour mettre en évidence l'existence de risques de blessures physiques et de dommages matériels provoqués par des niveaux de tension, de courant , de température et autres conditions dangeureuses.

Dans les situations où l'inattention pourrait provoquer soit des blessures personnelles, soit des dommages à l'équipement, un Avertissement est utilisé.

Précautions

Les avis de Précautions sont utilisés lorsque l'équipement pourrait être endommagé si des précautions ne sont pas prises.

Remarques

Les remarques attirent simplement l'attention sur les informations qui sont particulièrement significatives pour la compréhension et l'utilisation de l'équipement.

Ce document est basé sur des informations disponibles au moment de sa publication. Malgré tous les soins apportés quant à leur précision, les informations contenues ici ne prétendent pas couvrir tous les détails ou variations des matériels et logiciels, ni parer à toute éventualité concernant l'installation, l'utilisation ou la maintenance. Des caractéristiques peuvent être décrites ici, qui ne sont pas présentes dans tous les systèmes matériels et 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, l'état complet, la suffisance ou l'utilité des informations contenues ici. Par ailleurs, aucune garantie ne s'appliquera quant à l'adaptation à un marché ou à une application donnée.

Ce qui suit sont des marques déposées de GE Fanuc Automation North America, Inc.

Alarm Master	Genius	PowerTRAC	Series Six
CIMPLICITY	Helpmate	ProLoop	Series Three
CIMPLICITY 90–ADS	Logicmaster	PROMACRO	VersaMax
CIMSTAR	Modelmaster	Series Five	VersaPro
Field Control	Motion Mate	Series 90	VuMaster
Genet	PowerMotion	Series One	Workmaster

Ce manuel décrit le fonctionnement du système, la gestion des défauts et les instructions de programmation du Logicmaster 90™ pour les automates programmables Série 90™-30, Série 90-20 et Série 90 Micro. Les API Série 90-30, Série 90-20 et Série 90 Micro font partie de la famille d'automates programmables Série 90 de GE Fanuc Automation.

Révisions de ce manuel

- L'UC Modèle 364 (version 9.0 et ultérieure) supporte la connexion à un réseau Ethernet par l'un des deux ports Ethernet intégrés. Les ports AAUI et 10BaseT sont fournis. Le Modèle 364 est la seule UC Série 90-30 à supporter les Données Globales Ethernet (p. 2-39).
- La console de programmation de poche ne vous permet pas de changer l'horloge de la date et de l'heure tant que la protection par interrupteur à clé est active (p. 2-14).
- Des chiffres sur l'impact du balayage, pour la configuration DSM, ont été ajoutés au Chapitre 2.
- Des différences de fonctionnement des fonctions Logarithmique/Exponentielle (p. 6-12) et Tronquer (p. 11-11) ont été identifiées pour l'UC352.
- Les descriptions de l'Enregistreur d'Evénement Séquentiel (SER) ont été revues pour clarifier les possibilités de cette fonction. Des informations détaillées sur la synchronisation de cette fonction sont fournies dans l'Annexe A. Cette fonction est disponible avec la version 9 et ultérieure des UC série 35x et 36x (p. 12-8).
- Une nouvelle reqête de service, Sauter la scrutation des entrées et des sorties suivante (SVCREQ #45), est disponible avec la version 9.0 et ultérieure des UC série 35x et 36x (p. 12-63).
- Les listings des blocs paramètres pour les fonctions d'écriture et de lecture de la requête de service Accès rapide à l'état du fond de panier (SVCREQ #46) ont été corrigés (p. 12-64).
- Autres corrections et clarifications nécessaires.

Contenu de ce manuel

Chapitre 1. Introduction : fournit une vue d'ensemble des API Série 9030, Série 9020 et Série 90 Micro, le jeu d'instructions Série 9030/20/Micro.

Chapitre 2. Fonctionnement du système : décrit certaines opérations des API Série 9030, Série 9020 ou Série 90 Micro. Ceci comprend une description des séquences de balayage, de mise sous tension et hors tension, des horloges et des temporisateurs, de la sécurité, des E/S et de la gestion des défauts de l'API. Il comprend également des informations d'ordre général pour la compréhension basique de la programmation de la logique à relais.

Chapitre 3. Description et correction des défauts : fournit des informations sur le dépannage des API Série 9030, 9020 ou Micro. Il décrit les défauts de la tables des défauts de l'API et les catégories de défauts de la table des défauts des E/S.

Chapitre 4—12. Jeu d'instructions Série 9030/20/Micro : décrit les instructions de programmation disponibles pour les API Série 9030, Série 9020 et Série 90 Micro. Ces chapitres correspondent aux groupes de fonctions du programme principal.

Annexe A. Temps d'exécution des instructions : liste la taille des mémoires en octets et le temps d'exécution en microsecondes de chaque instruction de programmation. La taille mémoire est le nombre d'octets nécessités par la fonction dans un programme d'application de schéma relais.

Annexe B. Interprétation des tables de défauts : décrit comment interpréter le format de la structure des messages lors de la lecture des tables de défauts en utilisant le logiciel Logicmaster 9030/20/Micro.

Annexe C. Mnémoniques des instructions : liste les mnémoniques qui peuvent être tapés pour afficher les instructions de programmation en recherchant ou en éditant un programme.

Annexe D. Fonctions des touches : liste les affectations spéciales du clavier utilisées pour le logiciel Logicmaster 9030/20/Micro. Une *carte amovible de référence rapide* qui liste les fonctions des touches et les mnémoniques des instructions suit cette annexe.

Annexe E. Utilisation des nombres à virgule flottante : décrit des considérations spéciales pour l'utilisation de la virgule flottante dans les opérations mathématiques.

Publications relatives

Manuel de l'utilisateur du logiciel de programmation du Logicmaster™ 90 Série 90™30/20/Micro (GFK-0466).

Informations importantes sur le produit Logicmaster™ 90 Série 9030 et 9020 (GFK0468).

Manuel d'installation de l'automate programmable Série 90™30 (GFK0356).

Manuel d'installation de l'automate programmable Série 90™20 (GFK0551).

Manuel des caractéristiques du module d'E/S Série 90™30 (GFK0898).

Manuel de l'utilisateur du logiciel de support et du module coproceseur programmable Série 90™ (GFK0255).

Manuel de l'utilisateur du logiciel de développement (PCOP) du PCM Série 90™ (GFK-0487).

Manuel de l'utilisateur du système d'affichage alphanumérique CIMPLICITY™ 90ADS (GFK-0499).

Manuel de référence du système d'affichage alphanumérique CIMPLICITY™ 90ADS (GFK-0641).

Fiche technique du module coprocesseur d'affichage alphanumérique (GFK-0521).

Manuel de l'utilisateur de la console de programmation de poche des API Série 90™30 et 9020 (GFK0402).

Manuel de l'utilisateur du Power Mate APM pour l'API Série 90™Mode standard (GFK0840).

Manuel de l'utilisateur du Power Mate APM pour l'API Série 90™Mode suiveur (GFK-0781).

Manuel de l'utilisateur du Motion Mate™ DSM302 pour les API Série 90™-30 (GFK-1464)

Manuel de l'utilisateur des compteurs rapides Série 90™30 (GFK-0293).

Manuel de l'utilisateur du module de communication Genius Série 90™30 (GFK-0412).

Fiche technique du module de communication Genius (GFK-0272).

Manuel de l'utilisateur du contrôleur de bus Genius™ Série 90™30 (GFK1034).

Manuel de l'utilisateur du contrôleur de bus FIP Série 90™70 (GFK1038).

Manuel de l'utilisateur du scrutateur d'E/S déporté FIP Série 90™30 (GFK1037).

Manuel de l'utilisateur de l'ensemble interface de bus Genius™ pour les E/S distribuées et le contrôle système Field Control™ (GFK0825).

Manuel de l'utilisateur de l'Automate Programmable Série 90™ Micro (GFK1065).

Manuel de l'utilisateur des communications série de l'API Série 90™ (GFK0582).

Vos commentaires et suggestions sont les bienvenus

Chez GE Fanuc Automation, nous nous efforçons de produire une documentation technique de qualité. Après avoir utilisé ce manuel, nous vous remercions de bien vouloir remplir et nous retourner la Carte de Commentaires du Lecteur que vous trouverez page suivante.

Libby Allen
RédacteurTechnique

Préface

Table des matières

Table des matières

Table des matières

Table des matières

Table des matières

Chapitre 1 Introduction

Les API Série 90-30, 90-20, et Micro font partie de la gamme d'Automates Programmables Industriels (API) Série 90 de GE Fanuc. Ils sont faciles à installer et à configurer ; ils offrent des fonctions de programmation avancées et sont compatibles avec les API Série 90-70.

L'API Série 90-20 constitue une solution adaptée pour les applications comportant peu d'E/S. Les principaux objectifs de l'API Série 90-20 sont les suivants :

- Offrir un petit API facile à utiliser, à installer, à améliorer et à maintenir.
- Offrir un API da faible coût compatible avec la gamme.
- Faciliter l'intégration du système grâce au matériel et aux protocoles de communication standards.

L'API Série 90 Micro constitue également une solution adaptée pour les applications comportant peu d'E/S. Les principaux objectifs de l'API Micro sont les mêmes que ceux de la Série 90-20. De plus, le Micro offre:

- L'UC, l'alimentation, les entrées et les sorties sont toutes intégrées dans une petite unité.
- La plupart des modèles possèdent également un compteur rapide.
- Du fait que l'UC, l'alimentation, les entrées et les sorties sont toutes intégrées dans une seule unité, la configuration est très facile.

Les unités centrales des API 341, Série 90-30 et Série 90-20 utilisent une architecture de gestion de mémoire et de priorité d'exécution basée sur le microprocesseur 80188. Les modèles d'unité centrale 35x et 36x utilisent un microprocesseur 80386EX. L'API Série 90 Micro utilise le microprocesseur H8. Le cycle de ces API comprennent l'exécution du programme d'application et des tâches fondamentales comme les routines diagnostiques, la scrutation des entrées/sorties et le traitement des défauts. Le logiciel du système contient également des routines de communication avec la console de programmation. Ces routines permettent le chargement et le déchargement de programmes d'application, la récupération d'informations d'état et la commande de l'API.

Dans l'API Série 90-30, le programme d'application (logique utilisateur), qui gère le processus final pour lequel est destiné l'API, est commandé par un CoProcesseur Séquenceur d'Instructions (ISCP) dédié. L'ISCP est constitué d'un composant matériel dans les modèles 313 et supérieurs, et d'un composant logiciel dans le modèle 311 et l'API Micro. Le microprocesseur 80188 et l'ISCP peuvent travailler simultanément, ce qui permet au microprocesseur de gérer les communications pendant que l'ISCP exécute l'ensemble du programme d'application ; cependant, le microprocesseur doit exécuter les blocs fonctionnels non-Booléens.

Des défauts sont détectés dans l'automate Série 90-30, Série 90-20 ou Micro lorsqu'une anomalie pouvant affecter le fonctionnement et les performances du système survient. Lorsqu'un défaut non critique apparaît, comme une tension de pile faible indiquant que la tension de la pile protégeant la mémoire doit être remplacée, un simple avertissement est indiqué.

Les défauts sont gérés par une fonction processeur d'alarmes logicielle qui enregistre les défauts soit dans la table des défauts automate, soit dans la table des défauts des E/S. (Les UC des modèles 331 et supérieurs donnent également la date et l'heure des défauts). Ces tables peuvent être affichées par le logiciel de programmation Logicmaster 90-30/20/Micro en utilisant les fonctions de contrôle et d'état.

Remarque

Les possibilités de virgule flottante ne sont supportées *que* par les UC des modèles 35x et 36x, Version 9 ou plus et par toutes les versions d'UC 352.

L'UC du modèle 364 (Version 9.10 ou plus) est la seule UC Série 90-30 qui supporte EGD.

Remarque

Pour des informations supplémentaires, voir les annexes à la fin de ce manuel.

- L'Annexe A liste la taille mémoire en octets et le temps d'exécution en microsecondes de chaque instruction de programmation.
- L'Annexe B explique comment interpréter le format de la structure des messages lors de la lecture des tables de défauts de l'API et des E/S.
- L'Annexe C liste les mnémoniques des instructions pour les opérations de recherche et d'édition de programme.
- L'Annexe D liste les affectations spéciales du clavier utilisées par le logiciel du Logicmaster 90-30/20/Micro.
- L'Annexe E décrit l'utilisation des opérations mathématiques au format virgule flottante.

Chapitre 2 Fonctionnement du système

Ce chapitre décrit certaines opérations du système des API Série 90-30, 90-20 et Micro. Ces opérations du système comprennent :

Section 1 : Résumé du cycleAutomate

Le programme logique des API Série 90-30, 90-20 et Micro s'exécute de façon répétitive jusqu'à ce qu'il soit arrêté par une instruction de la console de programmation ou d'une commande provenant d'un autre équipment. La séquence des opérations nécessaires pour une éxécution du programme est appelée cycle. En plus de l'exécution du programme logique, le cycle inclut la lecture des données provenant des modules d'entrée, l'écriture de données vers les modules de sortie, la gestion interne, la gestion de la console de programmation et celle des autres fonctions de communication.

Les API Série 90-30, 90-20 et Micro fonctionnent habituellement en mode **CYCLE DE PROGRAMME STANDARD**. Les autres modes de fonctionnement comprennent le mode **STOP AVEC E/S DESACTIVEES**, le mode **STOP AVEC E/S ACTIVEES** et le mode **CYCLE CONSTANT**. Chacun de ces modes, décrits dans ce chapitre, est commandé par des évènements extérieurs et des réglages de configuration de l'application. L'API prend la décision concernant son mode de fonctionnement au début de chaque cycle.

Cycle de programme standard

Le mode **CYCLE DE PROGRAMME STANDARD** s'exécute normalement sous toutes les conditions. L'UC travaille en exécutant un programme d'application, en mettant les E/S à jour et en effectuant les communications et d'autres tâches. Ceci se produit dans un cycle répétitif appelé cycle de l'UC. Il existe sept parties dans la séquence d'exécution du cycle de programme standard :

1. Début du cycle de gestion interne
2. Scrutation des entrées
3. Exécution du programme d'application logique
4. Scrutation des sorties
5. Gestion de la console de programmation
6. Gestion des communications autres que la console de programmation
7. Diagnostics

Toutes ces étapes s'exécutent à chaque cycle. Bien que la fenêtre de communication vers console de programmation s'ouvre à chaque cycle, les services correspondants ne sont actifs que si un défaut de carte a été détecté ou si la console de programmation émet une demande de service ; autrement dit, la fenêtre de communication de la console de programmation vérifie l'existence de tâches à accomplir et se ferme sinon. La séquence du cycle de programme standard est illustrée par la figure suivante.

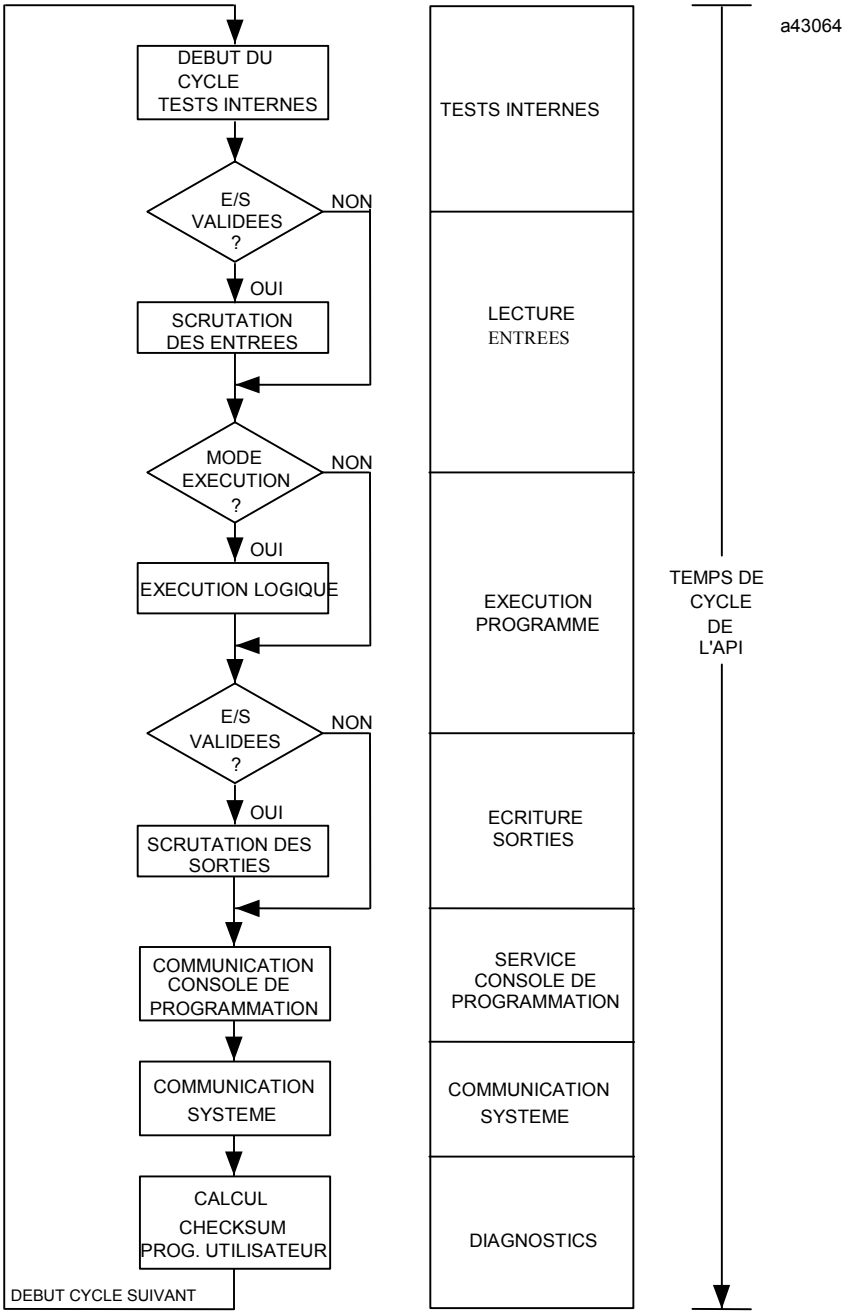


Figure 2-1. Cycle Automate

Comme montré dans la séquence décrivant le cycle automate, plusieurs éléments sont inclus dans le cycle. Ces éléments contribuent au temps de cycle global comme indiqué dans la table suivante.

Table 2-1. Contribution au temps de cycle

Elément de cycle	Description		Contribution en temps (ms) [4]
			Modèles 351, 352, 350 et 36x (les temps des modèles 350 et 36x estimés comme étant identiques)
Gestion interne	• Calcule le temps de cycle. • Prévoit le début du cycle suivant. • Détermine le mode du cycle suivant. • Met à jour les tables de référence de défauts. • Réinitialise la temporisation du chien de garde.		0.279
Entrée des données	Les données d'entrée sont reçues des modules d'entrées et optionnels.		Voir la Table 2-2 pour les contributions au temps de scrutation.
Exécution du programme	La logique utilisateur est éxécutée.		Le temps d'exécution dépend de la longueur du programme et du type des instructions utilisées dans le programme. Les temps d'exécution des instructions sont listés dans l'Annexe A.
Sortie des données	Les données de sortie sont envoyées aux modules de sorties et optionnels.		Voir la Table 2-2 pour les contributions au temps de scrutation.
Gestion des appareils externes	Les demandes de service des appareils de programmation et des modules intelligents sont traitées. [1]	HHP	0.334
		LM-90	0.517
		PCM [2]	0.482
Reconfiguration	Les emplacements comportant des modules défectueux et les emplacements vides sont lus.		0.319[6]
Diagnostics	Vérifient l'intégrité du programme utilisateur		0.010 par mot contrôlé par checksum à chaque cycle[3, 7]

1. La contribution au temps de scrutation du service lié à un équipement extérieur dépend du mode de configuration de la fenêtre de communication. Si le mode de la fenêtre est configuré en mode **LIMITE**, un maximum de 8 millisecondes pour les UC 311, 313, 323 et 331, et 6 millisecondes pour les UC 340 et supérieures, seront allouées à cette fenêtre. Si le mode de la fenêtre est en mode **RUN COMPLETE** , un maximum de 50 ms peut être alloué à cette fenêtre, en fonction des services à accomplir simultanément.
2. Ces mesures ont été prises avec une PCM présent physiquement mais non configurée et n'exécutant pas de tâche d'application
3. Le nombre de mots vérifiés par checksum à chaque cycle peut être changé par le bloc fonctionnel SVCREQ.
4. Ces mesures ont été prises avec un programme vide et la configuration par défaut. Les API Série 90-30 se trouvaient dans un châssis à 10 emplacements, sans châssis d'extension connecté. Egalement, les temps de cette table présument qu'il n'y a pas de sous-programme périodique actif ; les temps seront plus longs si un sous-programme périodique est actif.
5. Le temps d'entrées des données de l'API Micro peut être déterminé comme suit : 0,365 ms. (scrutation fixe) + 0,036 ms. (temps de filtrage) x (temps de cycle total)/0,5 ms.
6. Comme l'API Micro possède une configuration d'E/S figée, la reconfiguration n'est pas nécessaire.
7. Comme le programme utilisateur de l'API se trouve en mémoire Flash, son intégrité ne sera pas vérifiée.

Table 2-2. Contributions au temps de cycle des modules d'E/S pour les modèles 90-30 35x et 36x (en millisecondes)

Type de module		UC Série 35x et 36x		
		Châssis principal	Châssis d'extension	Châssis déporté
Entrées logiques, 8 points		.030	.055	.206
Entrées logiques, 16 points		.030	.055	.206
Entrées logiques, 32 points		.043	.073	.269
Sorties logiques, 8 points		.030	.053	.197
Sorties logiques, 16 points		.030	.053	.197
Sorties logiques, 32 points		.042	.070	.259
Combinaison entrées/sorties logiques		.060	.112	.405
Entrées analogiques, 4 voies		.075	.105	.396
Sorties analogiques, 2 voies		.058	.114	.402
Entrées analogiques, 16 voies (courant ou tension)		.978	1.446	3.999
Sorties analogiques, 8 voies		1.274	1.988	4.472
Combinaison entrées/sorties analogiques		1.220	1.999	4.338
Compteur rapide		1.381	2.106	5.221
Processeur d'E/S		1.574	2.402	6.388
Interface Ethernet (sans connexion)		.038	.041	.053
APM Power Mate (1 axe)		1.527	2.581	6.388
APM Power Mate (2 axes)		1.807	2.864	7.805
DSM 302 *	40 AI, 6 AQ	2.143	3.315	9.527
	50AI, 9 AQ	2.427	3.732	11.092
	64 AI, 12 AQ	2.864	4.317	13.138
GCM	sans dispositif	.911	1.637	5.020
	8 dispositifs 64 mots	8.826	16.932	21.179
GCM+	sans dispositif	.567	.866	1.830
	32 dispositifs 64 mots	1.714	2.514	5.783
GBC	sans dispositif	.798	1.202	2.540
	32 dispositifs 64 mots	18.382	25.377	70.777
PCM 311	non configuré ou sans tâche d'application	.476	N/D	N/D
	lecture 128 %R aussi vite que possible	.485	N/D	N/D
ADC (sans tâche)		.476	N/D	N/D
I/O Link Maître	sans dispositif	.569	.865	1.932
	16 dispositifs 64 points	4.948	7.003	19.908
I/O Link Esclave	32 points	.087	.146	.553
	64 points	.154	.213	.789

* Pour les applications où les contributions des DSM au temps de scrutation affecteront le fonctionnement de la machine, vous pouvez avoir recours au bloc fonctionnel Do I/O et les requêtes de service Suspendre E/S et Accès Rapide Etat Fond de Panier pour transférer les données nécessaires vers et du module de commande d'axe sans obtenir toutes les données à chaque scrutation. Se référer au *Manuel de l'utilisateur du Motion Mate DSM302 pour les API Série 90-30*, GFK-1464 pour plus de détails.

Table 2-3. Contribution au temps de cycle des modules d' E/S pour la Série 90-30 jusqu'à 341 (en millisecondes)

Type de module		Modèle d'UC						
		311/313	331			340/341		
			Châssis principal	Châssis d'extension	Châssis déporté	Châssis principal	Châssis d'extension	Châssis déporté
Entrées logiques, 8 points		.076	.054	.095	.255	.048	.089	.249
Entrées logiques, 16 points		.075	.055	.097	.257	.048	.091	.250
Entrées logiques, 32 points		.094	.094	.126	.335	.073	.115	.321
Sorties logiques, 8 points		.084	.059	.097	.252	.053	.090	.246
Sorties logiques, 16 points		.083	.061	.097	.253	.054	.090	.248
Sorties logiques, 32 points		.109	.075	.129	.333	.079	.114	.320
Combinaison d'entrées/sorties, 8 points		.165	.141	.218	.529	.098	.176	.489
Entrées analogiques, 4 voies		.151	.132	.183	.490	.117	.160	.462
Sorties analogiques, 2 voies		.161	.138	.182	.428	.099	.148	.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
APM Power Mate (2 axes)		3.181	3.647	4.497	9.303	2.154	3.097	7.729
DSM 302	40 AI, 6 AQ	3.613	4.081	5.239	11.430	2.552	3.648	9.697
	50AI, 9 AQ	4.127	4.611	5.899	13.310	2.911	4.170	11.406
	64 AI, 12 AQ	4.715	5.276	6.759	15.747	3.354	4.840	13.615
GCM	sans dispositif	.041	.054	.063	.128	.038	.048	.085
	8 dispositifs, 64 points	11.420	11.570	13.247	21.288	9.536	10.648	19.485
GCM+	sans dispositif	.887	.967	1.164	1.920	.666	.901	1.626
	32 dispositifs, 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/D	3.350	N/D	N/D	1.684	N/D	N/D
	lecture 128 %R aussi vite que possible	N/D	4.900	N/D	N/D	2.052	N/D	N/D
ADC 311		N/D	3.340	N/D	N/D	1.678	N/D	N/D
Entrées analogiques, 16 voies (courant ou tension)		1.370	1.450	1.937	4.186	1.092	1.570	3.796
I/O Link Maître	sans dispositif	1.910	2.030	1.169	1.925	.678	.904	1.628
	16 dispositifs, 64 points	6.020	6.170	8.399	21.291	4.992	6.985	20.010
I/O Link Esclave	32 points	.206	.222	.289	.689	.146	.226	.636
	64 points	.331	.350	.409	1.009	.244	.321	.926

* Pour les applications où les contributions des DSM au temps de scrutation affecteront le fonctionnement de la machine, vous pouvez avoir recours au bloc fonctionnel Do I/O et les requêtes de service Suspendre E/S et Accès Rapide Etat Fond de Panier pour transférer les données nécessaires vers et du module de commande d'axes sans obtenir toutes les données à chaque scrutation. Se référer au *Manuel de l'utilisateur du Motion Mate DSM302 pour les API Série 90-30*, GFK-1464 pour plus de détails.

Calcul du temps de cycle

La Table 2-1 liste les sept éléments qui constituent le temps de cycle automate. Le temps de cycle se compose de fonctions de durée fixe (gestion interne et diagnostics) et de tâches de durée variable. Les durées variables varient selon la configuration des E/S, la taille du programme utilisateur et le type d'outil de programmation connecté à l'API.

Exemple de calcul de temps de cycle

Un exemple de calculs pour la détermination du temps de cycle pour un API Série 90-30, modèle 331, se trouve dans la table ci-dessous.

Les modules et les instructions utilisés pour ces calculs sont listés ci-dessous :

- Modules d'entrées : cinq modules d'entrées, 16 points, Série 90-30.
- Modules de sorties : quatre modules de sorties, 16 points, Série 90-30.
- Instructions de programmation : Un programme de 1 200 pas comprenant 700 instructions Booléennes (LD, AND, OR, etc.), 300 bobines de sortie (OUT, OUTM, etc.) et 200 fonctions mathématiques (ADD, SUB, etc.).

Gestion interne

La partie gestion interne du cycle effectue toutes les tâches nécessaires pour préparer le début du cycle. Si l'API est en mode **`CYCLE CONSTANT`**, le cycle est retardé jusqu'à ce que le temps de correspondant au cycle constant se soit écoulé. Si le temps nécessaire s'est déjà écoulé, le contact OV_SWP %SA0002 est établi et le cycle se poursuit sans attendre. Ensuite, les valeurs du temporisateur (centaines, dizaines et secondes) sont mises à jour par le calcul de la différence entre le début du cycle précédent et la durée du cycle en cours. Pour avoir un bonne précision, le départ effectif du cycle est enregistré en incréments de 100 microsecondes. Chaque temporisateur possède un champ contenant le nombre d'incréments de 100 microsecondes qui se sont produits depuis la dernière fois que la valeur du temporisateur a été incrémentée.

Scrutation des entrées

Les entrées sont lues pendant la fenêtre de scrutation des entrées , juste avant l'exécution de la logique. Pendant cette partie du cycle, tous les modules d'entrées de la Série 90-30 sont scrutés et leurs valeurs mémorisées dans les références %I (entrées logiques) ou %AI (entrées analogiques) , selon leur format. Toutes les données globales reçues par un module de communication Genius, un module de communication Genius amélioré ou un contrôleur de bus Genius, sont mémorisées dans les références %G.

Les modules sont scrutés dans l'ordre des adresses de façon ascendante en commençant par le module de communication Genius, puis les modules d'entrées logiques et finalement, les modules d'entrées analogiques.

Si l'UC est en mode **`STOP sans scrutation des`** E/S , la scrutation des entrées est ignorée.

Scrutation ou exécution du programme d'application logique

Le programme d'application est exécuté pendant la fenêtre d'exécution du programme logique. Cette exécution commence toujours par la première instruction du programme utilisateur qui suit immédiatement la fin de la scrutation des entrées. L'exécution de la logique génère des nouvelles valeurs de sorties. La logique se termine lorsque l'instruction END est exécutée (END est invisible sauf si vous utilisez une console portable).

Le programme d'application est exécuté par l'ISCP et le microprocesseur 80C188. Dans les UC 313 et supérieures, l'ISCP exécute les instructions Booléennes ; le 80C188 ou le 80386EX exécute les temporisateurs, les compteurs et les blocs fonctionnels. Dans les UC des modèles 311 et 90-20 , le 80C188 exécute toutes les instructions Booléennes, de temporisation, de comptage et des blocs fonctionnels. Sur le Micro, le processeur H8 exécute tous les blocs Booléens et fonctionnels.

Une liste des temps d'exécution de chaque fonction de programmation se trouve dans l'Annexe A.

Scrutation des sorties

Les sorties sont raffraîchies pendant la fenêtre de scrutation des sorties , suivant immédiatement la solution logique. Les sorties sont mises à jour en utilisant les données de la mémoire %Q (pour les sorties logiques) et %AQ (pour les sorties analogiques), selon leur format. Si le module de communication Genius est configuré pour transmettre des données globales, les données de la mémoire %G sont envoyées au GCM, GCM+ ou GBC. Les scrutations des sorties des Séries 90-20 et Micro n'incluent que les sorties logiques.

Pendant la scrutation des sorties, tous les modules de sorties de la Série 90-30 sont scrutés dans l'ordre des adresses de façon ascendante.

Si l'UC est dans le mode **`STOP sans scrutation des E/S`**, la scrutation des sorties est ignorée. La scrutation des sorties est terminée lorsque toutes les données de sorties ont été envoyées à tous les modules de sorties de la Série 90-30.

Calcul du checksum du programme de la logique

Un calcul de checksum est effectué sur le programme utilisateur à la fin de chaque cycle. Comme il serait trop long de calculer la checksum du programme tout entier, vous pouvez spécifier le nombre de mots, de 0 à 32, dans la configuration de l'UC.

Si le checksum calculé ne correspond pas au checksum de référence, un défaut de défaillance de checksum du programme est reporté. Ceci provoque l'entrée d'un défaut dans la table de défauts automate et le passage de l'API en **`STOP`**. Si le calcul du checksum est faux, la fenêtre de communication de la console de programmation n'est pas affectée. Le nombre de mots vérifiés par défaut, par totalisation, est de 8.

Fenêtre de communication de la console de programmation

Cette partie du cycle est dédiée à la communication avec la console de programmation. S'il y a une console de programmation connectée, l'UC exécute la fenêtre de communication de la console de programmation. Cette fenêtre ne s'exécutera que s'il y a une console de programmation connectée ou une carte à scruter dans le système. Une seule carte est scrutée à chaque cycle.

Une connexion est prévue pour la console de poche et pour les autres outils de programmation via un port série utilisant le protocole SNP (Series Ninety Protocol). Une connexion est également prévue pour les communications entre la console de programmation et les modules optionnels intelligents.

En mode fenêtre limitée par défaut, l'UC effectue une opération, pour la console de programmation, à chaque cycle, ce qui signifie qu'elle accepte une requête de service ou une réponse à l'appui d'une touche. Si la console de programmation fait une requête qui nécessite plus de 6 (ou 8 selon l'UC—voir la remarque) millisecondes de traitement, le traitement de la requête est étalé sur plusieurs cycles de façon à ce que l'impact sur le cycle ne dépasse 6 (ou 8 selon l'UC—voir la remarque) millisecondes.

Remarque

La limite de temps, pour la fenêtre de communication, est de 6 millisecondes pour les UC des modèles 340 et supérieurs et de 8 millisecondes pour les UC des modèles 311, 313, 323 et 331.

La figure suivante décrit l' organigramme de gestion de la console de programmation lors d'un cycle automate.

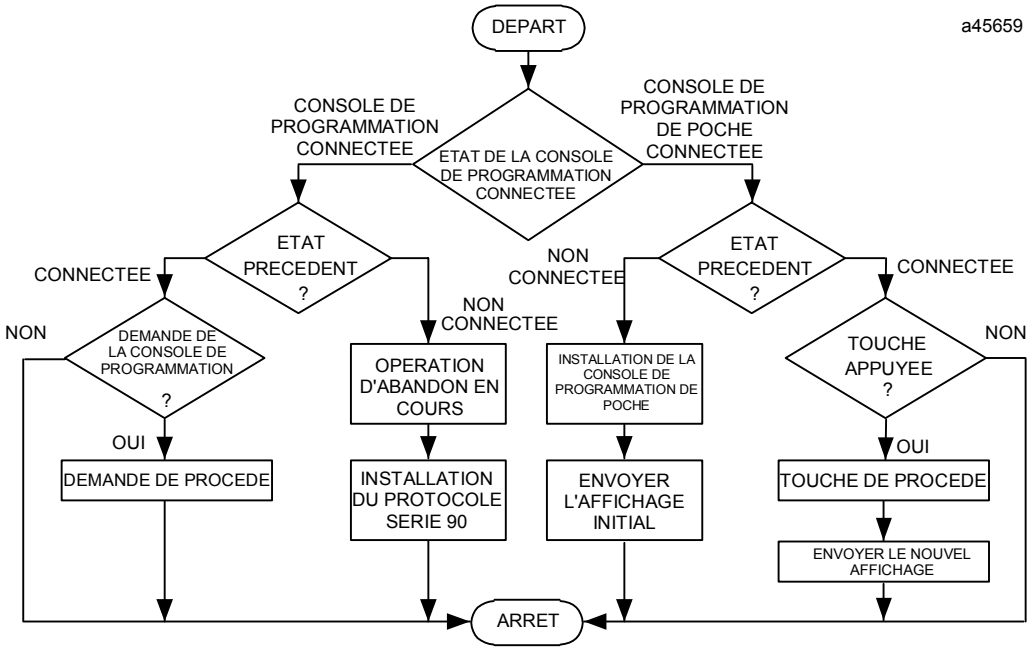


Figure 2-2. Organigramme de la fenêtre de communication de la console de programmation

Fenêtre de communication du système

Ceci corespond à la partie du cycle traitant les demandes de communication des modules d'option intelligents, comme le PCM ou le DSM (voir l'organigramme). Les demandes sont traitées sur la base du premier arrivé, premier servi. Cependant, comme les modules intelligents sont interrogés à tour de rôle, aucun module optionnel intelligent n'a la priorité sur un autre.

Dans le mode par défaut **`Exécution-complète`** , la longueur de la fenêtre de communication du système est limitée à 50 millisecondes. Si un module optionnel intelligent effectue une demande nécessitant plus de 50 millisecondes de traitement, la demande est traitée sur plusieurs cycles de façon à ce l'impact sur le cycle ne dépasse 50 millisecondes.

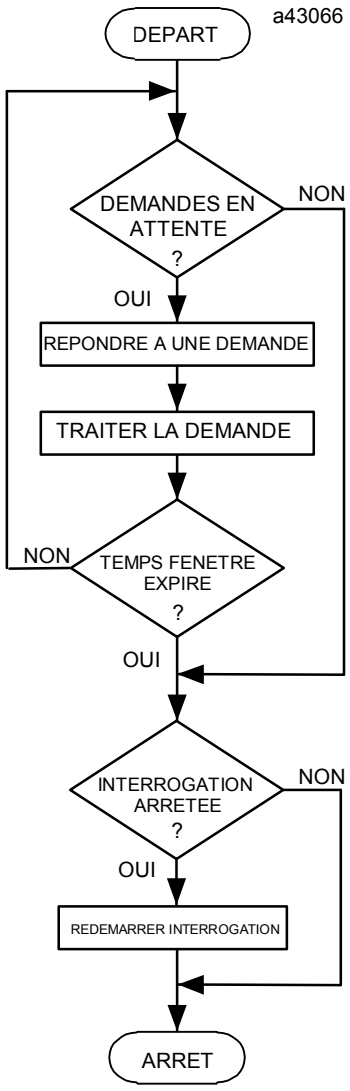


Figure 2-3. Organigramme de la fenêtre de communication système

Communication entre PCM et UC automate (Modèles 331 et supérieurs)

Les modules optionnels intelligents (IOM), comme la PCM, ne peuvent interrompre l'UC en cas de nécessité. L'UC doit interroger chaque module optionnel intelligent pour les requêtes de service. Cette interrogation se produit de façon asynchrone en tâche de fond, pendant le cycle (voir l'organigramme ci-dessous).

Lorsqu'un module optionnel intelligent est interrogé et envoie une requête de service à l'UC, cette requête est mise en file d'attente avant traitement pendant la fenêtre de communication du système.

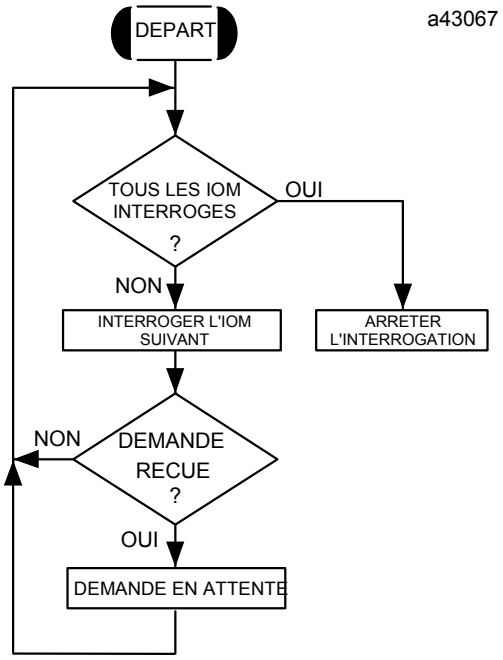


Figure 2-4. Communication entre PCM et UC automate

Communication entre DSM et UC automate

La DSM302 est un module intelligent fonctionnant de façon asynchrone avec l'UC de la Série 90-30. Les données sont échangées automatiquement entre l'UC et la DSM.

La DSM peut être configurée avec trois longueurs différentes de données %AI et %AQ. Une UC automate a besoin de temps pour lire et écrire les données, par le fond de panier, avec la DSM302. La Table 2-2 liste l'impact sur le temps de cycle automate des différentes configurations de données %AI et %AQ. Pour plus d'informations sur la synchronisation qui s'applique au module DSM302, se référer au manuel de l'utilisateur du *Motion Mate DSM302 pour la Série 90-30*, GFK-1464.

Les modes de scrutation autres que le mode standard

En plus du mode de scrutation standard, d'autres modes de scrutation sont disponibles. Ces modes, décrits dans les paragraphes suivants, peuvent être affichés et/ou modifiés à partir du logiciel de programmation.

Mode temps de cycle constant

Dans le mode de scrutation standard, chaque cycle automate est exécuté aussi rapidement que possible avec une durée variable pour chaque cycle. Le mode **TEMPS DE CYCLE CONSTANT** représente une alternative où chaque cycle est éxécuté avec la même durée. Vous pouvez réaliser ceci en validant le mode cycle constant qui deviendra ensuite le mode de scrutationpar défaut, prenant alors effet à chaque fois que l'API passe du mode **STOP** au mode **RUN**. Une valeur de 5 à 200 millisecondes (ou jusqu'à 500 millisecondes pour les UC des API modèles 35x et 36x), pour la temporisation de cycle constant (par défaut 100 millisecondes) est supportée.

A cause des variations du temps nécessaire à l'exécution des différentes parties du cycle automate, le temps de cycle constant doit être réglé à un valeur d'au moins 10 millisecondes supérieure au temps de cycle affiché sur la ligne d'état lorsque l'API est configuré en mode **TEMPS DE CYCLE NORMAL**. Ceci permet d'éviter que des défauts de dépassement de cycle ne se produisent.

Le mode temps de cycle constant est utile lorsque des points d'E/S ou des valeurs de registre doivent être interrogés à une fréquence constante, comme dans les algorithmes de commande. L'une des raisons d'utiliser le mode **TEMPS DE CYCLE CONSTANT** pourrait être de s'assurer que les E/S sont mises à jour à des intervalles réguliers. Une autre raison pourrait être de s'assurer qu'une certaine durée se soit écoulée entre la scrutation des sorties et la scrutation des entrées du cycle suivant, permettant aux entrées de se stabiliser après avoir l'écriture des valeurs de sortie venant du programme.

Si la temporisation du cycle constant expire avant que le cycle ne soit terminé, le cycle entier, incluant la fenêtre, est interrompu. Cependant, un défaut de dépassement de temps de cycle est enregistré au début du cycle suivant.

Remarque

Contrairement à la valeur du temps de cycle constant actif, qui ne peut être édité qu'en mode RUN, le mode temps de cycle constant ne peut être modifié qu'en mode STOP et il faudra "transférer la configuration de la console de programmation vers l'API" afin que la modification puisse prendre effet. Une fois transféré, ce mode devient le mode de scrutation par défaut.

Scrutation automate en mode STOP

Lorsque l'API est en mode **STOP**, le programme d'application n'est pas exécuté. La communication avec la console de programmation et les modules optionnels intelligents est maintenue. De plus, l'interrogation de carte défectueuse et l'exécution de la reconfiguration de carte se poursuit pendant le mode **STOP**. Pour plus d'efficacité, le système d'exploitation utilise des tranches de temps plus grandes que celles utilisées en mode **RUN** (habituellement, environ 50 millisecondes par fenêtre). Vous pouvez choisir si les E/S sont scrutées ou non. Les scrutations des E/S peuvent s'exécuter en mode **STOP** si le paramètre ***IOScan-Stop*** de l'écran de configuration de l'UC est mis sur **OUI**.

Modes fenêtre de communication

Le mode d'exécution par défaut de la fenêtre de communication vers console de programmation est le mode "Limité" . Ceci signifie que si une requête prend plus de 6 millisecondes à traiter, elle est traitée sur plusieurs cycles de façon à ce qu'aucun cycle ne soit affecté par plus de 6 millisecondes. Pour les UC 313, 323, et 331, l'impact sur le cycle peut aller jusqu'à 12 millisecondes pendant une mémorisation du mode **RUN**. Le mode d'exécution de cette fenêtre peut être changé en utilisant l'écran "Contrôle de cycle" du Logicmaster. Pour les instructions concernant ce changement de mode, se référer au Chapitre 5, "Etat et contrôle de l'API", dans le *Manuel de l'utilisateur du logiciel de programmation Logicmaster 90™ Série 90™-30/20/Micro* (GFK-0466).

Remarque

Si l'exécution de la fenêtre système est en mode Limité, les modules optionnels, comme la PCM ou le GBC, qui communiquent avec l'API en utilisant la fenêtre système, auront moins d'impact sur le temps de cycle automate, mais la réponse à leurs requêtes sera plus lente.

Interrupteur à clé sur les UC des modèles 35x et 36x : Changement de mode et protection de la mémoire Flash

Chaque UC des modèles 35x et 36x possède un interrupteur à clé, en face avant du module, permettant de protéger la mémoire Flash en écriture. Lorsque vous tournez cette clé sur la position **ON**/**RUN**, on ne peut modifier le contenu de la mémoire Flash sans tourner la clé sur la position OFF.

A partir de la version 7 des UC 351et 352, l'interrupteur à clé a une autre fonction : elle vous permet de mettre l'API en mode **STOP**, en mode **RUN** et d'effacer les défauts qui ne sont pas fatals, comme discuté dans la section suivante.

A partir de la version 8 des UC des modèles 35x et 36x, l'interrupteur à clé possède une fonction de protection avancée de la mémoire : Elle peut être utilisée pour fournir deux types supplémentaires de protection de la mémoire (voir la section “Utilisation de la protection mémoire sur les Versions 8 et supérieures”).

Si l'interrupteur à clé est activé et dans la position ON/RUN, vous ne pouvez modifier la date et l'heure qu'avec le logiciel de programmation. Dans ce cas, la console de programmation de poche ne vous permet pas de modifier la date et l'heure.

Utilisation de l'interrupteur à clé sur les Versions 7 et supérieures

Contrairement aux possibilités de protection de mémoire Flash des versions précédentes, si vous n'activez pas l'interrupteur à clé, par le paramètre d'interrupteur à clé **RUN**/**STOP** dans l'écran de configuration de l'UC, celui-ci ne disposera pas du contrôle avancé évoqué ici.

L’interrupteur à clé assure les mêmes fonctions de protections et de vérifications aussi bien pendant le mode STOP que lors du passage en RUN de l’automate.; Autrement dit, l'API ne pourra passer en mode **RUN** par action sur l'interrupteur à clé lorsqu'il se trouve en mode **STOP**/**DEFAUT**. Cependant, il est possible dans ce mode d’effacer les défauts non-fatals et mettre l'API en mode **RUN** en utilisant l'interrupteur à clé.

Si des défauts ***non-fatals*** se trouvent dans les tables de défauts (c'est-à-dire qu'ils ne provoquent pas le passage de l'UC en mode **STOP**/**DEFAUT**), l'UC sera alors placée en mode **RUN** lorsque vous tournerez la clé de Stop à Run pour la première fois et les tables de défauts NE SERONT PAS effacées.

Si des défauts fatals se trouvent dans la table des défauts (UC en mode **STOP**/**DEFAUT**), la première transition de l'interrupteur à clé de la position **STOP** à la position **RUN** provoquera le clignotement du voyant **RUN** de l'UC à une fréquence de 2 Hz et une temporisation de 5 secondes commencera. Le voyant **RUN** clignotant indique qu'il y a un(des) défaut(s) fatal(s) dans les tables de défauts. Dans ce cas, l'UC ne sera PAS mise en état **RUN**, même si l'interrupteur à clé se trouve dans la position **RUN**.

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

Si vous tournez la clé de **RUN** à **STOP**, puis, de nouveau sur **RUN** pendant les 5 secondes où le voyant **RUN** clignote, les défauts seront effacés et l'UC sera mise en mode **RUN**. Le voyant s'arrêtera de clignoter et restera allumé à partir de ce moment. L'interrupteur doit être maintenu en position **RUN** ou **STOP** pendant au moins 1/2 seconde avant d'être remis sur l'autre position.

Remarque

Si vous laissez les 5 secondes du temporisateur s'expirer (le voyant RUN s'arrête de clignoter), l'UC restera dans son état originel, le mode STOP/DEFAUT, avec les défauts dans la table des défauts. Si vous tournez à nouveau l'interrupteur à clé de la position STOP à la position RUN, le traitement sera répété, comme lors d'une première transition.

La table suivante résume l'incidence sur le fonctionnement de l'automate des deux paramètres R/S Switch et IOScan-Stop de l'UC, en fonction de la position physique de l'interrupteur à clé.

Paramètre de l'interrupteur à clé R/S dans la config. de l'UC	Position physique de l'interrupteur à clé	Paramètre IO Scan-Stop dans la configuration de l'UC	Fonctionnement de l'API
OFF	X	X	Tous les modes de la console de programmation de l'API sont permis.
ON	ON/RUN	X	Tous les modes de la console de programmation de l'API sont permis.
ON	OFF/STOP	X	L'API n'est pas autorisé à passer en RUN.
ON	Commuter l'interrupteur à clé de OFF/STOP à ON/RUN	X	L'API passe en RUN s'il n'y a aucun défaut fatal ; autrement, la LED RUN clignote pendant 5 secondes.
ON	Commuter l'interrupteur à clé de ON/RUN àOFF/STOP	NON	L'API passe en STOP–NO IO
ON	Commuter l'interrupteur à clé de ON/RUN à OFF/STOP	OUI	L'API passe en STOP–IO

X = Sans effet

Protection avancée de la mémoire avec les UC Versions 8 et supérieures

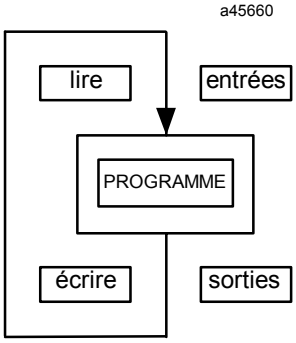
Dans les UC des Versions 8 et supérieures, l'interrupteur à clé possède en plus de toutes les fonctionnalités décrites ci-dessus, d'une fonction de protection de le RAM, si configuré comme tel. Deux types d'opération sont bloqués lorsque cette protection mémoire est activée : Le programme utilisateur et la configuration ne peuvent pas être modifiés, et le forçage et le bascuilement de points ne sont pas permis. Ceci est activé par le champ Mem Protect sur l'écran de configuration des UC des modèles 35x ou 36x du Logicmaster. Par défaut, elle est désactivée.

Section 2 : Organisation du programme et des données/références utilisateur

La taille totale de la logique des automates programmables Série 90-30 est listée dans la table suivante.

Modèles	Mémoire logique utilisateur (Koctets)
UC311	6
UC313, UC323	12
UC331	16
UC340	32
UC341	80
UC350	80 (versions 9 et supérieures) 32 (avant la version 9)
UC351, UC352, UC360, UC363, UC364	240 (versions 9 et supérieures) 80 (avant la version 9)

A partir des UC Version 9, les tailles mémoire des modèles 351, 352 et 36x sont configurables. (Pour des instructions détaillées et une description des tailles mémoire disponibles, se référer à la "Mémoire configurable des UC des modèles 351 et supérieurs" dans le Chapitre 10, Section 3 du *Manuel de l'utilisateur du logiciel de programmation du Logicmaster 90™, Série 90™-30/20/Micro* (GFK-0466K ou supérieur). Un programme pour un automate programmable Série 90-20 peut faire jusqu'à 2 Ko pour une UC du modèle Model 211. Le programme utilisateur contient la logique utilisée lorsqu'il est mis en route. Le nombre maximal de lignes de programme autorisé par bloc logique (principal ou sous-programme) est de 3000 ; pour les API 90-30, la taille maximale d'un bloc est de 80 kilo-octets pour les blocs C et de 16 kilo-octets pour les blocs LD et SFC. Dans un bloc SFC, une partie des 16 Ko est utilisée pour le bloc de données internes. La logique est exécutée de façon répétitive par l'API.



Se référer au *Manuel de l'utilisateur de l'automate programmable Série 90-30,* GFK-0356, ou au *Manuel de l'utilisateur de l'automate programmable Série 90-20*, GFK-0551, pour une liste des tailles de programme et des limites de référence pour chaque modèle d'UC.

Tous les programmes possèdent une table de variables qui liste les descriptions des variables et des références qui ont été affectées au programme utilisateur.

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

Blocs sous-programmes

Un programme peut "appeler" des blocs sous-programmes pendant son exécution. Un sous-programme doit être déclaré en edition dans la partie déclaration de bloc avant qu'une instruction CALL ne puisse être utilisée pour l'appeler. Un maximum de 64 déclarations de blocs sous-programmes dans le programme et 64 instructions CALL sont permises pour chaque bloc logique dans le programme. La taille maximale d'un bloc sous-programme est de 16 Ko ou 3000 circuits, mais le programme principal et tous les sous-programmes doivent respecter les contraintes de taille de logique pour le modèle d'UC utilisé.

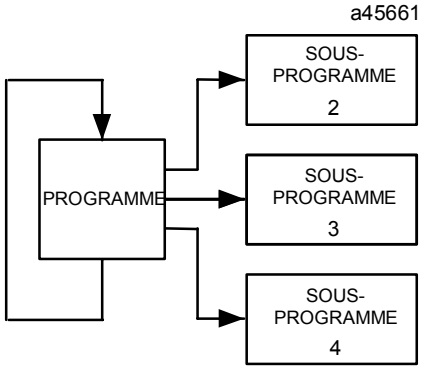
Remarque

Les blocs sous-programmes ne sont pas disponibles pour les API Série 90-20 et Micro.

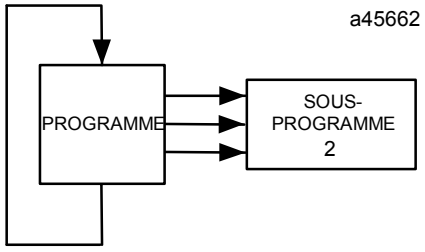
L'utilisation des sous-programmes est optionnelle. La division d'un programme en sous-programmes plus petits peut simplifier la programmation, améliorer la compréhension de l'algorithme de contrôle et optimise la taille du programme logique.

Exemples d'utilisation de blocs de sous-programmes

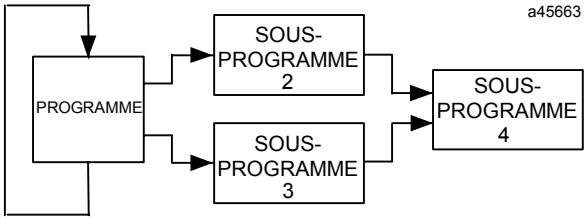
A titre d'exemple, la logique d'un programme est divisée en trois sous-programmes, chacun pouvant être appelé si nécessaire par le programme. Le programme principal contient une petite logique, servant uniquement à séquencer les appels de sous-programmes.



Un sous-programme peut être appelé à chaque fois que le programme s'exécute. Une logique répétitive peut être mise dans un bloc sous-programme qui devra être appelé à chaque fois que sa logique doit être exécutée. De cette façon, la taille totale du programme est réduite.



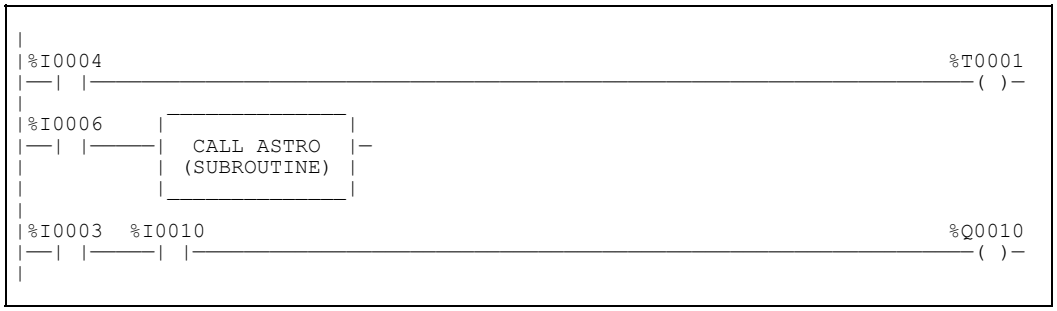
En plus d'être appelés par le programme, les sous-programmes peuvent également être appelés par d'autres sous-programmes. Un sous-programme ne peut s'appeler lui-même.



L'API autorisera huit appels imbriqués avant qu'un défaut “Dépassement de pile de l'application” ne soit enregistré et que l'API ne passe en mode **Stop/Défaut**. Le programme principal est considéré comme étant le niveau 1 dans le décompte des niveaux d’imbrication des appels.

Comment sont appelés les blocs

Un sous-programme s'exécute lorsqu'il est appelé par le programme principal ou par un autre bloc.



Cet exemple montre l'instruction CALL d'un sous-programme depuis le bloc appelant.

Sous-programmes périodiques

Les UC Version 4.20 ou supérieures du 340 supportent les sous-programmes périodiques. Veuillez noter les restrictions suivantes :

1. Les blocs fonctionnels de temporisation (TMR, ONDTR et OFDTR) ne s'exécuteront pas correctement dans un sous-programme périodique. Un bloc fonctionnel DOIO, dans un sous-programme périodique dont la plage de référence inclut les références affectées à un module Smart E/S (HSC, APM Power Mate, Genius, etc.), provoquera la perte de communication de l'UC avec le module. Les contacts FST_SCN et LST_SCN (%S1 et %S2) auront 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 latence d'un sous-programme périodique (c'est-à-dire, l'intervalle maximal entre le moment où le sous-programme périodique aurait dû s'exécuter et le moment où il s'exécute réellement) peut être d'environ 0,35 millisecondes s'il n'y a pas de module PCM, CMM ou ADC dans le châssis principal. S'il y a un module PCM, CMM ou ADC dans le châssis principal, même s'il n'est pas configuré ou utilisé, la latence peut atteindre 2,25 millisecondes. Pour cette raison, l'utilisation du sous-programme périodique avec des produits à base de PCM n'est ***pas*** recommandée.

Références utilisateur

Les données utilisées dans un programme d'application sont mémorisées comme références de registre ou logiques.

Table 2-4. Références de registre

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

Remarque

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

Table 2-5. Références logiques

Type	Description
%I	Le préfixe %I représente les références d'entrée TOR. 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 reçues des modules d'entrée après la dernière scrutation des entrées. Une adresse de référence est affectée aux modules d'entrées logiques en utilisant le logiciel de configuration ou la console de programmation de poche. A moins qu'une adresse de référence ne soit affectée, aucune donnée ne sera reçue du module. Les données %I peuvent être rémanentes ou non-rémanentes.
%Q	Le préfixe %Q représente les références de sortie TOR. La fonction de vérification de la bobine du logiciel Logicmaster 90-30/20/Micro vérifie s'il y a des utilisations multiples des références %Q avec les bobines de relais ou les sorties. A partir de la Version 3 du logiciel, vous pouvez sélectionner le niveau souhaité de vérification de la bobine (`SIMPLE`, `AVERTIR MULTIPLE` ou `MULTIPLE`). Se référer au Manuel de l'utilisateur du logiciel de programmation, GFK-0466, pour plus d'informations concernant cette fonction. 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 stocke l'état des références de sortie dernièrement établies par le programme d'application. Ces valeurs de la table d'état des sorties sont envoyées aux modules de sorties pendant la scrutation des sorties. Une adresse de référence est affectée aux modules de sorties logiques en utilisant le logiciel de configuration ou la console de programmation de poche. A moins qu'une adresse de référence soit affectée, aucune donnée ne sera envoyée au module. Une référence %Q particulière peut être rémanente ou non-rémanente. *
%M	Le préfixe %M représente les références internes. La fonction de vérification de bobine vérifie s'il y a des utilisations multiples des bobines de relais et des sorties. A partir de la Version 3 du logiciel, vous pouvez sélectionner le niveau souhaité de vérification de la bobine (`SIMPLE`, `AVERTIR MULTIPLE` ou `MULTIPLE`). Se référer à GFK-0466 pour plus d'informations concernant cette fonction. Une référence %M particulière peut être rémanente ou non-rémanente.*
%T	Le préfixe %T représente les références temporaires. Comme ces références ne sont jamais vérifiées quant à l'utilisation multiple de bobine, elles peuvent être utilisées de nombreuses fois dans le même programme, même lorsque la vérification d'utilisation de bobine est activée. %T peut être utilisé pour éviter des conflits d'utilisation des bobines tout en utilisant les fonctions couper/coller et écrire/inclure fichier. Comme cette mémoire est prévue pour une utilisation temporaire, elle n'est pas sauvegardée en cas de coupure d'alimentation ou de transitions **`RUN-A-STOP-A-RUN`** et ne peut pas être utilisée avec des bobines rémanentes.

* La rémanence est basée sur le type de bobine. Pour plus d'informations, se référer à “Rémanence des données” à la page 2-21.

Table 2-5. Références logiques - Suite

Type	Description
%S	Le préfixe %S représente les références d'état du système. Ces références sont utilisées pour accéder aux données API spéciales, comme les temporisateurs, les informations de scrutation et les informations de défauts. Les références du système incluent les références %S, %SA, %SB et %SC. %S, %SA, %SB et %SC peuvent être utilisés sur tous les contacts. %SA, %SB et %SC peuvent être utilisés sur les bobines rémanentes –(M)–. %S peut être utilisé comme argument d'entrée mot ou chaîne de bits pour les fonctions ou les blocs fonctionnels. %SA, %SB et %SC peuvent être utilisés comme argument d'entrée ou de sortie pour les fonctions et les blocs fonctionnels.
%G	Le préfixe %G représente les références de données globales. Ces références sont utilisées pour accéder aux données partagées par plusieurs API. Les références %G peuvent être utilisées sur les contacts et les bobines rémanentes car la mémoire %G est toujours rémanente. %G ne peut pas être utilisé sur des bobines non-rémanentes.

Transitions et forçages

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

Dans les UC des modèles 331 et ultérieurs, des bits de forçage peuvent être mis. Lorsque des bits de forçage sont mis, les références associées ne peuvent pas être modifiées par le programme ou le dispositif d'entrée ; ils ne peuvent être modifiés que par une commande de la console de programmation. Les UC des modèles 323, 321, 313 et 311, et les UC du Micro ne supportent pas le forçage des références logiques.

Rémanence des données

Les données sont dites rémanentes si elles sont sauvegardées par l'API lorsqu'il est arrêté. L'API Série 90 conserve la logique du programme, les tables de défauts et les diagnostics, les forçages, les données mot (%R, %AI, %AQ), les données bit (%I, %SC, %G, bits de défaut et bits réservés), les données %Q et %M (sauf si elles sont utilisées avec des bobines non-rémanentes) et les données mot stockées dans %Q et %M. Les données %T ne sont pas sauvegardées. Bien que, comme indiqué ci-dessus, les données bit %SC soient rémanentes, les défauts pour %S, %SA et %SB ne sont pas rémanents.

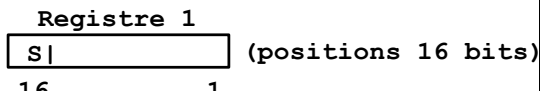
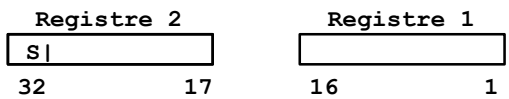
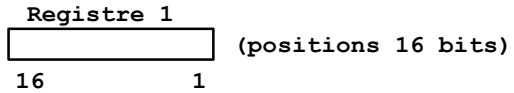
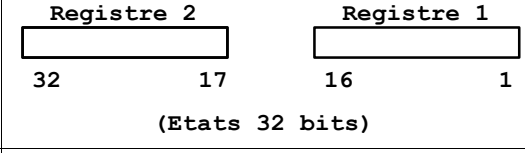
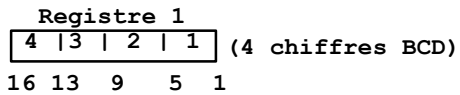
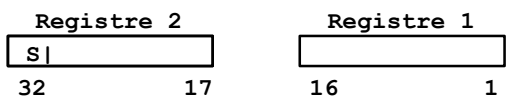
Les références %Q et %M ne sont pas rémanentes (c'est-à-dire, effacées à la mise sous tension lorsque l'API passe de **STOP** à **RUN**) si elles sont utilisées avec des bobines non-rémanentes. Les bobines non-rémanentes incluent les bobines —( )—, les bobines inversées —(/)—, les bobines SET —(S)— et les bobines RESET —(R)—.

Lorsque les références %Q ou %M sont utilisées avec des bobines rémanentes ou comme sorties de bloc fonctionnel, le contenu est sauvegardé en cas de coupure d'alimentation et de transitions **RUN-A-STOP-A-RUN**. Les bobines rémanentes incluent les bobines rémanentes —(M)—, les bobines rémanentes inversées —(/M)—, Les bobines SET rémanentes —(SM)— et les bobines RESET rémanentes —(RM)—.

La dernière fois qu'une référence %Q ou %M est programmée dans une instruction de bobine détermine si la référence %Q ou %M est rémanente ou non, selon le type de bobine. Par exemple, si %Q0001 a été programmé la dernière fois comme étant la référence d'une bobine rémanente, les données %Q0001 seront rémanentes. Cependant, si %Q0001 était programmé en dernier sur une bobine non-rémanente, les données %Q0001 seront non-rémanentes.

Types de données

Table 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 par la notation complément à 2. La plage valide des données de type INT est –32,768 à +32,767.	
DINT	Entier signé double précision	Les entiers signés double précision sont stockés dans un emplacement mémoire de 32 bits (actuellement deux emplacements mémoire consécutifs de 16 bits) et représentés en notation complément à 2. (Le bit 32 est le bit de signe). La plage valide des données de type DINT est – 2,147,483,648 à +2,147,483,647.	
BIT	Bit	Les données de type Bit sont les unités les plus petites de la mémoire. Elles ont deux états, 1 ou 0. Une chaîne BIT peut avoir une longueur N.	
BYTE	Octet	les données de type Octet ont une valeur de 8 bits. La plage valide est 0 à 255 (0 à FF en hexadécimal).	
WORD	Mot	Les données de type Mot utilisent 16 bits consécutifs de la mémoire de données ; mais à la place de bits dans l'emplacement de données représentant un nombre, les bits sont indépendants les uns des autres. Chaque bit représente son propre état binaire (1 ou 0) et les bits ne sont pas verrouillés ensembles pour représenter un nombre entier. La plage valide des valeurs Mot est 0 à FFFF.	
DWORD	Mot double	Les données de type Mot Double ont les mêmes caractéristiques que les données de type Mot, excepté qu'elles utilisent 32 bits consécutifs dans la mémoire au lieu de 16 bits.	
BCD-4	Décimal codé binaire à quatre chiffres	Les nombres BCD à quatre chiffres utilisent 16 bits d'emplacement mémoire. Chaque chiffre BCD utilise quatre bits et peut représenter des nombres compris entre 0 et 9. Cette codification BCD des 16 bits a une plage de valeurs valides de 0 à 9999.	
REAL	Virgule flottante	Les nombres réels utilisent 32 bits consécutifs (actuellement, deux emplacements mémoire de 16 bits consécutifs). La plage de nombres qui peuvent être mémorisés dans ce format est ± 1.401298E-45 à ± 3.402823E+38.	

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

Références d'état du système

Les références d'état du système de l'API Série 90 sont affectées aux mémoires %S, %SA, %SB et %SC. Elles ont chacune un symbole. Des références d'impulsions 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. Cependant, vous pouvez écrire dans les bits %SA, %SB et %SC.

Les références d'état du système listées ci-dessous sont disponibles et peuvent être utilisées dans un programme d'application. Lors de l'entrée de la logique, la référence ou le symbole peut être utilisé. Se référer au Chapitre 3, “Explications et correction des défauts”, pour des descriptions plus détaillées des défauts et des informations sur la correction des défauts.

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

Table 2-7. Références d'état du système

Référence	Symbole	Définition
%S0001	FST_SCN	Mis à 1 lorsque le cycle courant est le premier cycle.
%S0002	LST_SCN	Réinitialisé de 1 à 0 lorsque le cycle courant est le dernier cycle.
%S0003	T_10MS	Contact temporisé de 0,01 seconde.
%S0004	T_100MS	Contact temporisé de 0,1 seconde.
%S0005	T_SEC	Contact temporisé de 1,0 seconde.
%S0006	T_MIN	Contact temporisé de 1,0 minute.
%S0007	ALW_ON	Toujours à "1".
%S0008	ALW_OFF	Toujours à "0".
%S0009	SY_FULL	Mis à "1" lorsque la table des défauts de l'API est pleine. Remis à "0" lorsqu'une entrée est enlevée de la table des défauts de l'API et lorsque la table des défauts de l'API est effacée.
%S0010	IO_FULL	Mis à "1" lorsque la table des défauts d'E/S est pleine. Remis à "0" lorsqu'une entrée est enlevée de la table des défauts des E/S et lorsque la table des défauts des E/S est effacée.
%S0011	OVR_PRE	Mis à "1" lorsqu'un forçage existe dans la mémoire %I, %Q, %M ou %G.
%S0013	PRG_CHK	Mis à "1" lorsqu'une vérification de programme en arrière-plan est active.
%S0014	PLC_BAT	Mis à "1" pour indiquer une pile défectueuse dans l'UC Version 4 ou ultérieure. La référence de contact est rafraîchie une fois par cycle.

Table 2-7. Références d'état du système - Suite

Référence	Nom	Définition
%S0017	SNPXACT	L'hôte SNP-X est activement connecté à l'UC.
%S0018	SNPX_RD	L'hôte SNP-X a lu les données de l'UC.
%S0019	SNPX_WT	L'hôte SNP-X a écrit des données dans l'UC.
%S0020		Mis à "1" lorsqu'une fonction relationnelle, utilisant les données REAL, s'exécute correctement. Il est mis à "0" lorsqu'une des entrées n'est pas un nombre (NaN).
%S0032		Utilisation réservée 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 est dû à une défaillance temporaire, le bit logique peut être remis à "0" en mémorisant de nouveau le programme dans l'UC. Si le défaut était dû à une défaillance matérielle de la RAM, l'UC doit être remplacée.
%SA0002	OV_SWP	Mis à "1" lorsque l'API détecte que le cycle précédent a mis plus de temps que spécifié par l'utilisateur. Remis à "0" lorsque l'API détecte que le cycle précédent n'a pas été plus long que le temps spécifié. Il est également remis à "0" pendant la transition du mode **STOP** au mode **RUN**. Valide seulement si l'API est en mode **CYCLE CONSTANT**.
%SA0003	APL_FLT	Mis à "1" lorsqu'un défaut d'application se produit. Remis à "0" lorsque l'API passe du mode**STOP** au mode **RUN**.
%SA0009	CFG_MM	Mis à "1" lorsqu'une incohérence de configuration est détectée pendant la mise sous tension du système ou pendant la mémorisation de la configuration. Remis à "0" par la mise sous tension de l'API lorsqu'il y a correspondance ou pendant la mémorisation de la configuration qui correspond au matériel.
%SA0010	HRD_CPU	Mis à "1" lorsque les diagnostics détectent un problème matériel avec l'UC. Remis à "0" en remplaçant le module de l'UC.
%SA0011	LOW_BAT	Mis à "1" lorsqu'un défaut de pile faible se produit. Remis à "0" en remplaçant la pile et en s'assurant que l'API se met sous tension sans la condition de pile faible.
%SA0014	LOS_IOM	Mis à "1" lorsqu'un module d'E/S arrête de communique avec l'UC de l'API. Remis à "0" en remplaçant le module et en remettant le châssis principal sous tension.
%SA0015	LOS_SIO	Mis à "1" lorsqu'un module d'option arrête de communiquer avec l'UC de l'API. Remis à "0" en remplaçant le module et en remettant le châssis principal sous tension.
%SA0019	ADD_IOM	Mis à "1" lorsqu'un module d'E/S est ajouté à un châssis. Remis à "0" en remettant le châssis principal sous tension et lorsque la configuration correspond au matériel après une mémorisation.
%SA0020	ADD_SIO	Mis à "1" lorsqu'un module d'option est ajouté à un châssis. Remis à "0" en remettant le châssis principal sous tension et lorsque la configuration correspond au matériel après une mémorisation.
%SA0027	HRD_SIO	Mis à "1" lorsqu'une défaillance matérielle est détectée dans un module d'option. Remis à "0" en remplaçant le module et en remettant le châssis principal sous tension.
%SA0031	SFT_SIO	Mis à "1" lorsqu'un défaut logiciel irrécupérable est détecté dans un module d'option. Remis à "0" en remettant le châssis principal sous tension et lorsque la configuration correspond au matériel.
%SB0010	BAD_RAM	Mis à "1" lorsque l'UC détecte une mémoire RAM corrompue à la mise sous tension. Remis à "0" lorsque l'UC détecte que la mémoire RAM est valide après la mise sous tension.

Table 2-7. Références d'état du 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 effacée.
%SB0013	SFT_CPU	Mis à "1" lorsque l'UC détecte une erreur irrémédiable dans le logiciel. Remis à "0" en effaçant la table des défauts de l'API.
%SB0014	STOR_ER	Mis à "1" lorsqu'une erreur se produit pendant une opération de mémorisation dans la console de programmation. Remis à "0" lorsque l'opération de mémorisation s'est terminée correctement.
%SC0009	ANY_FLT	Mis à "1" lorsqu'un défaut se produit. Remis à "0" lorsque les deux tables de défauts sont vides
%SC0010	SY_FLT	Mis à "1" lorsqu'un défaut se produit, provoquant le placement d'une entrée dans la table de défauts de l'API. Remis à "0" lorsque la table de défauts de l'API est vide (aucune entrée).
%SC0011	IO_FLT	Mis à "1" lorsqu'un défaut se produit, provoquant le placement d'une entrée dans la table de défauts des E/S. Remis à "0" lorsque la table de défauts des E/S est vide.
%SC0012	SY_PRES	Mis à "1" tant qu'il y a au moins une entrée dans la table des défauts de l'API. Remis à "0" lorsque la table de défauts de l'API est vide (aucune entrée).
%SC0013	IO_PRES	Mis à "1" tant qu'il y a au moins une entrée dans la table des défauts des E/S. Remis à "0" lorsque la table de défauts des E/S est vide (aucune entrée).
%SC0014	HRD_FLT	Mis à "1" lorsqu'un défaut matériel se produit. Remis à "0" lorsque les deux tables de défauts sont vides
%SC0015	SFT_FLT	Mis à "1" lorsqu'un défaut logiciel se produit. Remis à "0" lorsque les deux tables de défauts sont vides

Remarque : Toute référence %S non listée ici est réservée et ne doit pas être utilisée dans le programme logique.

Structure des blocs fonctionnels

Chaque circuit logique se compose d'une ou plusieurs instructions de programmation. Il peut s'agir d'un simple relais ou de fonctions plus complexes.

Format des relais de la logique en échelle

Le logiciel de programmation comprend plusieurs types de fonctions relais. Ces fonctions fournissent le flux et le contrôle de base de la logique d'un programme. Les exemples ci-dessous montrent un contact et une bobine normalement ouverts. Chacun de ces contacts et bobines possède une entrée et une sortie laissant passer le flux logique.

Chaque contact ou bobine doit avoir une référence automate qui lui soit affectée lors de la sélection du relais. Pour un contact, la référence représente un emplacement de la mémoire qui détermine le passage de l'énergie dans le contact. Dans l'exemple suivant, si la référence %I0122 est mise à "1", l'énergie circulera à travers ce contact.

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

Pour une bobine, la référence représente un emplacement de la mémoire qui est contrôlé par le flux d'énergie dans la bobine. Dans cet exemple, si l'énergie entre dans le côté gauche de la bobine, la référence %Q0004 mise à "1".

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

Le logiciel de programmation et la console de programmation de poche disposent tous deux d'une fonction de vérification de bobine qui vérifient les utilisations multiples des références %Q ou %M avec les bobines ou les fonctions de sortie.

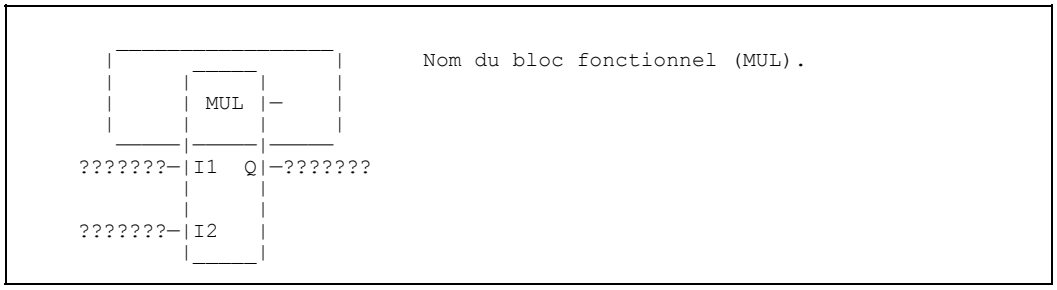
Format des blocs fonctionnels du programme

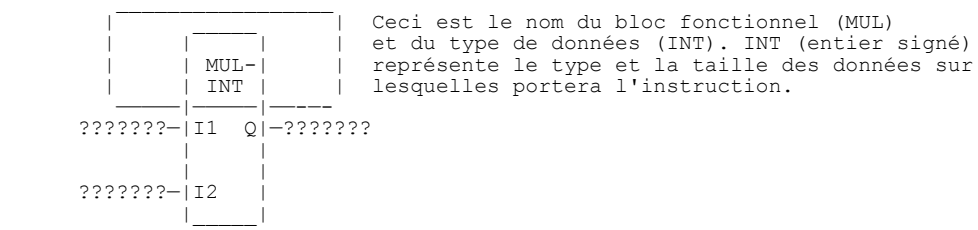
Certaines fonctions sont très simples, comme la fonction MCR qui est montrée avec l'abréviation du nom de la fonction entre crochets :

```
-[ MCR ]-
```

D'autres fonctions sont plus complexes. Elles peuvent avoir plusieurs paramètres à renseigner, indispensables à l'éxécution de la fonction.

Le bloc fonctionnel générique illustré ci-dessous est une multiplication (MUL) ; les paramètres varient avec le type de bloc fonctionnel. Les parties qui le composent sont typiques de nombreuses fonctions de programmation. La partie supérieure du bloc fonctionnel indique le nom de la fonction. Elle peut également indiquer un format de données ; dans ce cas, il s'agit d'un entier signé.



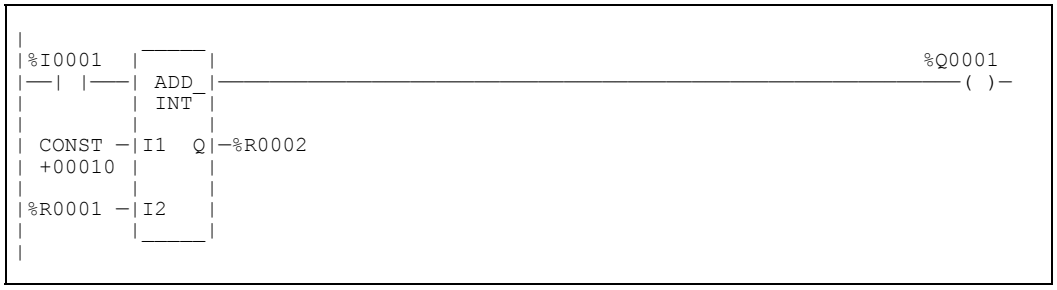


De nombreuses fonctions du programme vous permettent de choisir le type de données pour la fonction, après la sélection de la fonction. Pour cet exemple, le type de données de la fonction MUL pourrait être changé en entier signé double précision. Des informations supplémentaires sur les types de données sont fournies au début de ce chapitre.

Paramètres des blocs fonctionnels

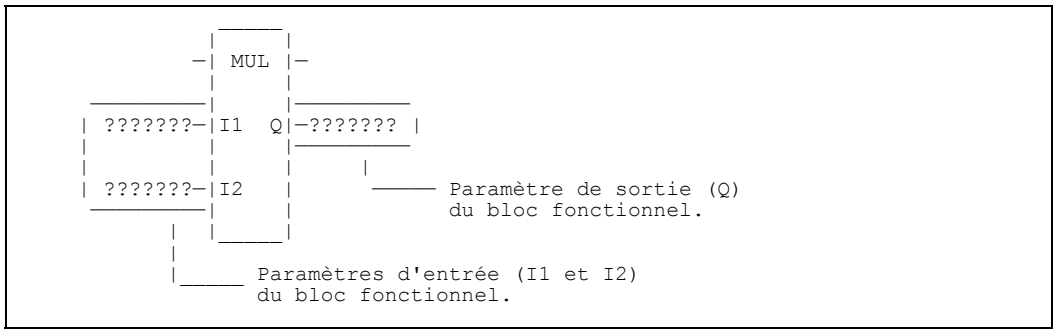
Chaque ligne entrant à gauche d'un bloc fonctionnel représente une entrée pour cette fonction. Il y a deux formes d'entrée qui peuvent passer dans un bloc fonctionnel : les constantes et les références. Une constante est une valeur explicite. Une référence est l'adresse d'une valeur.

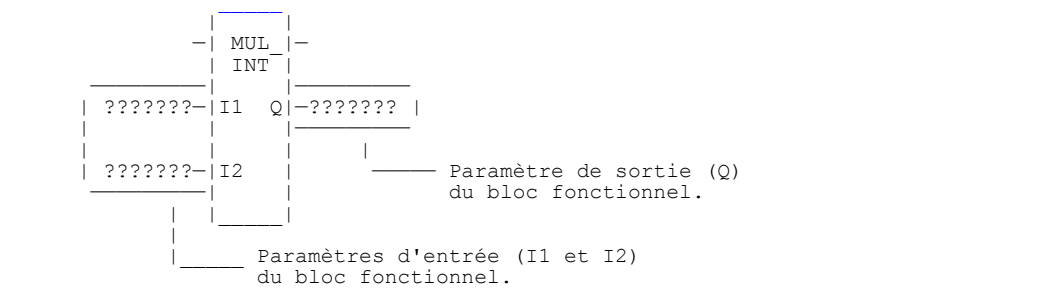
Dans l'exemple suivant, le paramètre d'entrée 11 du bloc fonctionnel ADD est une constante, et le paramètre d'entrée I2 est une référence.



Chaque ligne sortant à droite du bloc fonctionnel représente une sortie. Il n'y a qu'une forme de sortie provenant d'un bloc fonctionnel ou d'une référence. Les sorties ne peuvent jamais être des constantes.

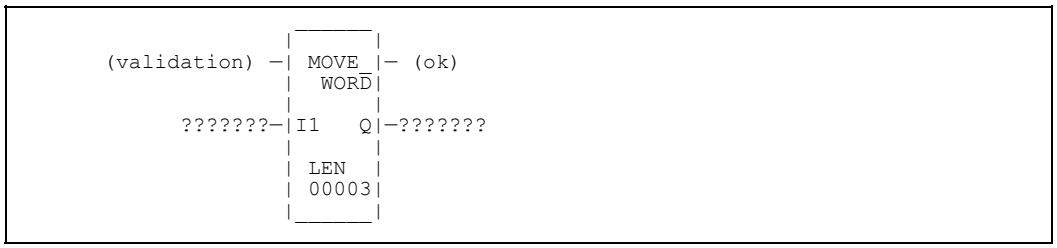
Lorsqu'un point d'interrogation apparaît à gauche d'un bloc fonctionnel, vous entrerez soit les données elles-mêmes, soit une référence où se trouvent les données, soit une variable représentant la référence où se trouvent les données. Lorsqu'un point d'interrogation apparaît à droite d'un bloc fonctionnel, il faudra saisir une référence ou une variable représentant l'emplacement de référence pour les données produites par le bloc fonctionnel.



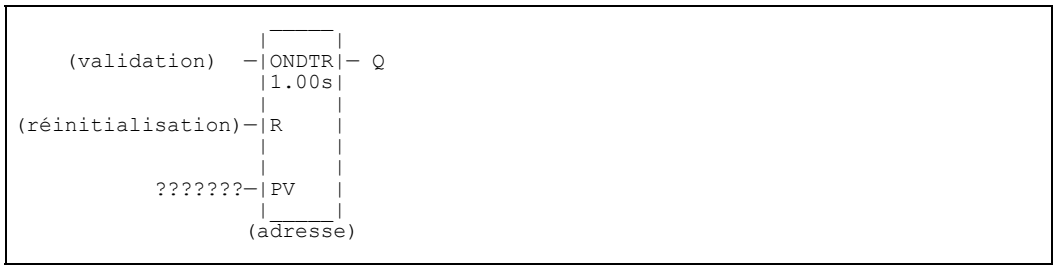


La plupart des blocs fonctionnels ne modifient pas les données d'entrée et placent le résultat de l'opération dans une référence de sortie.

Pour les fonctions utilisant les tables, une longueur peut être sélectionnée pour la fonction. Dans la fonction suivante, le paramètre LEN spécifie le nombre de mots à déplacer.

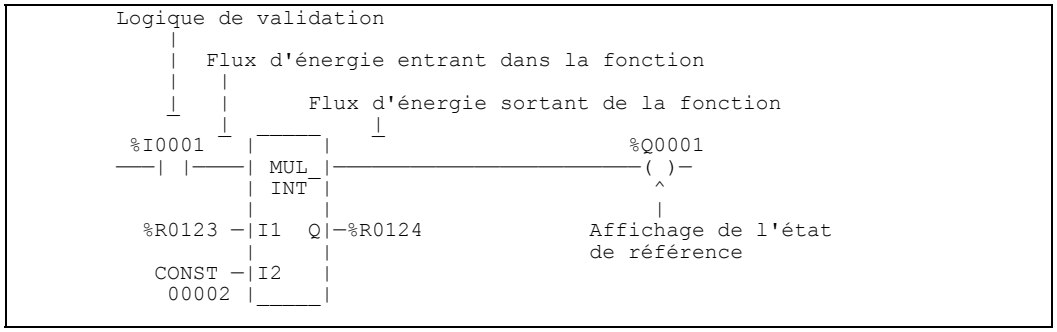


Les fonctions temporisateur, compteur, BITSEQ et ID nécessitent l'affectation de trois registres pour stocker la valeur courante, la valeur de pré-sélection et un mot de contrôle.



Flux d'énergie entrant et sortant d'une fonction

L'énergie entre dans un bloc fonctionnel en haut, à gauche. Souvent, une logique de validation est utilisée pour contrôler le flux d'énergie vers un bloc fonctionnel ; à défaut, le bloc fonctionnel est éxécuté de façon inconditionnelle à chaque cycle de l'UC.



Remarque

Les blocs fonctionnels ne peuvent être reliés directement au rail d'énergie gauche. Vous pouvez utiliser %S7, le bit ALW_ON (toujours à "1") avec un contact normalement ouvert relié au rail d'énergie pour appeler une fonction à chaque cycle.

Le flux d'énergie sort du bloc fonctionnel en haut, à droite. Il peut être transmis à une autre logique ou à une bobine (optionnel). Les blocs fonctionnels laissent passer l'énergie lorsqu'ils s'exécutent correctement.

Section 3 : Séquences de mise sous/hors tension

Il existe deux séquences de mise sous tension possibles pour l'API Série 90-30 ; la mise sous tension à froid et la mise sous tension à chaud. L'UC utilise habituellement la séquence de mise sous-tension à froid. Cependant, pour un API Modèle 331 ou supérieur, si le temps écoulé entre la mise hors tension et la mise sous tension est inférieur à cinq secondes, la séquence de mise sous tension à chaud est utilisée.

Mise sous tension

Une mise sous tension à froid se compose de la séquence d'évènements suivante. Une séquence de mise sous tension à chaud ignore l'étape 1.

1. L'UC effectuera des diagnostics internes. Ceci comprend la vérification d'une partie des RAM protégées par la pile afin de déterminer si les RAM contiennent ou non des données valides.
2. Si une EPROM, EEPROM ou une mémoire Flash est présente et que la configuration de mise sous tension indique que le contenu des PROM doit être utilisé, le contenu des PROM est copié dans la mémoire RAM. Si aucune EPROM, EEPROM ou mémoire Flash n'est présente, le contenu de la mémoire RAM n'est pas modifié.
3. L'UC interroge chaque emplacement du système pour déterminer quelles sont les cartes présentes.
4. La configuration matérielle est comparée à la configuration logicielle afin de vérifier qu'elles sont identiques. Toute incohérence est considérée comme étant un défaut et génère une alarme. Egalement, si une carte est spécifiée dans la configuration logicielle, mais qu'un module différent est présent dans la configuration matérielle réelle, , une alarme est générée.
5. S'il n'y a pas de configuration logicielle, l'UC utilisera la configuration par défaut.
6. L'UC établit le canal de communication vers tous les modules intelligents.
7. Dans l'étape finale de l'exécution, le mode du premier cycle est déterminé par rapport à la configuration de l'UC. En cas de mode **RUN**, le cycle se déroule comme décrit dans la "Transition de mode de**STOP**-à-**RUN**". La Figure 2-5 de la page suivante montre la séquence de décision de l'UC lorsqu'elle décide de copier les PROM ou de se mettre sous tension en mode **STOP** ou **RUN**.

Remarque

Les étapes 2 à 7 ci-dessus ne s'appliquent pas à l'API Série 90 Micro. Pour des informations concernant les séquences de mise sous/hors tension du Micro, se référer à la section "Séquences de mise sous/hors tension" du Chapitre 5, "Fonctionnement du système", dans le *Manuel de l'utilisateur de l'API Série 90 Micro* (GFK-1065).

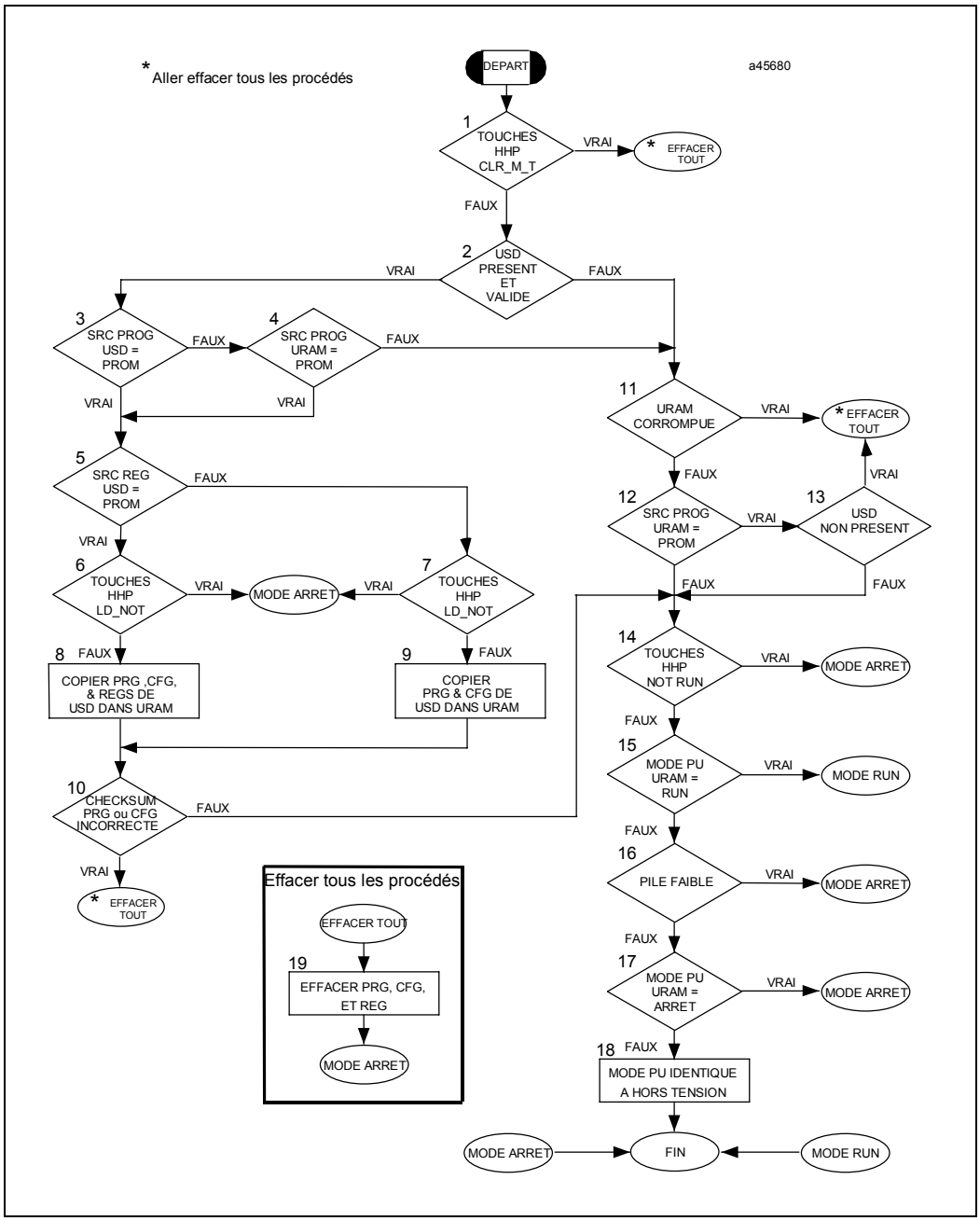


Figure 2-5. Séquence de mise sous tension

Avant l'instruction START de l'organigramme de mise sous tension, l'UC effectue les diagnostics de mise sous tension, qui testent les divers périphériques utilisés par l'UC, et teste les RAM. A la fin des diagnostics, les structures des données internes et les périphériques utilisés par l'UC sont initialisés. L'UC détermine alors si la RAM utilisateur a été corrompue. Si la RAM utilisateur est corrompue, le programme et la configuration de l'utilisateur sont effacés, remplacés par les valeurs par défaut, et tous les registres utilisateur sont effacés.

TERMES DE L'ORGANIGRAMME :

PRG = programme utilisateur

CFG = configuration utilisateur

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

USD = dispositif de stockage utilisateur, soit une EEPROM, soit une mémoire Flash.

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

TEXTE DE L'ORGANIGRAMME :

(1) Les touches <CLR> et <M_T> ont-elles été appuyées sur la HHP pendant la mise sous tension pour effacer toute l'URAM ?

(2) L'USD est-il présent (ne peut manquer que sur les modèles utilisant des EEPROM) et les informations sur USD sont-elles valides ?

(3) Le paramètre PRG SRC de l'USD est-il mis sur Prom, signifiant charger PRG et CFG à partir de l'USD ?

(4) Le paramètre PRG SRC de l'URAM est-il mis sur Prom, signifiant charger PRG et CFG à partir de l'USD ?

(5) Le paramètre REG SRC de l'USD est-il mis sur Prom, signifiant charger REGS à partir de l'USD ?

(6 & 7) Les touches <LD> et <NOT> ont-elles été appuyées sur la HHP pendant la mise sous tension pour éviter de charger PRG, CFG et REGS à partir de l'USD ?

(8) Copier PRG, CFG et REGS à partir de l'USD dans l'URAM.

(9) Copier PRG et CFG à partir de l'USD dans l'URAM.

(10) La checksum de PRG ou CFG, chargée de USD, est-elle invalide ?

(11) L'URAM est-elle corrompue ? Pourrait être provoqué par la mise hors tension sans pile ou avec une pile faible. Pourrait également être dû à la mise à jour du macro-programme.

(12) Le paramètre PRG SRC de l'URAM est-il mis sur Prom, signifiant charger PRG et CFG à partir de l'USD ?

(13) L'USD est-il présent ? Ne s'applique qu'aux modèles utilisant une EEPROM.

(14) Les touches <**NOT**> et <**RUN**> ont-elles été appuyées sur la HHP pendant la mise sous tension pour mettre sous tension inconditionnellement en mode Stop ?

(15) Le paramètre PWR UP de l'URAM est-il réglé sur **RUN**?

(16) La pile est-elle faible ?

(17) Le paramètre PWR UP de l'URAM est-il réglé sur **STOP**?

(18) Mettre le mode mise sous tension comme le mode hors tension, quel qu'il fut.

(19) Effacer PRG, CFG et REGS.

Remarque

La première partie du tableau de la page précédente ne s'applique pas à l'API Série 90 Micro. Pour des informations concernant les séquences de mise sous/hors tension du Micro, se référer à la section “Séquences de mise sous/hors tension” du Chapitre 5, “Fonctionnement du système”, dans le *Manuel de l'utilisateur de l'API Série 90 Micro* (GFK-1065).

Mise hors tension

La mise hors tension du système se produit lorsque l'alimentation détecte une chute de l'alimentation secteur CA pendant un certain temps ou que la sortie de l'alimentation 5 volts est tombée à moins de 4,9 volts CC.

Section 4 : Horloges et temporisateurs

Les horloges et les temporisateurs de l'API Série 90-30 incluent une horloge de temps écoulé, une horloge de date et heure (Modèles 331, 340/341, 351/352 et Micro 28 points), une temporisation chien de garde et une temporisation de cycle constant. Trois types de blocs fonctionnels de temporisation comprennent un temporisateur de retard à l'activation, un temporisateur de retard à la désactivation et un temporisateur de retard à l'activation rémanent ,également appelé temporisateur d'horloge de garde. Quatre contacts temporisés changeant d'état par intervalles de 0,01 seconde, 0,1 seconde, 1,0 seconde et 1 minute sont également disponibles.

Horloge de temps écoulé

L'horloge de temps écoulé utilise des "impulsions" de 100 microsecondes pour suivre le temps écoulé depuis la mise sous tension de l'UC. L'horloge n'est pas rémanente en cas de coupure d'alimentation ; Elle redémarre à chaque mise sous tension. Chaque seconde, le matériel interrompt l'UC pour permettre l'enregistrement du comptage des secondes. Ce comptage des secondes repart à zéro environ 100 ans après que l'horloge commence le comptage.

Comme l'horloge de temps écoulé constitue la base de fonctionnement du logiciel d'exploitation et des blocs fonctionnels de temporisation, elle ne peut pas être réinitialisée par le programme utilisateur ou la console de programmation. Cependant, le programme d'application peut lire la valeur courante de l'horloge de temps écoulé en utilisant la requête de service 16.

Horodateur

La date et l'heure des API Micro 28 points et Série 90-30, modèle 331 et supérieur, sont maintenues par une horloge matérielle. L'horodateur assure sept fonctions de temps :

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

L'horodateur est maintenu par une pile et conserve son état présent en cas de coupure d'alimentation. Cependant, à moins d'initialiser l'horloge, les valeurs qu'il contient sont sans signification. Le programme d'application peut lire et régler l'horodateur en utilisant la requête de service #7. L'horodateur peut également être lu et réglé à partir du logiciel de configuration de l'UC. Noter que la console de programmation de poche ne vous autorise pas à modifier l'horodateur lorsque la protection par interrupteur à clé est active.

L'horodateur est conçu pour gérer les transitions d'un mois à l'autre et d'une année sur l'autre. Il tient également compte des années bissextiles jusqu'à l'année 2079.

Temporisateur chien de garde

Le temporisateur chien de garde de l'API Série 90-30 est conçu pour anticiper les conditions de pannes extrêmes qui peuvent conduire à un temps de cycle anormalement long. La valeur de la temporisation du chien de garde est de 200 millisecondes (500 millisecondes pour les UC des API modèles 35x et 36x) ; ceci est une valeur fixe qui ne peut pas être modifiée. Le temporisateur chien de garde part toujours de zéro au début de chaque cycle.

Pour les UC du 331 et des modèles antérieurs 90-30, si le chien de garde est dépassé, la LED OK s'éteint ; l'UC est réinitialisée et coupée complètement ; les sorties prennent les états par défaut. Aucune communication, sous quelle que forme que ce soit, n'est possible et tous les microprocesseurs de toutes les cartes sont arrêtés. Pour la restauration, l'alimentation doit être rétablie sur le châssis principal. Dans les UC 90-20, Série 90 Micro, 340 et supérieurs, 90-30, un dépassement du chien de garde provoque la réinitialisation de l'UC, l'exécution de sa séquence de mise sous tension, la génération d'un défaut de chien de garde et le passage de son mode en **STOP**.

Temporisateur de temps de coupure écoulé

Le temporisateur de temps de coupure écoulé est utilisé pour déterminer la durée pendant laquelle l'API est resté hors tension. Lorsque l'API est mis hors tension, il se réinitialise à 0 et commence à minuter. Lorsque l'API est mis sous tension, le minutage s'arrête et la valeur est retenue. La requête de service #29, décrite dans le Chapitre 12, peut être utilisée pour lire la valeur de ce temporisateur.

Remarque

Cette fonction n'est disponible que pour les UC 331 ou supérieures de la Série 90-30.

Temporisateur de cycle constant

Le temporisateur de cycle constant contrôle la longueur du cycle d'un programme lorsque l'API Série 90-30 fonctionne en mode **Temps de cycle constant**. Dans ce mode de fonctionnement, chaque cycle utilise la même durée. Typiquement, pour la plupart des programmes d'application, la lecture des entrées, la scrutation du programme logique et l'écriture des sorties ne nécessitent pas la même durée d'exécution à chaque cycle. La valeur du temporisateur de cycle constant est réglée par la console de programmation et peut être toute valeur comprise entre 5 et la valeur du temporisateur du chien de garde (dont la valeur par défaut est de 100 millisecondes).

Si la temporisation de cycle constant expire avant la fin du cycle et que le cycle précédent n'a pas dépassé le temps alloué, l'API place une alarme de surcycle dans la table des défauts de l'API. Au début du cycle suivant, l'API met le contact de défaut OV_SWP à "1". Le contact OV_SWP est réinitialisé lorsque l'API n'est pas en mode **TEMPS DE CYCLE CONSTANT** ou lorsque le dernier cycle n'excède pas la temporisation de cycle constant.

Contacts d'impulsions d'horloge

L'API Série 90 fournit quatre contacts d'impulsions d'horloge ayant une durée de 0,01 seconde, 0,1 seconde, 1,0 seconde, et 1 minute. L'état de ces contacts ne change pas pendant l'exécution du cycle. Ces contacts fournissent des impulsions d'une durée de marche et d'arrêt égale. Les contacts sont référencés T_10MS (0,01 seconde), T_100MS (0,1 seconde), T_SEC (1,0 seconde) et T_MIN (1 minute).

Le chronogramme suivant représente la durée de marche/arrêt de ces contacts.

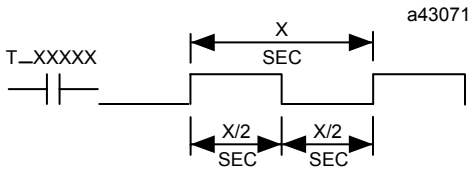


Figure 2-6. Chronogramme des contacts d'impulsions d'horloge

Section 5 : Sécurité du système

La sécurité des API Série 90-30, Série 90-20 et Micro permet d' éviter des modifications non-autorisées du contenu d'un API. Il existe quatre niveaux de sécurité dans l'API. Le premier niveau, qui est toujours disponible, donne seulement la possibilité de lire les données de l'API ; aucune modification de l'application n'est permise. Les trois autres niveaux ont un accès protégé par un mot de passe.

Chaque niveau de privilège supérieur permet davantage de possibilités de modification que le(s) niveau(x) inférieur(s). Les niveaux de privilège se cumulent, en ce sens que le privilège accordé à un niveau constitue une combinaison de ce niveau, plus tous les niveaux inférieurs. Les niveaux et leurs privilèges sont :

Niveau de privilège	Description
Niveau 1	Toutes les données, sauf les mots de passe, peuvent être lues. Ceci comprend toutes les mémoires de données (%I, %Q, %AQ, %R, etc.), les tables de défauts et tous les types de blocs programme (données, valeurs et constantes). Aucune valeur ne peut être modifiée dans l'API.
Niveau 2	Ce niveau permet d'écrire dans les mémoires de données (%I, %R, etc.).
Niveau 3	Ce niveau permet d'écrire dans le programme d'application en mode **STOP** seulement.
Niveau 4	Ceci est le niveau par défaut des systèmes qui n'ont pas de mot de passe établi. Le niveau par défaut d'un système avec mot de passe est le niveau le plus haut non-protégé. Ce niveau, le plus haut, permet la lecture et l'écriture de toutes les mémoires dans les deux 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 par privilège dans l'API. (Aucun mot de passe ne peut être établi pour accéder au niveau 1). Chaque mot de passe peut être unique ; cependant, le même mot de passe peut être utilisé pour plusieurs niveaux. La longueur des mots de passe est de un à quatre caractères ASCII ; ils ne peuvent être introduits ou modifiés que par le logiciel de programmation ou la console de programmation de poche.

Un changement de niveau de privilège n'a d'effet que tant que la communication, entre l'API et la console de programmation, est établie. Il n'est pas nécessaire qu'il y ait une activité quelconque, mais la liaison de communication ne doit pas être interrompue. S'il n'y a pas de communication pendant 15 minutes, le niveau de privilège revient au plus haut niveau non-protégé.

L'API étant connecté, le logiciel de programmation indique l'état de la protection de chaque niveau de privilège à l'API. Le logiciel de programmation met ensuite l'API au plus haut niveau non-protégé, donnant ainsi accès au niveau le plus haut non-protégé sans avoir à demander un niveau particulier. Lorsque la console de programmation de poche est connectée à l'API, ce dernier retourne au plus haut niveau non-protégé.

Demandes de changement de niveau de privilège

La console de programmation permet un changement de niveau de privilège en fournissant le mot de passe pour le niveau de privilège souhaité. Le changement de niveau de privilège est refusé si le mot de passe ne correspond pas au mot de passe mémorisé dans la table des mots de passe d'accès de l'API pour le niveau demandé. Le niveau de privilège courant est maintenu et aucune modification ne se produira. Si vous essayez d'accéder ou de modifier des informations dans l'API en utilisant la console de programmation de poche sans le niveau de privilège correct, la console de programmation de poche renverra un message d'erreur indiquant que l'accès est refusé.

Verrouillage/déverrouillage de sous-programmes

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

Type de verrou	Description
Visualisation	Une fois verrouillé, vous ne pouvez pas visualiser ce sous-programme.
Edition	Une fois verrouillé, les informations de ce sous-programme ne peuvent pas être éditées.

Un sous-programme dont la visualisation ou l'édition a été verrouillée peut être déverrouillé avec l'éditeur de déclaration de bloc sauf si la visualisation ou l'édition a été verrouillée en permanence.

Une fonction de recherche ou de recherche et remplacement peut être effectuée sur le sous-programme à visualisation verrouillée. Si la cible de la recherche est trouvée dans un sous-programme à visualisation verrouillée, l'un des messages suivants est affiché à la place de la logique :

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

ou

```
Impossible d'écrire dans un bloc verrouillé <nom_bloc> (Continuer/Quitter)
```

Vous pouvez continuer ou abandonner la recherche.

Les dossiers contenant des sous-programmes verrouillés peuvent être vidés ou supprimés. Si un dossier contient des sous-programmes verrouillés, ces blocs restent verrouillés lors de l'éxécution des fonctions Copier, Sauvegarder et Restaurer le logiciel de programmation .

Verrouillage permanent d'un sous-programme

En plus des verrouillages de visualisation et d'édition, il existe deux types de verrouillages permanents. Si un VERROUILLAGE DE VISUALISATION PERMANENT est demandé, plus aucune visualisation de ce sous-programme n'est posible. Si un VERROUILLAGE D'EDITION PERMANENT est mis en place, toutes les tentatives d'édition du bloc sont refusées.

Précautions

Les verrous permanents diffèrent des VERROU DE VISUALISATION et VERROU D'EDITION normaux en ce sens qu'une fois établis, ils ne peuvent plus être enlevés.

Lorsqu'un VERROU D'EDITION PERMANENT est établi, il ne peut être changé qu'en VERROU DE VISUALISATION PERMANENT. Un VERROU DE VISUALISATION PERMANENT ne peut pas être changé en un autre type de verrou.

Section 6 : Système d'E/S, Série 90-30, 90-20 *et Micro*

Le système d'E/S de l'API sert d'interface entre l'API Série 90-30 et les dispositifs et équipements fournis par l'utilisateur. Les E/S de la Série 90-30 sont appelés E/S Série 90-30. Les modules d'E/S Série 90-30 s'enfichent directement dans les emplacements du châssis principal ou des châssis d'extension de l'API Série 90-30, Modèle 331 ou supérieur. Les systèmes d'E/S des modèles 331, 340 et 341 supportent jusqu'à 49 modules d'E/S (5 châssis). Les systèmes d'E/S des modèles 351 et 352 supportent jusqu'à 79 modules d'E/S (8 châssis). Le châssis à 5 emplacements de l'API Série 90-30, Modèle 311 ou Modèle 313 supporte jusqu'à 5 modules d'E/S; Le châssis à 10 emplacements du Modèle 323 supporte jusqu'à 10 modules d'E/S.

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

Système d'E/S de l'API

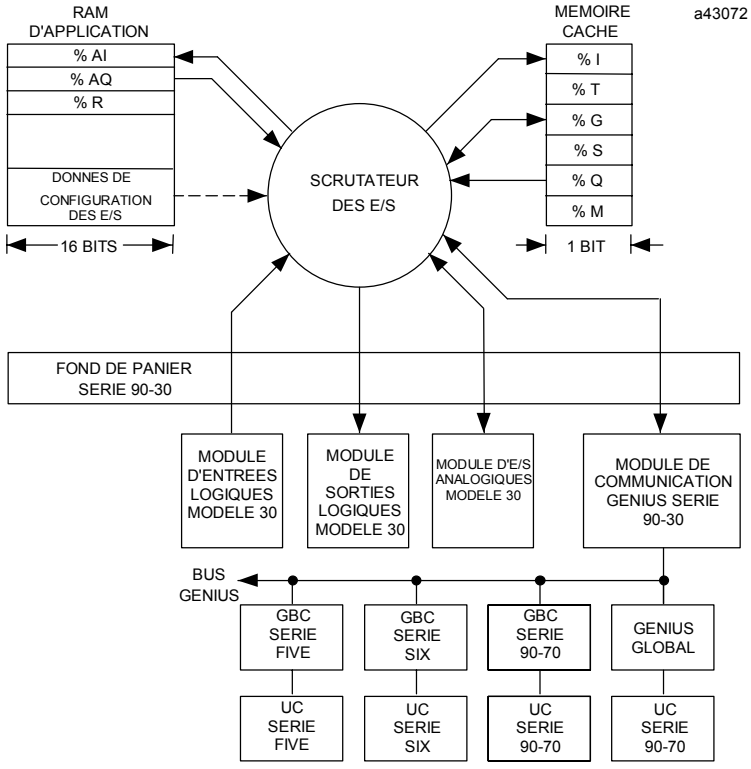


Figure 2-7. Structure des E/S Série 90-30

Remarque

Le schéma montré ci-dessus est spécifique à la structure des E/S du 90-30. Les modules intelligents et d'options ne font partie de la scrutation des E/S ; ils utilisent la fenêtre de communication du système. Pour des informations concernant la structure des E/S 90-20, se référer au *Manuel de l'utilisateur de l'automate programmable Série 90™-20* (GFK-0551). Pour des informations concernant la structure des E/S de l'API Micro, se référer au *Manuel de l'utilisateur de l'API Série 90™Micro* (GFK-1065).

Modules d'E/S Série 90-30

Les modules d'E/S Série 90-30 sont disponibles en cinq types : module d'entrées logiques, de sorties logiques, d'entrées analogiques, de sorties analogiques et d'option. La table suivante liste les modules d'E/S Série 90-30 par numéro de référence, par nombre de points d'E/S et une brève description de chaque module.

Remarque

Tous les modules d'E/S listés ci-dessous ne sont peut-être pas disponibles au moment où ce manuel est imprimé. Pour la disponibilité, consultez votre distributeur d'API GE Fanuc local ou le représentant commercial de GE Fanuc. Se référer au *Manuel des caractéristiques de chaque module d'E/S Série 90-30*, GFK-0898, pour les caractéristiques et les informations de câblage de chaque module d'E/S Série 90-30.

Figure 2-8. Modules d'E/S Série 90-30

Référence produit	Points	Description	Référence manuel
		Modules d'entrées logiques	
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 VCC, 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	24 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

Table 2-8. Modules d'E/S Série 90-30 - Suite

Référence produit	Points	Description	Référence manuel
		Modules de sorties logiques	
IC693MDL310	12	120 VCA, 0,5 A	GFK-0898
IC693MDL330	8	120/240 VCA, 2 A	GFK-0898
IC693MDL340	16	120 VCA, 0,5 A	GFK-0898
IC693MDL390	5	120/240 VCA, isolé, 2 A	GFK-0898
IC693MDL730	8	12/24 VCC logique positive, 2A	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, 0,3 A	GFK-0898
IC693MDL752	32	5/24 VCC (TTL), logique négative, 0,5 A	GFK-0898
IC693MDL753	32	12/24 VCC, logique positive/négative, 0.5 A	GFK-0898
IC693MDL930	8	Relais, normalement ouvert, 4 A, isolé	GFK-0898
IC693MDL931	8	Relais, BC, isolé	GFK-0898
IC693MDL940	16	Relais, normalement ouvert, 2 A	GFK-0898
		Modules d'entrées/sorties	
IC693MDR390	8/8	Entrée 24 VCC, sortie relais	GFK-0898
IC693MAR590	8/8	Entrée 120 VCC, sortie relais	GFK-0898
		Modules analogiques	
IC693ALG220	4 voies	Entrée analogique, tension	GFK-0898
IC693ALG221	4 voies	Entrée analogique, courant	GFK-0898
IC693ALG222	16	Entrée analogique, tension	GFK-0898
IC693ALG223	16	Entrée analogique, courant	GFK-0898
IC693ALG390	2 voies	Sortie analogique, tension	GFK-0898
IC693ALG391	2 voies	Sortie analogique, courant	GFK-0898
IC693ALG392	8 voies	Sortie analogique, courant/tension	GFK-0898
IC693ALG442	4/2	Analogique, combinaison d'entrées/sorties courant/tension	GFK-0898

Table 2-8. Modules d'E/S Série 90-30 - Suite

Référence produit	Description	Référence manuel
	Modules d'options	
IC693APU300	Compteur rapide	GFK-0293
IC693APU301	Module APM Power Mate, 1 axe, mode suiveur	GFK-0781
IC693APU301	Module APM Power Mate, 1 axe, mode standard	GFK-0840
IC693APU302	Module APM Power Mate, 2 axes, mode suiveur	GFK-0781
IC693APU302	Module APM Power Mate, 2 axes, mode standard	GFK-0840
IC693MCS001/2	Système de commande d'axes Power Mate J (1 et 2 axes)	GFK-1256
IC693APU305	Module co-processeur d'E/S	GFK-1028
IC693CMM321	Module de communication Ethernet	GFK-1084
IC693ADC311	Co-processeur d'affichage alpha-numérique	GFK-0521
IC693BEM331	Contrôleur de bus Genius	GFK-1034
IC693BEM320	Module interface I/O Link (esclave)	GFK-0631
IC693BEM321	Module interface I/O Link (maître)	GFK-0823
IC693CMM311	Module co-processeur de communication	GFK-0582
IC693CMM301	Module de communication Genius	GFK-0412
IC693CMM302	Module de communication Genius amélioré	GFK-0695
IC693PCM300	PCM, 160 Koctets (35 Koctets de programme MegaBasic utilisateur)	GFK-0255
IC693PCM301	PCM, 192 Koctets (47 Koctets de programme MegaBasic utilisateur)	GFK-0255
IC693PCM311	PCM, 640 Koctets (190 Koctets de programme MegaBasic utilisateur)	GFK-0255

Formats des données d'E/S

Les entrées et les sorties logiques sont stockées en bits dans une mémoire cache de bits (table d'états). Les données d'entrée analogique et de sortie analogique sont stockées en mots et résident en mémoire dans une partie de la mémoire RAM de l'application affectée à cette fin.

Conditions par défaut des modules de sortie Série 90-30

A la mise sous tension, les modules de sorties logiques Série 90-30 mettent les sorties à "0" par défaut. Ils conserveront cette condition par défaut jusqu'à la première scrutation des sorties de l'API. Les modules de sorties analogiques peuvent être configurés par un cavalier situé sur le bornier amovible du module, pour se mettre à zéro par défaut ou conserver leur dernier état. Egalement, les modules de sorties analogiques peuvent être alimentés par une source extérieure de façon à ce que, même si l'API n'est pas alimenté, le module de sorties analogiques continue à fonctionner dans son état par défaut.

Données de diagnostics

Des bits de diagnostics sont disponibles dans la mémoire %S pour indiquer la perte de module d'E/S ou qu'il existe une incohérence dans la configuration des E/S. Les informations de diagnostics ne sont pas disponibles pour chaque point d'E/S. Pour plus d'informations sur la gestion des défauts, voir le Chapitre 3, “Explications et correction des défauts”.

Données globales

Données globales Genius

l'API Série 90-30 supporte le partage très rapide des données entre plusieurs UC en utilisant les données globales Genius. Le contrôleur de bus Genius, IC693BEM331 dans l'UC, version 5 et ultérieure, et le module de communication amélioré Genius, IC693CMM302, peuvent diffuser jusqu'à 128 octets de données à d'autres API ou ordinateurs. Ils peuvent recevoir jusqu'à 128 octets de chacun des 30 autres contrôleurs Genius possibles sur le réseau. Les données peuvent être diffusées vers ou reçues de tout type de mémoire, et non pas seulement des bits globaux %G. Le module de communication Genius, IC693CMM301, est limité aux adresses fixes %G et ne peut échanger que 32 bits par adresse de bus série(SBA) allant de 16 à 23. Ce module présente donc des performances moindres puisque le GCM amélioré possède plus de 100 fois cette capacité.

Les données globales peuvent être partagées entre les API Série Five, Série Six et Série 90 connectés au même bus d'E/S Genius.

Communications Ethernet

L'UC Modèle 364 (version 9.0 et supérieure) supporte la connexion à un réseau Ethernet par l'un des deux ports intégrés Ethernet. Des ports AAUI et 10BaseT sont fournis. Le Modèle 364 (version 9.10 ou supérieure) est la seule UC Série 90-30 qui supporte les données globales sur Ethernet (EGD).

Les EGD sont similaires aux données globales Genius en ce sens qu'elle permet à un dispositif (le producteur) de transférer des données à un ou plusieurs dispositifs (les consommateurs) sur le réseau. Les EGD ne sont pas supportées par le logiciel Logicmaster 90 (nécessite la console de programmation à base de Windows pour les API Série 90).

Modules d'E/S du modèle 20

Les modules d'E/S suivants sont disponibles pour l'API Série 90-20. Chaque module est listé par numéro de référence, nombre de points d'E/S et une brève description. Les E/S sont intégrées dans un châssis avec l'alimentation électrique. Pour les caractéristiques et les informations de câblage de chaque module, se référer au Chapitre 5 du *Manuel de l'utilisateur de l'automate programmable Série 90-20*, GFK-0551.

Référence produit	Description	Points d'E/S
IC692MAA541	Module de base d'alimentation et d'E/S, Entrées 120 VCA/Sorties 120 VCA/Alimentation 120 VCA	16 entrées/12 sorties
IC692MDR541	Module de base d'alimentation et d'E/S, Entrées 24 VCC/Sorties relais/Alimentation 120 VCA	16 entrées/12 sorties
IC692MDR741	Module de base d'alimentation et d'E/S, Entrées 24 VCC/Sorties relais/Alimentation 240 VCA	16 entrées/12 sorties
IC692CPU211	Module UC, UC Modèle 211	Non applicable

Configuration et programmation

La configuration consiste à affecter des adresses locales, ainsi que d'autres caractéristiques, aux modules matériels du système. Elle peut s'effectuer soit avant, soit après la programmation, en utilisant le logiciel de configuration ou la console de programmation de poche ; cependant, il est recommandé d'effectuer cette configuration d'abord. Si elle n'a pas été effectuée, vous devez vous référer au *Manuel de l'utilisateur du logiciel de programmation*, GFK-0466, pour décider s'il vaut mieux commencer la programmation à ce moment là.

La programmation consiste à créer un programme d'application pour un API. Comme les API Série 90-30, 90-20, et Série 90 Micro ont un jeu d'instructions commun, les trois peuvent être programmés en utilisant le logiciel Logicmaster 90-30. Les Chapitres 4 à 12 décrivent les instructions de programmation qui peuvent être utilisées pour créer des programmes logique en échelle pour les automates programmables Série 90-30 et Série 90-20.

Si le logiciel de programmation Logicmaster 90-30/20/Micro n'est pas encore installé, se référer au *Manuel de l'utilisateur du logiciel de programmation*, GFK-0466, pour les instructions. Le manuel de l'utilisateur explique comment créer, transférer, éditer et imprimer des programmes.

Chapitre 3 Description et correction des défauts

Ce chapitre est une aide au dépannage des API Série 90-30, 90-20 et Micro s. Il décrit les défauts qui apparaissent dans la table des défauts de l'API et les catégories des défauts qui apparaissent dans la table des défauts des E/S.

Chaque explication de défaut de ce chapitre donne une description du défaut pour la table des défauts de l'API ou la catégorie de défaut pour la table des défauts des E/S. Trouver la description du défaut ou la catégorie du défaut correspondant à l'entrée de la table des défauts applicable, affichée sur l'écran de votre console de programmation. Une description de la cause du défaut et les instructions pour corriger le défaut se trouvent ci-dessous.

Le Chapitre 3 contient les sections suivantes :

Section 1 : Gestion des défauts

Remarque

Ces informations sur la gestion des défauts s'appliquent aux systèmes programmés en utilisant le logiciel Logicmaster 90-30/20/Micro.

Des défauts se produisent dans l'API Série 90-30, 90-20 ou Série 90 Micro lorsque certaines défaillances ou conditions apparaissent et affectent le fonctionnement et les performances du système. Ces conditions, comme la perte d'un module d'E/S ou d'un châssis, peuvent affecter la capacité de l'API à contrôler une machine ou un procédé. Ces conditions peuvent également avoir des effets bénéfiques, comme lorsqu'un nouveau module est connecté, prêt à être utilisé. Ou, ces conditions peuvent servir seulement d'avertissement, comme un signal de pile faible qui indique que la pile protégeant 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 gère ces conditions est appelé Processeur d'alarmes. L'interfaçage entre l'utilisateur et le processeur d'alarmes est le logiciel de programmation. Tout défaut détecté est enregistré dans la table des défauts et affiché soit sur l'écran de la table des défauts de l'API, soit sur l'écran de la table des défauts des E/S, selon le cas.

Classes de défauts

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

Classe de défaut	Exemples
Défaillances internes	Pas de réponses des modules. Seuil de pile bas. Erreurs de checksum de la mémoire.
Défauts d'E/S externes	Perte de châssis ou de module. Addition d'un châssis ou d'un module.
Défauts opérationnelles	Défauts de communication. Défauts de configuration. Défauts d'accès par mot de passe.

Remarque

Pour des informations sur la gestion des défauts de l'API Micro, se référer au *Manuel de l'utilisateur de l'API Série 90 Micro* (GFK-1065).

Réponse du système aux défauts

Les défauts matériels nécessitent la coupure du système 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 des API Série 90-30, 90-20 et Micro ont deux attributs :

Attribut	Description
Table des défauts concernée	Table des défauts des E/S Table des défauts de l'API
Catégorie du défaut	Fatal Diagnostic Informationnel

Tables des défauts

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

Table 3-1. Résumé des défauts

Groupe de défaut	Catégorie du défaut	Table des défauts	Références spéciales des défauts logiques			
Perte ou absence de module d'E/S	Diagnostic	E/S	io_flt	any_flt	io_pres	los_iom
Perte ou absence de module d'optionnel	Diagnostic	API	sy_flt	any_flt	sy_pres	los_sio
Configuration incohérente du système	Fatal	API	sy_flt	any_flt	sy_pres	cfg_mm
Défaut matériel de l'UC de l'API	Fatal	API	sy_flt	any_flt	sy_pres	hrd_cpu
Défaut de checksum du programme	Fatal	API	sy_flt	any_flt	sy_pres	pb_sum
Seuil de pile bas	Diagnostic	API	sy_flt	any_flt	sy_pres	low_bat
Table des défauts de l'API pleine	Diagnostic	—	sy_full			
Table des défauts des E.S pleine	Diagnostic	—	io_full			
Défaut d'application	Diagnostic	API	sy_flt	any_flt	sy_pres	apl_flt
Pas de programme utilisateur	Informationnel	API	sy_flt	any_flt	sy_pres	no_prog
RAM utilisateur altérée	Fatal	API	sy_flt	any_flt	sy_pres	bad_ram
Défaur d'accès par mot de passe	Diagnostic	API	sy_flt	any_flt	sy_pres	bad_pwd
Défaut du logiciel de l'API	Fatal	API	sy_flt	any_flt	sy_pres	sft_cpu
Défaut de stockage de l'API	Fatal	API	sy_flt	any_flt	sy_pres	stor_er
Temps de cycle constant dépassé	Diagnostic	API	sy_flt	any_flt	sy_pres	ov_swp
Défaut d'API inconnu	Fatal	API	sy_flt	any_flt	sy_pres	
Défaut d'E/S inconnu	Fatal	E/S	io_flt	any_flt	io_pres	

Catégorie du défaut

Les défauts peuvent être fatals, diagnostiques ou informationnels.

Les défauts fatals provoquent l'enregistrement du défaut dans la table appropriée, l'établissement des variables de diagnostic et l'arrêt du système. Les défauts diagnostiques sont enregistrés dans la table appropriée et les variables diagnostiques sont établies. Les défauts informationnels sont seulement enregistrés dans la table appropriée.

Les catégories des défauts possibles sont listées dans la table suivante.

Table 3-2. Réponses aux défauts

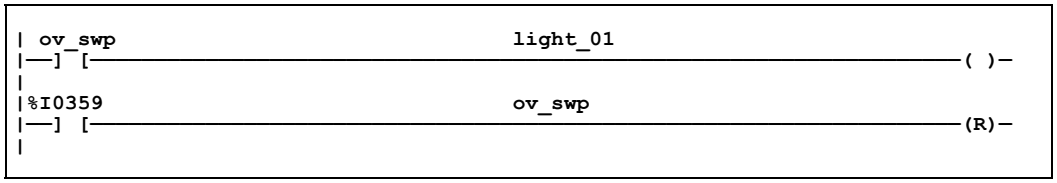
Catégorie du défaut	Réponse de l'UC
Fatal	Enregistre le défaut dans la table des défauts Etablit les références de défaut Passe en mode **ARRET**.
Diagnostic	Enregistre le défaut dans la table des défauts Etablit les références de défaut
Informationnel	Enregistre le défaut dans la table des défauts

Lorsqu'un défaut est détecté, l'UC utilise la réponse à ce défaut. Les réponses aux défauts ne sont pas configurables dans l'API Série 90-30, Serie 90-20 ou Série 90 Micro.

Références de défauts

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

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



Définitions des références de défauts

Le processeur d'alarmes maintient les états des 128 bits logiques du système dans la mémoire %S. Ces références de défauts peuvent être utilisées pour indiquer où un défaut s'est produit et de quel type de défaut il s'agit. Les références de défauts sont affectées aux mémoires %S, %SA, %SB et %SC, et elles possèdent chacune un symbole. Ces références sont disponibles dans le programme d'application selon les besoins. Se référer au Chapitre 2, "Fonctionnement du système", pour une liste des références d'état du système.

Effets secondaires des défauts

Deux défauts, décrits précédemment, ont des effets secondaires associés. Ils sont décrits dans la table suivante.

Effet secondaire	Description
Défaillance du logiciel de l'UC de l'API	Lorsqu'une défaillance du logiciel de l'UC de l'API est enregistrée, l'UC Série 90-30 ou 90-20 passe immédiatement dans un mode spécial **BALAYAGE D'ERREURS**. Aucune activité n'est permise dans ce mode. La seule méthode de suppression de cette condition est de réinitialiser l'API en le mettant hors/sous tension.
Défaillance de stockage de séquences de l'API	Pendant un stockage de séquences (un stockage de blocs programme et d'autres données précédés d'une instruction spéciale Début-de-séquence et finissant avec l'instruction Fin-de-séquence), si la communication avec l'appareil de programmation effectuant le stockage est interrompue ou si une autre défaillance se produit et termine le chargement, le défaut Défaillance de stockage de séquences est enregistré. Tant que ce défaut est présent dans le système, l'API ne passera pas en mode **EXECUTION**.

Affichage de la table des défauts de l'API

L'écran de la table des défauts de l'API affiche les défauts de l'API comme les violations de mot de passe, les incohérences de la configuration de l'API, les erreurs de parité et les erreurs de communication.

Le logiciel de programmation peut être dans n'importe quel mode de fonctionnement. Si le logiciel de programmation est en mode **HORS LIGNE**, aucun défaut n'est affiché. Dans le mode **EN LIGNE** ou **MONITEUR**, les données des défauts de l'API sont affichées. Dans le mode **EN LIGNE**, les défauts peuvent être supprimés (ceci peut être protégé par un mot de passe).

Une fois supprimés, les défauts qui sont toujours présents ne sont pas à nouveau enregistrés dans la table (sauf pour le défaut "Pile faible").

Affichage de la table des défauts des E/S

L'écran de la table des défauts des E/S affiche les défauts des E/S comme les défauts de circuits, les conflits d'adresses, les circuits forcés et les défauts de bus d'E/S.

Le logiciel de programmation peut être dans n'importe quel mode de fonctionnement. Si le logiciel de programmation est en mode **HORS LIGNE**, aucun défaut n'est affiché. Dans le mode **EN LIGNE** ou **MONITEUR**, les données des défauts d'E/S sont affichées. Dans le mode **EN LIGNE**, les défauts peuvent être supprimés (cette fonction peut être protégée par un mot de passe).

Une fois effacés, les défauts qui sont toujours présents ne sont pas à nouveau enregistrés dans la table.

Accès aux informations supplémentaires concernant les défauts

Les tables des défauts contiennent des informations de base sur les défauts. Des informations supplémentaires concernant chaque défaut peuvent être affichées par le logiciel de programmation. De plus, le logiciel de programmation peut fournir un vidage hexadécimal du défaut.

La dernière entrée, Correction, pour chaque explication de défaut de ce chapitre liste les actions à prendre pour corriger le défaut. Noter que l'action corrective de certains défauts inclut le message :

Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.

Ce deuxième message signifie que vous devez donner au S.A.V. les informations lisibles directement dans la table des défauts **et** les informations en hexadécimal. Le personnel du S.A.V. vous donnera d'autres instructions pour l'action appropriée à prendre.

Section 2 : Explications de la table des défauts de l'API

Chaque explication de défaut contient une description du défaut et des instructions pour corriger le défaut. De nombreuses descriptions de défauts ont des causes multiples. Dans ces cas, le code d'erreur, affiché avec les infirmations supplémentaires sur le défaut, est utilisé pour distinguer les différentes conditions du défaut partageant la même description de défaut. Le code d'erreur est représenté par les deux premiers chiffres hexadécimaux du cinquième groupe de nombres, comme montré dans l'exemple suivant.

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

Certains défauts peuvent se produire à cause d'une défaillance de la RAM de la carte UC de l'API. Ces mêmes défauts peuvent également se produire si le système a été coupé et que la tension de la pile est (ou était) trop faible pour maintenir la mémoire. Pour éviter une duplication excessive des instructions lorsqu'une mémoire corrompue peut être la cause de l'erreur, la correction indique simplement :

Effectuer les corrections pour une mémoire corrompue.

ceci signifie :

1. Si le système a été coupé, remplacer la pile. La tension de la pile est peut-être insuffisante pour maintenir le contenu de la mémoire.
2. Remplacer la carte de l'UC de l'API. Les circuits intégrés de la carte de l'UC de l'API sont peut-être défaillants.

La table suivante vous permet de trouver rapidement l'explication du défaut particulier de l'API dans cette section. Chaque entrée est listée comme elle apparaît sur l'écran de la console de programmation.

Actions sur les défauts

Les défauts **Fatals** provoquent le passage de l'API dans une forme de mode **ARRET** à la fin du balayage dans lequel l'erreur s'est produite. Les défauts **Diagnostics** sont enregistrés et les contacts de défaut correspondants sont mis à "1". Les défauts **Informationnels** sont simplement enregistrés dans la table des défauts de l'API.

Perte ou absence de module d'option

Le groupe de défauts **Perte ou absence de module d'option** se produit lorsqu'un PCM, CMM ou ADC ne répond pas. La défaillance peut se produire à la mise sous tension si le module manque ou pendant le fonctionnement si le module ne répond pas. L'action pour ce défaut dans ce groupe est **Diagnostic**.

Code d'erreur :	1, 42
Nom :	Défaillance de la réinitialisation du module d'option
Description :	UC de l'API incapable de ré-établir la communication avec un module d'option après une réinitialisation.
Correction :	(1) Essayer de réinitialiser une deuxième fois. (2) Remplacer le module d'option. (3) Mettre le système hors tension. Vérifier que le PCM est bien installé dans le châssis et que tous les câbles sont bien connectés et installés. (4) Remplacer les câbles.
Code d'erreur :	Tous les autres
Nom :	Défaillance du module pendant la configuration
Description :	Le logiciel d'exploitation de l'API génère cette erreur lorsqu'un module est défaillant pendant la mise sous tension ou le stockage de la configuration.
Correction :	(1) Mettre le système hors tension. Remplacer le module situé dans cet emplacement du châssis.

Réinitialisation, addition d'un module d'option ou module d'option en trop

Le groupe de défauts **Réinitialisation, addition d'un module d'option ou module d'option en trop** se produit lorsqu'un module d'option (PCM, ADC, etc.) vient en ligne ou qu'un module est trouvé dans le châssis alors qu'aucun n'est spécifié dans la configuration. L'action pour ce défaut dans ce groupe est **Diagnostic**. Trois octets de données spécifiques au défaut fournissent des informations supplémentaires concernant le défaut.

Correction :	(1) Mettre à jour la configuration pour inclure le module. (2) Enlever le module du système.

Incohérence de la configuration du système

Le groupe de défauts **Incohérence de la configuration** se produit lorsque le module occupant un emplacement est différent de celui qui est spécifié dans le fichier de configuration. L'action pour ce défaut dans ce groupe est **Fatale**.

Code d'erreur :	1
Nom :	Configuration incohérente du système
Description :	Le logiciel d'exploitation de l'API (qui configure le système) génère ce défaut lorsque le module occupant un emplacement n'est pas du même type que ce qui est indiqué par le fichier de configuration ou lorsque le type de châssis configuré ne correspond pas au châssis présent actuellement.
Correction :	Identifier l'incohérence et reconfigurer le module ou le châssis.
Code d'erreur :	6
Nom :	Configuration incohérente du système
Description :	Ceci est identique au code d'erreur 1 en ce sens que ce défaut se produit lorsque le module occupant un emplacement n'est pas du même type que ce que le fichier de configuration indique ou lorsque le type de châssis configuré ne correspond pas au châssis présent actuellement.
Correction :	Identifier l'incohérence et reconfigurer le module ou le châssis.
Code d'erreur :	18
Nom :	Matériel non supporté
Description :	Un module du type PCM ou PCM est présent dans un 311, 313 ou 323, ou dans un châssis d'extension.
Correction :	Corriger physiquement la situation en enlevant le module du type PCM ou PCM ou en installant une UC qui supporte le PCM.
Code d'erreur :	26
Nom :	Configuration du module pas encore acceptée par le module
Description :	Le module ne peut pas accepter la nouvelle configuration car il est occupé par un procédé différent.
Correction :	Laisser le module terminer son opération courante et re-stocker la configuration.
Code d'erreur :	51
Nom :	Fonction FIN exécutée par l'action SFC
Description :	Le placement d'une fonction FIN dans la logique SFC ou dans la logique appelée par SFC produira ce défaut.
Correction :	Supprimer la fonction FIN de la logique SFC ou de la logique appelée par la logique SFC.

Défaillance du logiciel du module d'option

Le groupe de défauts **Défaillance du logiciel du module d'option** se produit lorsqu'une défaillance irrécupérable du logiciel se produit sur un module PCM ou ADC. L'action pour ce défaut dans ce groupe est **Fatale**.

Code d'erreur :	Tous
Nom :	Fréquence COMMREQ trop élevée
Description :	Les COMMREQs sont envoyées à un module plus vite qu'il ne peut les traiter.
Correction :	Modifier le programme de l'API pour envoyer les COMMREQs au module affecté à une vitesse moins élevée.

Défaillance du checksum d'un bloc programme

Le groupe de défauts **Défaillance du checksum d'un bloc programme** se produit lorsque l'UC de l'API détecte des conditions d'erreur dans des blocs programme reçus par l'API. Il se produit également lorsque l'UC de l'API détecte des erreurs de checksum pendant la vérification de la mémoire à la mise sous tension ou pendant une vérification en arrière-plan en mode **`EXECUTION`**. L'action pour ce défaut dans ce groupe est **Fatale.**

Code d'erreur :	Tous
Nom :	défaillance du checksum d'un bloc programme
Description :	Le logiciel d'exploitation de l'API génère cette erreur lorsqu'un bloc programme est corrompu.
Correction :	(1) Effacer la mémoire de l'API et essayer à nouveau de stocker. (2) Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.

Signal de pile faible

Le groupe de défauts **Signal de pile faible** se produit lorsque l'UC de l'API détecte une pile faible sur l'alimentation de l'API ou un module, comme le PCM et rapporte une condition de pile faible. L'action pour ce défaut dans ce groupe est **Diagnostic**.

Code d'erreur :	0
Nom :	Signal de pile défectueuse
Description :	La pile du module de l'UC (ou un autre module disposant d'une pile) est défectueuse.
Correction :	Remplacer la pile. Ne pas couper l'alimentation du châssis.
Code d'erreur :	1
Nom :	Signal de pile faible
Description :	Une pile de l'UC ou d'un autre module a un signal faible.
Correction :	Remplacer la pile. Ne pas couper l'alimentation du châssis.

Temps de cycle constant dépassé

Le groupe de défauts **Temps de balayage constant dépassé** se produit lorsque l'UC de l'API fonctionne en mode **BALAYAGE CONSTANT** et qu'il détecte que le balayage a dépassé la temporisation de balayage constant. Les données supplémentaires du défaut contiennent la durée actuelle du balayage dans les deux premiers octets et le nom du programme dans les huit octets suivants. L'action pour ce défaut dans ce groupe est **Diagnostic**.

Correction :	(1)	Augmenter le temps de balayage constant.
	(2)	Enlever de la logique du programme d'application.

Défaut d'application

Le groupe de défauts **Défaut d'application** se produit lorsque l'UC de l'API détecte un défaut dans le programme utilisateur. L'action pour ce défaut dans ce groupe est **Diagnostic**, sauf lorsque l'erreur est Pile d'appels de sous-programme dépassée, auquel cas, elle est **Fatale**.

Code d'erreur :	7
Nom :	Pile d'appels de sous-programmes dépassée
Description :	Les appels de sous-programmes sont limités à une profondeur de 8. Un sous-programme peut en appeler un autre qui, à son tour, peut appeler un autre sous-programme jusqu'à ce que les 8 niveaux d'appels soient atteints.
Correction :	Modifier le programme afin que la profondeur d'appel de sous-programmes ne dépasse pas 8.
Code d'erreur :	1B
Nom :	CommReq non traitée à cause des limitations mémoire de l'API
Description :	Les demandes de communication sans attente peuvent être placées dans la queue plus vite qu'elles ne peuvent être traitées (par exemple, une par balayage). Dans une telle situation, lorsque les demandes de communications s'accumulent au point que l'API a moins qu'une quantité minimale de mémoire disponible, la demande de communication sera mise en défaut et ne sera pas traitée
Correction :	Emettre moins de demandes de communication ou sinon, réduire la quantité de courrier échangé dans le système.
Code d'erreur :	5A
Nom :	Mise hors tension utilisateur demandée
Description :	Le logiciel d'exploitation de l'API (blocs fonctionnels) génère cette alarme informationnelle lorsque la Requête de service #13 (Mise hors tension utilisateur) s'exécute dans le programme d'application.
Correction :	Aucune. Alarme informationnelle seulement.

Pas de programme utilisateur présent

Le groupe de défauts **Pas de programme utilisateur présent** se produit lorsque l'UC de l'API doit passer du mode **ARRET** au mode **EXECUTION** ou s'il existe un stockage vers l'API et qu'aucun programme utilisateur n'existe dans l'API. L'UC de l'API détecte l'absence de programme utilisateur à la mise sous tension. L'action pour ce défaut dans ce groupe est **Informationnelle**

Correction :	Charger un programme d'application avant d'essayer de passer en mode **EXECUTION**.

Programme utilisateur corrompu à la mise sous tension

Le groupe de défauts **Programme utilisateur corrompu à la mise sous tension** se produit lorsque l'UC de l'API détecte une RAM utilisateur corrompue. L'UC de l'API restera en mode **ARRET** jusqu'à ce qu'un programme utilisateur et un fichier de configuration valides soient chargés. L'action pour ce défaut dans ce groupe est **Fatale**.

Code d'erreur :	1
Nom :	RAM utilisateur corrompue à la mise sous tension
Description :	Le logiciel d'exploitation de l'API (logiciel d'exploitation) génère cette erreur lorsqu'il détecte une RAM utilisateur corrompue à la mise en route.
Correction :	(1) Recharger le fichier de configuration, le programme utilisateur et les références (s'il y en a). (2) Remplacer la pile de l'UC de l'API. (3) Remplacer la carte d'extension mémoire sur l'UC de l'API. (4) Remplacer l'UC de l'API.
Code d'erreur :	2
Nom :	Instruction Booléenne illégale détectée
Description :	Le logiciel d'exploitation de l'API (logiciel d'exploitation) génère cette erreur lorsqu'il détecte une mauvaise instruction dans le programme utilisateur.
Correction :	(1) Restaurer le programme utilisateur et les références (s'il y en a). (2) Remplacer la carte d'extension mémoire sur l'UC de l'API. (3) Remplacer l'UC de l'API.

Défaillance d'accès par mot de passe

Le groupe de défauts **Défaillance d'accès par mot de passe** se produit lorsque l'UC de l'API reçoit une demande de changement pour un nouveau niveau de privilège et que le mot de passe inclus dans cette demande n'est pas valide pour ce niveau. L'action pour ce défaut dans ce groupe est **Informationnelle**

Correction :	Refaire la demande avec le mot de passe correct.

Défaillance du logiciel de l'UC de l'API

Les défauts de ce groupe de défauts **Défaillance du logiciel de l'UC de l'API** sont générés par le logiciel d'exploitation de l'UC de l'API Série 90-30, 90-20 ou Micro. Ils se produisent à de nombreux points différents du fonctionnement du système. Lorsqu'un défaut **Fatal** se produit, l'UC de l'API passe **immédiatement** en mode spécial **BALAYAGE D'ERREURS**. Aucune activité n'est permise lorsque l'API se trouve dans ce mode. Le seul moyen de supprimer cette condition est de mettre l'API hors tension, puis de le remettre sous tension. L'action pour ce défaut dans ce groupe est **Fatale**.

Code d'erreur :	1 à B
Nom :	La mémoire utilisateur n'a pas pu être allouée
Description :	Le logiciel d'exploitation de l'API (gestionnaire de mémoire) génère ces erreurs lorsque le logiciel demande au gestionnaire de mémoire d'allouer ou désallouer un bloc ou des blocs de mémoire dans la RAM utilisateur, qui ne sont pas légaux. Ces erreurs ne doivent *pas* se produire dans un système de production.
Correction :	Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.
Code d'erreur :	D
Nom :	Mémoire système non disponible
Description :	Le logiciel d'exploitation de l'API (scrutation des E/S) génère cette erreur lorsque sa demande de bloc de mémoire système est refusée par le gestionnaire de mémoire car aucune mémoire n'est disponible dans l'ensemble de la mémoire du système. Elle est *Informationnelle* si elle se produit pendant l'exécution d'un bloc fonctionnel d'E/S. Elle est fatale si elle se produit pendant l'initialisation à la mise sous tension ou l'autoconfiguration.
Correction :	Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.
Code d'erreur :	E
Nom :	La mémoire système ne peut pas se libérer
Description :	Le logiciel d'exploitation (scrutation des E/S) génère cette erreur lorsqu'il demande au gestionnaire de mémoire de désallouer un bloc de mémoire système et que la désallocation n'a pas réussi. Cette erreur ne peut se produire que pendant l'exécution d'un bloc fonctionnel d'E/S.
Correction :	(1) Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut. 2 Effectuer les corrections pour une mémoire corrompue.
Code d'erreur :	10
Nom :	Demande de scrutation invalide du scrutateur des E/S
Description :	Le logiciel d'exploitation (scrutation des E/S) génère cette erreur lorsque le système d'exploitation ou le bloc fonctionnel d'E/S ne demande ni une scrutation totale, ni une scrutation partielle des E/S. Ceci ne doit *pas* se produire dans un système de production.
Correction :	Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.
Code d'erreur :	13
Nom :	Erreur du logiciel d'exploitation de l'API
Description :	Le logiciel d'exploitation de l'API génère cette erreur lorsque certains problèmes de logiciel d'exploitation de l'API se produisent. Cette erreur ne doit *pas* se produire dans un système de production.
Correction :	(1) Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut. (2) Effectuer les corrections pour une mémoire corrompue.

Code d'erreur :	14, 27
Nom :	mémoire programme de l'API corrompue
Description :	Le logiciel d'exploitation de l'API génère ces erreurs lorsque certains problèmes de logiciel d'exploitation de l'API se produisent. Celles-ci ne doivent *pas* se produire dans un système de production.
Correction :	(1) Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut. 2 Effectuer les corrections pour une mémoire corrompue.
Code d'erreur :	27 à 4E
Nom :	Erreur du logiciel d'exploitation de l'API
Description :	Le logiciel d'exploitation de l'API génère ces erreurs lorsque certains problèmes de logiciel d'exploitation de l'API se produisent. Ces erreurs ne doivent *pas* se produire dans un système de production.
Correction :	Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.
Code d'erreur :	4F
Nom :	Communications défaillantes
Description :	Le logiciel d'exploitation de l'API(processeur de requête de service) génère cette erreur lorsqu'il essaie de se conformer à une requête qui nécessite une communication de fond de panier et reçoit une réponse rejetée.
Correction :	(1) Vérifier si l'activité du bus est normale. (2) Remplacer le module d'option intelligent auquel la requête était dirigée.
Code d'erreur :	50, 51, 53
Nom :	Erreurs de mémoire système
Description :	Le logiciel d'exploitation de l'API génère ces erreurs lorsque sa demande de bloc de mémoire système est refusée par le gestionnaire de mémoire à cause de mémoire non disponible ou qui contient des erreurs.
Correction :	(1) Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut. (2) Effectuer les corrections pour une mémoire corrompue.
Code d'erreur :	52
Nom :	Communication de fond de panier défaillante
Description :	Le logiciel d'exploitation de l'API(processeur de requête de service) génère cette erreur lorsqu'il essaie de se conformer à une requête qui nécessite une communication de fond de panier et reçoit une réponse rejetée.
Correction :	(1) Vérifier si l'activité du bus est normale. (2) Remplacer le module d'option intelligent auquel la requête était dirigée. (3) Vérifier la bonne fixation du câble parallèle de la console de programmation.
Code d'erreur :	Tous les autres
Nom :	Erreur du système interne de l'UC de l'API
Description :	Une erreur système interne s'est produite, ce qui ne doit **pas** se produire dans un système de production.
Correction :	Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.

Défaillance des communications pendant le stockage

Le groupe de défauts **Défaillance des communications pendant le stockage** se produit pendant le stockage de blocs programme et d'autres données dans l'API. Le flux d'instructions et de données pour le stockage de blocs programme et de données commence par une instruction spéciale de début-de-séquence et se termine par une instruction de fin-de-séquence. Si la communication avec l'appareil de programmation effectuant le stockage est interrompue ou si une autre défaillance se produit, qui met fin au chargement, ce défaut est enregistré. Tant que ce défaut est présent dans le système, l'automate programmable ne passera pas en mode **EXECUTION**.

Ce défaut n'est *pas* automatiquement supprimé à la mise en route ; l'utilisateur doit spécifiquement ordonner la suppression de la condition. L'action pour ce défaut dans ce groupe est **Fatale**.

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

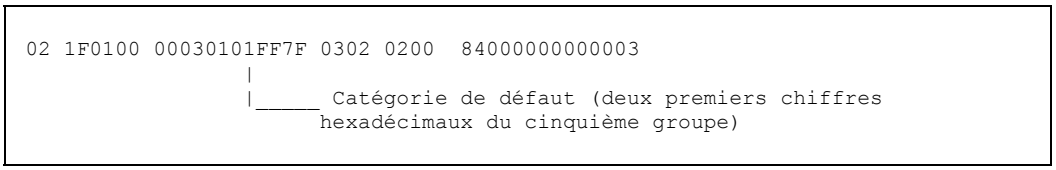
Section 3 : Explications de la table des défauts d'E/S

La table des défauts des E/S rapporte les données concernant les défauts selon trois classifications :

- Catégorie de défaut.
- Type de défaut.
- Description du défaut.

Les défauts décrits à la page suivante ont une catégorie de défaut, mais n'ont pas de type de défaut ou de groupe de défauts.

Chaque explication de défaut contient une description du défaut et des instructions pour corriger le défaut. De nombreuses descriptions de défauts ont des causes multiples. Dans ces cas, le code d'erreur, affiché avec les informations supplémentaires sur le défaut obtenues en appuyant sur CTRL-F, est utilisé pour distinguer les différentes conditions du défaut partageant la même description de défaut. (Pour plus d'informations concernant l'utilisation de CTRL-F, se référer à l'Annexe B, *"*Interprétation des tables de défauts," dans ce manuel). La catégorie de défaut est représentée par les deux premiers chiffres hexadécimaux du cinquième groupe de nombres, comme montré dans l'exemple suivant.



La table suivante vous permet de trouver rapidement l'explication du défaut d'E/S particulier dans cette section. Chaque entrée est listée comme elle apparaît sur l'écran de la console de programmation.

Perte de module d'E/S

La catégorie de défaut **Perte de module d'E/S** s'applique aux modules d'E/S logiques et analogiques du Modèle 30. Il n'y a pas de types de défaut ou de descriptions de défaut associés à cette catégorie. L'action pour ce défaut est **Diagnostic**.

Description :	Le logiciel d'exploitation de l'API génère cette erreur lorsqu'il détecte qu'un module d'E/S du Modèle 30 ne répond plus aux instructions de l'UC de l'API ou lorsque le fichier de configuration indique qu'un module d'E/S occupe un emplacement et qu'il n'y a pas de module dans cet emplacement.
Correction :	(1) Remplacer le module. (2) Corriger le fichier de configuration. (3) Afficher la table des défauts de l'API sur la console de programmation. Contacter le S.A.V. API GE Fanuc en donnant toutes les informations contenues dans l'entrée du défaut.

Addition d'un module d'E/S

La catégorie de défauts **Addition d'un module d'E/S** s'applique aux modules d'E/S logiques et analogiques du Modèle 30. Il n'y a pas de types de défaut ou de descriptions de défaut associés à cette catégorie. L'action pour ce défaut est **Diagnostic**.

Description :	Le logiciel d'exploitation de l'API génère cette erreur lorsqu'un module d'E/S, qui a été mis en défaut, fonctionne à nouveau.
Correction :	(1) Aucune action nécessaire si le module a été enlevé ou remplacé, ou que le châssis déporté a été mis hors tension, puis remis sous tension. (2) Mettre à jour le fichier de configuration ou enlever le module.
Description :	Le logiciel d'exploitation de l'API génère cette erreur lorsqu'il détecte un module d'E/S du Modèle 30 dans un emplacement dont le fichier de configuration indique qu'il devrait être vide.
Correction :	(1) Enlever le module. (Il est peut être dans le mauvais emplacement). (2) Mettre à jour et restaurer le fichier de configuration pour inclure le module supplémentaire.

Chapitre 4 Fonction relais

Ce chapitre explique l'utilisation des contacts, des bobines et des liaisons dans les programmes en schéma à relais.

Utilisation des contacts

Un contact représente l'état d'une référence automate. Un contact laissera passer le flux d'énergie ou non selon l'état de la référence correspondante et du type de contact. Une référence est activée si son état est à "1" ; elle est désactivée si son état est à "0".

Table 4-1. Types de contacts

Type de contact	Affichage	Le contact transmet le flux à droite
Normalement ouvert	—| |—	Lorsque la référence est à "1".
Normalement fermé	—|/|—	Lorsque la référence est à "0".
Contact de continuation	<+>———	Si la bobine de continuité précédente est à "1".

Utilisation des bobines

Les bobines sont utilisées pour commander les références logiques. Une logique conditionnelle doit être utilisée pour contrôler le flux d'énergie vers une bobine. Les bobines agissent directement et ne transmettent pas le flux d'énergie vers la droite. Si une logique supplémentaire du programme doit être exécutée après une bobine, une référence interne doit être utilisée pour cette bobine ou une combinaison bobine/contact de continuité peut être utilisée.

Les bobines sont toujours situées à l'extrême droite d'une ligne de logique. Un circuit peut contenir jusqu'à huit bobines.

Le type de bobine utilisé dépend du type d'action désiré. Les états des bobines rémanentes sont sauvegardés lors d'une mise hors tension ou lorsque l'API passe du mode **ARRET** au mode **EXECUTION**. Les états des bobines non-rémanentes sont mis à "0" lors d'une mise hors tension ou lorsque l'API passe du mode **ARRET** au mode **EXECUTION**.

Table 4-2. Types de bobines

Type de bobine	Affichage	Bobine alimentée	Résultat
Normalement ouverte	—()—	OUI	Met la référence à "1".
		NON	Met la référence à "0".
Inversée	—(/)—	OUI	Met la référence à "0".
		NON	Met la référence à "1".
Rémanente	—(M)—	OUI	Met la référence à "1", rémanente.
		NON	Met la référence à "0", rémanente.
Inversée Rémanente	—(/M)—	ON	Met la référence à "0", rémanente.
		NON	Met la référence à "1", rémanente.
Transition positive	—(↑)—	NON→ OUI	Si la référence est à "0", la met à "1" pour un balayage.
Transition négative	—(↓)—	OUI← NON	Si la référence est à "0", la met à "1" pour un balayage.
SET	—(S)—	OUI	Met la référence à "1" jusqu'à sa remise à "0" par —(R)—.
		NON	Ne change pas l'état de la bobine.
RESET	—(R)—	OUI	Met la référence à "0" jusqu'à sa mise à "1" par —(S)—.
		NON	Ne change pas l'état de la bobine.
SET rémanent	—(SM)—	OUI	Met la référence à "1" jusqu'à sa remise à "0" par —(RM)—, rémanente.
		NON	Ne change pas l'état de la bobine.
Rémanente RESET	—(RM)—	OUI	Met la référence à "0" jusqu'à sa mise à "1" par —(SM)—, rémanente.
		NON	Ne change pas l'état de la bobine.
Continuité Bobine	——<+>	OUI	Met le contact de continuité suivant à "1".
		NON	Met le contact de continuité suivant à "0".

Contact normalement ouvert —| |—

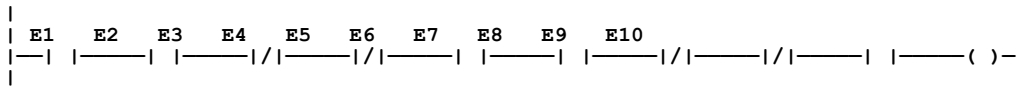
Un contact normalement ouvert agit comme un interrupteur qui transmet le flux d'énergie si la référence associée est à "1" .

Contact normalement fermé —|/|—

Un contact normalement fermé agit comme un interrupteur qui transmet le flux d'énergie si la référence associée est à "0".

Exemple

L'exemple suivant montre un échelon de 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".
To be adjusted (one element is missing)

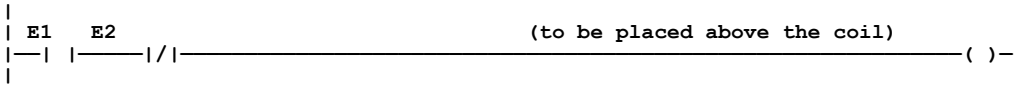


Bobine —( )—

Une bobine met une référence logique à "1" lorsqu'elle reçoit le flux d'énergie. Elle n'est pas rémanente et ne peut donc être utilisée avec les références système (%SA, %SB, %SC ou %G).

Exemple

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

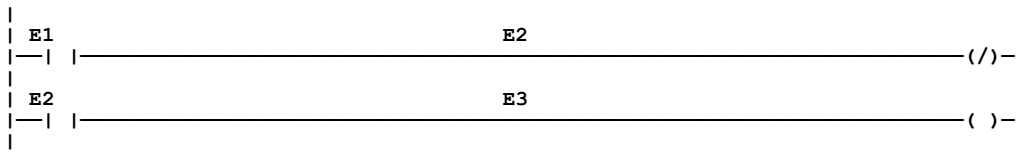


Bobine inversée —(/)—

Une bobine inversée met une référence logique à "1" lorsqu'elle n'est pas alimentée. Elle n'est pas rémanente et ne peut être utilisée avec les références 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 normalement ouverte, la bobine rémanente met une référence logique à "1" lorsqu'elle reçoit le flux d'énergie. L'état de la bobine rémanente est maintenu en cas de coupure d'alimentation. Par conséquent, elle ne peut pas être utilisée avec des références automate strictement non-rémanentes (%T).

Bobine rémanente inversée —(/M)—

La bobine rémanente inversée met une référence logique à "1" lorsqu'elle ne reçoit pas de flux d'énergie. L'état de la bobine rémanente inversée est maintenu en cas de coupure d'alimentation. Par conséquent, elle ne peut pas être utilisée avec des références automate strictement non-rémanentes (%T).

Bobine de transition positive —(↑)—

Si la référence associée à une bobine de transition positive est à "0", la bobine est mise à "1" lorsqu'elle reçoit le flux d'énergie. Tous les contacts associés à cette bobine changeront d'état pendant le cycle automate en cours. (Si le circuit contenant la bobine n'est pas validé les cycles suivants, elle restera à "1"). Cette bobine peut être utilisée comme impulsion.

Chaque référence ne doit être utilisée comme bobine de transition qu'une fois dans le programme d'application afin de conserver la nature impulsionnelle de la bobine.

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

Bobine de transition négative —(↓)—

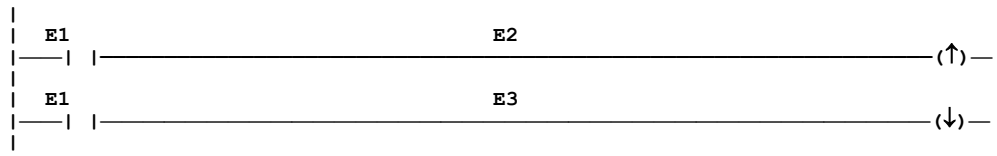
Si la référence associée à cette bobine est à "0", lorsque la bobine ***cesse*** de recevoir le flux d'énergie, la référence est mise à "1" et tous les contacts associés à cette bobine changeront d'état pendant un balayage.

Chaque référence ne doit être utilisée comme bobine de transition qu'une fois dans le programme d'application afin de conserver la nature impulsionnelle de la bobine.

Les bobines de transition peuvent être utilisées avec des références mémoire rémanentes ou non-rémanentes (%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 d'énergie, mettant E2 à "1" pendant un un cycle automate. Lorsque E1 passe de "1" à "0", le flux d'énergie est enlevé de E2 et E3, mettant la bobine E3 à "1" pendant un cycle.



Bobine SET —(S) —

Les bobines SET et RESET sont des bobines non-rémanentes qui peuvent être utilisées pour maintenir (“verrouiller”) l'état d'une référence (par exemple, E1) soit à "1", soit à "0". Lorsqu'une bobine SET reçoit le flux d'énergie, sa référence reste à "1" (que la bobine elle-même reçoive ou non le flux d'énergie) jusqu'à ce que la référence soit remise à "0" par une autre bobine.

Les bobines SET écrivent un résultat indéfini dans les tables de transition. (Se référer aux informations de “Transitions et Forçages” dans le Chapitre 2, “Fonctionnement du système”).

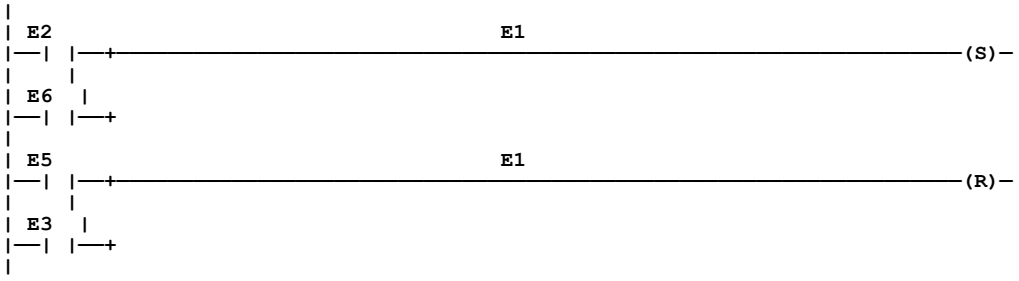
Bobine RESET —(R)—

La bobine RESET met une référence logique à "0" si la bobine reçoit le flux d'énergie. La référence reste à "0" jusqu'à ce qu'elle soit remise à "1" par une autre bobine. La dernière bobine SET ou RESET résolue d'une paire, prend la priorité.

Les bobines RESET écrivent un résultat indéfini dans les tables de transition. (Se référer aux informations de “Transitions et Forçages” dans le Chapitre 2, “Fonctionnement du système”).

Exemple

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



Remarque

Lorsque l'utilisation d'une bobine est configurée pour être UNIQUE, vous pouvez utiliser une référence %M ou %Q spécifique avec une seule bobine, mais vous pouvez l'utiliser avec une bobine SET et une bobine RESET simultanément. Lorsque le niveau de vérification de bobine est WARN MULTIPLE ou MULTIPLE, alors chaque référence peut être utilisée avec des bobines, bobines SET et bobines RESET multiples. Avec un usage multiple, une référence peut être mise à "1" soit par une bobine SET, soit par une bobine normale, et peut être mise à "0" par une bobine RESET ou par une bobine normale.

Bobine SET rémanente —(SM)—

Les bobines SET et RESET rémanentes sont similaires aux bobines SET et RESET, mais elles sont maintenues en cas de coupure d'alimentation ou lorsque l'API passe du mode **ARRET** au mode **EXECUTION**. Une bobine SET rémanente met une référence logique à "1" lorsqu'elle reçoit le flux d'énergie. La référence reste à "1" jusqu'à ce qu'elle soit remise à "0" par une bobine RESET rémanente.

Les bobines SET rémanentes écrivent un résultat indéfini dans le bit de transition pour la référence données. (Se référer aux informations de "Transitions et Forçages" dans le Chapitre 2, "Fonctionnement du système").

Bobine RESET rémanente —(RM)—

Cette bobine met une référence logique à "0" lorsqu'elle reçoit le flux d'énergie. La référence reste à "0" jusqu'à ce qu'elle soit mise à "1" par une bobine SET rémanente. L'état de cette bobine est maintenu en cas de coupure d'alimentation ou lorsque l'API passe du mode **ARRET** au mode **EXECUTION**.

Les bobines RESET rémanentes écrivent un résultat indéfini dans la table de transition pour la référence correspondante. (Se référer aux informations de "Transitions et Forçages" dans le Chapitre 2, "Fonctionnement du système").

Liaisons

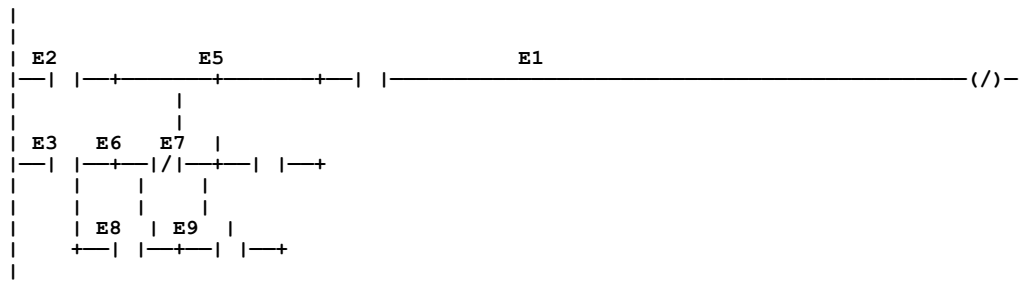
Les liaisons horizontales et verticales sont utilisées pour relier les éléments d'un circuit entre eux. Elles servent à terminer le flux de logique ("énergie") de la gauche vers la droite, dans un circuit.

Remarque

Vous ne pouvez pas utiliser une liaison horizontale pour relier une fonction ou une bobine vers la barre d'alimentation de gauche. Cependant, vous pouvez utiliser %S7, le bit système ALW_ON (toujours à "1") avec un contact normalement ouvert relié à labarre d'alimentation pour appeler une fonction à chaque cycle.

Exemple

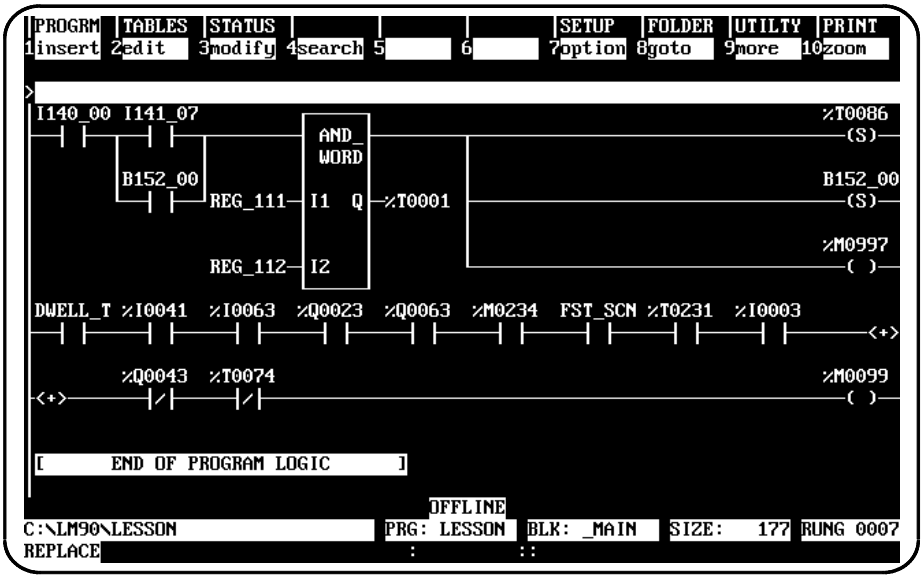
Dans l'exemple suivant, deux liaisons horizontales sont utilisées pour connecter les contacts E2 et E5. Une liaison verticale est utilisée pour connecter les contacts E3, E6, E7, E8 et E9 à E2.



Bobines (———<+>) et contacts (<+>———) de continuité

Les bobines de continuité (———<+>) et les contacts de continuité (<+>———) sont utilisés pour poursuivre le circuit au-delà de la limite de dix colonnes. L'état de la dernière bobine de continuité exécutée est l'état du flux qui sera pris en compte sur le contact de continuité suivant. Une bobine de continuité doit précéder un contact de continuité. L'état du contact de continuité est mis à "0" lorsque l'API passe de **Arrêt** à **Exécution**, et il n'y aura pas de flux si la bobine de transition n'a pas été mise à "1" depuis le passage en mode **Exécution**.

Il ne peut y avoir qu'une bobine et un contact par circuit ; le contact de continuité doit être dans la colonne 1 et la bobine de continuité doit être dans la colonne 10. Un exemple de bobine et de contact de continuité est montré ci-dessous :



Chapitre 5 Temporisateurs et compteurs

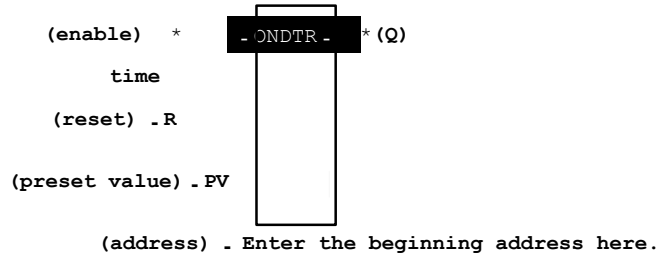
Ce chapitre explique comment utiliser les temporisateurs à l'activation/désactivation et les compteurs/décompteurs. Les données associées à ces fonctions sont rémanentes pendant les coupures d'alimentation.

Paramètres fonctionnels nécessaires aux temporisateurs et compteurs

Chaque temporisateur et compteur utilise trois registres %R pour stocker les informations suivantes :

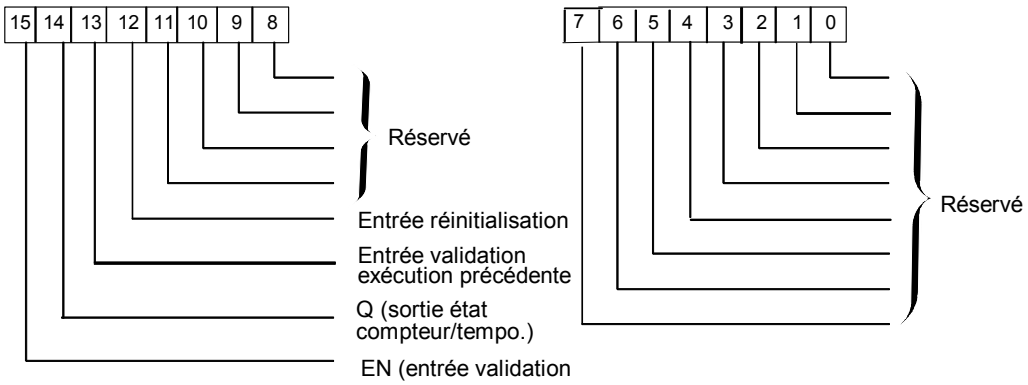
valeur de comptage (CV)	mot 1
valeur de pré-selection (PV)	mot 2
mot de contrôle	mot 3

Lorsque vous entrez un temporisateur ou un compteur, vous devez entrer une adresse de début pour ces trois registres directement en dessous du graphique représentant la fonction. Par exemple :



Remarque

Veiller à ne pas réutiliser la zone de trois registres d'un temporisateur/compteur. Le logiciel Logicmaster ne vérifie *pas* et ne vous avertit pas si les blocs registres se chevauchent. Les temporisateurs et les compteurs ne fonctionneront pas si vous mettez la valeur de comptage d'un bloc par-dessus la valeur de pré-sélection du bloc précédent.Le mot de contrôle mémorise l'état des entrées et des sorties Booléennes du bloc fonctionnel associé, comme montré dans l'illustration 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

Faire attention si vous utilisez la même adresse pour la valeur de pré-sélection et le deuxième mot du bloc de trois mots. Si la valeur de pré-sélection n'est pas une constante, elle est généralement chargée dans un emplacement mémoire différent de celui du deuxième mot. Certaines applications imposent d'utiliser l'adresse du deuxième mot pour la valeur de pré-sélection, par exemple %R0102 pour un bloc commençant à %R0101. Ceci permet à l'application de changer la valeur de pré-sélection pendant le fonctionnement du temporisateur ou du compteur. Les applications peuvent lire la valeur de comptage ou le mot de contrôle, mais elles ne doivent pas écrire dans ces mots.

Remarque spéciale sur certaines opérations de bits

Lors de l'utilisation de la fonction Bit Test, Bit Set, Bit Clear ou Bit Position, les bits sont indexés de 1 à 16, *ET NON* de 0 à 15 comme montré ci-dessus.

ONDTR

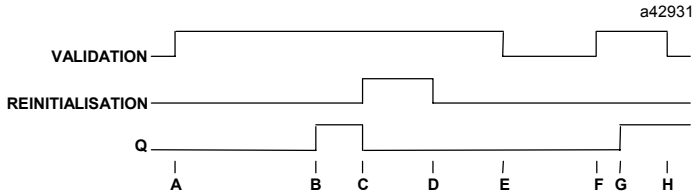
Un temporisateur à l'activation rémanent (ONDTR) s'incrémente lorsqu'il reçoit un flux d'énergie et maintient sa valeur lorsque le flux d'énergie s'arrête. Le temps peut se compter en dixièmes de seconde (la sélection par défaut), en centièmes de seconde ou en millièmes de seconde. La plage est de 0 à +32,767 en 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 la fonction ONDTR reçoit le flux d'énergie, il commence à accumuler le temps (valeur courante). Lorsque ce temporisateur est rencontré dans le programme, sa valeur courante est rafraîchie.

Remarque

Si de multiples évènements de ce même temporisateur, avec la même adresse, sont validés pendant un cycle automate, la valeur courante du temporisateur ne sera pas modifiée.

Lorsque la valeur courante est égale ou excède la valeur de pré-sélection, la sortie Q est excitée. Tant que le temporisateur est alimenté, il continue d'accumuler jusqu'à ce que la valeur maximale soit atteinte. Lorsque la valeur maximale est atteinte, elle est mémorisée et la sortie Q reste excitée sans tenir compte de l'état de l'entrée validation.



A = VALIDATION passe à "1" ; le temporisateur commence à accumuler.

B = la valeur courante atteint la valeur présélectionnée ; Q passe à "1".

C = REINITIALISATION passe à "1" ; Q passe à "0", le temps accumulé est réinitialisé.

D = REINITIALISATION passe à "0" ; le temporisateur commence à nouveau à accumuler.

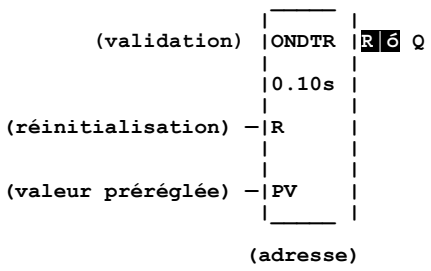
E = VALIDATION passe à "0" ; le temporisateur arrête d'accumuler. Le temps accumulé reste le même.

F = VALIDATION repasse à "1" ; le temporisateur continue d'accumuler le temps.

G = la valeur courante devient égale à la valeur présélectionnée ; Q passe à "1". Le temporisateur continue d'accumuler le temps jusqu'à ce que
VALIDATION passe à "0", REINITIALISATION passe à "1" ou que la valeur courante devienne égale au temps maximal.

H = VALIDATION passe à "0" ; le temporisateur arrête d'accumuler le temps.

Lorsque le flux d'énergie vers le temporisateur s'arrête, la valeur courante se fige et est maintenue. La sortie Q reste activée si tel était le cas. Lorsque la fonction reçoit à nouveau le flux d'énergie, la valeur courante s'incrémente à nouveau, en repartant de la valeur mémorisée. Lorsque la réinitialisation R reçoit le flux d'énergie, la valeur courante est remise à "0" et la sortie Q est désactivée. Sur les UC des modèles 35x et 36x, si la validation du ONDTR est à "0", la valeur présélectionnée = 0 et la réinitialisation R reçoit le flux d'énergie ; alors, la sortie sera à "0". Cependant, sur les UC du modèle 311 au modèle 341 , sous les mêmes conditions, elle sera à "1".



Paramètres

Paramètre	Description
adresse	La fonction ONDTR utilise trois registres consécutifs correspondants à : • Valeur courante = mot 1. • Valeur présélectionnée = mot 2. • Mot de contrôle = mot 3. Lorsque vous entrez dans un ONDTR, vous devez entrer une adresse pour l'emplacement de ces trois registres consécutifs directement en dessous du graphique représentant la fonction. **Remarque :** Ne pas utiliser cette adresse avec d'autres instructions. **Précautions :** Des références se chevauchant résulteront en un fonctionnement irrégulier du temporisateur.
validation	Lorsque la validation reçoit le flux d'énergie, la valeur courante du temporisateur est incrémentée.
R	Lorsque R reçoit le flux d'énergie, elle réinitialise la valeur courante à "0".
PV	PV est la valeur à copier dans la valeur présélectionnée du temporisateur lorsque le temporisateur est validé ou réinitialisé.
Q	La sortie Q est excitée lorsque la valeur courante est supérieure ou égale à la valeur présélectionnée.
temps	La base de temps est en dixièmes (0,1), centièmes (0,01) ou millièmes (0,001) de seconde pour la valeur de pré-sélection PV.

Types de mémoire valides

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

• Référence valide ou transmision du flux à travers la fonction.

Exemple

Dans l'exemple suivant, un temporisateur d'activation rémanent est utilisé pour créer un signal (%Q0011) qui s'active 8,0 secondes après que %Q0010 s'active et se désactive lorsque %Q0010 se désactive.

```
|
| %Q0010 |      |                                                          %Q0011
|--| |---|ONDTR|----------------------------------------------------------( )-
|        | 0.1s|
| %Q0010 |      |
|--|/|---|R     |
|        |      |
|        |      |
| CONST -|PV    |
| +00080 |______|
|
|        %R0004
```

TMR

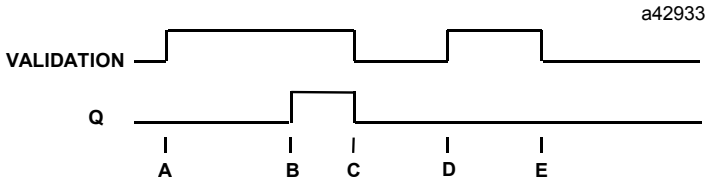
Le temporisateur d'activation simple (TMR) s'incrémente pendant qu'il reçoit le flux d'énergie et se réinitialise à "0" lorsque le flux d'énergie s'arrête. Le temps peut se compter en dixièmes de seconde (la sélection par défaut), en centièmes de seconde ou en millièmes de seconde. La plage est de 0 à +32,767 en unités de temps, par conséquent, la plage de temporisation est de 0,001 à 3276,7 secondes. 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 le TMR reçoit le flux d'énergie, il commence à accumuler le temps (valeur courante). La valeur courante est rafraîchie à chaque scrutation de la logique afin de refléter le temps total écoulé pendant lequel le temporisateur a été validé depuis sa dernière réinitialisation.

Remarque

Si de multiples occurences de ce même temporisateur, avec la même adresse, sont validés pendant un cycle automate, la valeur courante du temporisateur sera la même.

Ce rafraîchissement se produit tant que la logique de validation reste à "1". Lorsque la valeur courante atteint ou excède la valeur présélectionnée PV, la fonction commence à laisser passer le flux vers la droite. Le temporisateur continue à accumuler le temps jusqu'à ce que la valeur maximale soit atteinte. Lorsque le paramètre de validation passe de "1" à "0", le temporisateur arrête d'accumuler le temps et la valeur courante est remise à "0".



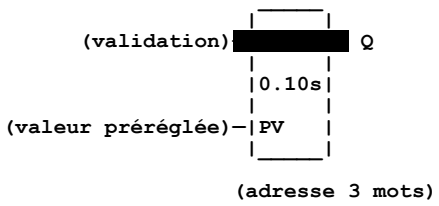
A = VALIDATION passe à "1" ; le temporisateur commence à accumuler le temps.

B = La valeur courante atteint la valeur présélectionnée PV ; Q passe à "1" et le temporisateur continue à accumuler le temps.

C = VALIDATION passe à "0" ; Q passe à "0" ; le temporisateur arrête d'accumuler le temps et le temps courant est effacé.

D = VALIDATION passe à "1" ;le temporisateur commence à accumuler le temps.

E = VALIDATION passe à "0" avant que la valeur courante n'atteigne la valeur présélectionnée PV ; Q reste à "0" ; le temporisateur arrête d'accumuler le temps et il est remis à "0".



Paramètres

Paramètre	Description
adresse	Le TMR utilise trois mots consécutifs de la mémoire %R pour stocker ce qui suit : • Valeur courante = mot 1. • Valeur présélectionnée = mot 2. • Mot de contrôle = mot 3. Lorsque vous entrez dans un TMR, vous devez entrer une adresse pour l'emplacement de ces trois mots consécutifs directement en dessous du graphique représentant la fonction. **Remarque :** Ne pas utiliser cette adresse dans d'autres instructions. **Précautions :** Des références se chevauchant résulteront en un fonctionnement irrégulier du temporisateur.
validation	Lorsque la validation reçoit le flux d'énergie, la valeur courante du temporisateur est incrémentée. Lorsque le TMR n'est pas validé, la valeur courante est remise à "0" et Q est désactivée.
PV	PV est la valeur à copier dans la valeur présélectionnée du temporisateur lorsque le temporisateur est validé ou réinitialisé.
Q	La sortie Q est excitée lorsque le TMR est validé et que la valeur courante est supérieure ou égale à la valeur présélectionnée.

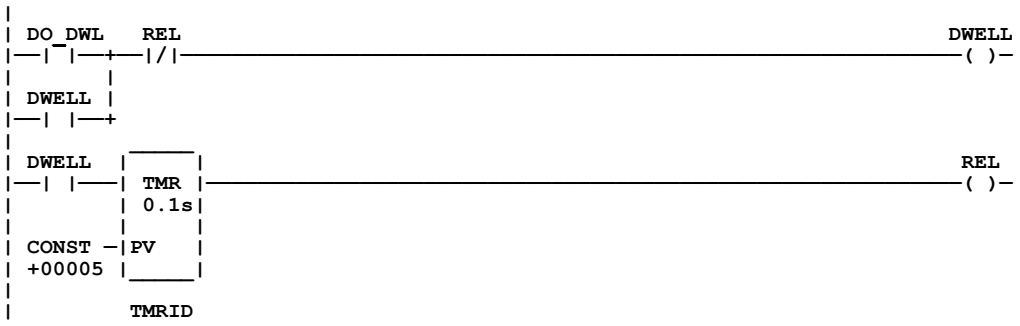
Types de mémoire valides

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

• Référence valide ou transmission de flux à travers la fonction.

Exemple

Dans l'exemple suivant, un temporisateur de retard adressé à TMRID est utilisé pour contrôler la durée pendant laquelle la bobine DWELL est à "1". Lorsque le contact normalement ouvert DO_DWL est à "1", la bobine DWELL est excitée. Le contact de la bobine DWELL garde la bobine DWELL excitée (même lorsque le contact DO_DWL est relâché) et démarre également le temporisateur TMRID. Lorsque TMRID atteint sa valeur présélectionnée d'une demi-seconde, la bobine REL est excitée, interrompant la condition de maintien de la bobine DWELL. Le contact DWELL interrompt le flux d'énergie vers TMRID, remettant à "0" sa valeur courante et désactivant la bobine REL. Le circuit est alors prêt pour une autre activation momentanée du contact DO_DWL.



OFDT

Le temporisateur de désactivation (OFDT) s'incrémente pendant que le flux d'énergie est à "0" et se remet à "0" lorsque le flux d'énergie est à "1". Le temps peut se compter en dixièmes de seconde (la sélection par défaut), en centièmes de seconde ou en millièmes de seconde. La plage est de 0 à +32,767 en 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 l'OFDT reçoit le flux d'énergie, il fait passer l'énergie vers la droite et la valeur courante (CV) est mise à "0". L'OFDT utilise le premier registre comme accumulateur CV—voir la section “Paramètres” à la page suivante pour plus d'informations. La sortie reste à "1" tant que la fonction reçoit le flux d'énergie. Si la fonction ne reçoit plus le flux d'énergie de la gauche, elle continue de faire passer l'énergie vers la droite et le temporisateur commence à accumuler le temps dans la valeur courante.

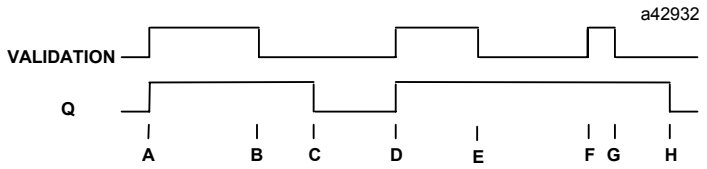
Remarque

Si de multiples occurences de ce même temporisateur, avec la même adresse, sont validés pendant un cycle automate, la valeur courante du temporisateur sera la même.

L'OFDT ne laisse pas passer le flux d'énergie si la valeur présélectionnée est à "0" ou négative.

A chaque fois que la fonction est éxécutée avec la logique de validation mise à "0", la valeur courante est rafraîchie pour refléter le temps écoulé depuis que le temporisateur a été mis à "0". Lorsque la valeur courante (CV) atteint ou excède la valeur présélectionnée (PV), la fonction arrête de laisser passer le flux vers la droite. Lorsque ceci se produit, le temporisateur arrête d'accumuler le temps—voir la Partie C ci-dessous.

Lorsque la fonction reçoit le flux d'énergie à nouveau, la valeur courante se remet à "0".



A = VALIDATION et Q passent tous les deux à "1" ; le temporisateur est remis à "0" (CV = 0).

B = VALIDATION passe à "0" ; le temporisateur commence à accumuler le temps.

C = CV atteint PV ; Q passe à "0" et le temporisateur arrête d'accumuler le temps.

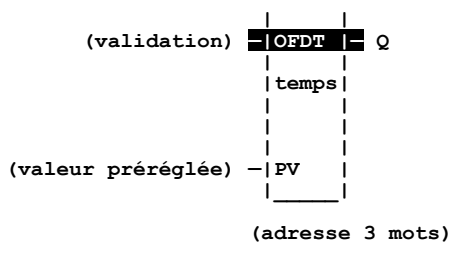
D = VALIDATION passe à "1" ; le temporisateur est remis à "0" (CV = 0).

E = VALIDATION passe à "0" ; le temporisateur commence à accumuler le temps.

F = VALIDATION passe à "1" ; le temporisateur est remis à "0" (CV = 0).

G = VALIDATION passe à "0" ; le temporisateur commence à accumuler le temps.

H = CV atteint PV ; Q passe à "0" et le temporisateur arrête d'accumuler le temps.



Lorsque l'OFDT est utilisé dans un bloc programme qui n'est ***pas*** appelé à chaque cycle, le temporisateur accumule le temps entre les appels du bloc programme, sauf s'il est remis à "0". Ceci signifie qu'il fonctionne comme un temporisateur qui fonctionne dans un programme avec une scrutation beaucoup plus lent que le temporisateur dans un bloc du programme principal. Pour les blocs programme qui restent inactifs pendant longtemps, le temporisateur doit être programmé pour permettre cette fonction de rattrapage. Par exemple, si un temporisateur, dans un bloc programme, est remis à "0" et que le bloc programme n'est pas appelé pendant quatre minutes, lorsque le bloc programme est appelé, quatre minutes de temps auront déjà été accumulées. Ce temps est appliqué au temporisateur lorsqu'il est validé, sauf si le temporisateur est d'abord remis à "0".

Paramètres

Paramètre	Description
adresse	L'OFDT utilise trois mots consécutifs de la mémoire %R pour stocker ce qui suit : • Valeur courante (CV) = mot 1. • Valeur présélectionnée = mot 2. • Mot de contrôle = mot 3. Lorsque vous entrez dans un OFDT, vous devez entrer une adresse pour l'emplacement de ces trois registres consécutifs directement en dessous du graphique représentant la fonction. **Remarque :** Ne pas utiliser cette adresse dans d'autres instructions. **Précautions :** Des références se chevauchant résulteront en un fonctionnement irrégulier du temporisateur.
validation	Lorsque la validation reçoit le flux d'énergie, la valeur courante du temporisateur est incrémentée.
temps	L'incrément de temps est en dixièmes (0,1), centièmes (0,01) ou millièmes (0,001) de seconde pour le bit inférieur de la valeur présélectionnée PV.
PV	PV est la valeur à copier dans la valeur présélectionnée du temporisateur lorsque le temporisateur est validé ou réinitialisé.
Q	La sortie Q est excitée lorsque la valeur courante est inférieure à la valeur présélectionnée. L'état de Q est rémanent en cas de coupure d'alimentation ; aucune initialisation automatique ne se produit à la mise sous tension.

Types de mémoire valides

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

• Référence valide ou transmission de flux à travers la fonction.

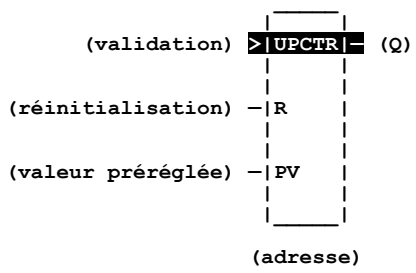
Exemple

Dans l'exemple suivant, un temporisateur OFDT est utiliser pour mettre une sortie (%Q0001) à "0" pendant qu'une entrée (%I0001) est mise à "1". La sortie est remise à "1" de nouveau, 0,3 secondes après que l'entrée est mise à "0".

UPCTR

Le compteur (UPCTR) est utilisé pour compter jusqu'à une valeur désignée. La plage est de 0 à +32,767 comptes. Lorsque la réinitialisation du compteur est à "1", la valeur courante du compteur est remise à "0". A chaque fois que l'entrée 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 à "1" lorsque la valeur courante est supérieure ou égale à la valeur présélectionnée.

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



Paramètres

Paramètre	Description
adresse	L'UPCTR utilise trois mots consécutifs (registres) de la mémoire %R pour stocker ce qui suit : • Valeur courante = mot 1. • Valeur présélectionnée = mot 2. • Mot de contrôle = mot 3. Lorsque vous entrez dans un UPCTR, vous devez entrer une adresse pour l'emplacement de ces trois mots consécutifs (registres) directement en dessous du graphique représentant la fonction. **Remarque :** Ne pas utiliser cette adresse avec un autre compteur, décompteur ou une autre instruction car un mauvais fonctionnement en résulterait. **Précautions :** Des références se chevauchant résulteront en un fonctionnement irrégulier du compteur.
validation	Sur une transition positive de validation, le compte courant est incrémenté de 1.
R	Lorsque R reçoit le flux d'énergie, elle remet la valeur courante à "0".
PV	PV est la valeur à copier dans la valeur présélectionnée du compteur lorsque le compteur est validé ou réinitialisé.
Q	La sortie Q est excitée lorsque la valeur courante est supérieure ou égale à la valeur présélectionnée.

Types de mémoire valides

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

• Référence valide ou transmission dub flux à travers la fonction.

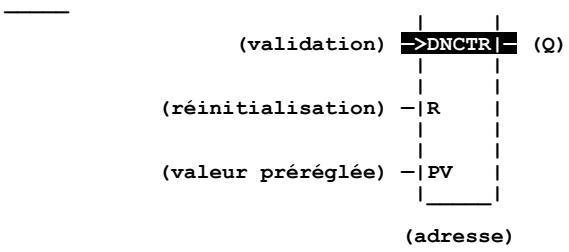
Exemple

Dans l'exemple suivant, chaque fois que l'entrée %I0012 passe de "0" à "1", le compteur PRT_CNT s'incrémentede 1; une fois à 100, la bobine interne %M0001 est excitée . Lorsque %M0001 est à "1", l'accumulateur est remis à zéro.

DNCTR

Le décompteur (DNCTR) est utilisé pour décompter à partir d'une valeur présélectionnée. La valeur présélectionnée minimale est zéro ; la valeur maximale est +32,767. La valeur de comptage minimale est –32,768. Lors de la réinitialisation, la valeur courante du compteur est mise à la valeur présélectionnée PV. Lorsque l'entrée validation passe de "0" à "1", la valeur courante est décrémentée de 1. La sortie est à "1" lorsque la valeur courante est inférieure ou égale à zéro.

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



Paramètres

Paramètre	Description
adresse	Le DNCTR utilise trois mots consécutifs de la mémoire %R pour stocker ce qui suit : Valeur courante = mot 1 Valeur présélectionnée = mot 2. Mot de contrôle = mot 3. Lorsque vous entrez un DNCTR, vous devez entrer une adresse pour l'emplacement de ces trois mots consécutifs directement en dessous du graphique représentant la fonction. **Remarque :** Ne pas utiliser cette adresse avec un autre compteur, décompteur ou une autre instruction car un mauvais fonctionnement en résulterait. **Précautions :** Des références se chevauchant résulteront en un fonctionnement irrégulier du compteur.
validation	Sur une transition positive de validation, la valeur courante est décrémentée de 1.
R	Lorsque R reçoit le flux d'énergie, elle remet la valeur courante à la valeur présélectionnée.
PV	PV est la valeur à copier dans la valeur présélectionnée du compteur lorsque le compteur est validé ou réinitialisé.
Q	La sortie Q est excitée lorsque la valeur courante est inférieure ou égale à zéro.

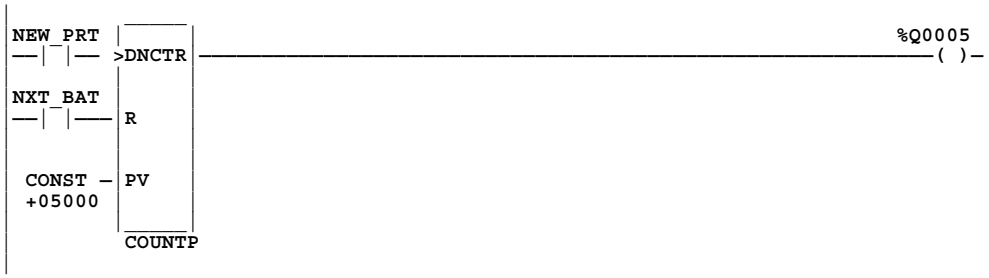
Types de mémoire valides

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

• Référence valide ou transmission du flux à travers la fonction.

Exemple

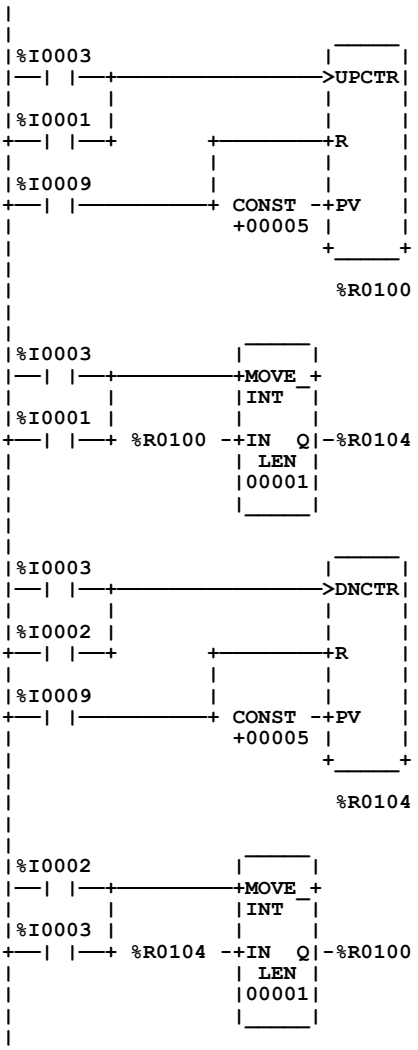
Dans l'exemple suivant, le décompteur appelé COUNTP compte jusqu'à 5000 avant d'exciter la sortie %Q0005.



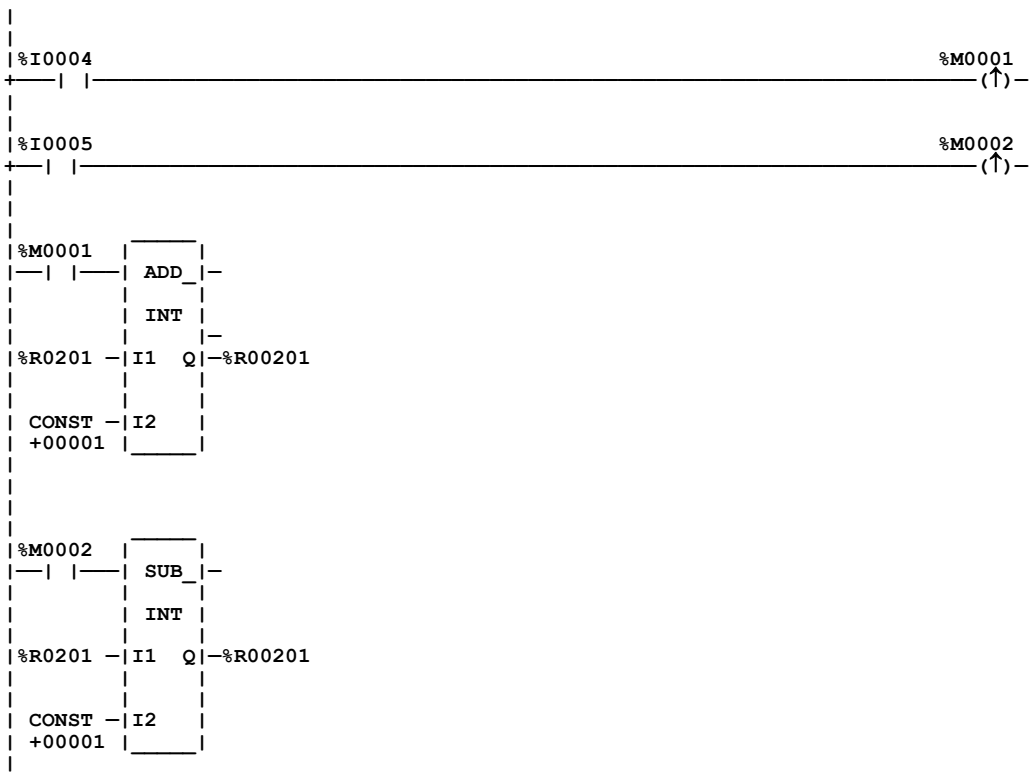
Exemple

Dans l'exemple suivant, l'API est utilisé pour conserver une trace du nombre de pièces contenues dans une zone de stockage temporaire. Il y a deux façons d'accomplir cette fonction en utilisant le jeu d'instructions de la Série 90-30/20/Micro.

La première méthode consiste à utiliser une paire de compteur/décompteur avec un registre partagé pour la valeur de comptage. Lorsque les pièces entrent dans la zone de stockage, le compteur s'incrémente de 1, augmentant la valeur courante de pièces en stock de 1. Lorsqu'une pièce quitte la zone de stockage, le décompteur se décrémente de 1, diminuant la valeur de l'inventaire du stock de 1. Afin d'éviter des conflits avec le registre partagé, les deux compteurs utilisent des adresses de registre différentes. Lorsqu'un registre compte, sa valeur courante doit être copiée vers le registre de valeur courante de l'autre compteur.



La deuxième méthode, montrée ci-dessous, utilise les fonctions ADD et SUB pour gérer l'état du stock.



Chapitre 6

Fonctions mathématiques

Ce chapitre décrit les fonctions mathématiques du jeu d'instructions de la Série 90-30/20/Micro :

Abréviation	Fonction	Description	Page
ADD	Addition	Additionne deux nombres.	6-2
SUB	Soustraction	Soustrait un nombre à un autre.	6-2
MUL	Multiplication	Multiplie deux nombres.	6-2
DIV	Division	Divise un nombre par un autre, donnant un quotient.	6-2
MOD	Division modulo	Divise un nombre par un autre, donnant un reste.	6-6
SQRT	Racine carrée	Trouve la racine carrée d'un entier ou d'une valeur réelle.	6-8
SIN, COS, TAN, ASIN, ACOS, ATAN	Fonctions trigonométriques †	Effectue la fonction appropriée sur la valeur réelle d'une entrée IN.	6-10
LOG, LN EXP, EXPT	Fonctions logarithmique/exponentielle †	Effectue la fonction appropriée sur la valeur réelle d'une entrée IN.	6-12
RAD, DEG	Conversion en radians †	Effectue la fonction appropriée sur la valeur réelle d'une entrée IN.	6-13

† Les fonctions trigonométriques, logarithmique/exponentielle et conversion de radian ne sont disponibles que sur les UC des modèles 35x et 36x, version 9 ou ultérieures et dans toutes les versions de l'UC352.

Remarque

La division et la division modulo sont des fonctions similaires qui diffèrent dans leur sortie ; la division trouve un quotient alors que la division modulo trouve un reste.

Fonctions mathématiques standards (ADD, SUB, MUL, DIV)

Les fonctions mathématiques comprennent l'addition, la soustraction, la multiplication et la division. Lorsqu'une fonction reçoit le flux d'énergie, la fonction mathématique appropriée est effectuée sur les paramètres d'entrée I1 et I2. Ces paramètres doivent être du même type de données. La sortie Q est du même type de données que I1 et I2.

Remarque

DIV arrondit à l'entier inférieur ; elle n'arrondit pas à l'entier le plus proche. (Par exemple, 24 DIV 5 = 4).

Les fonctions mathématiques s'effectuent sur ce type de donnée :

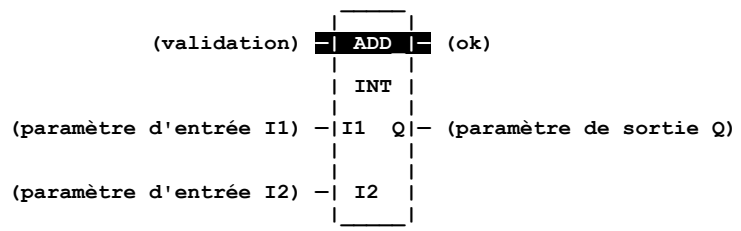
Type de donnée	Description
INT	Entier signé
DINT	Entier signé double précision
REAL	Virgule flottante

Remarque

Le type de données REAL n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou ultérieure, ou sur toutes les versions de l'UC352.

Le type de données par défaut est l'entier signé ; cependant, il peut être changé après avoir sélectionné la fonction. Pour plus d'informations sur les types de données, se référer au Chapitre 2, section 2, Organisation du programme et références/données utilisateur".

Si une opération sur INT ou DINT résulte en un dépassement, la référence de sortie est mise à la valeur la plus grande possible pour le type de données. Pour les nombres signés, le signe est mis pour montrer le sens du dépassement. Si l'opération ne résulte pas en un dépassement (et que les entrées sont des nombres valides), la sortie OK est mise à "1" ; autrement, elle est mise à "0". Si des entiers signés ou des entiers à double précision sont utilisés, le signe du résultat des fonctions DIV et MUL dépend des signes de I1 et I2.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
I1	I1 contient une constante ou une référence pour la première valeur utilisée dans l'opération. (I1 est à gauche de l'équation mathématique, comme dans I1 — I2).
I2	I2 contient une constante ou une référence pour la deuxième valeur utilisée dans l'opération. (I2 est à droite de l'équation mathématique, comme dans I1 — I2).
ok	La sortie OK est excitée lorsque la fonction est effectuée sans dépassement, sauf si une opération invalide se produit.
Q	La sortie Q contient le résultat de l'opération.

Types de mémoire 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	•	•	•		

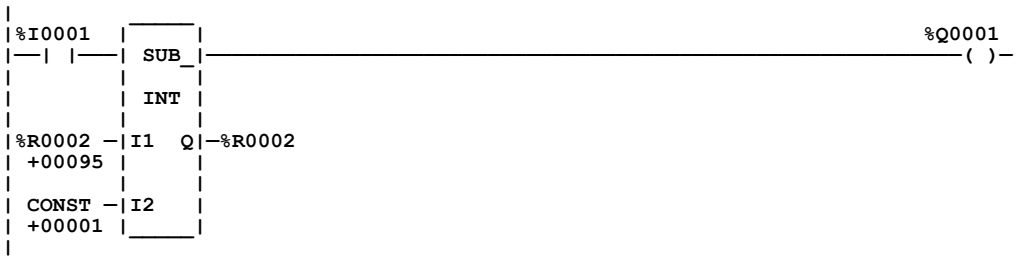
• Référence ou emplacement valide pour une transmission du flux dans la fonction.
o Référence valide que pour les données INT ; invalide pour DINT ou REAL.
† Les constantes sont limitées aux valeurs situées entre –32,768 et +32,767 pour les opérations d'entier signé à double précision.

Remarque

Le type par défaut est INT pour les instructions à 16 bits ou simple. Appuyer sur `F10` pour changer la sélection du type pour DINT, double mot 32 bits, ou REAL (pour les UC modèles 35x et 36x seulement). Les valeurs INT de l'API occupent un seul registre 16 bits, %R, %AI ou %AQ. Les valeurs DINT nécessitent deux registres consécutifs avec les 16 bits inférieurs dans le premier mot et les 16 bits supérieurs avec le signe dans le deuxième mot. Les valeurs REAL, dans les UC modèles 35x et 36x (Version 9 ou ultérieure) et toutes les versions de l'UC352, occupent également un registre double de 32 bits avec le signe dans le bit haut suivi de l'exposant et de la mantisse.

Exemple

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



Fonctions mathématiques et types de données

Fonction	Opération	Affiché
ADD INT	Q(16 bits) = I1(16 bits) + I2(16 bits)	Nombre de 5 chiffres décimaux avec signe
ADD DINT	Q(32 bits) = I1(32 bits) + I2(32 bits)	Nombre de 8 chiffres décimaux avec signe
ADD REAL*	Q(32 bits) = I1(32 bits) + I2(32 bits)	Nombre de 7 chiffres décimaux avec signe et décimales
SUB INT	Q(16 bits) = I1(16 bits) - I2(16 bits)	Nombre de 5 chiffres décimaux avec signe
SUB DINT	Q(32 bits) = I1(32 bits) - I2(32 bits)	Nombre de 8 chiffres décimaux avec signe
SUB REAL*	Q(32 bits) = I1(32 bits) - I2(32 bits)	Nombre de 7 chiffres décimaux avec signe et décimales
MUL INT	Q(16 bits) = I1(16 bits) * I2(16 bits)	Nombre de 5 chiffres décimaux avec signe
MUL DINT	Q(32 bits) = I1(32 bits) * I2(32 bits)	Nombre de 8 chiffres décimaux avec signe
MUL REAL*	Q(32 bits) = I1(32 bits) * I2(32 bits)	Nombre de 7 chiffres décimaux avec signe et décimales
DIV INT	Q(16 bits) = I1(16 bits) / I2(16 bits)	Nombre de 5 chiffres décimaux avec signe
DIV DINT	Q(32 bits) = I1(32 bits) / I2(32 bits)	Nombre de 8 chiffres décimaux avec signe
DIV REAL*	Q(32 bits) = I1(32 bits) / I2(32 bits)	Nombre de 7 chiffres décimaux avec signe et décimales

* UC modèles 35x et 36x seulement, version 9 ou supérieure, ou toutes les versions de l'UC352

Remarque

Les types de données d'entrée et de sortie doivent être les mêmes. Les fonctions MUL et DIV ne supportent pas un mode mixte comme sur les API 90-70. Par exemple, la MUL INT de 2 entrées 16 bits produit un résultat 16 bits et non 32 bits. L'utilisation de MUL DINT pour 32 bits nécessite deux entrées 32 bits. La DIV INT divise un I2 16 bits pour un résultat 16 bits alors que DIV DINT divise un I1 32 bits par un I2 32 bits pour un résultat 32 bits.

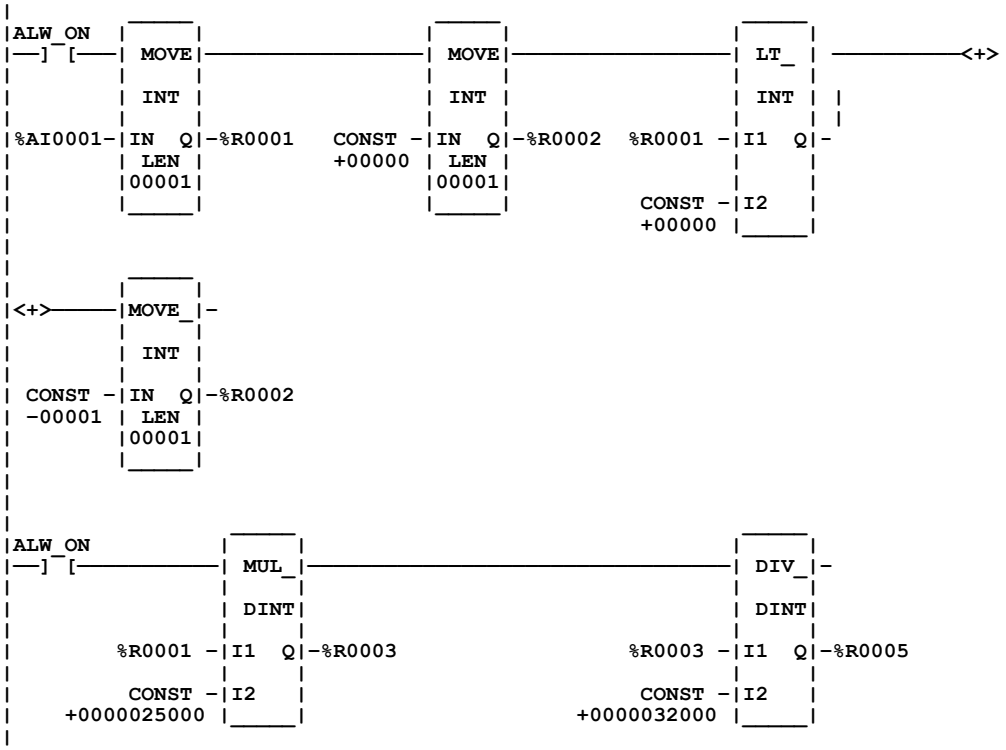
Ces fonctions passent l'énergie s'il n'y a pas de dépassement mathématique. Si un dépassement se produit, le résultat sera la plus grande valeur avec le signe correct et pas de flux d'énergie.

Veillez à éviter les dépassements lors de l'utilisation des fonctions MUL et DIV. Si vous devez convertir des valeurs INT en DINT, se rappeler que l'UC utilise le complément à 2 standard avec le signe étendu au bit le plus haut du deuxième mot. Vous devez vérifier le signe du mot de 16 bits inférieur et l'étendre au second mot de 16 bits. Si le bit le plus significatif d'un mot INT de 16 bits est 0 (positif), mettre un 0 dans le second mot. Si le bit le plus significatif d'un mot de 16 bits est –1 (négatif), mettre un –1 ou 0FFFFh en Hex dans le second mot. La conversion de DINT en INT est plus facile car le mot inférieur de 16 bits est la partie INT d'un mot DINT de 32 bits. Les 16 bits supérieurs ou le second mot doivent avoir une valeur soit de 0 (positive), soit de –1 (négative) ou le nombre DINT est trop grand pour être converti en 16 bits.

Exemple

Une application habituelleconsiste à convertir des valeurs d'entrée analogique avec une opération MUL suivie d'une DIV, et éventuellement d'une opération ADD. Avec une plage allant jusqu'à 32000, l'utilisation d'une MUL INT provoquera un dépassement. L'utilisation d'une valeur %AI pour une MUL DINT ne marchera pas non plus car le I1 de 32 bits combinera 2 sorties analogiques en même temps. Vous devez déplacer l'entrée analogique dans le mot inférieur d'un double registre, tester ensuite le signe et mettre le deuxième registre à 0 s'il est positif ou –1 s'il est négatif. Utiliser le double registre avec MUL DINT pour un résultat de 32 bits, pour la fonction DIV suivante.

Par exemple, la logique suivante pourrait être utilisée pour convertir une entrée de +/–10 volts %AI1 en unités ingénieur de +/– 25000 dans %R5.



MOD (INT, DINT)

La fonction modulo (MOD) est utilisée pour diviser une valeur par une autre valeur du même type de données afin d'obtenir le reste. Le signe du résultat est toujours identique au signe du paramètre d'entrée I1.

La fonction MOD fonctionne sur ces types de données :

Type de donnée	Description
INT	Entier signé
DINT	Entier signé double précision

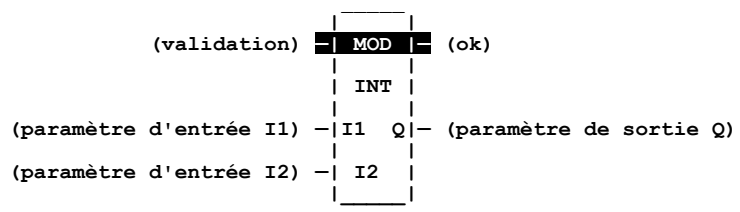
Le type de données par défaut est un entier signé ; cependant, il peut être changé après avoir sélectionné la fonction. Pour plus d'informations sur les types de données, se référer au Chapitre 2, section 2, Organisation du programme et références/données utilisateur".

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

```
Q = I1 - ((I1 DIV I2) * I2)
```

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

OK est toujours à "1" lorsque la fonction reçoit le flux d'énergie, sauf s'il y a une tentative de division par zéro. Dans ce cas, il est mis à "0".



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
I1	I1 contient une constante ou une référence pour la valeur à diviser par I2.
I2	I2 contient une constante ou une référence pour la valeur à diviser en I1.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans dépassement.
Q	La sortie Q contient le résultat de la division de I1 par I2 pour obtenir un reste.

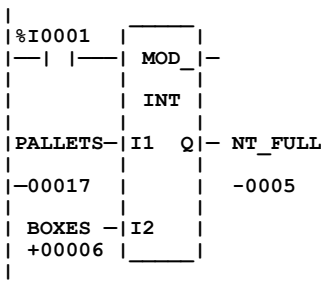
Types de mémoire valides

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 emplacement valide pour la transmission du flux dans la fonction.
o Référence valide que pour les données INT ; non valide pour DINT.
† Les constantes sont limitées aux valeurs situées entre –32,768 et +32,767 pour les opérations d'entier signé à double précision.

Exemple

Dans l'exemple suivant, le reste de la division entière de PALLETS par BOXES est placé dans NT_FULL si %I0001 est à "1".



SQRT (INT, DINT, REAL)

La fonction racine carrée (SQRT) est utilisée pour trouver la racine carrée d'une valeur. Lorsque la fonction reçoit le flux d'énergie, la partie entière de la racine carrée de l'entrée IN est mise dans la sortie Q. La sortie Q doit avoir le même format de données que IN.

La fonction SQRT fonctionne sur ces formats de données :

Type de donnée	Description
INT	Entier signé
DINT	Entier signé double précision
REAL	Virgule flottante

Remarque

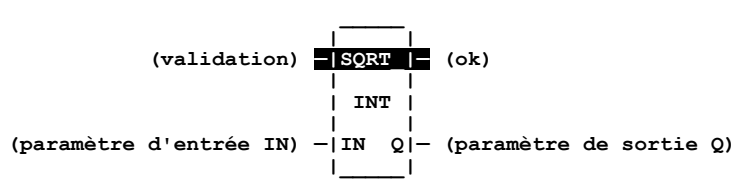
Le format de données REAL n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou supérieure, ou sur toutes les versions de l'UC352.

Le type de données par défaut est l'entier signé ; cependant, il peut être changé après avoir sélectionné la fonction. Pour plus d'informations sur les types de données, se référer au Chapitre 2, section 2, Organisation du programme et références/données utilisateur".

OK est mis à "1" si la fonction est effectuée sans dépassement, sauf dans les cas suivants :

- IN < 0.
- IN est NaN (Pas un Nombre).

Autrement, OK est mis à "0".



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
IN	IN contient une constante ou une référence pour la valeur dont la racine carrée doit être extraite. Si IN est inférieure à zéro, la fonction ne laissera pas passer le flux d'énergie.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans dépassement, sauf si une opération invalide se produit.
Q	La sortie Q contient la racine carrée de IN.

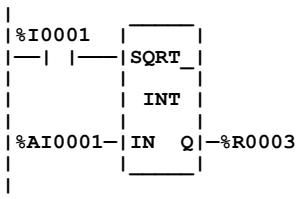
Types de mémoire valides

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

• Référence ou emplacement valide lorsque l'énergie peut passer dans la fonction.
o Référence valide que pour les données INT ; invalide pour DINT et REAL.
† Les constantes sont limitées aux valeurs situées entre –32,768 et +32,767 pour les opérations d'entier signé à double précision.

Exemple

Dans l'exemple suivant, la racine carrée du nombre entier situé dans %AI001 est placée dans le résultat situé dans %R0003 lorsque %I0001 est à "1".



Fonctions trigonométriques (SIN, COS, TAN, ASIN, ACOS, ATAN)

Les fonctions SIN, COS et TAN sont utilisées pour trouver le sinus, le cosinus et la tangente trigonométriques de leur entrée. Lorsque l'une de ces fonctions reçoit le flux d'énergie, elle calcule le sinus (ou le cosinus ou la tangente) de IN, dont les unités sont les radians, et stocke les résultats dans la sortie Q. IN et Q sont des valeurs à virgule flottante.

Les fonctions ASIN, ACOS et ATAN sont utilisées pour trouver le sinus, le cosinus et la tangente inverses de leur entrée. Lorsque l'une de ces fonctions reçoit le flux d'énergie, elle calcule le sinus inverse (ou le cosinus ou la tangente) de IN, dont les unités sont les radians, et stocke les résultats dans la sortie Q. IN et Q sont des valeurs à virgule flottante.

Les fonctions SIN, COS et TAN acceptent une large gamme de valeurs d'entrée, avec $-2^{63} < IN < +2^{63}$, ($2^{63} \approx 9.22 \times 10^{18}$).

Les fonctions ASIN et ACOS acceptent une large gamme de valeurs d'entrée, avec $-1 \leq IN \leq 1$. Pour une valeur valide du paramètre IN, la fonction ASIN_REAL produira un résultat Q semblable à celui-ci :

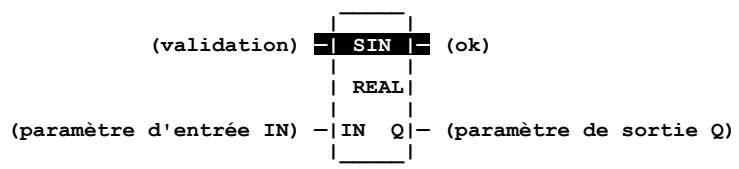
$$\mathbf{ASIN\ (IN)} = -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$

La fonction ACOS_REAL produira un résultat Q semblable à celui-ci :

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

La fonction ATAN accepte la plus large gamme de valeurs d'entrée, avec $-\infty \leq IN \leq +\infty$. Pour une valeur valide du paramètre IN, la fonction ATAN_REAL produira un résultat Q semblable à celui-ci :

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



Remarque

Les fonctions TRIG ne sont disponibles que pour les UC modèles 35x et 36x, Version 9 ou supérieure, ou pour toutes les versions de l'UC352.

Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
IN	IN contient la valeur réelle constante ou de référence sur laquelle effectuer l'opération.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans dépassement, sauf si une opération invalide se produit et/ou IN est NaN (Pas un Nombre).
Q	La sortie Q contient la valeur trigonométrique de IN.

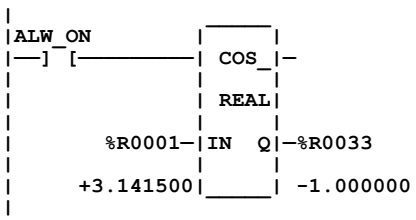
Types de mémoire valides

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

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Exemple

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

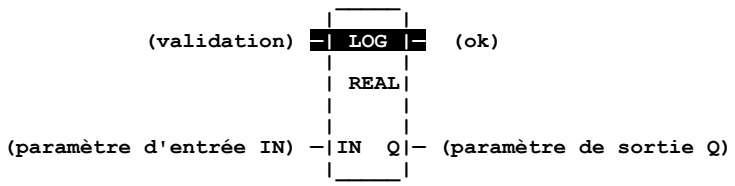


Fonctions logarithmique/exponentielle (LOG, LN, EXP, EXPT)

Les fonctions LOG, LN et EXP possèdent deux paramètres d'entrée et deux paramètres de sortie. Lorsque la fonction reçoit le flux d'énergie, elle applique l'opération logarithmique/exponentielle appropriée sur la valeur réelle de l'entrée IN et place le résultat dans la sortie Q.

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

La sortie ok recevra le flux d'énergie, sauf si IN est NaN (Pas un Nombre) ou négatif.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
IN	IN contient la valeur réelle sur laquelle effectuer l'opération.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans dépassement, sauf si une opération invalide se produit et/ou IN est NaN (Pas un Nombre) ou négatif.
Q	La sortie Q contient la valeur logarithmique/exponentielle de IN.

Remarque

Les fonctions LOG, LN, EXP et EXPT ne sont disponibles que pour les UC modèles 35x et 36x, Version 9 ou supérieure, et pour toutes les versions de l'UC352.

Remarque

Lorsque la valeur d'entrée, IN, pour la fonction EXP est une infinité négative (-∞), la fonction retourne une valeur de(à compléter!), comme prévu. Dans ce cas, pour l'UC352, la fonction ne laisse *pas* passer l'énergie. Toutes les autres UC 90-30 *laissent* passer l'énergie, même si la sortie est de 0.

Types de mémoire valides

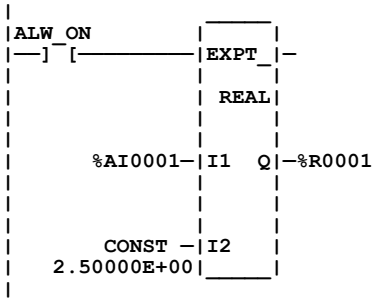
Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN*								•	•	•	•	
ok	•											•
Q								•	•	•		

* Pour la fonction EXPT, l'entrée IN est remplacée par les paramètres d'entrée I1 et I2.

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Exemple

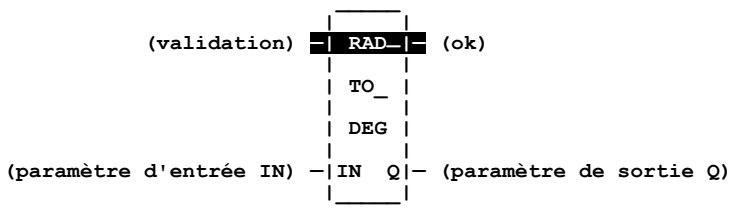
Dans l'exemple suivant, la valeur de %AI0001 est élevée à la puissance 2,5 et le résultat est placé dans %R0001.



Conversion en radians (RAD, DEG)

Lorsque la fonction reçoit le flux d'énergie, la conversion appropriée (RAD_TO_DEG ou DEG_TO_RAD, c'est-à-dire, Radian en Degré ou vice versa) est effectuée sur la valeur réelle de l'entrée IN et le résultat est placé dans la sortie Q.

La sortie ok recevra le flux d'énergie, sauf si IN est NaN (Pas un Nombre).



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
IN	IN contient la valeur réelle sur laquelle effectuer l'opération.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans dépassement, sauf si IN est NaN (Pas un Nombre).
Q	La sortie Q contient la valeur convertie de IN.

Remarque

Les fonctions Conversion en radians ne sont disponibles que pour les UC modèles 35x et 36x, Version 9 ou supérieure, ou pour toutes les versions de l'UC352.

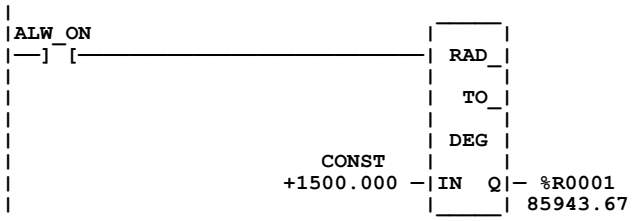
Types de mémoire valides

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

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Exemple

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



Chapitre 7 Fonctions relationnelles

Les fonctions relationnelles sont utilisées pour déterminer la relation entre deux valeurs. Ce chapitre décrit les fonctions relationnelles suivantes :

Abréviation	Fonction	Description	Page
EQ	Egal à	Teste l'égalité de deux nombres	7-2
NE	Différent de	Teste la différence entre deux nombres	7-2
GT	Supérieur à	Teste si un nombre est supérieur à un autre.	7-2
GE	Supérieur ou égal à	Teste si un nombre est supérieur ou égal à un autre.	7-2
LT	Inférieur à	Teste si un nombre est inférieur à un autre.	7-2
LE	Inférieur ou égal à	Teste si un nombre est inférieur ou égal à un autre.	7-2
RANGE	Plage	Détermine si un nombre se trouve dans une plage spécifiée (disponible pour les UC Version 4.5 ou supérieure).	7-4

Fonctions relationnelles standards (EQ, NE, GT, GE, LT, LE)

Lorsque la fonction reçoit le flux d'énergie, elle compare le paramètre d'entrée I1 au paramètre d'entrée I2, qui doivent être du même format. Les fonctions relationnelles s'effectuent sur ces types de données :

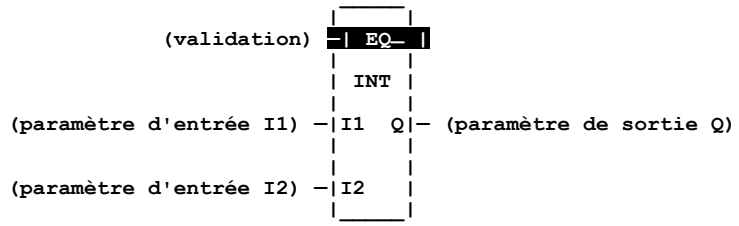
Type de donnée	Description
INT	Entier signé
DINT	Entier signé double précision
REAL	Virgule flottante

Remarque

Le type de données REAL n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou supérieure, et sur toutes les versions de l'UC352. Le bit %S0020 est mis à "1" lorsqu'une fonction relationnelle utilisant la donnée REAL s'exécute correctement. Il est mis à "0" lorsque l'une des entrées est NaN (Pas un Nombre). Le bloc fonctionnel Range n'accepte pas le type REAL.

Le type de données par défaut est l'entier signé ; Pour comparer les nombres entiers signés, les nombres entiers signés à double précision ou les nombres réels, sélectionner le format correspondant après avoir sélectionné la fonction relationnelle. Pour comparer des données d'un autre format ou de deux formats différents, utiliser d'abord la fonction de conversion appropriée (décrite dans le Chapitre 11, "Fonctions de conversion") pour mettre les données dans l'un des formats supportés.

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



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
I1	I1 contient une constante ou une référence pour la première valeur à comparer. (I1 est à gauche de l'équation relationnelle, comme dans I1 < I2).
I2	I2 contient une constante ou une référence pour la deuxième valeur à comparer. (I2 est à droite de l'équation relationnelle, comme dans I1 < I2).
Q	La sortie Q est excitée lorsque I1 et I2 correspondent à la relation spécifiée.

Remarque

I1 et I2 doivent être des nombres valides, c'est-à-dire, pas NaN (Pas un Nombre).

Description étendue

Fonction	Description
Egal à	Lorsqu'elle est validée, si la valeur à l'entrée I1 est égale à la valeur à l'entrée I2, la sortie Q est excitée.
Différent de	Lorsqu'elle est validée, si la valeur à l'entrée I1 n'est PAS égale à la valeur à l'entrée I2, la sortie Q est excitée.
Supérieur à	Lorsqu'elle est validée, si la valeur à l'entrée I1 est supérieure à la valeur à l'entrée I2, la sortie Q est excitée.
Supérieur ou égal à	Lorsqu'elle est validée, si la valeur à l'entrée I1 est supérieure ou égale à la valeur à l'entrée I2, la sortie Q est excitée.
Inférieur à	Lorsqu'elle est validée, si la valeur à l'entrée I1 est inférieure à la valeur à l'entrée I2, la sortie Q est excitée.
Inférieur ou égal à	Lorsqu'elle est validée, si la valeur à l'entrée I1 est inférieure ou égale à la valeur à l'entrée I2, la sortie Q est excitée.

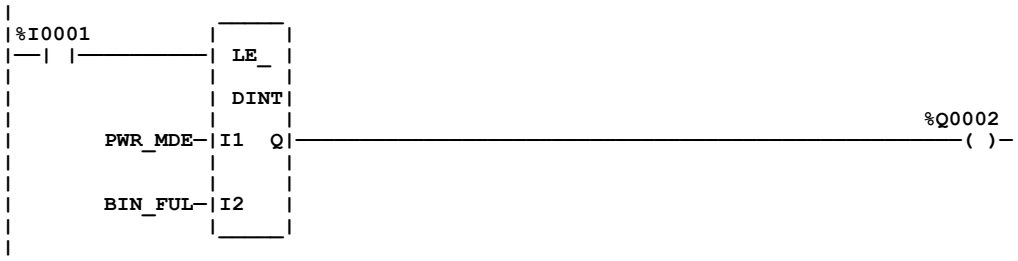
Types de mémoire valides

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 emplacement valide pour la transmission du flux dans la fonction.
o Référence valide uniquement pour les données INT ; invalide pour DINT ou REAL.
† Les constantes sont limitées aux valeurs entières pour les opérations sur entiers signés à double précision.

Exemple

Dans l'exemple suivant, deux entiers signés à double précision, PWR_MDE et BIN_FUL, sont comparés lorsque %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)

La fonction RANGE est utilisée pour déterminer si une valeur se trouve dans la plage déterminée par deux nombres.

Remarque

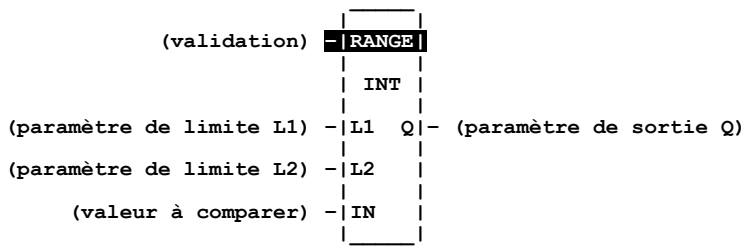
Cette fonction n'est disponible *qu'avec* les UC Version 4.41 ou supérieure.

La fonction RANGE fonctionne sur ces types de données :

Type de donnée	Description
INT	Entier signé.
DINT	Entier signé double précision.
WORD	Type de donnée Mot.

Le type de données par défaut est l'entier signé ; cependant, il peut être changé après avoir sélectionné la fonction. Pour plus d'informations sur les types de données, se référer au Chapitre 2, section 2, Organisation du programme et références/données utilisateur".

Lorsque la fonction est validée, le bloc fonctionnel RANGE comparera la valeur du paramètre d'entrée IN à la plage spécifiée par les paramètres de limites L1 et L2. Lorsque la valeur se trouve dans la plage spécifiée par L1 et L2, de façon inclusive, le paramètre de sortie Q est mis à "1". autrement, il est mis à "0".



Remarque

Les paramètres de limite L1 et L2 représentent les limites d'une plage. Il n'y a pas de notion de limite haute ou limite basse affectée à ces paramètres. Ainsi, une plage souhaitée de 0 à 100 pourrait être spécifiée en affectant 0 à L1 et 100 à L2 ou 0 à L2 et 100 à L1.

Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
L1	L1 contient le début de la plage.
L2	L2 contient la fin de la plage.
IN	IN contient la valeur à comparer avec la plage spécifiée par L1 et L2.
Q	La valeur Q est excitée lorsque la valeur de IN se trouve dans la plage spécifiée par L1 et L2, inclusive.

Types de mémoire 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 emplacement valide lpour la transmission du flux dans la fonction.
o Référence valide uniquement pour les données INT ou WORD ; invalide pour DINT.
‡ Les constantes sont limitées aux valeurs entières pour les opérations sur entiers signés à double précision.

Exemple 1

Dans l'exemple suivant, on vérifie si %AI0001 se trouve dans la plage spécifiée par deux constantes, 0 et 100.

Table de vérité de RANGE				
Etat de la validation %I0001	**Valeur constante L1**	**Valeur constante L2**	**Valeur IN %AI0001**	**Etat Q %Q0001**
"1"	100	0	< 0	"0"
"1"	100	0	0 — 100	"1"
"1"	100	0	> 100	"0"
"0"	100	0	Non applicable	"0"

Exemple 2

Dans cet exemple, on vérifie si %AI0001 se trouve dans la plage spécifiée par deux valeurs de registre.

Table de vérité de RANGE				
Etat de la validation %I0001	**Valeur L1 %R0001**	**Valeur L2 %R0002**	**Valeur IN %AI0001**	**Etat Q %Q0001**
"1"	500	0	< 0	"0"
"1"	500	0	0 — 500	"1"
"1"	500	0	> 500	"0"
"0"	500	0	Non applicable	"0"

Chapitre 8

Fonctions d'opérations logiques

Les fonctions d'opérations logiques permettent de faire des opérations de comparaison, de logique et de déplacement sur des chaînes de bits. Les instructions AND, OR, XOR et NOT fonctionnent sur un seul mot. Les autres fonctions d'opérations logiques peuvent fonctionner sur des mots multiples, avec une longueur de chaîne maximale de 256 mots. Toutes les fonctions d'opérations logiques nécessitent des données au format MOT.

Bien que toutes les données doivent être spécifiées en incréments de 16 bits, ces instructions fonctionnent sur les données comme une chaîne continue de bits, le bit 1 du premier mot étant le bit le moins significatif (LSB). Le dernier bit du dernier mot est le bit le plus significatif (MSB). Par exemple, si vous avez spécifié trois mots de données commençant à la référence %R0100, ils seront traités comme 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

Tout chevauchement d'adresses de références d'entrées et de sorties dans des fonctions utilisant plusieurs mots peuvent produire des résultats imprévus.

Les fonctions d'opérations logiques suivantes sont décrites dans ce chapitre :

Abréviation	Fonction	Description	Page
AND	ET logique	Si un bit de la chaîne de bits I1 et le bit correspondant de la chaîne de bits I2 sont tous les deux à 1, met un 1 dans l'emplacement correspondant de la chaîne de sortie Q.	8-3
OR	OU logique	Si un bit de la chaîne de bits I1 et/ou le bit correspondant de la chaîne de bits I2 sont tous les deux à 1, met un 1 dans l'emplacement correspondant de la chaîne de sortie Q.	8-3
XOR	OU logique exclusif	Si un bit de la chaîne de bits I1 et le bit correspondant de la chaîne de bits I2 sont différents, met un 1 dans l'emplacement correspondant de la chaîne de bits de sortie.	8-5
NOT	Inversion logique	Met l'état de chaque bit de la chaîne de bits de sortie Q dans l'état opposé du bit correspondant dans la chaîne de bits I1.	8-7
SHL	Décalage à gauche	Décale tous les bits d'un mot ou d'une chaîne de mots vers la gauche d'un nombre spécifique d'emplacements.	8-8
SHR	Décalage à droite	Décale tous les bits d'un mot ou d'une chaîne de mots vers la droite d'un nombre spécifique d'emplacements.	8-8
ROL	Rotation à gauche	Fait tourner tous les bits d'une chaîne d'un nombre spécifique d'emplacements vers la gauche.	8-10
ROR	Rotation à droite	Fait tourner tous les bits d'une chaîne d'un nombre spécifique d'emplacements vers la droite.	8-57
BTST	Test de bit	Teste un bit dans une chaîne de bits pour déterminer si ce bit est actuellement à 1 ou à 0.	8-12
BSET	Bit à 1	Met un bit, dans une chaîne de bits, à 1.	8-14
BCLR	Bit à 0	Efface un bit, dans une chaîne, en mettant ce bit à 0.	8-14
BPOS	Position de bit	Situe un bit mis à 1 dans une chaîne de bits.	8-16
MSKCMP	Comparaison masquée	Compare le contenu de deux chaînes de bits séparées avec la possibilité de masquer des bits sélectionnés (disponible pour les UC Version 4.5 ou supérieures).	8-18

AND et OR (WORD)

A chaque scrutation, la fonction AND ou OR examine chaque bit de la chaîne de bits I1 et le bit correspondant dans la chaîne de bits I2, en commençant par le bit le moins significatif .

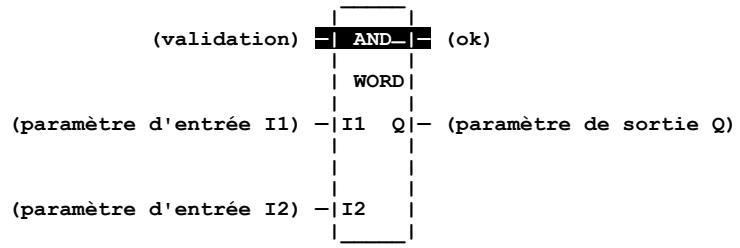
Pour chaque bit examiné pour la fonction AND, si les deux sont à 1, un 1 est mis dans l'emplacement correspondant dans la chaîne de sortie Q. Si l'un des bits ou les deux sont à 0, alors un 0 est placé dans la chaîne Q à cet emplacement.

La fonction AND est utile pour construire des masques, où seuls certains bits peuvent passer (ceux qui sont face à un 1 dans le masque), et où tous les autres bits sont mis à 0. La fonction peut également être utilisée pour effacer la zone sélectionnée de la mémoire mot en exécutant l'instruction ET sur les bits avec une autre chaîne de bits connue pour ne contenir que des 0. Les chaînes de bits spécifiées I1 et I2 peuvent se chevaucher.

Pour chaque bit examiné pour la fonction OR, si l'un des bits ou les deux sont à 1, un 1 est mis dans l'emplacement correspondant dans la chaîne de sortie Q. Si les deux bits sont à 0, alors un 0 est placé dans la chaîne Q à cet emplacement.

La fonction OR est utile pour associer des chaînes et contrôler de nombreuses sorties en utilisant une simple structure logique. La fonction équivaut à deux contacts en parallèle, multipliés par le nombre de bits de la chaîne. Elle peut être utilisée pour commander des voyants directement à partir des états des entrées ou de superposer des conditions de clignotement sur les voyants d'état.

La fonction laisse passer le flux d'énergie quand le flux est reçu.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
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 deuxième chaîne.
ok	La sortie ok est excitée lorsque validation est excitée.
Q	La sortie Q contient le résultat de l'opération.

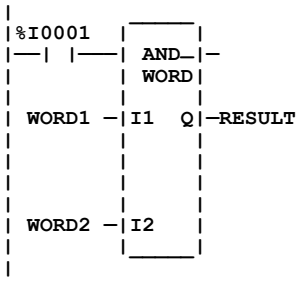
Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction.
† %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0001 est mise à "1", les chaînes de 16 bits représentées par les symboles WORD1 et WORD2 sont examinées. Les résultats du AND 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**

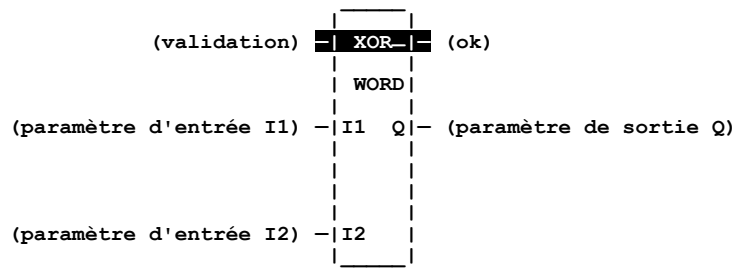
XOR (WORD)

La fonction OR exclusif (XOR) est utilisée pour comparer chaque bit de la chaîne bits I1 au bit correspondant de la chaîne I2. Si les bits sont différents, un 1 est mis dans la position correspondante de la chaîne de bits de sortie.

A chaque scrutation où l'énergie est reçue, la fonction examine chaque bit de la chaîne de bits I1 et le bit correspondant dans la chaîne de bits I2, en commençant par le bit le moins significatif de chacune. Comme les deux bits sont examinés, si un seul est à 1, alors un 1 est placé dans l'emplacement correspondant de la chaîne de bits Q. La fonction XOR laisse passer le flux d'énergie vers la droite lorsque le flux est reçu.

Si la chaîne I2 et la chaîne de sortie Q commence à la même référence, un 1 mis dans la chaîne I1 provoquera l'alternance du bit correspondant dans la chaîne I2 entre 0 et 1, changeant d'état à chaque scrutation aussi longtemps que l'énergie est reçue. Des cycles plus longs peuvent être programmés en impulsant le flux d'énergie vers la fonction, deux fois plus vite que la fréquence souhaitée de clignotement ; l'impulsion du flux d'énergie doit durer le temps d'une scrutation (bobine impulsionnelle ou temporisateur à réinitialisation automatique).

La fonction XOR est utile pour comparer rapidement deux chaînes de bits ou pour faire clignoter un groupe de bits à la fréquence d'un état 1 toutes les deux scrutations.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
I1	I1 contient une constante ou une référence pour le premier mot traité par XOR.
I2	I2 contient une constante ou une référence pour le deuxième mot traité par XOR.
ok	La sortie ok est excitée lorsque validation est excitée.
Q	La sortie Q contient le résultat de I1 traité par XOR avec I2.

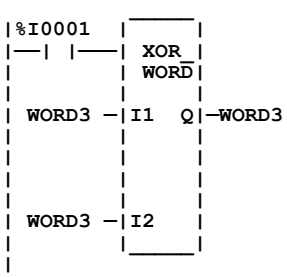
Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction.
† %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple

Dans l'exemple suivant, lorsque %I0001 est à "1", la chaîne de bits représentée par le symbole WORD3 est remise à zéro.



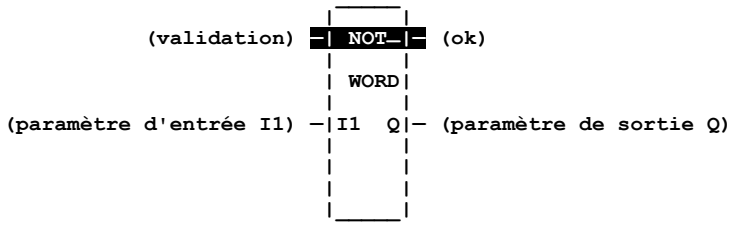
I1 (WORD3)	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**
I2 (WORD3)	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

Q (WORD3)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

NOT (WORD)

La fonction NOT est utilisée pour mettre l'état de chaque bit de la chaîne de bits de sortie Q à l'opposé de l'état du bit correspondant dans la chaîne de bits I1.

Tous les bits sont modifiés à chaque scrutation où l'énergie est reçue, faisant de la chaîne de sortie Q le complément logique de I1. La fonction fait passer le flux d'énergie vers la droite lorsque l'énergie est reçue.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
I1	I1 contient une constante ou une référence pour le mot à inverser.
ok	La sortie ok est excitée lorsque validation est excitée.
Q	La sortie Q contient le NON (négation) de I1.

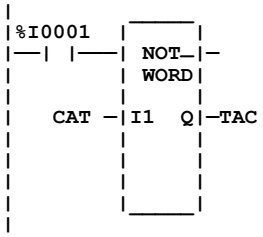
Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction. † %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

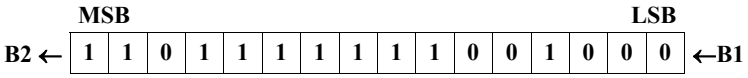
Exemple

Dans l'exemple suivant, lorsque l'entrée %I0001 est à "1", la chaîne de bits représentée par le symbole TAC est mise à l'inverse de la chaîne de bits CAT.

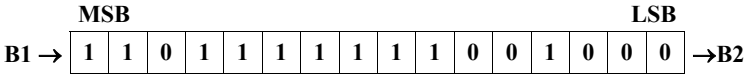


SHL et SHR (WORD)

La fonction de Décalage à gauche (SHL) est utilisée pour décaler tous les bits d'un mot ou d'un groupe de mots vers la gauche, d'un nombre spécifié d'emplacements. Lorsque le décalage se produit, le nombre spécifié de bits est décalé hors de la chaîne de sortie vers la gauche. Comme les bits sont décalés hors de l'extrémité haute de la chaîne, le même nombre de bits est décalé dans l'extrémité basse.



La fonction de Décalage à gauche (SHR) est utilisée pour décaler tous les bits d'un mot ou d'un groupe de mots vers la droite, d'un nombre spécifié d'emplacements. Lorsque le décalage se produit, le nombre spécifié de bits est décalé hors de la chaîne de sortie vers la droite. Comme les bits sont décalés hors de l'extrémité basse de la chaîne, le même nombre de bits est décalé dans l'extrémité haute.



Une longueur de chaîne de 1 à 256 mots peut être sélectionnée pour ces fonctions.

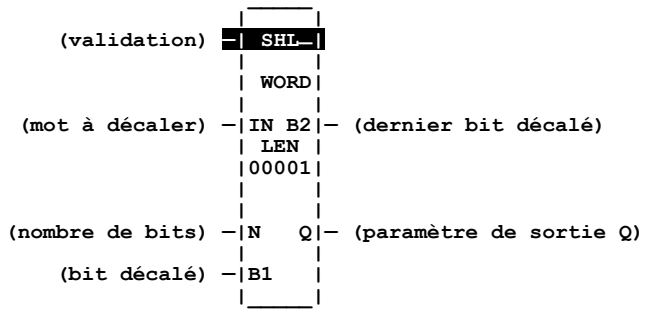
Si le nombre de bits à décaler (N) est supérieur au nombre de bits du groupe (LEN) * 16 ou si le nombre de bits à décale est de zéro, le groupe (Q) est rempli de copies du bit d'entrée (B1) et le bit de sortie est copié sur le flux d'énergie de la sortie (B2). Si le nombre de bits à décaler est zéro, aucun décalage n'est effectué ; le groupe d'entrée est copié dans le groupe de sortie ; le groupe d'entrée (B1) est copié dans le flux d'énergie.

Les bits décalés au début de la chaîne sont spécifiés par le paramètre d'entrée B1. Si une longueur supérieure à 1 a été spécifiée comme nombre de bits à décaler, chaque bit est mis à la même valeur (0 ou 1). Ceci peut être :

- La sortie Booléenne d'une autre fonction de programme.
- Tous à 1. Pour faire ceci, utiliser le symbole de référence spécial ALW_ON comme permissif vers la sortie B1.
- Tous à 0. Pour faire ceci, utiliser le symbole de référence spécial ALW_OFF comme permissif vers la sortie B1.

La fonction SHL ou SHR fait passer le flux d'énergie vers la droite, sauf si le nombre de bits spécifié à décaler est de zéro.

La sortie Q est la copie décalée de la chaîne d'entrée. Si vous voulez décaler la chaîne d'entrée, le paramètre de sortie Q doit utiliser le même emplacement mémoire que le paramètre d'entrée IN. La chaîne entière décalée est écrite à chaque scrutation où l'énergie est reçue. La sortie B2 est le dernier bit décalé. Par exemple, si quatre bits ont été décalés, B2 sera le quatrième bit décalé.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, le décalage est effectué.
IN	IN contient le premier mot à décaler.
N	N contient le nombre de décalage en bits dans le groupe.
B1	B1 contient la valeur du bit à décaler dans le groupe.
B2	B2 contient la valeur du dernier bit à décaler hors du groupe.
Q	La sortie Q contient le premier mot du groupe décalé.
LEN	LEN est le nombre de mots dans le groupe à décaler.

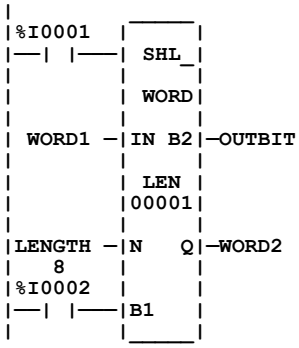
Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction.
† %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0001 est à "1", la chaîne de bits de sortie représentée par le symbole WORD2 reçoit une copie de WORD1, décalée vers la gauche du nombre de bits représentés par le mnémonique LENGTH. Les bits au début de la chaîne de sortie sont mis à la valeur de %I0002.



ROL et ROR (WORD)

La fonction de Rotation à gauche (ROL) est utilisée pour faire tourner tous les bits d'une chaîne, d'un nombre spécifié d'emplacements vers la gauche. Lorsque la rotation se produit, le nombre de bits spécifiés sort de la chaîne d'entrée vers la gauche et revient dans la chaîne par la droite.

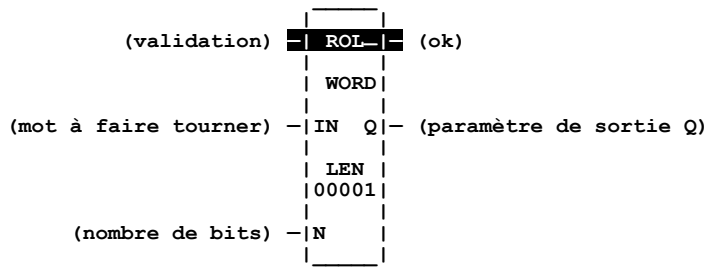
La fonction Rotation à droite (ROR) fait tourner les bits de la chaîne vers la droite. Lorsque la rotation se produit, le nombre de bits spécifiés sort de la chaîne d'entrée vers la droite et revient dans la chaîne par la gauche.

Une longueur de chaîne de 1 à 256 mots peut être sélectionnée pour ces fonctions.

Le nombre d'emplacements spécifiés pour la rotation doit être supérieur à zéro et inférieur au nombre de bits de la chaîne. Autrement, aucun déplacement ne se produit et aucun flux d'énergie n'est généré.

La fonction ROL ou ROR fait passer le flux d'énergie vers la droite, sauf si le nombre de bits spécifié est supérieur à la longueur totale de la chaîne ou inférieur à zéro.

Le résultat est mis dans la chaîne de sortie Q. Si vous voulez faire tourner la chaîne d'entrée, le paramètre de sortie Q doit utiliser le même emplacement mémoire que le paramètre d'entrée IN. La chaîne entière tournée est écrite à chaque scrutation où l'énergie est reçue.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, la rotation est effectuée.
IN	IN contient le premier mot de la rotation.
N	N contient le nombre de rotation en bits dans le groupe.
ok	La sortie ok est validée lorsque la rotation est excitée et que la longueur de la rotation n'est pas supérieure à la taille du groupe.
Q	La sortie Q contient le premier mot du groupe après rotation.
LEN	LEN est le nombre de mots dans le groupe de rotation.

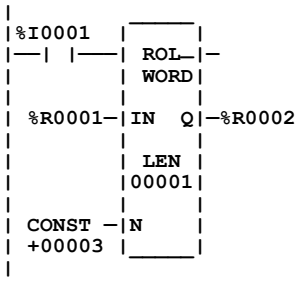
Types de mémoire valides

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

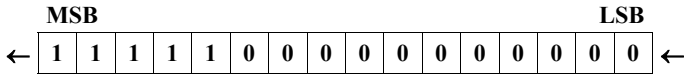
• Référence valide ou transmission possible du flux à travers la fonction.
† %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple

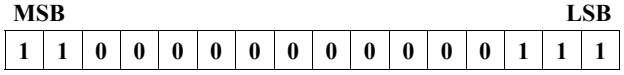
Dans l'exemple suivant, lorsque l'entrée %I0001 est à "1", la chaîne de bits d'entrée %R0001 est tournée de 3 bits et le résultat est placé dans %R0002. Après l'exécution de cette fonction, la chaîne de bits d'entrée %R0001 est inchangée. Si la même référence est utilisée pour IN et Q, une rotation s'effectuera sur place.



%R0001 :



%R0002 (après que %I0001 est soit passé à "1") :

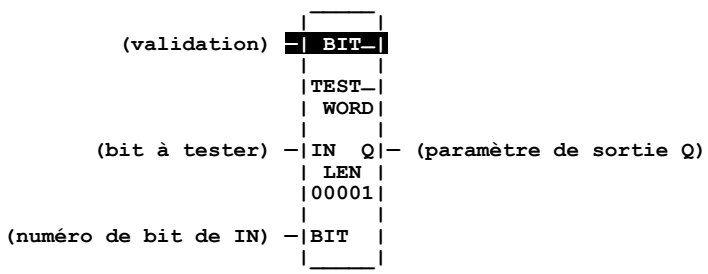


BTST (WORD)

La fonction Test de bit (BTST) est utilisée pour tester un bit dans une chaîne de bits afin de déterminer si ce bit est à "1" ou à "0". Le résultat du test est placé dans la sortie Q.

A chaque balayage où l'énergie est reçue, la fonction BTST met sa sortie Q dans le même état que le bit spécifié. Si un registre est utilisé à la place d'une constante pour spécifier le nombre de bits, le même bloc fonctionnel peut tester différents bits lors de cycles successifs. Si la valeur d'un BIT se trouve à l'extérieur de la plage (1 ≤ BIT ≤ (16 * LEN)), Q est mise à "0".

Une longueur de chaîne de 1 à 256 mots peut être sélectionnée.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, le test de bit est effectué.
IN	IN contient le premier mot de données à traiter.
BIT	BIT contient le numéro de bit de IN qui doit être testé. La plage valide est (1 ≤ BIT ≤ (16 * LEN)).
Q	La sortie Q est excitée si le bit testé est à "1".
LEN	LEN est le nombre de mots dans la chaîne à tester.

Remarque

Lors de l'utilisation de la fonction Bit Test, Bit Set, Bit Clear ou Bit Position, les bits sont numérotés de 1 à 16, *PAS* de 0 à 15.

Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction.

Exemple

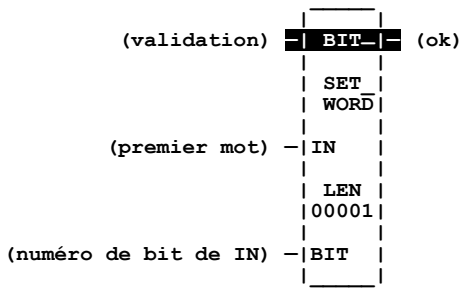
Dans l'exemple suivant, lorsque l'entrée %I0001 est à "1", le bit situé dans l'emplacement contenu dans la référence PICKBIT est testé. Le bit fait partie de la chaîne PRD_CDE. S'il est à "1", la sortie Q laisse passer le flux d'énergie et la bobine %Q0001 est mise à "1".

BSET et BCLR (WORD)

La fonction de mise à "1" de bit (BSET) est utilisée pour mettre un bit d'une chaîne de bits à "1". La fonction de mise à "0" de bit (BCLR) est utilisée pour mettre un bit d'une chaîne de bits à "0".

A chaque cycle où le flux d'énergie est reçu, la fonction met le bit spécifié à "1" pour la fonction BSET ou à "0" pour la fonction BCLR. Si une variable (registre) est utilisée à la place d'une constante pour spécifier le nombre de bits, le même bloc fonctionnel peut mettre différents bits à "1" lors de cycles successifs.

Une longueur de chaîne de 1 à 256 mots peut être sélectionnée. La fonction laisse passer le flux d'énergie vers la droite, sauf si la valeur de BIT est en dehors de la plage (1 ≤ BIT ≤ (16 * LEN)). Dans ce cas, ok est mis à "0".



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération logique est effectuée.
IN	IN contient le premier mot de données à traiter.
BIT	BIT contient le numéro de bit de IN qui doit être mis à "1" ou à "0". La plage valide est (1 ≤ BIT ≤ (16 * LEN)).
ok	La sortie ok est excitée lorsque validation est excitée.
LEN	LEN est le nombre de mots de la chaîne de bits.

Remarque

Lors de l'utilisation de la fonction Bit Test, Bit Set, Bit Clear ou Bit Position, les bits sont numérotés de 1 à 16, *PAS* de 0 à 15.

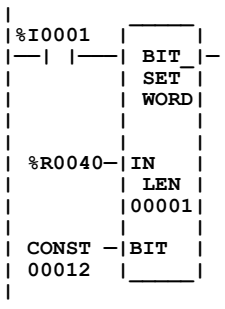
Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction.
† %SA, %SB ou %SC seulement ; %S ne peut pas être utilisé.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0001 est mise à "1", le bit 12 de la chaîne commençant à la référence %R0040 est mis à "1".



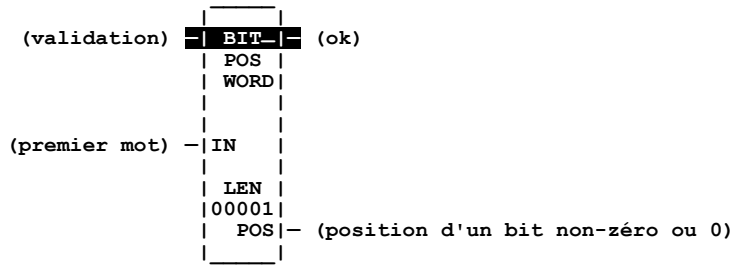
BPOS (WORD)

La fonction de position de bit (BPOS) est utilisée pour localiser un bit à "1" dans une chaîne de bits.

A chaque cycle où l'énergie est reçue, la fonction scrute la chaîne de bits en commençant par IN. Lorsque la fonction arrête la scrutation, soit un bit égal à 1 est trouvé, soit la chaîne entière a été scrutée.

POS est mis à la position du premier bit à 1 de la chaîne de bits ; POS est mis à "0" si aucun bit à 1 n'a été trouvé.

Une longueur de chaîne de 1 à 256 mots peut être sélectionnée. La fonction ne laisse passer le flux d'énergie que si la validation est à "1".



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, une opération de recherche de bit à 1 est effectuée.
IN	IN contient le premier mot de données à traiter.
ok	La sortie ok est excitée lorsque validation est excitée.
POS	La position du premier bit à 1 trouvé ou zéro si aucun bit à 1 n'est trouvé.
LEN	LEN est le nombre de mots de la chaîne de bits.

Remarque

Lors de l'utilisation de la fonction Bit Test, Bit Set, Bit Clear ou Bit Position, les bits sont numérotés de 1 à 16, *PAS* de 0 à 15.

Types de mémoire valides

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

• Référence valide ou transmission possible du flux à travers la fonction.

Exemple

Dans l'exemple suivant, si %I0001 est à "1", la chaîne de bits commençant à %M0001 est parcourue jusqu'à ce qu'un bit égal à 1 soit trouvé ou que les 6 mots aient été parcourus. La bobine %Q0001 est mise à "1". Si un bit égal à 1 est trouvé, son emplacement dans la chaîne de bits est écrit dans %AQ001. Si %I0001 est à "1", le bit %M0001 est à "0" et le bit %M0002 est à "1", alors, la valeur écrite dans %AQ001 est 2.

MSKCMP (WORD, DWORD)

La fonction comparaison masquée (MSKCMP) (*disponible pour les UC Version 4.41 ou supérieure*) est utilisée pour comparer le contenu de deux chaînes de bits avec la possibilité de masquer certains bits. La longueur des chaînes de bits à comparer est spécifiée par le paramètre LEN (où la valeur de LEN spécifie le nombre de mots de 16 bits pour la fonction MSKCMPW et de mots de 32 bits pour la fonction MSKCMPD).

Lorsque la logique validant la fonction laisse passer le flux d'énergie vers l'entrée validation (EN), la fonction commence à comparer les bits de la première chaîne de bits avec les bits correspondants dans la deuxième chaîne. La comparaison se poursuit jusqu'à ce qu'une différence soit trouvée ou jusqu'à ce que la fin de la chaîne soit atteinte.

L'entrée BIT est utilisée pour stocker le numéro de bit à partir du quel la comparaison suivante doit commencer (un *0* indique le premier bit de la chaîne). La sortie BN est utilisée pour stocker le numéro de bit où la dernière comparaison s'est produite (un *1* indique le premier bit de la chaîne). L'utilisation de la même référence pour BIT et BN provoque le départ de la comparaison à la position de bit suivant une différence ; ou, si tous les bits sont comparés correctement à l'invocation suivante du bloc fonctionnel, la comparaison commence au début.

Si vous voulez commencer la comparaison suivante à un autre emplacement de la chaîne, vous pouvez entrer des références différentes pour BIT et BN. Si la valeur de BIT est un emplacement situé au-delà de la fin d'une chaîne, BIT est mis à "0" avant de commencer la comparaison suivante.

Si tous les bits de I1 et I2 sont identiques

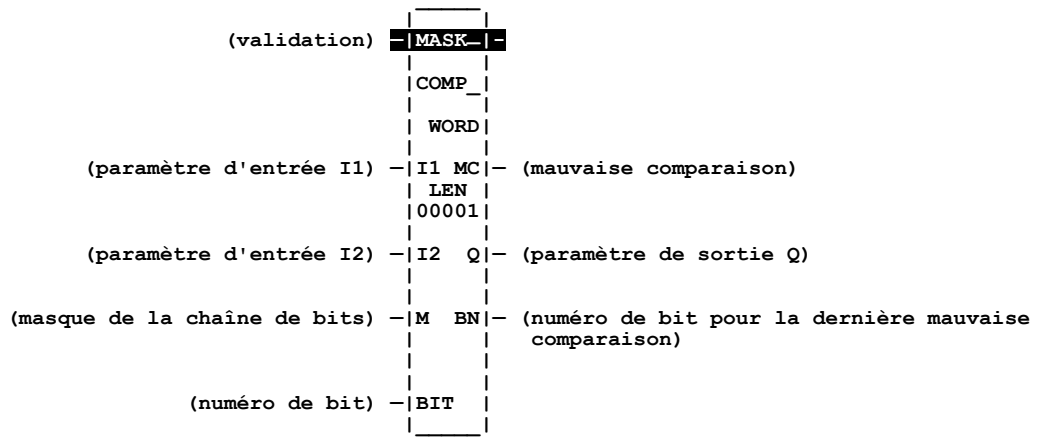
Si tous les bits correspondants des chaînes I1 et I2 sont identiques, la fonction met la sortie de "mauvaise comparaison" MC à 0 et BN au numéro de bit le plus haut dans les chaînes d'entrée. La comparaison s'arrête alors. A la prochaine invocation de MSKCMPW, il sera remis à "0".

Si une différence est trouvée

Lorsque les deux bits comparés sont différents, la fonction vérifie l'état du bit correspondant dans le masque M. Si le bit de masquage est un "1", la comparaison se poursuit jusqu'à ce qu'elle atteigne une autre différence ou jusqu'à la fin des chaînes d'entrée.

Si une différence est détectée et que le bit de masquage correspondant est un "0", la fonction effectue ce qui suit :

1. Met le bit de masquage correspondant dans M à "1".
2. Met la sortie de détection de différence (MC) à "1".
3. Met à jour la chaîne de bits de sortie Q pour correspondre au nouveau contenu de la chaîne de masquage M.
4. Met la sortie de numéro de bit (BN) au rang du bit présentant une différence..
5. Arrête la comparaison.



Paramètres

Paramètre	Description
validation	Logique permissive pour valider la fonction.
I1	Référence pour la première chaîne de bits à comparer.
I2	Référence pour la deuxième chaîne de bits à comparer.
M	Référence pour le masque de la chaîne de bits.
BIT	Référence pour le numéro de bit où la comparaison suivante doit commencer.
MC	Logique utilisateur pour déterminer si une mauvaise comparaison s'est produite.
Q	Copie de sortie de la chaîne de bits de masquage (M).
BN	Numéro du bit où la dernière comparaison s'est produite.
LEN	LEN est le nombre de mots de la chaîne de bit.

Types de mémoire 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 emplacement valide lorsque l'énergie peut passer dans la fonction.
o Référence valide que pour les données MOT ; invalide pour DWORD.
† %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.
‡ Valeur constante maximale de 4095 pour WORD et 2047 pour DWORD.

Exemple

Dans l'exemple suivant, après la première scrutation, le bloc fonctionnel MSKCMPW est exécuté. Les bits %M0001 à %M0016 sont comparés avec les bits %M0017 à %M0032, tandis que les bits %M0033 à %M0048 contiennent la valeur du masque. La valeur de %R0001 détermine à quelle position de bit la comparaison commence dans les deux chaînes d'entrée. Le contenu des références ci-dessus, avant éxécution du bloc fonctionnel, est le suivant :

(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

Le contenu de ces références, après éxécution du bloc fonctionnel, est le suivant :

(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 schéma à relais

Noter que, dans l'exemple ci-dessus, nous avons utilisé le contact FST_SCN pour se limiter à une seule exécution ; autrement, la comparaison masquée se répéterait, sans donner nécessairement le résultat souhaité.

Chapitre 9

Fonctions de transfert de données

Les fonctions de transfert de données permettent des déplacements de données de base. Ce chapitre décrit les fonctions de transfert de données suivantes :

Abréviation	Fonction	Description	Page
MOVE	Transfert	Copie des données en tant que bits individuels. La longueur maximale permise est de 256 mots, sauf pour MOVE_BIT qui est de 256 bits. Les données peuvent être transférées dans différents types de données sans conversion préalable.	9-2
BLKMOV	Transfert de bloc	Copie un bloc de sept constantes dans un emplacement mémoire spécifié. Les constantes sont entrées comme faisant partie de la fonction.	9-5
BLKCLR	Effacement de bloc	Remplace le contenu d'un bloc par des zéros. Cette fonction peut être utilisée pour effacer une zone de bits (%I, %Q, %M, %G ou %T) ou de mémoire mot (%R, %AI ou %AQ). La longueur maximale permise est de 256 mots.	9-7
SHFR	Registre de décalage	Décale un ou plusieurs mots dans une table. La longueur maximale permise est de 256 mots.	9-8
BITSEQ	Séquenceur de bits	Effectue une rotation de séquence de bits dans un groupe de bits. La longueur maximale permise est de 256 mots.	9-11
COMMREQ	Demande de communication	Permet au programme de communiquer avec un module intelligent comme un module de communication Genius ou un module co-processeur programmable.	9-14

MOVE (BIT, INT, WORD, REAL)

Utiliser la fonction MOVE pour copier des données en tant que bits individuels d'un emplacement à un autre. Comme les données sont copiées en format bit, la destination ne doit pas être nécessairement du même type de donnée que l'emplacement d'origine.

La fonction MOVE possède deux paramètres d'entrée et deux paramètres de sortie. Lorsque la fonction reçoit le flux d'énergie, elle copie les données du paramètre d'entrée IN dans le paramètre de sortie Q comme bits. Si les données sont transférées d'un emplacement de mémoire logique à un autre, (par exemple, de la mémoire %I à la mémoire %T), l'information de transition associée aux éléments de la mémoire logique est mise à jour pour indiquer si l'instruction MOVE provoque ou non un changement d'état d'un élément de la mémoire logique. Les données du paramètre d'entrée ne changent pas, sauf s'il y a un chevauchement dans la source et la destination.

Pour le type BIT, il y a une autre considération. Si un groupe BIT, spécifié dans le paramètre Q, n'englobe pas tous les bits d'un octet, les bits de transition associés à cet octet (qui ne sont pas dans le groupe) seront effacés lorsque MOVE_BIT recevra le flux d'énergie.

L'entrée IN peut être soit une référence pour les données à transférer, soit une constante. Si une constante est spécifiée, sa valeur est placée dans l'emplacement spécifié par la référence de sortie. Par exemple, si une valeur de 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 constante est placée dans l'emplacement mémoire spécifié par Q et les emplacements suivants, jusqu'à la longueur spécifiée. Par exemple, si la valeur de constante 9 est spécifiée pour IN et que la longueur est 4, alors la valeur 9 sera copié dans l'emplacement mémoire spécifié par Q et les trois emplacements suivants.

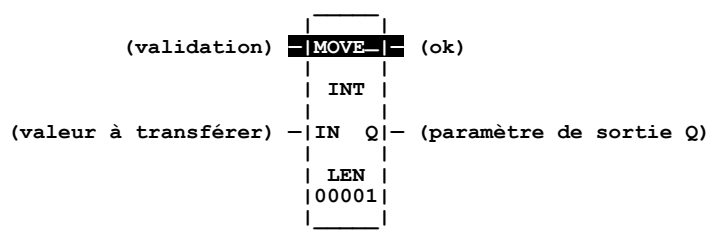
L'instruction LEN spécifie le nombre de :

- Mots à transférer pour MOVE_INT et MOVE_WORD.
- Bits à transférer pour MOVE_BIT.
- Nombres réels à transférer pour MOVE_REAL.

Remarque

Le type de données REAL n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou supérieure, ou sur toutes les versions de l'UC352.

La fonction laisse passer le flux d'énergie lorsque l'énergie est reçue.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, le transfert est effectué.
IN	IN contient la valeur à transférer. Pour MOVE_BIT, n'importe quelle référence logique peut être utilisée ; elle n'a pas besoin d'être alignée sur l'octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
ok	La sortie ok est excitée lorsque la fonction est validée.
Q	Lorsque le transfert est effectué, la valeur de IN est écrite dans Q. Pour MOVE_BIT, n'importe quelle référence logique peut être utilisée ; elle n'a pas besoin d'être alignée sur l'octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
LEN	LEN spécifie le nombre de mots ou de bits à transférer. Pour MOVE_WORD et MOVE_INT, LEN doit être entre 1 et 256 mots. Pour MOVE_BIT, lorsque IN est une constante, LEN doit être entre 1 et 16 bits ; autrement, LEN doit être entre 1 et 256.

Remarque

Sur les UC des modèles 351, 352 et 36x, les fonctions MOVE_INT et MOVE_WORD ne supportent pas le chevauchement des paramètres IN et Q, lorsque la référence IN est inférieure à la référence Q. Par exemple, avec les valeurs suivantes :
Avec IN=%R0001, Q=%R0004, LEN=5 (mots), les contenus de %R0007 et %R0008 seront indéterminés ; cependant, en utilisant les valeurs suivantes : Q=%R0001, IN=%R0004, LEN=5 (mots) donneront des contenus valides.

Noter également que seules les UC des modèles 35x et 36x (Version 9 et supérieures, plus toutes les versions de l'UC352) ont les possibilités de virgule flottante et, par conséquent, sont les seules à accepter MOVE_REAL.

Types de mémoire valides

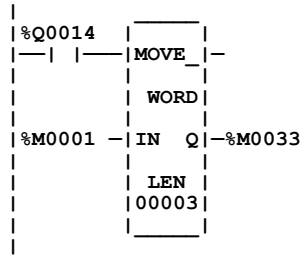
Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•	o	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	o†	•	•	•	•		

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

- • Référence valide pour les données ou emplacements BIT, INT ou MOT lorsque l'énergie peut passer dans la fonction. Pour MOVE_BIT, les références utilisateur logiques %I, %Q, %M et %T n'ont pas besoin d'être alignées sur octet.
- o Référence valide uniquement pour les données BIT ou MOT ; invalide 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 de 16 bits, le transfert est effectué correctement (sauf pour les UC 351 et 352, comme indiqué précédemment).



Avant utilisation de la fonction Transfert :

INPUT (%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 utilisation de la fonction Transfert :

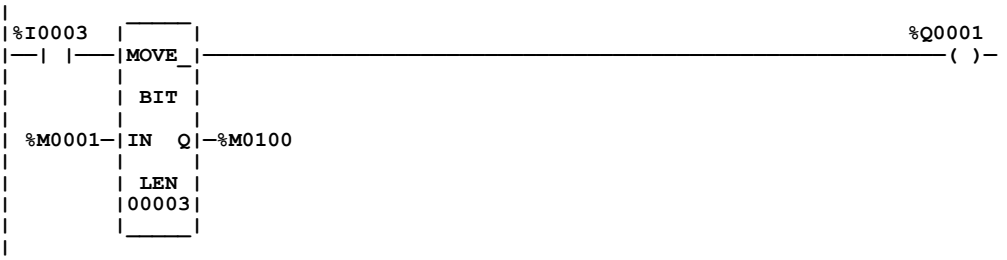
INPUT (%M0033 à %M0080)

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, lorsque %I0001 est à "1", les trois bits %M0001, %M0002 et %M0003 sont transférés dans %M0100, %M0101 et %M0102, respectivement. La bobine %Q0001 est mise à "1".



BLKMOV (INT, WORD, REAL)

Utiliser la fonction Transfert de données (BLKMOV) pour copier un bloc de sept constantes dans un emplacement spécifique.

Remarque

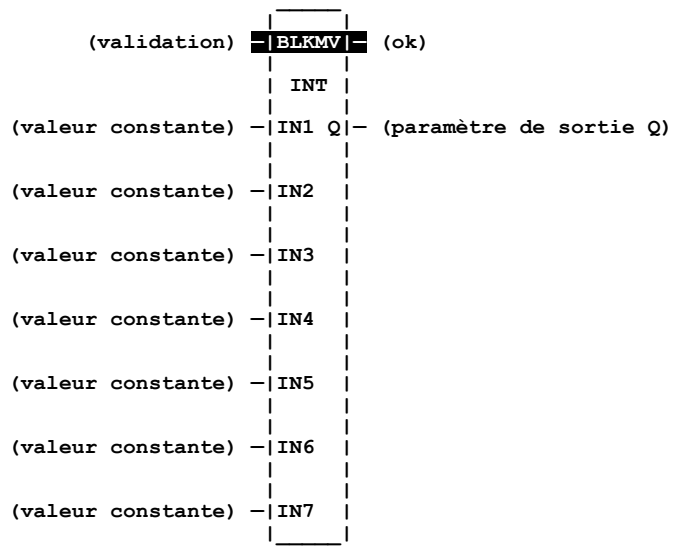
Le type de données REAL n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou supérieure, ou sur toutes les versions de l'UC352.

La fonction BLKMOV possède huit paramètres d'entrée et deux paramètres de sortie. Lorsque la fonction reçoit le flux d'énergie, elle copie les valeurs de constante dans des emplacements consécutifs, en commençant par la destination spécifiée dans la sortie Q. La sortie Q ne peut pas être l'entrée d'une autre fonction de programme.

Remarque

Pour BLKMOV_INT, les valeurs de IN1 — IN7 sont affichées comme décimales signées. Pour BLKMOV_WORD, IN1 — IN7 sont affichées en hexadécimal. Pour BLKMOV_REAL, IN1 — IN7 sont affichées en format Réel.

La fonction laisse passer le flux d'énergie lorsque l'énergie est reçue.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, le transfert de bloc est effectué.
IN1— IN7	IN1 à IN7 contiennent sept valeurs de constante.
ok	La sortie ok est excitée lorsque la fonction est validée.
Q	La sortie Q contient le premier entier du groupe transféré. IN1 est transféré dans to Q.

Types de mémoire 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 valide pour l'emplacement où l'énergie peut passer dans la fonction.

o Référence valide que pour les données MOT ; invalide pour INT ou REAL.

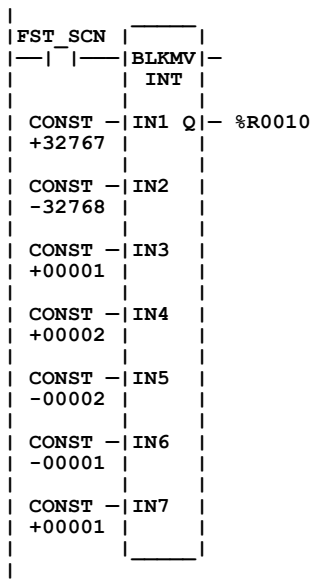
† %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Remarque

Il n'existe des possibilités de virgule flottante que pour les UC modèles 35x et 36x, Version 9 ou supérieure, ou pour toutes les versions de l'UC352. Ces UC 90-30 sont les seules à accepter BLKMV_REAL.

Exemple

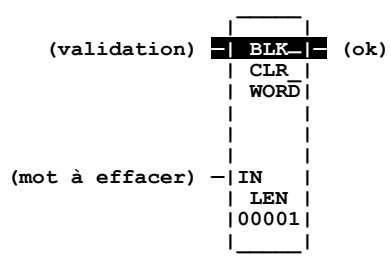
Dans l'exemple suivant, lorsque l'entrée de validation, représentée par le symbole FST_SCN, est à "1", la fonction BLKMOV copie les sept constantes d'entrée dans les emplacements mémoire %R0010 à %R0016.



BLKCLR (WORD)

Utiliser la fonction Effacement de bloc (BLKCLR) pour remplir de zéros un bloc spécifié de données.

La fonction BLKCLR possède deux paramètres d'entrée et un paramètre de sortie. Lorsque la fonction reçoit le flux d'énergie, elle écrit des zéros dans l'emplacement mémoire en commençant par la référence spécifiée par IN. Lorsque les données à effacer proviennent de la mémoire logique (%I, %Q, %M, %G ou %T), les informations de transition, associées aux références, sont également effacées. La fonction laisse passer l'énergie vers la droite.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, le groupe est effacé.
IN	IN contient le premier mot du groupe à effacer.
ok	La sortie ok est excitée lorsque la fonction est validée.
LEN	LEN doit comporter entre 1 et 256 mots.

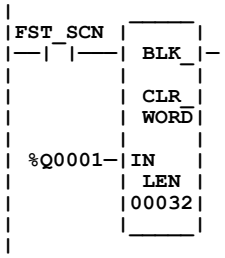
Types de mémoire valides

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

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction. † %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Exemple

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



SHFR (BIT, WORD)

Utiliser la fonction Registre à décalage (SHFR) pour décaler un ou plusieurs mots de données, ou des bits de données, d'un emplacement de référence vers une zone spécifiée de la mémoire. Par exemple, un mot pourrait être décalé dans une zone de mémoire d'une longueur spécifiée de cinq mots. Comme résultat de ce décalage, un autre mot de données serait décalé hors de l'extrémité de la zone mémoire.

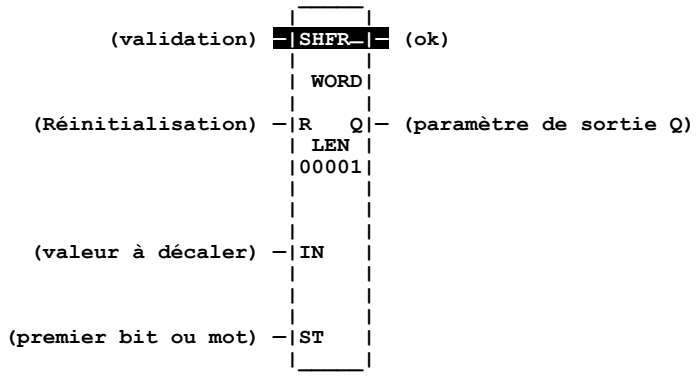
Remarque

Lors de l'affectation des adresses de référence, le chevauchement des plages d'adresses de référence d'entrée et de sortie dans les fonctions multimots, peut produire des résultats imprévus.

La fonction SHFR possède quatre paramètres d'entrée et deux paramètres de sortie. L'entrée validation (R) est prioritaire sur l'entrée validation de la fonction. Lorsque la réinitialisation est active, toutes les références commençant au registre de décalage (ST) jusqu'à la longueur spécifiée pour LEN, sont remplies de zéros.

Si la fonction reçoit le flux d'énergie et que la réinitialisation n'est pas à "1", chaque bit ou mot du registre de décalage est déplacé vers la référence suivante la plus haute. Le dernier élément du registre de décalage est décalé dans Q. La référence la plus haute de l'élément du registre de décalage de IN est décalé dans l'élément vacant commençant à ST. Le contenu du registre de décalage est accessible dans le programme car il est recouvert dans des emplacements absolus de la mémoire adressable de la logique.

La fonction fait passer l'énergie vers la droite lorsque cette énergie est reçue de la logique de validation.



Paramètres

Paramètre	Description
validation	Lorsque validation est à "1" et que R ne l'est pas, le décalage est effectué.
R	Lorsque R est excité, le registre de décalage, situé à ST, est rempli de zéros.
IN	IN contient la valeur à décaler dans le premier bit ou mot du registre de décalage. Pour SHFR_BIT, toute référence logique peut être utilisée ; elle n'a pas besoin d'être alignée sur octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
ST	ST contient le premier bit ou mot du registre de décalage. Pour SHFR_BIT, toute référence logique peut être utilisée ; elle n'a pas besoin d'être alignée sur octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
ok	La sortie ok est excitée lorsque la fonction est validée et que R n'est pas validé.
Q	La sortie Q contient le bit ou le mot décalé hors du registre de décalage. Pour SHFR_BIT, toute référence logique peut être utilisée ; elle n'a pas besoin d'être alignée sur octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
LEN	LEN détermine la longueur du registre de décalage. Pour SHFR_WORD, LEN doit comporter entre 1 et 256 mots. Pour SHFR_BIT, LEN doit comporter entre 1 et 256 bits.

Types de mémoire valides

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

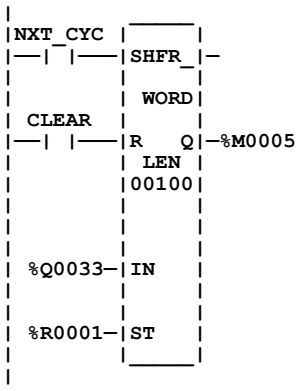
• Référence valide pour les données ou emplacements BIT ou MOT où l'énergie peut passer dans la fonction.
Pour SHFR_BIT, les références utilisateur logiques %I, %Q, %M et %T n'ont pas besoin d'être alignées sur octet.

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

Exemple 1

Dans l'exemple suivant, le registre à décalage fonctionne sur les emplacements mémoire du registre de décalage %R0001 à %R0100. Lorsque la référence réinitialisation CLEAR est à "1", les mots du registre de décalage sont mis à "0".

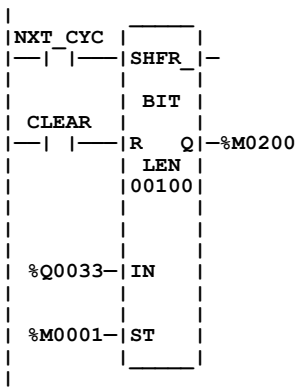
Lorsque la référence NXT_CYC est à "1" et que CLEAR est à "0", le mot de l'emplacement %Q0033 de la table d'état des sorties est décalé dans le registre de décalage à %R0001. Le mot décalé sortant du registre de décalage de %R0100 est stocké dans la sortie %M0005.



Exemple 2

Dans cet exemple, le registre à décalage fonctionne sur les emplacements mémoire %M0001 à %M0100. Lorsque la référence de réinitialisation CLEAR est à "1", la fonction SHFR remplit %M0001 à %M0100 de zéros.

Lorsque NXT_CYC est à "1" et que CLEAR est à "0", la fonction SHFR décale les données dans %M0001 à %M0100 d'un bit vers le bas. Le bit de %Q0033 est décalé dans %M0001 alors que le bit décalé hors de %M0100 est écrit dans %M0200.



BITSEQ (BIT)

La fonction Séquenceur de bits (BITSEQ) effectue un décalage séquentiel de bits dans un groupe de bits. La fonction BITSEQ possède cinq paramètres d'entrée et un paramètre de sortie. Le fonctionnement de la fonction dépend de la valeur précédente du paramètre EN, comme montré dans la table suivante.

Exécution courante R	Exécution précédente EN	Exécution courante EN	Exécution du séquenceur de bits
INACTIVE	INACTIVE	INACTIVE	Le séquenceur de bits ne s'exécute pas.
INACTIVE	INACTIVE	ACTIVE	Le séquenceur de bits incrémente/décrémente de 1.
INACTIVE	ACTIVE	INACTIVE	Le séquenceur de bits ne s'exécute pas.
INACTIVE	ACTIVE	ACTIVE	Le séquenceur de bits ne s'exécute pas.
ACTIVE	ACTIVE/INACTIVE	ACTIVE/INACTIVE	Le séquenceur de bits réinitialise.

L'entrée réinitialisation (R) prend la priorité sur la validation (EN) et réinitialise toujours le séquenceur. Lorsque R est à "1", le numéro de pas courant est mis à la valeur passée par le paramètre de numéro de pas. S'il n'y a pas de numéro de pas, le pas est mis à "1". Tous les bits du séquenceur sont mis à "0", sauf le bit indiqué par le pas courant, qui est mis à "1".

Lorsque EN est à "1" et que R est à "0", le bit indiqué par le numéro de pas courant est effacé. Le numéro de pas courant est soit incrémenté, soit décrémenté, selon le paramètre de direction. Ensuite, le bit indiqué par le nouveau numéro de pas est mis à "1".

- Lorsque le numéro de pas est incrémenté et qu'il sort de la plage de (1 ≤ numéro de pas ≤ LEN), il est remis à "1".
- Lorsque le numéro de pas est décrémenté et qu'il sort de la plage de (1 ≤ numéro de pas ≤ LEN), il est mis à LEN.

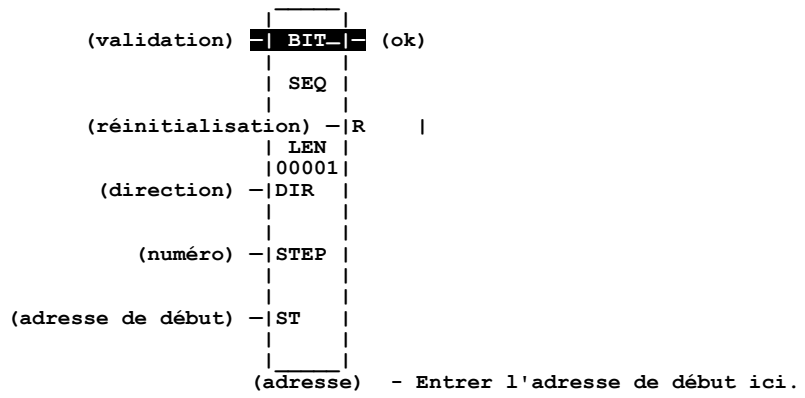
Le paramètre ST est optionnel. S'il n'est pas utilisé, BITSEQ fonctionne comme décrit ci-dessus, sauf qu'aucun bit n'est mis à "1" ou à "0". A la base, BITSEQ fait circuler le numéro de pas courant dans sa plage légale.

Mémoire nécessaire pour un séquenceur de bits

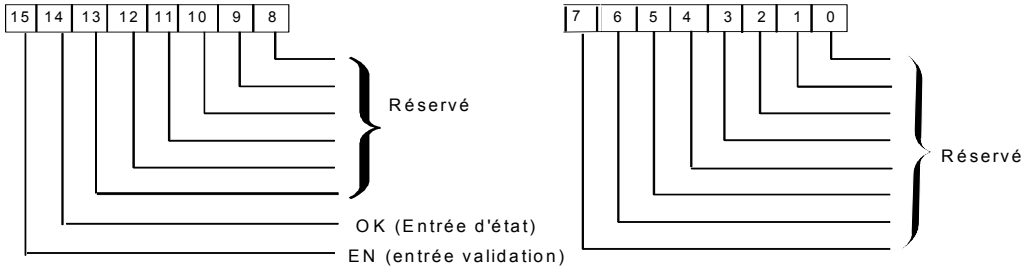
Chaque séquenceur de bits utilise trois mots (registres) de mémoire %R pour stocker les informations suivantes :

numéro de pas courant	mot 1
longueur de la séquence (en bits)	mot 2
mot de contrôle	mot 3

Lorsque vous utilisez un séquenceur de bits, vous devez entrer une adresse de début pour ces trois mots (registres) directement en dessous du graphique représentant la fonction (voir l'exemple à la page suivante).



Le mot de contrôle mémorise l'état des entrées et des sorties Booléennes de son bloc fonctionnel associé, comme montré dans l'illustration suivante :



Remarque

Les bits 0 à 13 ne sont pas utilisés. Egalement, noter que ces bits doivent être entrés de 1 à 16, *PAS* 0 à 15 dans le paramètre STEP.

Paramètres

Paramètre	Description
adresse	L'adresse est l'emplacement du pas courant, de la longueur et des derniers états validation et ok du séquenceur de bits.
validation	Lorsque la fonction est validée, si elle n'a pas été validée lors du balayage précédent et si R n'est pas excité, le décalage séquentiel de bits est effectué.
R	Lorsque R est excité, le numéro de pas du séquenceur de bits est mis à la valeur de STEP (par défaut = 1), et le séquenceur de bits est rempli de zéros, sauf pour le bit de numéro de pas courant.
DIR	Lorsque DIR est excité, le numéro de pas du séquenceur de bits est incrémenté avant le décalage. Autrement, il est décrémenté.
STEP	Lorsque R est excité, le numéro de pas est mis à cette valeur.
ST	ST contient le premier mot du séquenceur de bits.
ok	La sortie ok est excitée lorsque la fonction est validée.
LEN	LEN doit comporter entre 1 et 256 bits.

Remarque

La vérification de bobine, pour la fonction BITSEQ, vérifie les 16 bits du paramètre ST, même lorsque LEN est inférieur à 16.

Types de mémoire valides

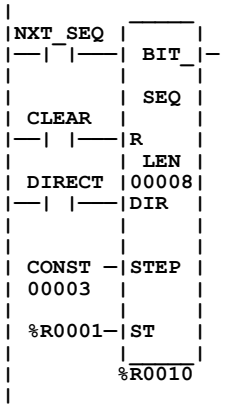
Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
adresse								•				
validation	•											
R	•											
DIR	•											
STEP		•	•	•	•		•	•	•	•	•	•
ST		•	•	•	•	•†	•	•	•	•		•
ok	•											•

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.
† %SA, %SB, %SC seulement ; %S ne peut pas être utilisé.

Exemple

Dans l'exemple suivant, le séquenceur fonctionne sur la mémoire registre %R0001. Ses données statiques sont stockées dans les registres %R0010, %R0011 et %R0012. Lorsque CLEAR est actif, le séquenceur est réinitialisé et le pas courant est mis au numéro de pas 3. Les 8 premiers bits de %R0001 sont mis à zéro.

Lorsque NXT_SEQ est actif et que CLEAR est inactif, le bit du numéro de pas 3 est effacé et le bit du numéro de pas 2 ou 4 (selon que DIR est excité ou pas) est mis à "1".



COMMREQ

Utiliser la fonction Requête de communication (COMMREQ) si le programme a besoin de communiquer avec un module intelligent ou un module co-processeur programmable.

Remarque

Les informations présentées sur les pages suivantes montrent le format de la fonction COMMREQ. Vous aurez besoin d'informations supplémentaires pour programmer COMMREQ pour chaque type de dispositif. Les exigences de programmation de chaque module, qui utilise la fonction COMMREQ, sont décrites dans la documentation du module.

La fonction COMMREQ possède trois paramètres d'entrée et un paramètre de sortie. Lorsque la fonction COMMREQ reçoit le flux d'énergie, un bloc de commande de données est envoyé au module intelligent. Le bloc de commande commence à la référence spécifiée en utilisant le paramètre IN. Le châssis et le N° d'emplacement du module intelligent sont spécifiés dans SYSID.

COMMREQ peut soit envoyer un message et attendre une réponse, soit envoyer un message et continuer sans attendre de réponse. Si le bloc de commande spécifie que le programme n'attendra pas de réponse, le contenu du bloc de commande est envoyé au dispositif récepteur et l'exécution du programme se poursuit immédiatement. (La valeur de dépassement de temps est ignorée). Ceci est désigné comme mode **SANS ATTENTE** .

Si le bloc de commande spécifie que le programme attendra une réponse, le contenu du bloc de commande est envoyé au dispositif récepteur et l'UC attend une réponse. La durée maximale que l'API attendra une réponse du dispositif est spécifiée dans le bloc de commande. Si le dispositif ne répond pas dans cette durée, l'exécution du programme se poursuit. Ceci est désigné comme mode **ATTENTE** .

La sortie Fonction défaillante (FT) peut être mise à "1" si :

1. L'adresse cible spécifiée est absente (SYSID).
2. La tâche spécifiée est invalide pour le dispositif (TASK).
3. La longueur de donnée est 0.
4. L'adresse du pointeur d'état du dispositif (partie du bloc de commande) n'existe pas. Ceci peut être dû à une sélection de type de mémoire incorrecte ou à une adresse de ce type de mémoire qui est hors de la plage.

Bloc de commande

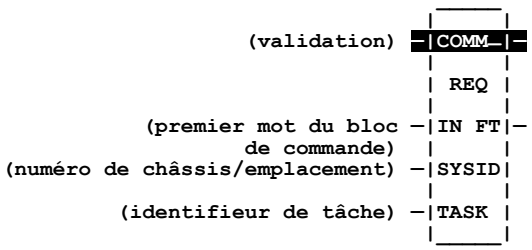
Le bloc de commande fournit des informations au module intelligent concernant la commande à effectuer.

L'adresse du bloc de commande est spécifiée pour l'entrée IN de la fonction COMMREQ. Cette adresse peut être toute zone mémoire mot (%R, %AI ou %AQ). La longueur du bloc de commande dépend de la quantité de données envoyée au dispositif.

Le bloc de commande a la structure suivante :

Longueur (en mots)	adresse
Mode Attente/Sans attente	adresse + 1
Mémoire pointeur d'état	adresse + 2
Décalage de pointeur d'état	adresse + 3
Valeur de dépassement de temps	adresse + 4
Temps de communication maximal	adresse + 5
Bloc de données	adresse + 6 à adresse + 133

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



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la requête de communication est effectuée.
IN	IN contient le premier mot du bloc de commande.
SYSID	SYSID contient le numéro du châssis (octet le plus significatif) et le numéro d'emplacement (octet le moins significatif) du dispositif cible.
TASK	TASK contient l'identifieur de tâche du procédé sur le dispositif cible.
FT	FT est excité si une erreur est détectée en traitant COMMREQ.

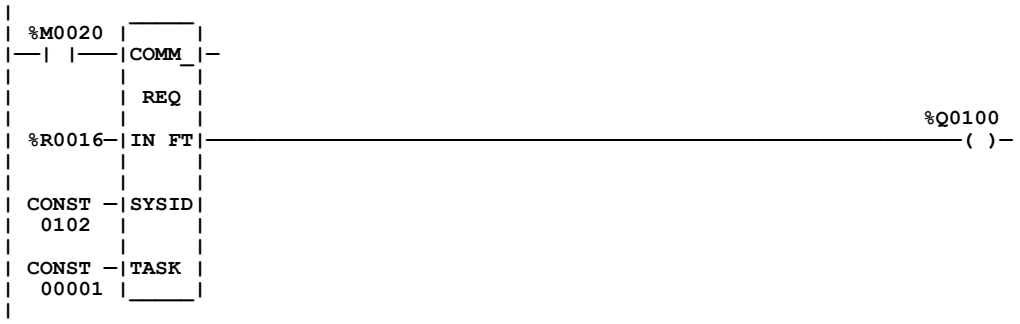
Types de mémoire valides

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

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Exemple

Dans l'exemple suivant, lorsque l'entrée validation %M0020 est à "1", un bloc de commande situé à partir de %R0016 est envoyé à la tâche 1 de communication du dispositif situé dans le châssis 1, emplacement 2 de l'API. Si une erreur se produit, pendant le traitement de COMMREQ, %Q0100 est mis à "1".



Remarque

Pour les systèmes qui n'ont pas de châssis d'extension, SYSID doit être zéro pour le châssis principal.

Chapitre 10

Fonctions Tableau

Les fonctions Tableau sont utilisées pour effectuer les fonctions suivantes :

Abréviation	Fonction	Description	Page
ARRAY_MOVE	Transfert de groupe	Copie un nombre spécifié d'éléments de données d'un groupe source dans un groupe destination.	10-2
SRCH_EQ	Recherche égal à	Recherche toutes les valeurs d'un groupe qui sont égales à une valeur spécifiée.	10-6
SRCH_NE	Recherche différent de	Recherche toutes les valeurs d'un groupe qui sont différentes d'une valeur spécifiée.	10-6
SRCH_GT	Recherche supérieur à	Recherche toutes les valeurs d'un groupe qui sont supérieures à une valeur spécifiée.	10-6
SRCH_GE	Recherche supérieur ou égal à	Recherche toutes les valeurs d'un groupe qui sont supérieures ou égales à une valeur spécifiée.	10-6
SRCH_LT	Recherche inférieur à	Recherche toutes les valeurs d'un groupe qui sont inférieures à une valeur spécifiée.	10-6
SRCH_LE	Recherche inférieur ou égal à	Recherche toutes les valeurs d'un groupe qui sont inférieures ou égales à une valeur spécifiée.	10-6

La longueur maximale permise pour ces fonctions est de 32 767 octets ou mots, ou 262 136 bits (les bits ne sont disponibles que pour ARRAY_MOVE).

Les fonctions Tableau s'effectuent sur ces types de données :

Type de donnée	Description
INT	Entier signé
DINT	Entier signé double précision
BIT *	Type de donnée Bit
BYTE	Type de donnée Octet
WORD	Type de donnée Mot.

* Disponible seulement pour ARRAY_MOVE.

Le type de donnée par défaut est l'entier signé. Le type de donnée peut être changé après avoir sélectionné la fonction tableau de données spécifique. Pour comparer des données d'autres types ou de deux types différents, utiliser d'abord la fonction de conversion appropriée (décrite dans le Chapitre 11, “Fonctions de conversion”) pour changer les données dans l'un des types de donnée listés ci-dessus.

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

Utiliser la fonction Transfert de groupe (ARRAY_MOVE) pour copier un nombre spécifié d'éléments de données d'un groupe source dans un groupe destination.

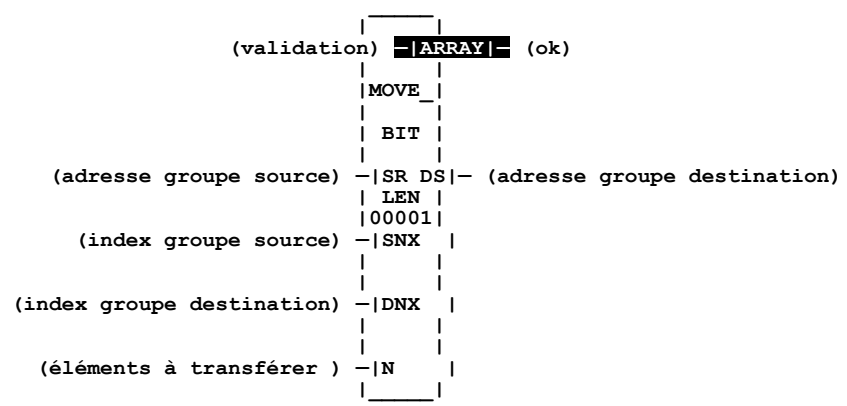
La fonction ARRAY_MOVE possède cinq paramètres d'entrée et deux paramètres de sortie. Lorsque la fonction reçoit le flux d'énergie, le nombre d'éléments de données de l'indicateur de compte (N) est extrait du groupe d'entrée commençant par l'emplacement indexé (SR + SNX — 1). Les éléments de données sont écrits dans le groupe de sortie en commençant par l'emplacement indexé (DS + DNX — 1). L'instruction LEN spécifie le nombre d'éléments qui constituent chaque groupe.

Pour ARRAY_MOVE_BIT, lorsque la mémoire mot est sélectionnée pour les paramètres du groupe source et/ou l'adresse de début du groupe destination, le bit le moins significatif du mot spécifié est le premier bit du groupe. La valeur affichée contient 16 bits, quelle que soit la longueur du groupe.

Les indices d'une instruction ARRAY_MOVE sont à base de 1. En utilisant ARRAY_MOVE, aucun élément extérieur au groupe source ou destination (comme spécifié par son adresse de début et sa longueur) ne peut être référencé.

La sortie ok recevra le flux d'énergie, sauf si l'une des conditions suivantes se produit :

- Validation est à "0".
- (N + SNX – 1) est supérieur à LEN.
- (N + DNX – 1) est supérieur à LEN.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
SR	SR contient l'adresse de début du groupe source. Pour ARRAY_MOVE_ BIT, toute référence peut être utilisée ; elle n'a pas besoin d'être alignée sur octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
SNX	SNX contient l'index du groupe source.
DNX	DNX contient l'index du groupe destination
N	N fournit un indicateur de compte.
ok	La sortie ok est excitée lorsque validation est excitée.
DS	DS contient l'adresse de début du groupe destination. Pour ARRAY_MOVE_ BIT, toute référence peut être utilisée ; elle n'a pas besoin d'être alignée sur octet. Cependant, 16 bits, commençant par l'adresse de référence spécifiée, sont affichés en ligne.
LEN	LEN spécifie le nombre d'éléments commençant à SR et DS qui constituent chaque groupe.

Types de mémoire valides

Paramètre	flux	%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 valide ou emplacement où l'énergie peut passer à travers la fonction.
Pour ARRAY_MOVE_BIT, les références utilisateur logiques %I, %Q, %M et %T n'ont pas besoin d'être alignées sur octet.

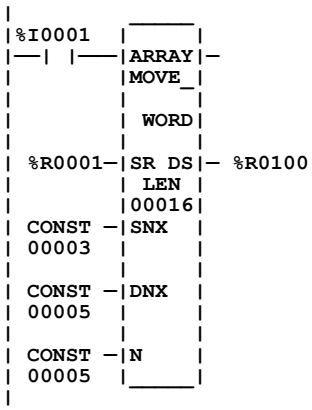
o Référence valide uniquement pour les données INT, BIT, OCTET ou MOT ; invalide pour DINT.

Δ Type de donnée valide pour BIT, OCTET ou MOT seulement ; invalide pour INT ou DINT.

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

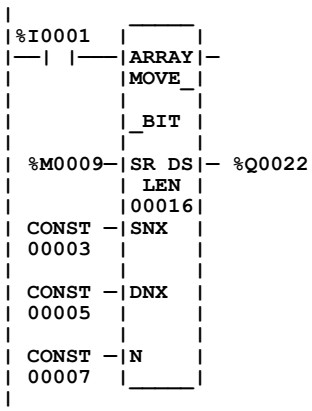
Exemple 1

Dans cet exemple, %R0003 — %R0007 du groupe %R0001 — %R0016 sont lus, puis écrits dans %R0104 — %R0108 du groupe %R0100 — %R0115.



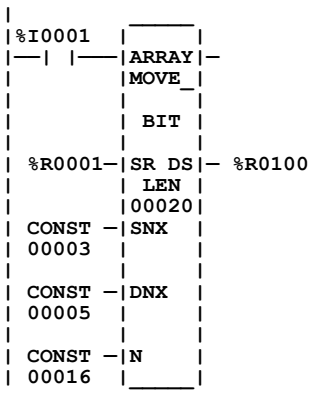
Exemple 2

En utilisant la mémoire bit pour SR et DS, %M0011— %M0017 du groupe %M009 — %M0024 sont lus, puis écrits dans %Q0026 — %Q0032 du groupe %Q0022 — %Q0037.



Exemple 3

En utilisant la mémoire mot, pour SR et DS, le troisième bit le moins significatif de %R0001 jusqu'au deuxième bit le moins significatif de %R0002 du groupe contenant les 16 bits de %R0001 et les quatre bits de %R0002, sont lus, puis écrits dans le cinquième bit le moins significatif de %R0100 jusqu'au quatrième bit le moins significatif de %R0101 du groupe contenant les 16 bits de %R0100 et les quatre bits de %R0101.



Fonctions de recherche

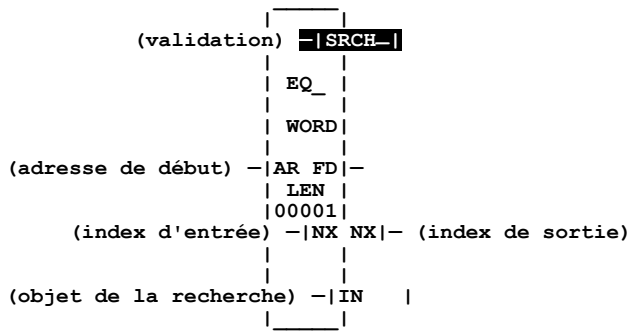
Utiliser la fonction Recherche appropriée listée ci-dessous pour rechercher toutes les valeurs d'un groupe pour cette opération particulière.

Abréviation	Fonction	Description
SRCH_EQ	Recherche égal à	Recherche toutes les valeurs d'un groupe qui sont égales à une valeur spécifiée.
SRCH_NE	Recherche différent de	Recherche toutes les valeurs d'un groupe qui sont différentes d'une valeur spécifiée.
SRCH_GT	Recherche supérieur à	Recherche toutes les valeurs d'un groupe qui sont supérieures à une valeur spécifiée.
SRCH_GE	Recherche supérieur ou égal à	Recherche toutes les valeurs d'un groupe qui sont supérieures ou égales à une valeur spécifiée.
SRCH_LT	Recherche inférieur à	Recherche toutes les valeurs d'un groupe qui sont inférieures à une valeur spécifiée.
SRCH_LE	Recherche inférieur ou égal à	Recherche toutes les valeurs d'un groupe qui sont inférieures ou égales à une valeur spécifiée.

Chaque fonction possède quatre paramètres d'entrée et deux paramètres de sortie. Lorsque la fonction reçoit l'énergie, le groupe est recherché en commençant à (AR + entrée NX). Ceci est l'adresse de début du groupe (AR) plus l'index dans ce groupe (entrée NX).

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

Les valeurs valides pour l'entrée NX sont 0 à LEN — 1. NX doit être mis à "0" 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 1 à LEN. Si la valeur de l'entrée NX est en dehors de la plage, (< 0 ou ≥LEN), elle est mise à la valeur par défaut de zéro.



Paramètres

Paramètre	Description
validation	Lorsque cette fonction est validée, l'opération est effectuée.
AR	AR contient l'adresse de début du groupe à rechercher.
Entrée NX	L'entrée NX contient l'index dans le groupe où commencer la recherche.
IN	IN contient l'objet de la recherche.
Sortie NX	La sortie NX maintient la position, dans le groupe, de la cible recherchée.
FD	FD indique que l'élément du groupe a été trouvé et la fonction a réussi.
LEN	LEN spécifie le nombre d'éléments commençant à AR qui constituent le groupe à rechercher. Il peut être de 1 à 32 767 octets ou mots.

Types de mémoire 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 emplacement valide lorsque l'énergie peut passer dans la fonction.
o Référence valide que pour les données INT, OCTET ou MOT ; invalide pour DINT.
Δ Référence valide pour les données OCTET ou MOT seulement ; invalide pour INT ou DINT.

Exemple 1

Le groupe AR est défini comme adresses mémoire %R0001 — %R0005. Lorsque EN est à "1", la partie du groupe située entre %R0004 et %R0005 est recherchée pour un élément dont la valeur est égale à IN. If %R0001 = 7, %R0002 = 9, %R0003 = 6, %R0004 = 7, %R0005 = 7 et %R0100 = 7, ensuite, la recherche commencera à %R0004 et se terminera à %R0004 lorsque FD sera mis à "1" et qu'un 4 sera écrit dans %R0101.

Exemple 2

Le groupe AR est défini comme adresses mémoire %AI0001 — %AI0016. Les valeurs des éléments du groupe sont 100, 20, 0, 5, 90, 200, 0, 79, 102, 80, 24, 34, 987, 8, 0 et 500. Initialement, %AQ0001 est 5. Lorsque EN est à "1", chaque balayage recherchera dans le groupe toute correspondance avec la valeur 0 de IN. Le premier balayage commencera la recherche à %AI0006 et trouvera une correspondance à %AI0007, ainsi FD est à "1" et %AQ0001 est 7. Le deuxième balayage commencera à rechercher à %AI0008 et trouvera une correspondance à %AI0015, ainsi FD reste à "1" et %AQ0001 est 15. Le balayage suivant commencera à %AI0016. Comme la fin du groupe est atteinte sans correspondance, FD est mis à "0" et %AQ0001 est mis à "0". Le balayage suivant commencera la recherche en commençant au début du groupe.

Chapitre
11

Fonctions de conversion

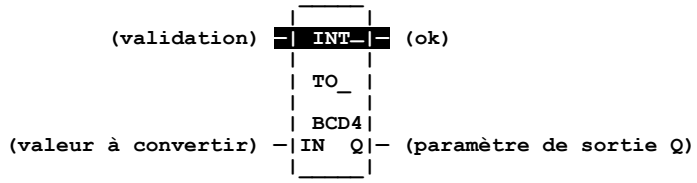
Utiliser les fonctions de conversion pour convertir une donnée d'un format à un autre. De nombreuses instructions de programmation, comme les fonctions mathématiques, doivent être utilisées avec des données d'un format spécifique. Cette section décrit les fonctions de conversion suivantes :

Abréviation	Fonction	Description	Page
BCD-4	Conversion en BCD 4 chiffres	Convertit un entier signé en format BCD-4.	11-2
INT	Conversion en entier signé	Convertit BCD-4 ou REAL en entier signé.	11-3
DINT	Conversion en entier signé à double précision	Convertit REAL en entier signé à double précision.	11-5
REAL	Conversion en REAL	Convertit INT, DINT, BCD-4 ou WORD en REAL.	11-7
WORD	Conversion en WORD	Convertit REAL en WORD.	11-9
TRUN	Troncation	Arrondit un nombre réel à la valeur par défaut	11-11

—>BCD-4 (INT)

La fonction Conversion en BCD-4 est utilisée pour générer un BCD à 4 chiffres à partir d'une donnée en entier signé. La donnée d'origine n'est pas modifiée par cette fonction. Les données peuvent être converties en format BCD pour commander des affichages à LED codées BCD ou prérégler des dispositifs extérieurs comme les compteurs rapides.

Lorsque la fonction reçoit le flux d'énergie, elle effectue la conversion et met le résultat dans la sortie Q. La fonction laisse passer le flux d'énergie lorsqu'elle est alimentée, sauf si la conversion spécifiée donne une valeur en dehors de la plage de 0 à 9999.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la conversion est effectuée.
IN	IN contient une référence pour la valeur entière à convertir en BCD-4.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans erreur.
Q	La sortie Q contient la forme BCD-4 de la valeur d'origine dans IN.

Types de mémoire valides

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

- Référence valide ou transmission du flux à travers la fonction.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0002 est mise à "1" et qu'il n'y a aucune erreur, l'entier se trouvant dans le paramètre d'entrée %I0017 à %I0032 est converti en BCD à quatre chiffres et le résultat est stocké dans les emplacements mémoire %Q0033 à %Q0048. La bobine %Q1432 est utilisée pour vérifier l'exactitude de la conversion.

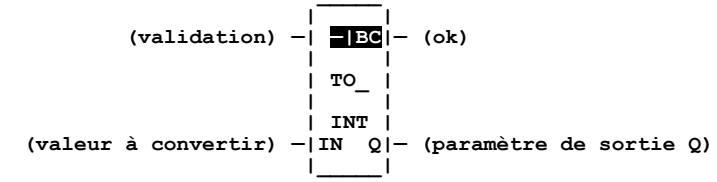
—>INT (BCD-4, REAL)

La fonction Conversion en entier signé est utilisée pour générer une valeur entière à partir d'une donnée BCD-4 ou REAL. La donnée d'origine n'est pas modifiée par cette fonction.

Remarque

Le type de données REAL n'est disponible que sur les UC des modèles 35x et 36x, Version 9 ou supérieure, ou sur toutes les versions de l'UC352.

Lorsque la fonction reçoit le flux d'énergie, elle effectue la conversion et met le résultat dans la sortie Q. La fonction laisse toujours passer le flux d'énergie lorsqu'elle est alimentée, sauf si la donnée est en dehors de la plage.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la conversion est effectuée.
IN	IN contient une référence pour la valeur BCD-4, REAL ou Constante à convertir en entier.
ok	La sortie ok est excitée lorsque validation est excitée, sauf si la donnée est en dehors de la plage ou NaN (Pas un Nombre).
Q	La sortie Q contient la forme entière de la valeur d'origine dans IN.

Types de mémoire valides

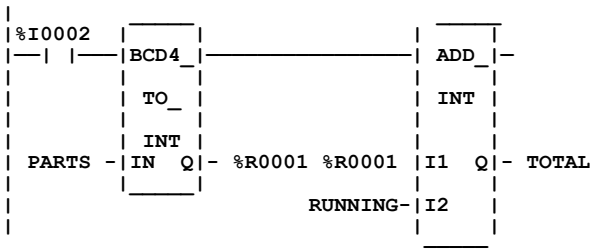
Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
IN		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

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

• Référence valide ou transmission du flux à travers la fonction.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0002 est mise à "1", la valeur BCD-4 de PARTS est convertie en entier signé et passée à la fonction ADD où elle est ajoutée à l'entier signé représenté par la référence RUNNING. Le résultat de la fonction ADD est mis dans la référence TOTAL.



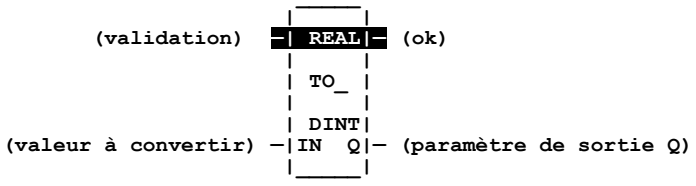
—>DINT (REAL)

La fonction Conversion en entier signé à double précision est utilisée pour générer un entier signé à double précision à partir d'une donnée REAL. La donnée d'origine n'est pas modifiée par cette fonction.

Remarque

Le type de données REAL n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou supérieure, ou sur toutes les versions de l'UC352.

Lorsque la fonction reçoit le flux d'énergie, elle effectue la conversion et met le résultat dans la sortie Q. La fonction laisse toujours passer le flux d'énergie lorsqu'elle est alimentée, sauf si la valeur réelle est en dehors de la plage.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la conversion est effectuée.
IN	IN contient une référence pour la valeur à convertir en entier à double précision.
ok	La sortie ok est excitée lorsque validation est excitée, sauf si la valeur réelle est en dehors de la plage.
Q	La sortie Q contient la forme entière signée à double précision de la valeur d'origine dans IN.

Remarque

Une perte de précision peut se produire lors de la conversion REAL en DINT, car REAL est constitué de 24 bits significatifs.

Types de mémoire valides

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

• Référence valide ou transmission du flux à travers la fonction.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0002 est à "1", la valeur réelle du paramètre d'entrée %R0017 est convertie en entier signé à double précision et le résultat est mis dans l'emplacement %R0001. La sortie %Q1001 est mise à "1" lorsque la fonction s'exécute correctement.

—>REAL (INT, DINT, BCD-4, WORD)

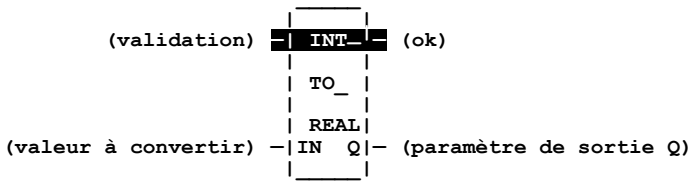
La fonction Conversion en réel est utilisée pour générer la valeur réelle de la donnée d'entrée. La donnée d'origine n'est pas modifiée par cette fonction.

Lorsque la fonction reçoit le flux d'énergie, elle effectue la conversion et met le résultat dans la sortie Q. La fonction laisse passer le flux d'énergie lorsqu'elle est alimentée, sauf si la conversion spécifiée donne une valeur en dehors de la plage.

Une perte de précision peut se produire lors de la conversion DINT en REAL car le nombre de bits significatifs est réduit à 24.

Remarque

Cette fonction n'est disponible que pour les UC des modèles 35x et 36x, Version 9 ou supérieure, ou sur toutes les versions de l'UC352.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la conversion est effectuée.
IN	IN contient une référence pour la valeur entière à convertir en REAL.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans erreur.
Q	La sortie Q contient la forme REAL de la valeur d'origine dans IN.

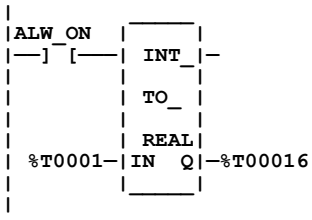
Types de mémoire valides

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

• Référence ou emplacement valide où l'énergie peut passer dans la fonction.
o Invalide pour DINT_TO_REAL.

Exemple

Dans l'exemple suivant, la valeur entière de l'entrée IN est 678. La valeur résultante mise dans %T0016 est 678.000.



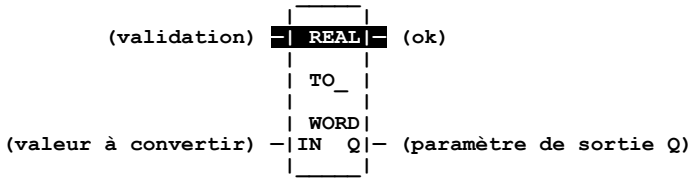
—>WORD (REAL)

La fonction Conversion en mot est utilisée pour générer le mot équivalant à la donnée réelle. La donnée d'origine n'est pas modifiée par cette fonction.

Remarque

Cette fonction n'est disponible que sur les UC modèles 35x et 36x.

Lorsque la fonction reçoit le flux d'énergie, elle effectue la conversion et donne le résultat par la sortie Q. La fonction laisse passer le flux d'énergie lorsqu'elle est alimentée, sauf si la conversion spécifiée donne une valeur en dehors de la plage de 0 à FFFFh.



Paramètres

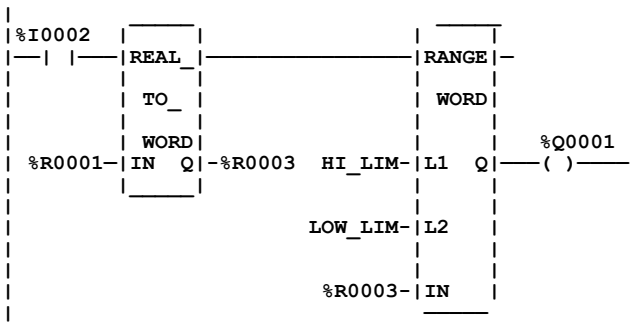
Paramètre	Description
validation	Lorsque la fonction est validée, la conversion est effectuée.
IN	IN contient une référence pour la valeur à convertir en WORD.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans erreur.
Q	La sortie Q contient la forme entière non signé de la valeur d'origine dans IN.

Types de mémoire valides

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

• Référence valide ou transmission du flux à travers la fonction.

Exemple


%I0002
REAL_
TO_
WORD
IN
Q
%R0001
%R0003
HI_LIM
LOW_LIM
%R0003
RANGE
WORD
L1
Q
L2
IN
%Q0001

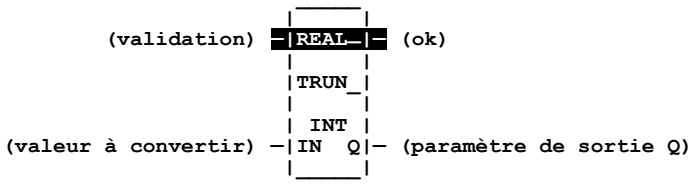
TRUN (INT, DINT)

La fonction troncation est utilisée pour arrondir un nombre réel à la valeur par défaut. La donnée d'origine n'est pas modifiée par cette fonction.

Remarque

Les UC des modèles 35x et 36x (Version 9 ou supérieure et toutes les versions de l'UC352) sont les seules UC de la Série 90-30 ayant la possibilité de virgule flottante ; par conséquent, la fonction TRUN ne s'applique pas aux autres UC 90-30.

Lorsque la fonction reçoit le flux d'énergie, elle effectue la conversion et met le résultat dans la sortie Q. Pour l'UC 352, la fonction laisse passer le flux d'énergie lorsqu'elle est alimentée, sauf si la conversion spécifiée résulte en une valeur qui se trouve en dehors de la plage ou si IN est NaN (Pas un Nombre). Pour toutes les autres UC modèles 35x et 36x, la fonction ne laisse *pas* passer l'énergie.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la conversion est effectuée.
IN	IN contient une référence pour la valeur réelle à tronquer.
Ok	La sortie ok est excitée lorsque la fonction est effectuée sans erreur, sauf si la valeur est en dehors de la plage ou si IN est NaN (Pas un Nombre).
Q	Q contient la valeur INT ou DINT tronquée de la valeur d'origine de IN.

Remarque

Une perte de précision peut se produire lors de la conversion REAL en DINT, car REAL a 24 bits significatifs.

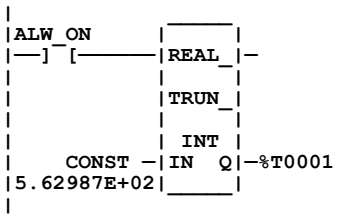
Types de mémoire valides

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

• Référence ou emplacement valide où l'énergie peut passer dans la fonction.
o Valide pour REAL_TRUN_INT seulement.

Exemple

Dans l'exemple suivant, la constante affichée est tronquée et l'entier résultant 562 est mis dans %T0001.



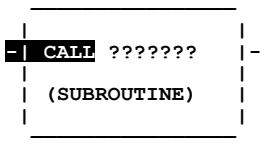
Chapitre 12 Fonctions de commande

Ce chapitre décrit les fonctions de commande qui peuvent être utilisées pour limiter l'exécution d'un programme et modifier la façon dont l'UC exécute le programme d'application. (Se référer au Chapitre 2, section 1, “Résumé du cycle de l'API,” pour des informations sur le cycle de l'API.

Fonction	Description	Page
CALL	Provoque le saut de l'exécution du programme dans un bloc sous-programme spécifié.	12-2
DOIO	Pendant un cycle, rafraîchit immédiatement un groupe spécifié d'entrées et de sorties. (Toutes les entrées et sorties d'un module sont rafraîchies si des adresses de référence de ce module sont inclus dans la fonction DOIO. Des rafraîchissements partiels des E/S du module ne sont pas effectués). Optionnellement, une copie des E/S scrutées peut être placée dans une mémoire interne, plutôt que les points d'entrée réels.	12-3
SER	Enregistreur d’événements Séquentiels— collecte une série d'échantillons. Un bloc de commande fonctionnel contient la configuration fournie par l'utilisateur de l'exécution du bloc fonctionnel, une configuration d'échantillon et des paramètres d'opération.	12-8
END	Donne une fin temporaire de la logique. Le programme exécute du premier circuit au dernier ou l'instruction END, quel que soit le premier rencontré. Cette instruction est utile pour le déverminage, mais elle n'est pas permise en programmation SFC (se référer à la Remarque de la page 12-8).	12-21
MCR et MCRN	Programme un relais de contrôle maître. Un MCR provoque l'exécution de tous les circuits, entre le MCR et son ENDMCR, sans flux d'énergie. Le logiciel Logicmaster 90-30/20/Micro supporte deux formes de fonction MCR ; une forme imbriquée (MCRN)et une forme non imbriquée (MCR).	12-22
ENDMCR et ENDMCRN	Indique que la logique suivante doit être exécutée avec un flux d'énergie normal. Le logiciel Logicmaster 90-30/20/Micro supporte deux formes de fonction ENDMCR ; une forme imbriquée (ENDMCRN)et une forme non imbriquée (ENDMCR).	12-25
JUMP et JUMPN	Provoque le saut de l'exécution d'un programme vers un emplacement spécifié (indiqué par une étiquette (LABEL), voir ci-dessous) dans la logique. Le logiciel Logicmaster 90-30/20 /Micro supporte deux formes de fonction JUMP ; une forme non imbriquée (JUMP) et une forme imbriquée (JUMPN).	12-26
LABEL et LABELN	Spécifie l'emplacement cible d'une instruction JUMP. Le logiciel Logicmaster 90-30/20 /Micro supporte deux formes de fonction LABEL ; une forme non imbriquée (LABEL) et une forme imbriquée (LABELN).	12-28
COMMENT	Met un commentaire (explication de circuit) dans le programme. Après la programmation d'une instruction, le texte peut être tapé en "zoomant" dans l'instruction.	12-29
SVCREQ	Demande un service API spécial. (Voir la liste des demandes de service à la page 12-30).	12-30
PID	Fournit deux algorithmes de commande en boucle fermée PID (proportionnelle/intégrale/dérivée) : • Algorithme PID standard ISA (PIDISA). • Algorithme à expression indépendante (PIDIND).	12-64

CALL

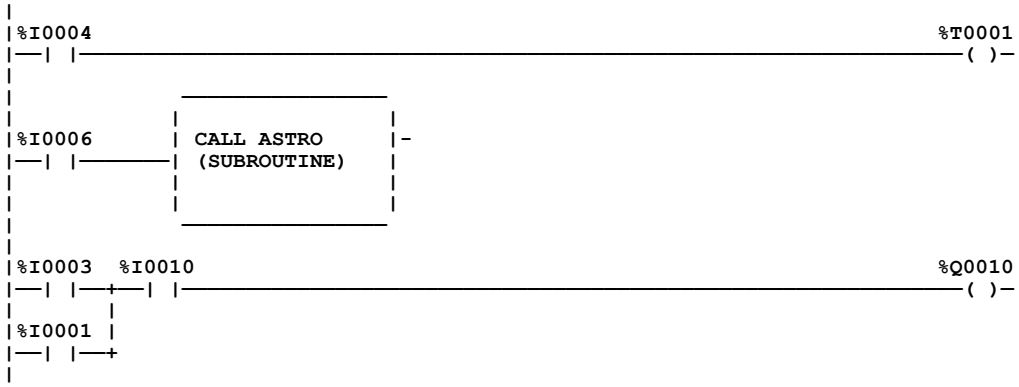
Utiliser la fonction CALL pour provoquer un saut de l'exécution du programme vers un bloc de sous-programme spécifié.



Lorsqu'une fonction CALL reçoit le flux d'énergie, elle dévie scrutation immédiatement dans le bloc sous-programme désigné et l'exécute. Lorsque l'exécution du bloc sous-programme est terminée, la commande retourne au point de la logique suivant immédiatement l'instruction CALL.

Exemple

L'exemple suivant montre l'instruction CALL d'un sous-programme telle qu'elle apparaît dans le bloc d'appel. En positionnant le curseur dans l'instruction, vous pouvez appuyer sur **F10** pour zoomer dans le sous-programme.



Remarque

Les API Micro ne supportent pas les sous-programmes ; par conséquent, la fonction CALL ne peut être utilisée avec un API Micro.

DOIO

La fonction DO I/O (DOIO) est utilisée pour rafraîchir les entrées ou les sorties pendant un cycle parallèlement au programme. La fonction DOIO peut également être utilisée pour rafraîchir des E/S sélectionnées pendant le programme, en plus de la scrutation normale des E/S.

Si des références d'entrée sont spécifiées, la fonction permet d'obtenir les valeurs d'entrée les plus récentes dans le programme logique. Si des références de sortie sont spécifiées, DO I/O rafraîchit les sorties avec les valeurs les plus récentes stockées dans la mémoire d' E/S. Les E/S sont rafraîchies par modules d'E/S; l'API ajuste les références, si nécessaire, pendant que la fonction s'exécute.

La fonction DOIO possède quatre paramètres d'entrée et un paramètre de sortie. Lorsque la fonction reçoit le flux d'énergie et que des références d'entrée sont spécifiées, les points d'entrée commençant à la référence (ST) et finissant à END sont scrutés. Si une référence est spécifiée pour ALT, une copie des nouvelles valeurs d'entrée est mise en mémoire, en commençant à cette référence et les points d'entrée réels ne sont pas rafraîchis. ALT doit être de la même taille que le type de référence scruté. Si une référence logique est utilisée pour ST et END, ALT doit également être logique. Si aucune référence n'est spécifiée pour ALT, les points d'entrée réels sont rafraîchis.

Lorsque la fonction DOIO reçoit le flux d'énergie et que des références de sortie sont spécifiées, les points de sortie commençant à la référence (ST) et finissant à END sont écrits dans les modules de sorties. Si les sorties doivent être écrites dans les modules de sorties à partir de la mémoire interne, autre que %Q ou %AQ, la référence de début peut être spécifiée pour ALT. Le groupe de sorties écrites dans les modules de sorties est spécifié par la référence de début (ST) et la référence de fin (END).

L'exécution de la fonction se poursuit jusqu'à ce que toutes les entrées du groupe sélectionné soient lues ou que toutes les sorties soient écrites sur les cartes d'E/S. l'exécution du programme reprend alors à la ligne suivant le DO I/O.

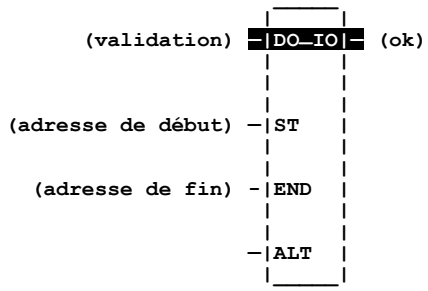
Si la plage de références inclut un module optionnel (HSC, APM, etc.), toutes les données d'entrée (%I et %AI) ou toutes les données de sorties (%Q et %AQ) de ce module seront scrutées. Le paramètre ALT est ignoré pendant la scrutation des modules optionnel. De même, la plage de référence ne doit pas inclure de module GCM amélioré (voir la Remarque ci-dessous).

Remarque

Pour les UC Version 9 ou supérieures, la fonction DOIO *peut* être utilisée avec un module GCM amélioré.

La fonction laisse passer le flux d'énergie, sauf si :

- Toutes les références du type spécifié ne sont pas présentes dans la plage sélectionnée.
- L'UC n'est pas capable de gérer correctement la liste temporaire des E/S créée par la fonction.
- La plage spécifiée inclut des modules d'E/S qui sont associés à un défaut de “Perte d'E/S”.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, une scrutation limitée d'entrées ou de sorties est effectuée.
ST	ST est l'adresse de début ou le groupe de points d'entrée ou de sortie ou de mots à desservir.
END	END est l'adresse de fin ou le groupe de points d'entrée ou de sortie ou de mots à desservir.
ALT	Pour la scrutation des entrées, ALT spécifie l'adresse pour stocker les points d'entrées scrutés/valeurs sur mot. Pour la scrutation des sorties, ALT spécifie l'adresse où obtenir les points de sortie/valeurs sur mot à envoyer aux modules d'E/S. Pour les UC Modèle 331 et supérieurs, le paramètre ALT peut avoir un effet sur la vitesse d'exécution du bloc fonctionnel DOIO (voir la Remarque ci-dessous et la section sur la fonction DO I/O améliorée pour les UC 331 et supérieures à la page 12-4).
ok	La sortie ok est excitée lorsque la scrutation des entrées ou des sorties se termine normalement.

Remarque

Pour les UC Modèle 331 et supérieures, le paramètre ALT du bloc fonctionnel DOIO peut être utilisé pour entrer l'emplacement d'un module dans le châssis principal. Lorsque ceci est fait, le bloc fonctionnel s'exécutera en 80 microsecondes au lieu des 236 microsecondes nécessaires lorsque le bloc est programmé sans paramètre ALT. Aucune vérification d'erreur n'est effectuée pour éviter le chevauchement des adresses de référence ou les incohérences de types de module.

Types de mémoire valides

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

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Exemple d'entrée 1

Dans l'exemple suivant, lorsque l'entrée 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 de la référence %M0001 à %M0064. Les points d'entrée réels ne sont pas rafraîchis. Cette forme peut être utilisée pour comparer les valeurs courantes des points d'entrée avec les valeurs des points d'entrée en début de scrutation.

Exemple d'entrée 2

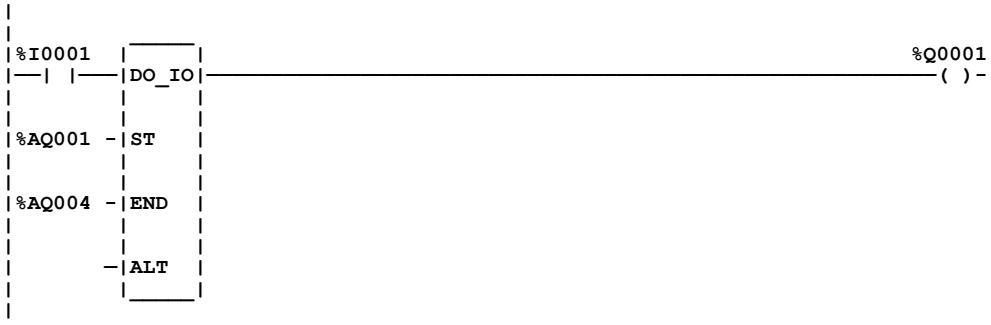
Dans l'exemple suivant, lorsque l'entrée validation %I0001 est à "1", les références %I0001 à %I0064 sont scrutées et %Q0001 est mise à "1". Les entrées scrutées sont placées dans la mémoire d'état des entrées de la référence %I0001 à %I0064. Cette forme permet de scruter une ou plusieurs fois les points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

Exemple de sortie 1

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



Fonction DO I/O améliorée pour les UC 331 et supérieures

Précautions

Si la fonction DO I/O améliorée est utilisée dans un programme, le programme ne pourra être chargé avec une version du logiciel Logicmaster 90-30/20 antérieure à 4.01.

Une version améliorée de la fonction DO I/O (DOIO) est disponible pour la Version 4.20 ou supérieure des UC des Modèles 331 et supérieurs. Cette version améliorée de la fonction DOIO ne peut être utilisée qu'avec un seul module d'entrées logiques ou de sorties logiques, 8 points, 16 points ou 32 points.

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

Remarque

La seule vérification effectuée par le bloc fonctionnel DOIO amélioré est de vérifier l'état du module dans l'emplacement spécifié pour voir si le module est OK.

La fonction DOIO améliorée ne s'applique qu'aux modules situés dans le châssis principal. Par conséquent, le paramètre ALT doit être entre 2 et 5 pour un châssis à cinq emplacements ou entre 2 et 10 pour un châssis à 10 emplacements.

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

Le tableau suivant compare les temps d'exécution d'un bloc fonctionnel DOIO normal pour un module d'entrées/sorties logiques de 8 points, 16 points o 32 points avec ceux d'un bloc fonctionnel DOIO amélioré.

Module	Temps d'exécution DOIO normal	Temps d'exécution DOIO amélioré
Module d'entrées logiques 8 points Module de sorties logiques 8 points	224 microsecondes 208 microsecondes	67 microsecondes 48 microsecondes
Module d'entrées logiques 16 points Module de sorties logiques 16 points	224 microsecondes 211 microsecondes	68 microsecondes 47 microsecondes
Module d'entrées logiques 32 points Module de sorties logiques 32 points	247 microsecondes 226 microsecondes	91 microsecondes 50 microsecondes

SER

Caractéristiques

- Le bloc fonctionnel SER (Enregistreur d'Evénements Séquentiels) collecte une série d'échantillons. Un bloc fonctionnel SER collecte jusqu'à 32 bits contigus ou non par échantillons lorsque l'entrée Validation reçoit le flux d'énergie.
- Chaque SER peut capturer jusqu'à 1024 échantillons.
- Si le bloc fonctionnel SER est imbriqué dans un sous-programme périodique, la vitesse d'échantillonnage est déterminée par la vitesse d'exécution du sous-programme périodique.
- Seul l'échantillon de déclenchement est horodaté. L'échantillon de déclenchement peut être horodaté en BCD (la résolution maximale est 1s) ou en format POSIX (la résolution maximale est 10ms). SER ne supporte pas plus d'un horodatage par enregistrement.
- SER peut être configuré pour les modes pré-, mi- ou post-déclenchement. (Voir page 12-14).
- L'opération SER est configurée par un bloc de paramètres que vous pouvez créer en utilisant une série de commandes Block Move (BLKMV). (Voir page 12-10

Remarque

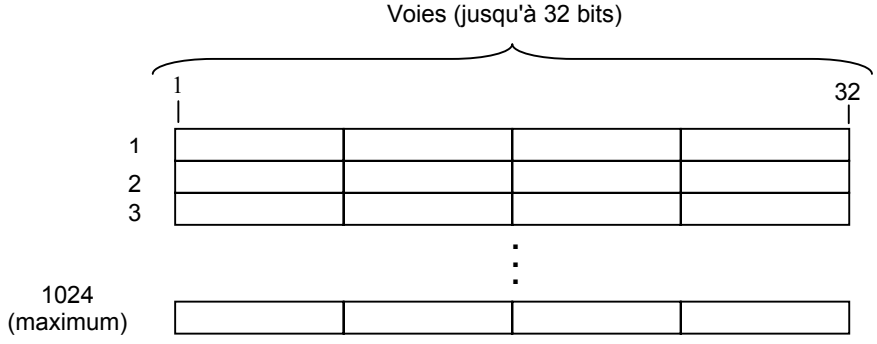
La synchronisation API-à-API n'est pas supportée.

Le bloc fonctionnel possède une sortie et trois entrées : validation, réinitialisation et déclenchement.

Exemple de bloc fonctionnel SER

Remarque

Cette fonction nécessite un macro-programme d'UC Version 9.00 ou supérieure et elle n'est disponible *que pour* les UC 350 et supérieures.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée et que l'entrée réinitialisation est à "0", le bloc fonctionnel SER collecte un échantillon de toutes les voies configurées.
R	Lorsque l'entrée réinitialisation reçoit le flux d'énergie, la fonction SER est réinitialisée sans tenir compte de l'état de l'entrée validation. Tampon d'échantillon, Décalage d'échantillon de déclenchement, Temps de déclenchement et Décalage d'échantillon courant sont tous mis à "0". Le bloc fonctionnel reste dans le même état de réinitialisation jusqu'à ce que le flux d'énergie soit enlevé de l'entrée réinitialisation. La sortie OK est mise à "0" pendant l'état réinitialisation. Lorsque le flux d'énergie est enlevé de l'entrée réinitialisation, l'échantillonnage se poursuit.
T	Si le mode Entrée déclenchement est sélectionné et que le bloc fonctionnel est validé, lorsque l'entrée déclenchement passe à "1", SER passe à l'état déclenché. Le Temps de déclenchement, le Décalage d'échantillon de déclenchement et un échantillon sont enregistrés. L'échantillon de déclenchement sera enregistré sans tenir compte du nombre d'échantillons prélevés. Après le déclenchement, l'enregistreur d'évènement continue l'échantillonnage jusqu'à ce que le nombre d'Echantillons après déclenchement soit satisfait, auquel moment, il s'arrête de collecter des échantillons jusqu'à ce que le flux d'énergie apparaisse sur l'entrée réinitialisation. Si le mode Déclenchement est mis sur Tampon plein, le signal de déclenchement est ignoré. Pour des informations sur la configuration du Mode déclenchement, voir “Bloc de commande” à la page 12-10.
Référence de début	Le groupe de 78 mots du bloc de commande commence à cette référence. Le bloc de commandedéfinit l'exécution du bloc fonctionnel, la configuration de l'échantillon et les commande de l'opération. Pour les détails, voir “Bloc de commande” à la page 12-10
ok	La sortie ok est excitée lorsque les conditions de déclenchement sont satisfaites (particulièrement par le paramètre Mode Déclenchement) et que tout l'échantillonnage est terminé. La sortie continue à recevoir le flux d'énergie sans tenir compte de l'entrée validation jusqu'à ce que réinitialisation reçoive le flux d'énergie.

Types de mémoire valides

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

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Bloc de commande

Le bloc de commande est un groupe de 78 mots qui définit les informations concernant la capture de données et le mécanisme de déclenchement pour la fonction SER. Dans un programme particulier, un seul bloc de commande Enregistreur d'évènement séquentiel peut être associé à chaque bloc de commande et bloc de données.

Effectuer les étapes suivantes pour configurer les paramètres du bloc fonctionnel SER :

1. Placer les valeurs stockées du groupe comme défini dans la table ci-dessous. Vous pouvez utiliser des transferts de blocs pour initialiser les registres ou initialiser les données de la table des registres, et stocker la table avant l'activation de la fonction SER.
2. Ajouter le bloc fonctionnel SER à votre logique en échelle.

Remarque

Si vous avez besoin de x voies où x n'est pas égal à 8, 16, 24, mais est inférieur à 32, vous devez sélectionner un nombre de voies supérieur à x et un multiple de 8, et remplir avec une description de voie nulle pour les voies inutilisées. Une description de voie nulle correspond à un sélecteur de segment de 0xFFh, un paramètre de longueur égal au nombre de voies inutilisées et un décalage de 0.

Mot	Paramètre	Description
0 (référence de début)	Etat	Variable à lecture seule qui indique l'état courant du bloc fonctionnel SER. Des informations supplémentaires sont fournies dans Données supplémentaires d'état, (Mot 1). **Remarque :** Si une erreur est détectée dans le bloc de commande, l'état sera mis à 6, la sortie OK sera effacée et aucune action n'aura lieu. Les réglages d'Etat comprennent : 0 = Réinitialisation 1 = Inactif 2 = Actif 3 = Déclenché 4 = Terminé 5 = Erreur dépassement 6 = Erreur de paramètre
1	Données supplémentaires d'état	Une variable à lecture seule qui fournit des informations d'état supplémentaires concernant la fonction SER. Voir “Etats des données supplémentaires d'état” à la page 12-12 pour les réglages de ce paramètre.
2	Mode Déclenchement	Définit les conditions pour le bloc fonctionnel SER pour passer à l'état déclenché. Les réglages valides sont : 0 = Mode Entrée déclenchement 1 = Mode Tampon plein Dans le mode Entrée déclenchement, si le bloc fonctionnel est validé, un horodatage est généré lorsque le signal de déclenchement est activé. L'échantillonnage continue jusqu'à ce que le nombre d'échantillons après déclenchement soit satisfait. Lorsque ceci se produit, la sortie OK est activée. Dans le mode Tampon plein, le signal de déclenchement est ignoré. Lorsque le bloc fonctionnel est validé, l'échantillonnage continue jusqu'à ce que le tampon d'échantillons soit rempli. Lorsque ceci se produit, la sortie OK est activée. Le paramètre Nombre d'échantillons règle la taille du tampon size.
3	Format de l'heure de déclenchement	Détermine comment l'heure de déclenchement sera affichée. Pour un affichage BCD, mettre ce paramètre à 0. Pour un affichage POSIX, mettre ce paramètre à 1. (Pour les détails, voir la page 12-20).
4—7	Réservé	Les mots 4 à 7 sont réservés et doivent être mis à zéro.

Mot	Paramètre	Description
8	Nombre de voies (bits par échantillon)	Spécifie le nombre de bits de données qui seront échantillonnés et renvoyés dans le tampon d'échantillons à chaque exécution du bloc fonctionnel. Les choix valides sont 8, 16, 24 ou 32 bits. L'incrément est en taille d'octet (8 bits) et toutes les voies inutilisées doivent être configurées avec une description de voie nulle. (Voir Mots 14—77).
9	Nombre d'échantillons	Spécifie la taille du tampon d'échantillons. Les choix valides sont 1 à 1024 échantillons. (La taille actuelle en bits du tampon est le nombre d'échantillons multiplié par le nombre de voies).
10	Nombre d'échantillons après déclenchement	Spécifie le nombre d'échantillons collectés après que la condition de déclenchement soit devenue vraie. Ce paramètre peut être réglé à une valeur comprise entre 0 et (nombre d'échantillons – 1). Ce paramètre n'est valide que lorsque le mode Déclenchement est mis sur l'entrée Déclenchement (0).
11	Emplacement de module d'entrées	Spécifie l'emplacement du module d'entrées pour l'échantillonnage des données (emplacement dans le châssis principal). Si la valeur est 0, la scrutation du module d'entrées est désactivée. Lorsqu'un module d'entrées est scruté, ses valeurs sont stockées localement et les valeurs des adresses de référence configurées pour le module ne sont pas affectées. Pour stocker les valeurs du module d'entrées scruté dans le tampon d'échantillons de blocs de données, une description de voie doit être fournie. Si le module n'est pas présent ou est défectueux au moment de la scrutation, les données renvoyées seront zéro. Un défaut ne sera pas enregistré dans la table des défauts si cela se produit ; l'indication de défaut sera laissée au scrutateur des E/S.
12	Sélecteur de segment de bloc de données	Spécifie le type de données allouées au bloc de données. Par exemple, si vous voulez commencer à %R0100, vous devez entrer 08 pour ce paramètre. Les réglages valides de ce paramètre sont : %R (08h), %AI (0Ah), %AQ (0Ch). Pour des détails sur le bloc de données, voir page 12-13.
13	Décalage de bloc de données	Spécifie la référence de début du bloc de données. Ce paramètre est à base de zéro. Par exemple, si vous voulez commencer à %R0100, vous devez entrer 99 pour ce paramètre. S'assurer de laisser suffisamment de mémoire pour le bloc de données entier.
14—77	Description de voie	Spécifie l'emplacement de référence (Sélecteur de segment, Longueur et Décalage) associé à une voie particulière. Il peut y avoir de 1 à 32 descriptions de voie, selon le nombre de voies échantillonnées et la longueur des données. Les données sont renvoyées dans l'ordre défini dans cette section.
	Sélecteur de segment/longueur de voie	Entré en valeur hexadécimale, ce mot définit le sélecteur de segment et la longueur des données (en bits). MSB = Sélecteur de segment. LSB = Longueur des données. La longueur des données est utile pour les échantillons qui sont contigus. Le sélecteur de segment peut être réglé pour tout type de données logiques : %I (46h), %Q (48h), %M (4Ch), %T (4Ah), %G (56h), %S (54h), %SA (4Eh), %SB (50h), %SC (52h), Sélecteur nul (FFh) et Sélecteur de module d'entrées (00h).
		Le paramètre de longueur peut être de 1—32, mais la somme de toutes les longueurs ne doit pas être supérieure au paramètre Nombre de voies. Une longueur supérieure à 1 permet de configurer des voies contiguës multiples avec une seule description de voie.
	Décalage de voie	Entré comme valeur hexadécimale, ce mot définit le décalage de BIT pour le type de données ou le module d'entrées spécifié dans le Sélecteur de segment. Ce décalage est à base de zéro. La plage de ce paramètre varie selon le Sélecteur de segment (type et longueur de données). Le décalage indique l'emplacement, dans la table des données ou dans le module d'entrées, où doit se faire l'échantillonnage.

Données supplémentaires d'état Etats

Les données supplémentaires d'état (Mot 1 dans le bloc de commande) fournissent des informations d'état supplémentaires pour la fonction SER.

Valeur	Etat	Description
0	Etat réinitialisation	L'entrée réinitialisation reçoit le flux d'énergie. Tampon d'échantillons, Décalage d'échantillons déclenchés, Heure de déclenchement et Décalage d'échantillons courant sont tous mis à zéro. La sortie est maintenue sans flux d'énergie. Le passage en ***Etat inactif*** se produit lorsque le flux d'énergie de la réinitialisation est enlevé. Les Données supplémentaires d'état n'ont aucune signification et seront effacées à zéro.
1	Inactif	Etat entre Réinitialisé et Actif. Aucune action n'est effectuée dans cet état. La sortie de SER est maintenue sans flux d'énergie. Le passage à l'état Actif se produit lorsque le bloc fonctionnel reçoit le flux d'énergie.
2	Actif	L'entrée validation a reçu le flux d'énergie, mais le bloc fonctionnel n'est pas réinitialisé, est en erreur ou déclenché. Un échantillon est enregistré à chaque exécution lorsque le bloc fonctionnel est validé. La sortie est maintenue sans flux d'énergie. La condition de déclenchement (spécifiée par le paramètre Mode Déclenchement) est surveillée et provoquera le passage à l'état déclenché si les conditions sont vraies. Si davantage d'échantillons que le “Nombre d'échantillons” ont été prélevés, les Données supplémentaires d'état seront mises à 0x01, autrement, elles seront à 0x00.
3	Déclenché	Etat si la condition de déclenchement, définie par le Mode Déclenchement, est vraie. Des échantillons supplémentaires sont prélevés selon les réglages du mode déclenchement et du paramètre. La sortie est maintenue sans flux d'énergie. Le passage à l'état se produira lorsque l'ensemble de l'échantillonnage sera terminé. Si davantage d'échantillons que le “Nombre d'échantillons” ont été prélevés, les Données supplémentaires d'état seront mises à 0x01, autrement, elles seront à 0x00.
4	Terminé	L'échantillonnage est terminé. La sortie reçoit le flux d'énergie. Seul le passage à l'état Réinitialisé est permis. Si davantage d'échantillons que le “Nombre d'échantillons” ont été prélevés, les Données supplémentaires d'état seront mises à 0x01, autrement, elles seront à 0x00.
5	Erreur de dépassement	Le bloc de commande/données a dépassé la fin de son type de mémoire. La sortie est maintenue sans flux d'énergie. Seul le passage à l'état Réinitialisé est permis. Les Données supplémentaires d'état n'ont aucune signification et seront effacées à zéro.
6	Erreur de paramètre	Il y a une erreur dans le bloc de commande ou dans d'autres paramètres de l'opération. La sortie est maintenue sans flux d'énergie. Seul le passage à l'état Réinitialisé est permis. Le mot de Données supplémentaires d'état contient le décalage dans le bloc de commande auquel l'erreur de paramètre s'est produite.
7	Erreur d'état	Le paramètre Etat est invalide. La sortie est maintenue sans flux d'énergie. Seul le passage à l'état Réinitialisé est permis. La valeur d'état invalide sera stockée dans l'emplacement Données supplémentaires d'état, dans le bloc de commande.

Bloc de données SER Format

Le bloc de données SER contient les informations sur le tampon d'échantillons, les décalages d'échantillons et le déclenchement. Ces informations sont fournies par l'UC et vous pouvez seulement lire dans cette zone de données. Il est de votre responsabilité d'allouer suffisamment d'espace registre pour le bloc de données. Le format du bloc est le suivant :

Mot*	Description du paramètre
0	Numéro de décalage d'échantillon courant. Référence l'emplacement où l'échantillon le plus récent a été placé. Le paramètre est à base de zéro. La plage valide est –1 à 1023. Emplacement du registre de l'échantillon = (Octets par échantillons) * (Paramètre de décalage)/2 + (Registre de début du tampon d'échantillons). **Remarque :** Cette valeur n'est pas valide tant qu'un échantillon n'est pas prélevé. Cette valeur est mise à –1 lorsque la fonction SER est réinitialisée par l'entrée réinitialisation.
1	Numéro de décalage de l'échantillon déclenché. Référence l'emplacement de stockage de l'échantillon obtenu lorsque la condition de déclenchement est passée à l'état Vrai. Le paramètre est à base de zéro. La plage valide est 0 à 1023. Emplacement du registre de l'échantillon = (Octets par échantillons) * (Paramètre de décalage)/2 + (Registre de début du tampon d'échantillons). **Remarque :** Cette valeur n'est pas valide tant que la condition de déclenchement n'est pas satisfaite. Cette valeur est mise à 0 lorsque la fonction SER est réinitialisée par l'entrée réinitialisation.
2 à 5	Heure de déclenchement : Indique l'heure, selon l'horloge interne de l'API, à laquelle la condition de déclenchement est passée à l'état vrai dans le bloc fonctionnel. L'heure peut être affichée en BCD (par défaut) ou en format POSIX. Le format est déterminé par le paramètre *Format de l'heure de déclenchement* dans le bloc de commande. Cette valeur est initialisée à zéro à l'activation de l'entrée réinitialisation.
6 à la fin du tampon d'échantillons	Tampon d'échantillons. La zone mémoire qui conserve les échantillons de données. Cette zone est mise à zéro lorsque le paramètre de réinitialisation est excité. La taille du tampon d'échantillons varie selon le nombre de voies et la taille de l'échantillon. Le tampon d'échantillons est un tampon circulaire ; lorsque le dernier emplacement est écrit, l'échantillon suivant écrasera l'échantillon du premier registre. Fin du tampon d'échantillons = 5 + ({[(Nombre d'échantillons à prélever) * (Nombre de voies à échantillonner / 8)] +1} / 2

*Décalage à partir de la référence de départ définie par le Sélecteur de segment de bloc de données (Mot 12) et le décalage de bloc de données (Mot 13) dans le bloc de commande.

Fonctionnement de SER

Si SER est validé lorsqu'il est scruté, il lit les points échantillons configurés et les met dans une liste circulaire. Après que le nombre d'échantillons configurés est prélevé, la sortie est mise à "1". La transition de la sortie peut être utilisée pour enregistrer le moment où le dernier échantillon est prélevé ou pour initier un échantillonnage supplémentaire. (Voir “Modes Echantillonnage”).

Le bloc fonctionnel SER doit être réinitialisé (valider le flux d'énergie dans l'entrée réinitialisation) avant de commencer l'échantillonnage. La réinitialisation initialise la zone bloc de données. Si l'état du bloc fonctionnel n'est pas réinitialisé, il s'exécutera avec les valeurs courantes du bloc de données, ce qui rendra le décalage d'échantillon incorrect et donnera des données invalides dans le bloc de données.

Le bloc de commande du bloc fonctionnel SER est scruté à chaque fois que le bloc fonctionnel est exécuté dans l'état Réinitialisé, Actif ou Déclenché. Si vous changez un paramètre de configuration du

bloc de commande pendant l'exécution du programme, le changement prendra effet la prochaine fois que le bloc fonctionnel SER, associé à ce bloc de commande, sera scruté. Si une erreur est rencontrée, l'opération s'arrête et le bloc fonctionnel passe à l'état d'erreur approprié. Vous devez corriger l'erreur et réinitialiser le bloc fonctionnel (valider le flux d'énergie de l'entrée réinitialisation) pour recommencer l'échantillonnage.

Si vous sélectionnez un module d'entrées à scruter, l'API ne vérifiera *pas* si le module est un module d'entrées logiques ou si les descriptions de voies associées au module ont des longueurs et des décalages valides par rapport à la taille du module. Vous devez régler correctement l'échantillonnage d'un module d'entrées. Bien que de multiples descriptions de voies puissent cibler un module d'entrées, le module ne reste scruté qu'une fois par exécution du bloc fonctionnel.

Le bloc fonctionnel SER peut être placé dans le programme logique utilisateur normal ou dans un sous-programme périodique. S'il est placé dans le programme logique utilisateur, la résolution de l'intervalle entre les scrutations est la résolution du temps de scrutation, qui peut varier selon le nombre et les types de fonctions actives lors d'une scrutation particulière. S'il est placé dans un sous-programme d'interruption, l'intervalle peut être réglé aussi bas que 1 ms et la résolution aura une répétition élevée à 1 ms avec quelques variations.

Le temps d'exécution d'un bloc fonctionnel avec un sous-programme périodique de 1 ms peut épuiser jusqu'à 50% des ressources de l'UC. Vous ne devez pas prévoir l'exécution de plus de deux fonctions SER avec un sous-programme périodique de 1 ms.

Modes Echantillonnage

Le mode échantillonnage de SER est déterminé par le mode Déclenchement (Mot 2 du bloc de commande) et les paramètres Nombre d'échantillons après déclenchement (Mot 10). Vous devrez interpréter le contenu du tampon d'échantillons en fonction de la façon dont vous avez configuré ces paramètres.

Echantillonnage contrôlé par déclenchement

Pour configurer les modes d'échantillonnage pré-, mi- et post déclenchement, le mode Déclenchement (Mot 2 = 0) est sélectionné. Le mode d'échantillonnage est contrôlé par le Nombre d'échantillons après déclenchement (Mot 10). Dans tous les cas, l'échantillonnage commence lorsque le signal Validation passe à "1". Lorsque le signal Déclenchement passe à "1", l'échantillonnage continue jusqu'à ce que le Nombre d'échantillons après déclenchement (Mot 10) soit prélevé. Le signal de sortie du bloc fonctionnel passe à "1" lorsque l'échantillonnage est terminé.

Si un nombre supérieur au Nombre d'échantillons configuré (Mot 9) est prélevé avant que la condition Nombre d'échantillons après déclenchement soit satisfaite, le tampon "se boucle" ce qui signifie que SER revient au début du tampon et écrase les échantillons initiaux.

Lorsque le déclenchement passe de "0" à "1", le temps du déclenchement est placé dans un emplacement configuré.

Pré-déclenchement

Prélève des échantillons jusqu'à ce que le déclenchement soit détecté.

Pour configurer ce mode, mettre le Mot 10 à la valeur 0, de façon à ce que lorsque le signal de déclenchement est activé, l'échantillonnage s'arrête et un horodatage est généré. (Tous les échantillons prélevés avant le déclenchement).

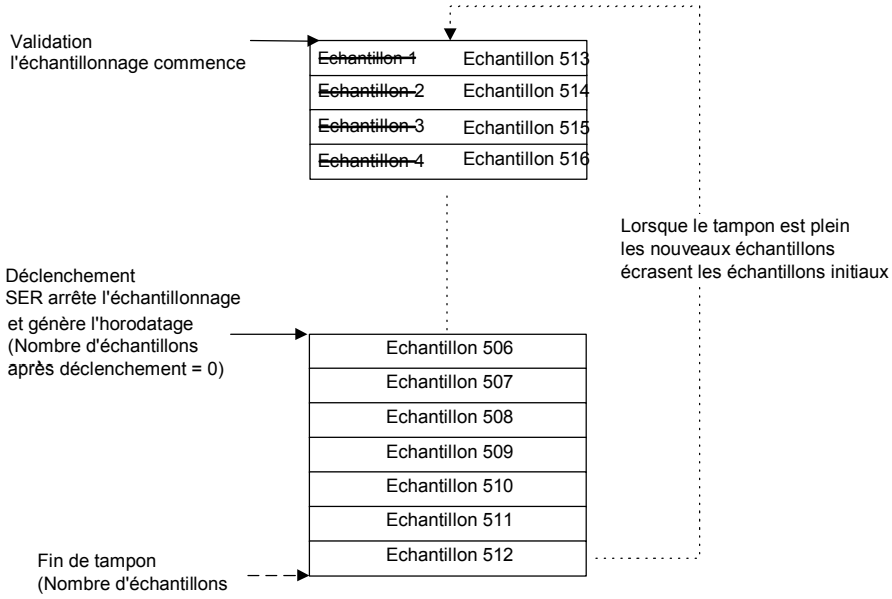


Figure 12-1. Exemple d'échantillonnage SER pré-déclenché

Mi-déclenchement

Prélève des échantillons continuellement jusqu'à ce que le nombre d'échantillons après déclenchements soit prélevé.

Pour configurer ce mode, mettre le Mot 10 à une valeur comprise entre 1 et le Nombre d'échantillons (Mot 9). Lorsque le signal de déclenchement est activé, l'échantillonnage continue jusqu'à ce que le nombre configuré soit prélevé. Dans l'exemple suivant, le Nombre d'échantillons après déclenchement est 12. Lorsque l'échantillonnage est terminé, le tampon contiendra 500 échantillons pré-déclenchés et 12 échantillons post-déclenchés.

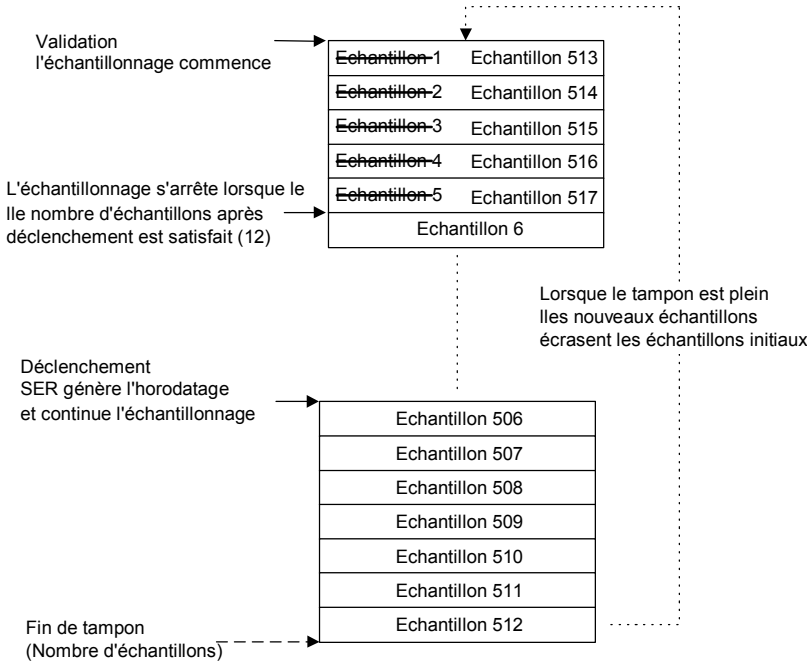


Figure 12-2. Exemple d'échantillonnage mi-déclenché de SER

Post-déclenchement

Prélève des échantillons continuellement jusqu'à ce que le nombre d'échantillons soit atteint.

Pour configurer ce mode, mettre le Mot 10 à une valeur égale au Nombre d'échantillons (Mot 9). Lorsque le signal de déclenchement est activé, l'échantillonnage continue jusqu'à ce que le nombre configuré soit prélevé. (Tous les échantillons prélevés après le déclenchement).

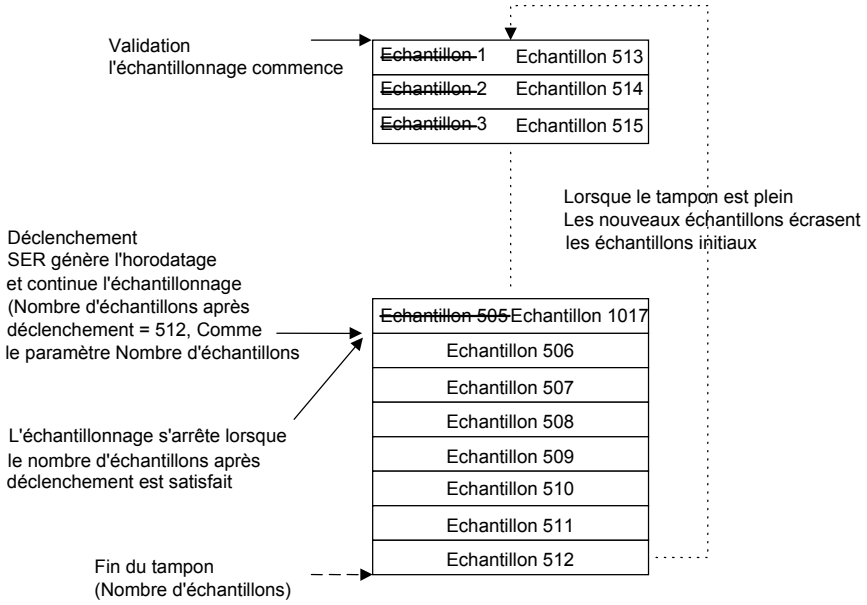


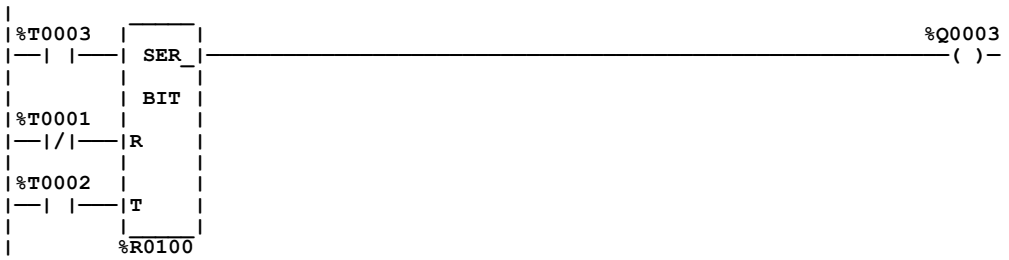
Figure 12-3. Echantillonnage post-déclenché de SER

Tampon plein (le déclenchement de commande pas l'échantillonnage)

Si le mode Déclenchement est mis à "1", le paramètre Nombre d'échantillons après déclenchement (Mot 10) est ignoré et le signal d'entrée Déclenchement n'a aucun effet sur l'opération du bloc fonctionnel. Lorsque le bloc fonctionnel est validé, l'échantillonnage continue jusqu'à ce que le nombre d'échantillons (Mot 8) soit prélevé, remplissant le tampon d'échantillons. Lorsque le tampon est plein, l'échantillonnage s'arrête, un horodatage du déclenchement est généré et la sortie OK du bloc fonctionnel passe à "1".

Exemple de SER

Dans l'exemple suivant, le bloc de commande a été réglé comme décrit dans la Table 12-1.



Exemple de bloc de commande

Dans cet exemple, le système possède un module d'entrées logiques de 16 points dans le châssis 0, emplacement 4 et le bloc de commande s'est exécuté suffisamment longtemps pour que 572 échantillons (512 + 60) soient prélevés. L'entrée Validation reçoit le flux d'énergie mais les entrées Réinitialisation et Déclenchement ne le reçoivent pas.

Table 12-1. Exemple de bloc de commande pour SER

Mot	Registre	Paramètre	Valeur (déc)	Valeur (hex)	Description
0	%R0100	Etat	2	0002	Le bloc fonctionnel est à l'état actif. Ceci signifie que le bloc fonctionnel s'exécute normalement et prélève un échantillon à chaque fois qu'il est rencontré dans la logique programme.
1	101	Etat de données excédentaires	1	0001	L'état de données excédentaires indique que plus de 512 échantillons ont été prélevés et que le tampon s'est déjà bouclé au moins une fois.
2	102	Mode Déclenchement	0	0000	L'enregistreur d'évènement est configuré pour se déclencher en fonction de l'entrée Déclenchement.
3	103	Format heure de déclenchement	0	0000	BCD
4	104	Réservé	0	0000	Les paramètres Réservés sont toujours mis à "0".
5	105	Réservé	0	0000	
6	106	Réservé	0	0000	
7	107	Réservé	0	0000	
8	108	Nombre de voies	24	0018	La configuration d'échantillon comporte des données de 24 bits.
9	109	Nombre d'échantillons à prélever	512	0200	La taille du tampon d'échantillons est de 512 échantillons. Noter que le tampon d'échantillons n'a pas 512 octets. Il est de 512 x (24/8) = 1536 octets ou 768 mots. (Chaque échantillon fait 3 octets de long).
10	110	Nombre d'échantillons après déclenchement	12	000C	Le nombre d'échantillons à prélever après le déclenchement est de 12.
11	111	Emplacement du module d'entrées	4	0004	Le module d'entrées du châssis 0, emplacement 4, sera scruté de façon à ce que ses données courantes soient disponibles pour l'échantillonnage.
12	112	Sélecteur de segment de bloc de données	8	0008	Le segment de données est 0x08 (%R).
13	113	Décalage de bloc de données	200	00C8	Le décalage est de 200 ce qui place le début du bloc de données à %R0201. Le décalage est une valeur à base de 0, mais les tables de registres commencent à %R0001. Par conséquent, le point de départ du bloc de données est %R0001 + 200 = %R0201.

Mot	Registre	Paramètre	Valeur (déc)	Valeur (hex)	Description
Mot	**Registre**	**Paramètre**	**Valeur (déc)**	**Valeur (hex)**	**Description**
Description des voies		Les mots restants contiennent les descriptions des voies. Dans cet exemple, six descriptions de voies ont été définies.			
14	114	Seg. Sél. : Longueur	17921	4601	Description de voie 1 : La première description de voie sélectionne le segment %I avec une longueur de 0. ceci choisit %I0001 pour la voie 1.
15	115	Décalage	0	0000	
16	116	Seg. Sél. : Longueur	-253	FF03	Description de voie 2 : La description de la deuxième voie sélectionne le Sélecteur NULL avec une longueur de 3 et un décalage de 0. Le sélecteur NULL fera que les voies 2 - 4 seront ignorées ou "sautées". Ces voies contiendront toujours une valeur d'échantillon de zéro.
17	117	Décalage	0	0000	
18	118	Seg. Sél. : Longueur	3	0003	Description de voie 3 : La troisième description de voie sélectionne le Sélecteur de module d'entrées avec une longueur de 3 et un décalage de 12. Le Sélecteur de module d'entrées provoque le prélèvement d'échantillons dans le module d'entrées. Cette description de voie choisit les valeurs des points 13, 14 et 15 du module d'entrées pour les voies 5 - 7.
19	119	Décalage	12	0012	
20	120	Seg. Sél. : Longueur	18434	4802	Description de voie 4 : La quatrième description de voie sélectionne le segment %Q avec une longueur de 2 et un décalage de 8. Ceci choisit %Q0009 et %Q0010 pour les voies 8 et 9.
21	121	Décalage	8	0008	
22	122	Seg. Sél. : Longueur	8	0008	Description de voie 5 : La cinquième description de voie est un autre sélecteur de module d'entrées. Il a une longueur de 8 et un décalage de 0. Ceci provoque le placement des points 1 à 8 du module d'entrées dans les voies 10 - 17.
23	123	Décalage	0	0000	
24	124	Seg. Sél. : Longueur	-249	FF07	Description de voie 6 : La sixième description de voie est un autre sélecteur NULL. Il a une longueur de 7 et un décalage de 0. Cette description de voie NULL fera que les voies 18 - 24 seront remplies de zéros. Cette dernière description de voie est nécessaire pour remplir le tampon d'échantillons aux 24 bits spécifiés dans le paramètre de nombre de voies. Comme les 24 voies sont configurées, aucune autre description de voie n'est nécessaire.
25	125	Décalage	0	0000	

Contenu de l'échantillon

La Table 12-2 résume les valeurs contenues dans un seul échantillon sur la base des descriptions de voie du bloc de commande d'échantillon.

Table 12-2. Exemple de contenu d'échantillon pour SER

Numéro de voie	Contenu de la voie
1	%I0001
2 - 4	Zéros
5	Point 13 du module d'entrées
6	Point 14 du module d'entrées
7	Point 15 du module d'entrées
8	%Q0009
9	%Q0010
10 - 17	Points 1 - 8 du module d'entrées
18 - 24	Zéros

Exemple de bloc de données pour le bloc de commande

La Table 12-3 liste le format du bloc de données résultant de l'exemple de bloc de commande donné à la page 12-17. Noter qu'il commence au registre 201 comme décrit par les paramètres de décalage de segment (Mots 12 et 13) dans le bloc de commande.

Table 12-3. Exemple de bloc de données pour le bloc de commande de SER

Décalage	Registre	Description du paramètre	Valeur (déc)	Valeur (hex)
0	%R0201	Numéro de décalage d'échantillon courant.	59	003B
1	202	Numéro de décalage d'échantillon déclenché	0	0000
2 - 5	203 – 206	Heure du déclenchement	0 0 0 0	0000 0000 0000 0000
6 - 768	207 – 975	Tampon d'échantillons.	données d'échantillon	données d'échantillon

Le décalage d'échantillon courant est 59 signifiant que le 59ème échantillon est le dernier échantillon placé dans le tampon d'échantillons (pas 59 registres). Avec 3 octets par échantillon, le décalage courant est réellement de 59 * 3 = 177 octets ou l'octet haut du 89ème registre. Comme les conditions de déclenchement n'ont pas été respectées, l'échantillon déclenché et l'heure de déclenchement sont mis à 0 et la sortie n'est pas mise à "1". Le tampon d'échantillons contient 512 échantillons où 59 est l'échantillon le plus récent et 60 est l'échantillon le plus ancien.

Formats de l'horodatage du déclenchement du bloc fonctionnel

Exemple d'heure de déclenchement du 3 novembre 1998 à 8:34:05:16.

Format BCD :

```
struct time_of_day_clk_rec {
        caractère non signé    secondes ;
        caractère non signé    minutes ;
        caractère non signé    heures ;
        caractère non signé    jour du mois ;
        caractère non signé    mois ;
        caractère non signé    année ;
} ;
```

Registre	Paramètre	Valeur (déc)	Valeur (hex)
%R0203	Minutes/Secondes	13317	3405
%R204	Jour du mois/Heures	776	0308
%R205	Année/Mois	-26607	9811
%R206	Inutilisé	0	0

Format POSIX :

```
struct timespec {
        long    tv_sec ;        /* Nombre de secondes depuis le 1er
janvier 1970 */
        long    tv_nsec ;/* Nombre de nanosecondes dans les secondes
suivantes */
} ;
```

Registre	Paramètre	Valeur (déc)	Valeur (hex)
%R0203	Mot Faible Secondes	-7811	e17d
%R204	Mot Fort Secondes	13845	3615
%R205	Mot Faible Nano-secondes	26624	6800
%R206	Mot Fort Nano-secondes	2441	0989

END

La fonction END donne une fin temporaire de la logique. Le programme s'exécute du premier circuit au dernier circuit ou à la fonction END, quel que soit le premier rencontré.

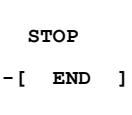
La fonction END termine l'exécution d'un programme de façon inconditionnelle. Il ne peut rien y avoir après la fonction END dans l'circuit. Aucune logique, au-delà de la fonction END, n'est exécutée et le contrôle est transféré au début du programme pour le cycle suivant.

La fonction END est utile pour le déverminage car elle évite d'exécuter la logique entièrement.

Le logiciel de programmation Logicmaster fournit un repère [END OF PROGRAM LOGIC] pour indiquer la fin du programme. Ce repère est utilisé s'il n'y a aucune fonction END programmée dans la logique.

Exemple

Dans l'exemple suivant, END est programmé pour terminer le cycle courant.



Remarque

Le fait de mettre une fonction END dans la logique SFC ou dans la logique appelée par les procédures SFC, produit un défaut "Fonction END exécutée par l'action SFC” dans les UC de la Version 7 ou supérieure. (Dans les UC précédant la Version 7, elle ne fonctionne pas correctement mais aucun défaut n'est généré). Pour des informations concernant ce défaut, se référer à la partie “Différence de configuration système” du Chapitre 3, Section 2.

MCRN/MCR

Un relais de contrôle maître (MCR) doit être utilisé avec une fonction Fin de relais de contrôle maître (ENDMCR). Les fonctions doivent avoir le même nom. Tous les circuits entre un MCR et sa fonction ENDMCR correspondante sont exécutés sans transmission de flux d'énergie aux bobines. La fonction ENDMCR associée au MCR permet de reprendre une exécution normale du programme. Contrairement à l'instruction JUMP, les MCR ne peuvent se dérouler que vers l'avant. Une instruction ENDMCR doit apparaître après son instruction MCR correspondante.

Les contrôles suivants sont imposés par un MCR :

- Les temporisateurs n'incrémentent/décrémentent pas. Les types TMR sont réinitialisés. Pour un bloc fonctionnel ONDTR, l'accumulateur maintient sa valeur.
- Les sorties normales sont à "0" ; les sorties inversées sont à "1".

Remarque

Lorsqu'un MCR est excité, la logique qu'il contrôle est scrutée et l'état des contacts est affiché, mais les sorties ne sont pas excitées. Si vous ne savez pas qu'un MCR contrôle la logique visualisée, ceci pourrait apparaître comme un défaut. Pour indiquer qu'une partie de la logique en échelle est sous contrôle MCR, le logiciel affiche un double rail d'énergie sur l'écran.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes de fonction MCR : une forme imbriquée et une forme non-imbriquée.

Compatibilité de l'UC

Type d'UC	Instruction de comparaison masquée
UC Série 35x et 36x (version 2 et supérieures)	N'utilise que la forme imbriquée (MCRN)
UC Série 90, Version 1	N'utilise que la forme non-imbriquée (MCR)

Fonctionnement de MCRN

Une fonction MCRN peut être placée n'importe où dans un programme tant qu'elle est correctement imbriquée par rapport aux autres MCRN et ne s'active pas dans la plage de tout MCR non-imbriqué ou JUMP non-imbriqué.

Si une paire MCRN/ENDMCRN est imbriquée avec une autre paire MCRN/ENDMCRN, elle doit être contenue entièrement dans l'autre paire. Jusqu'à huit niveaux d'imbrication sont permis. Pour un exemple, voir la page 12-24.

Remarque

N'utiliser qu'un seul MCRN pour chaque ENDMCRN avec les UC modèles 35x et 36x.

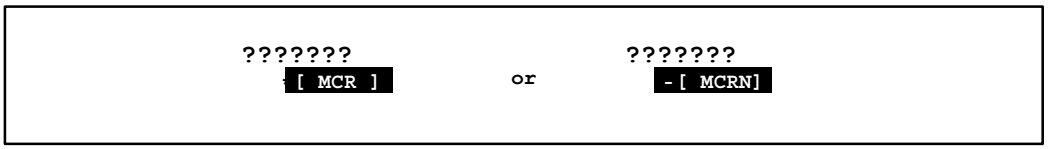
Il peut y avoir plusieurs fonctions MCRN correspondant à une seule ENDMCRN (sauf pour les UC modèles 35x et 36x comme noté ci-dessus). Ceci est analogue au JUMP imbriqué, pour lequel il est possible d' avoir des JUMP multiples au même LABEL. Pour les différences entre la fonction JUMP et la fonction MCR, se référer à la section “Différences entre les MCR et les JUMP”, à la page 12-23.

Fonctionnement du MCR

Il ne peut y avoir qu'une seule instruction MCR par instruction ENDMCR. Les plages de MCR et de ENDMCR ne peuvent pas se chevaucher ou contenir la plage de toute autre paire d'instructions MCR/ENDMCR ou de toute autre paire d'instructions JUMP/LABEL. Les MCR non-imbriqués ne peuvent se trouver dans le champ d'action de toute paire JUMP/LABEL.

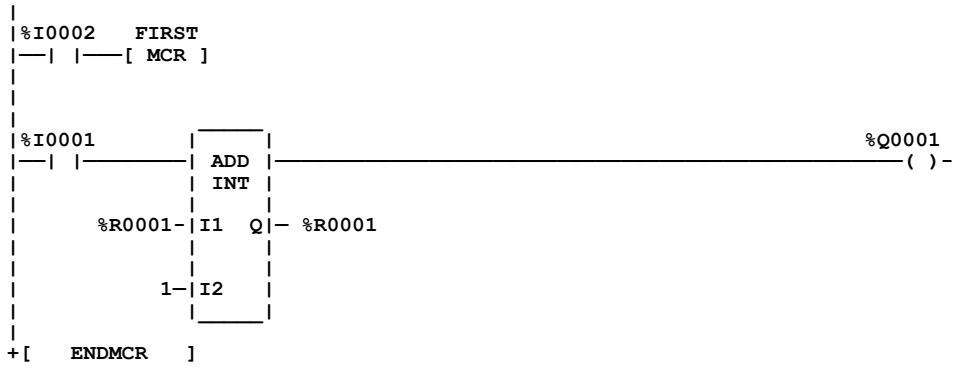
Paramètres

Les deux formes de fonction MCR ont les mêmes paramètres. Elles ont toutes les deux une entrée de validation Booléenne EN et un nom qui identifie le MCR. Ce nom est à nouveau utilisé avec une instruction ENDMCR. Ni la fonction MCR, ni la fonction MCRN ne possèdent de sorties ; il ne peut rien y avoir après un MCR dans un circuit.

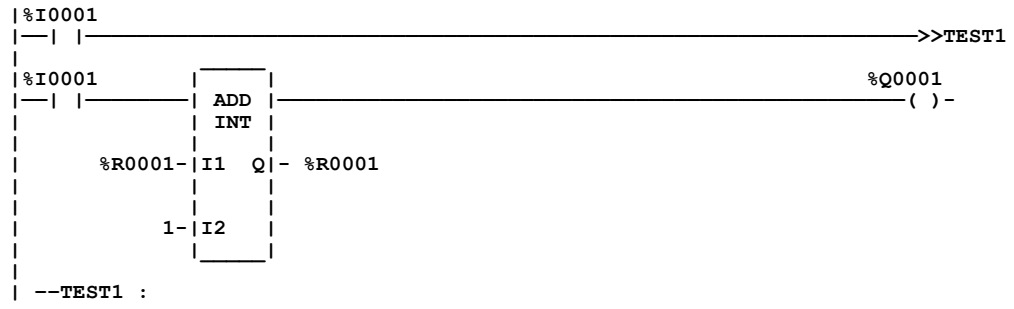


Différences entre les MCR et les JUMP

Avec une fonction MCR, les blocs fonctionnels se trouvant dans le champ d'action du MCR sont exécutés ***sans flux d'énergie*** et les bobines ***sont mises à "0"***. Dans l'exemple suivant, lorsque %I0002 est à "1", le MCR est validé. Lorsque le MCR est validé, même si %I0001 est à "1", le bloc fonctionnel ADD ne reçoit pas de flux d'énergie (c'est-à-dire que l'addition n'est pas effectuée) et %Q0001 est mis à "0".



Avec une fonction JUMP, tous les blocs fonctionnels, situés entre JUMP et LABEL, ***ne sont pas*** exécutés et les bobines ***ne sont pas pilotées***. Dans l'exemple suivant, lorsque %I0002 est à "1", le JUMP est pris. Comme la logique, entre le JUMP et le LABEL , est sautée, %Q0001 n'est pas piloté (c'est-à-dire que s'il était à "1", il reste à "1" ; s'il était à "0", il reste à "0").



Exemple

L'exemple suivant montre un MCRN appelé “Second” imbriqué dans un MCRN appelé “First.” Que %I0002 laisse passer ou non le flux d'énergie dans la fonction MCRN, l'exécution du programme continuera sans flux d'énergie vers les bobines jusqu'à ce que le ENDMCRN associé soit atteint. Si %I0001 et %I0003 sont à "1", %Q0001 est mis à "0" et %Q0003 reste à "1".

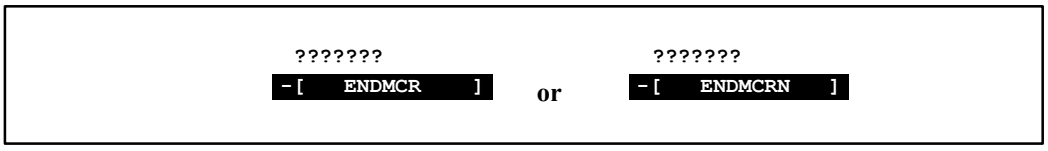
Pour aider au dépannage des programmes en échelle, un double rail identifie la logique qui est contrôlée par un MCR.

ENDMCRN/ENDMCR

Utiliser la fonction Fin de relais de contrôle maître (ENDMCR) pour poursuivre l'exécution normale du programme après une fonction MCR. Lorsque le MCR associé au ENDMCR est actif, le ENDMCR provoque la poursuite de l'exécution du programme avec un flux d'énergie normal. Lorsque le MCR associé au ENDMCR est inactif, le ENDMCR n'a pas d'effet.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes de fonction ENDMCR : une forme imbriquée et une forme non-imbriquée. La forme non-imbriquée de ENDMCR doit être utilisée avec la fonction MCR non-imbriqué. La forme imbriquée de ENDMCRN doit être utilisée avec la fonction MCR imbriqué.

La fonction ENDMCR possède une entrée inversée Booléenne EN. La validation de l'instruction doit être fournie par le rail d'énergie ; l'exécution ne peut pas être conditionnelle. La fonction ENDMCR a également un nom qui identifie le ENDMCR et l'associe au(x) MCR correspondant(s). La fonction ENDMCR n'a pas de sortie ; il ne peut rien y avoir avant ou après une instruction ENDMCR dans un circuit.



Exemple

Dans les exemples suivants, une instruction ENDMCR est programmée pour indiquer la fin du MCR.

Exemple d'un ENDMCR non-imbriqué

```
|
|  CLEAR
|
|-[   ENDMCR    ]
|
```

Exemple d'un ENDMCR imbriqué :

```
|
|  CLEAR
|
|-[   ENDMCRN   ]
|
```

JUMP

Utiliser l'instruction JUMP pour ignorer une partie de la logique programme. L'exécution du programme continuera au LABEL spécifié. Lorsque JUMP est actif, toutes les bobines se trouvant dans son champ d'action sont laissées dans leurs états précédents. Ceci inclut les bobines associées à des temporisateurs, compteurs, verrous et relais.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes de fonction JUMP : une forme imbriquée et une forme non-imbriquée. La forme non-imbriquée a été disponible depuis la Version 1 du logiciel et a la forme ————————>>LABEL01, où LABEL01 est le nom de l'instruction LABEL non-imbriquée correspondante.

Pour des JUMP non-imbriqués, il ne peut y avoir qu'une seule instruction JUMP par instruction LABEL. Le JUMP peut être un JUMP en avant ou en arrière.

Les plages des JUMP et des LABEL non-imbriqués ne peuvent pas chevaucher la plage de toute autre paire JUMP/LABEL ou toute autre paire MCR/ENDMCR d'instructions. Les JUMP non-imbriqués et leurs LABEL correspondants ne peuvent pas se trouver dans le champ d'action de toute autre paire JUMP/LABEL ou toute autre paire MCR/ENDMCR. De plus, une paire MCR/ENDMCR ou une autre paire JUMP/LABEL ne peut pas se trouver dans le champ d'action d'une paire JUMP/LABEL non-imbriquée.

Remarque

La forme non-imbriquée de l'instruction JUMP est la seule instruction JUMP qui puisse être utilisée dans un API Série 90-30, Version 1. La fonction JUMP imbriquée peut être utilisée (et son utilisation est recommandée) pour toutes les applications nouvelles.

Egalement, veuillez noter que les UC modèles 35x et 36x ne supportent que les sauts imbriqués. Les sauts non-imbriqués ne sont pas supportés par les UC modèles 35x et 36x.

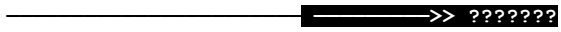
La forme imbriquée de l'instruction JUMP est ————N——>>LABEL01, où LABEL01 est le nom de l'instruction LABEL imbriquée correspondante. Elle est disponible dans les Versions 2 et supérieures du logiciel Logicmaster 90-30/20/Micro et dans le macro-programme de l'API.

Une instruction JUMP imbriquée peut être placée n'importeoù dans un programme, dans la mesure où elle ne s'active pas dans la plage de tout MCR non-imbriqué ou JUMP non-imbriqué.

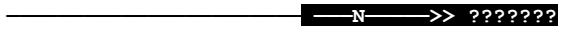
Il peut y avoir de multiples instructions JUMP imbriquées correspondant à un seul LABEL imbriqué. Les JUMP imbriqués peuvent être des JUMP vers l'avant ou vers l'arrière.

Les deux formes de l'instruction JUMP sont toujours placées dans les colonnes 9 et 10 de la ligne de circuit courante ; Il ne peut rien y avoir après la fonction JUMP dans l'circuit. Le flux d'énergie saute directement de l'instruction vers l'circuit ayant le label nommé.

JUMP non-imbriqué :



JUMP imbriqué :



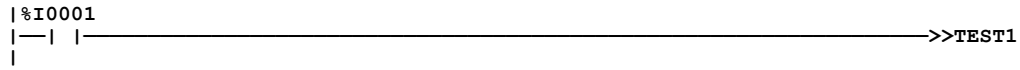
Précautions

Afin d'éviter de créer une boucle sans fin avec des instructions JUMP avant et arrière, un JUMP arrière être conditionnel.

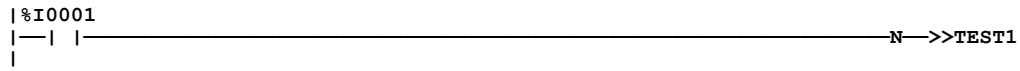
Exemples

Dans les exemples suivants, lorsque JUMP TEST1 est actif, le flux d'énergie est transféré à LABEL TEST1.

Exemple d'un JUMP non-imbriqué :



Exemple d'un JUMP imbriqué :



LABEL

L'instruction LABEL fonctionne comme étant la destination cible d'un JUMP. Utiliser l'instruction LABEL pour poursuivre l'exécution normale du programme après une instruction JUMP.

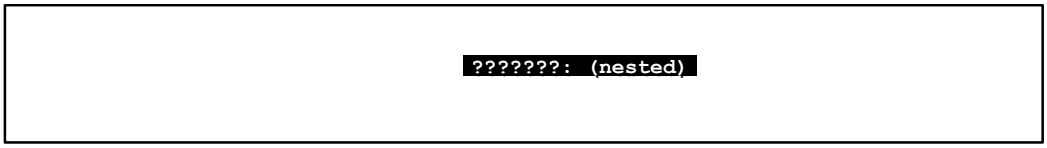
Il ne peut y avoir qu'un seul LABEL avec un nom de label particulier dans un programme. Des programmes sans paire JUMP/LABEL peuvent être créés et stockés dans l'API, mais ne peuvent pas être exécutés.

Le logiciel Logicmaster 90-30/20/Micro supporte deux formes de fonction LABEL : une forme imbriquée et une forme non-imbriquée. La forme non-imbriquée, LABEL01, doit être utilisée avec la fonction JUMP non-imbriquée, ————————>>LABEL01. La forme imbriquée, LABEL01 :(imbriquée), doit être utilisée avec la fonction JUMP imbriquée, ———N——>>LABEL01.

L'instruction LABEL n'a ni entrées, ni sorties ; il ne peut rien y avoir avant ou après une instruction LABEL dans un circuit.

LABEL non-imbriqué :

LABEL imbriqué :



Exemple

Dans les exemples suivants, le flux d'énergie de JUMP TEST1 se poursuit, en partant de LABEL TEST1.

Exemple de LABEL non-imbriqué :

```
|
| TEST1   :
|
```

Exemple d'un LABEL imbriqué :

```
|
| TEST1   :(imbriqué)
|
```

COMMENT

Utiliser la fonction COMMENT pour entrer un commentaire (explication de circuit) dans le programme. Un commentaire peut avoir jusqu'à 2 048 caractères de texte. Il est représenté dans la logique en échelle comme ceci :

Le texte peut être lu ou édité en déplaçant le curseur vers (* COMMENT *) après avoir validé le circuit et sélectionné le zoom (F10). Le texte du commentaire peut également être imprimé.

Un texte plus long peut être inclus dans les impressions en utilisant un fichier de texte d'annotation, comme décrit ci-dessous :

1. Créer le commentaire :
   A. Entrer le texte à l'endroit où le texte de l'autre fichier doit commencer.
   B. Déplacer le curseur au début d'une nouvelle ligne et entrer **\I** ou **\i**, le lecteur suivi de deux points, le sous-répertoire ou dossier et le nom du fichier, comme montré dans cet exemple :

      ```
      \I d:\text\commnt1
      ```

      La désignation du lecteur n'est pas nécessaire si le fichier est situé sur le même lecteur que le dossier du programme.
   C. Continuer l'édition du programme ou quitter vers MS-DOS.
2. Après avoir quitté la console de programmation, créer un fichier texte en utilisant tout progiciel compatible MS-DOS. Donner au fichier le nom de fichier entré dans le commentaire et le mettre sur le lecteur spécifié dans le commentaire.

SVCREQ

Utiliser la fonction requête de service (SVCREQ) pour demander l'un des services spéciaux suivants de l'API :

Table 12-4. Fonctions Requête de service

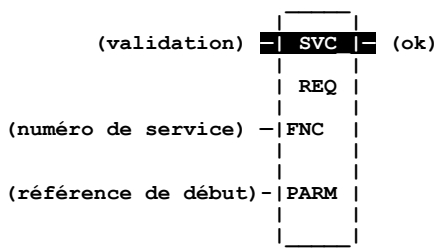
Fonction	Description
1	Changer/Lire le temporisateur de cycle constant.
2	Lire les valeurs de la fenêtre.
3	Changer le mode fenêtre de communication et la valeur du temporisateur de la console de programmation.
4	Changer le mode fenêtre de communication et la valeur du temporisateur du système.
6	Changer/Lire l'état de la tâche du checksum et le nombre de mots à totaliser.
7	Changer/Lire l'horloge interne.
8	Réinitialiser le temporisateur chien de garde.
9	Lire la durée du cycle à partir du début du cycle.
10	Lire le nom de dossier.
11	Lire l'identificateur de l'API.
12	Lire l'état d'exécution de l'API.
13	Couper l'API.
14	Effacer les Tables de défauts.
15	Lire le dernier défaut enregistré de la Table de défauts.
16	Lire l'horloge de temps écoulé.
18	Lire l'état de forçage des E/S.
23	Lire le checksum maître.
26/30	Interroger les E/S.
29	Lire le temps d'arrêt écoulé.
45	Sauter la scrutation des entrées et des sorties suivante. (Suspendre les E/S).
46	Etat de l'accès rapide au fond de panier.

Vue d'ensemble de SVC REQ

La fonction SVCREQ possède trois paramètres d'entrée et un paramètre de sortie. Lorsque SVCREQ reçoit le flux d'énergie, l'API doit effectuer la fonction FNC indiquée. Les paramètres de la fonction commencent à la référence donnée pour PARM. La fonction SVCREQ laisse passer le flux d'énergie sauf si un numéro de fonction incorrect, des paramètres incorrects ou des références en dehors de la plage sont spécifiés. D'autres causes de défaillances sont décrites dans les pages suivantes.

La référence données pour PARM peut représenter tout type de mémoire mot (%R, %AI ou %AQ). Cette référence est la première d'un groupe qui compose le "bloc paramètre" pour la fonction. Des adresses 16 bits successifs stockent les paramètres supplémentaires. Le nombre total de références nécessaires dépendra du type de fonction SVCREQ utilisée.

Les paramètres sont utilisés comme données d'entrée pour la fonction et comme adresse où récupérer les résultats de la fonction. Par conséquent, les données renvoyées par la fonction sont accessibles à l'adresse spécifiée par PARM.



Paramètres

Paramètre	Description
validation	Lorsque la fonction est validée, la requête de service est effectuée.
FNC	FNC contient la constante ou la référence pour le service demandé.
PARM	PARM contient la référence de début pour le bloc paramètre du service demandé.
ok	La sortie OK est excitée lorsque la fonction est effectuée sans erreur.

Types de mémoire valides

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

• Référence ou emplacement valide où l'énergie peut passer à travers la fonction.

Exemple

Dans l'exemple suivant, lorsque l'entrée validation %I0001 est à "1", la fonction SVCREQ numéro 7 est appelée avec le bloc paramètre situé à partir de %R0001. La bobine de sortie %Q0001 est mise à "1" si l'opération réussit.

SVCREQ #1 : Changer/Lire le temporisateur de cycle constant

A partir de l'UC 90-30 version 8, il est possible d'utiliser la fonction SVCREQ #1 pour :

- Désactiver le mode **CYCLE CONSTANT**.
- Valider le mode **CYCLE CONSTANT** et utiliser l'ancienne valeur du temporisateur.
- Valider le mode **CYCLE CONSTANT** et utiliser la nouvelle valeur du temporisateur.
- Régler une nouvelle valeur de temporisateur seulement.
- Lire l'état du mode **CYCLE CONSTANT** et la valeur du temporisateur.

Remarque

La requête de service 1 n'est supportée *que* par les UC 90-30, à partir de la Version 8.0.

Le bloc paramètre a une longueur de deux mots.

Pour désactiver le mode **CYCLE CONSTANT**, entrer la fonction SVCREQ #1 avec le bloc paramètre :

0	adresse
ignoré	adresse + 1

Pour valider le mode **CYCLE CONSTANT**, entrer la fonction SVCREQ #1 avec le bloc paramètre :

1	adresse
0 ou valeur du temporisateur	adresse + 1

Remarque

Toute nouvelle valeur du temporisateur doit être entrée dans le deuxième mot. Si la valeur du temporisateur ne doit pas être changée, entrer 0 dans le deuxième mot. Si la valeur du temporisateur n'existe pas encore, une valeur de 0 mettra la sortie OK de la fonction à "0".

Pour modifier la valeur du temporisateur **sans** changer la sélection pour l'état du mode de scrutation, entrer la fonction SVCREQ #1 avec le bloc paramètre :

2	adresse
nouvelle valeur de temporisateur	adresse + 1

Pour lire l'état et la valeur courants du temporisateur sans les changer, entrer la fonction SVCREQ #1 dans le bloc paramètre :

3	adresse
ignoré	adresse + 1

Remarque

Après l'utilisation de la fonction SVCREQ #1 avec le bloc paramètre de la page précédente, les UC Version 8 et supérieures renverront les valeurs 0 pour un mode de scrutation normal, 1 pour un mode constant. A ne pas confondre avec les valeurs *d'entrée* montrées ci-dessous.

L'exécution s'effectuera correctement sauf si :

1. Un nombre autre que 0, 1, 2 ou 3 est entré dans le champ opération:

0	Désactiver le mode **CYCLE CONSTANT**.
1	Valider le mode **CYCLE CONSTANT**.
2	Régler une nouvelle valeur de temporisateur seulement.
3	Lire le mode **CYCLE CONSTANT** et la valeur du temporisateur. (Voir Remarque ci-dessus).

2. Si la valeur est supérieure à 2 550 ms (2,55 secondes).
3. Le temps de cycle constant est validé sans valeur de temporisateur programmée ou avec une valeur de temporisation de 0.

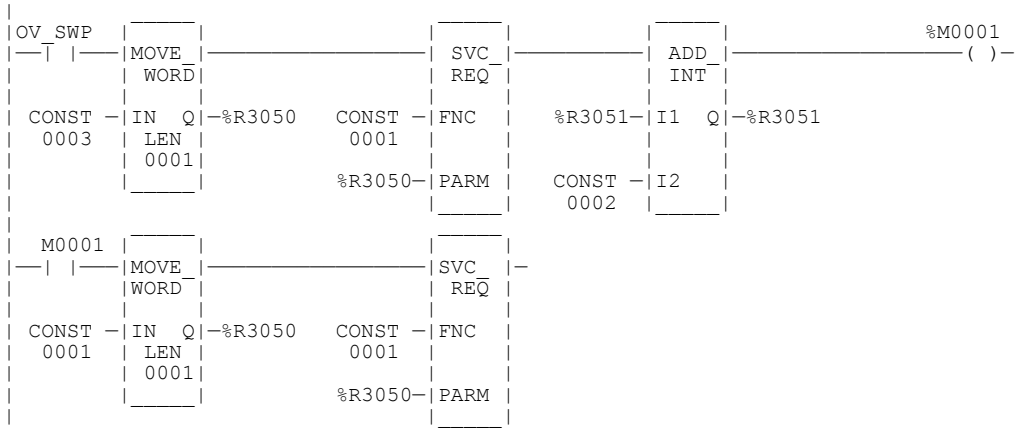
Après l'exécution de la fonction, la fonction renvoie l'état et la valeur du temporisateur dans les mêmes références de bloc paramètre :

0 = désactivé	
1 = activé	adresse
valeur courante du temporisateur	adresse + 1

Si l'adresse mot + 1 contient la valeur hexadécimale FFFF, c'est qu'aucune valeur de temporisateur n'a jamais été programmée.

Exemple

Cet exemple montre la logique dans un bloc programme. Lorsque le contact validation OV_SWP est à "1", le temps de cycle constant est lu, il est incrémenté de deux millisecondes avant d'être renvoyé à l'API. Le bloc paramètre est en mémoire locale à l'adresse %R3050. Comme les fonctions MOVE et ADD nécessitent trois positions de contact horizontales, une bobine interne %M0001 est utilisée pour mémoriser l'état d'exécution de la première ligne . A chaque cycle où OV_SWP n'est pas à "1", %M0001 est mis à "0".



SVCREQ #2 : Lire les valeurs de fenêtre

Utiliser la fonction SVCREQ #2 pour obtenir les valeurs de temps du mode fenêtre courant pour la fenêtre de communication de la console de programmation, la fenêtre de communication du système et la fenêtre de tâche en arrière-plan.

Remarque

La requête de service 2 n'est supportée que par les UC 90-30, à partir de la Version 8.0.

Il existe trois modes pour chaque fenêtre :

Nom du mode	Valeur	Description
Mode limité	0	Le temps d'exécution de la fenêtre est limité à sa valeur par défaut ou à une valeur définie en utilisant la fonction SVCREQ #3 pour la fenêtre de communication de la console de programmation ou la fonction SVCREQ #4 pour la fenêtre de communication du système. La fenêtre prendra fin lorsque plus aucune tâche ne subsiste
Mode constant	1	Chaque fenêtre fonctionnera en mode **EXECUTION COMPLETE** et l'API alternera parmi les trois fenêtres pendant une durée égale à la somme des valeurs de temps respectives de chaque fenêtre. Si une fenêtre est mise en mode **CONSTANT** , les deux fenêtres restantes sont automatiquement mises en mode **CONSTANT** . Si l'API fonctionne en mode **FENETRE CONSTANTE** et qu'un temps d'exécution particulier de la fenêtre n'est pas défini en utilisant la fonction SVCREQ associée, le temps par défaut pour cette fenêtre est utilisé dans le calcul de temps de fenêtre constant.
Mode exécution complète	2	Sans tenir compte du temps de fenêtre associé à une fenêtre particulière, qu'il soit par défaut ou défini en utilisant une requête de service, la fenêtre fonctionnera jusqu'à ce que toutes ses tâches soient accomplies.

Une fenêtre est désactivée lorsque la valeur du temps est de zéro.

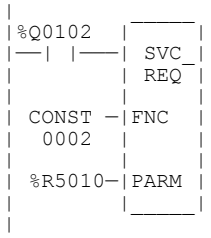
Le bloc paramètre a une longueur de trois mots :

	Octet haut	Octet bas	
Fenêtre de la console de programmation	Mode	Valeur en ms	adresse
Fenêtre de communication du système	Mode	Valeur en ms	adresse + 1
Fenêtre d'arrière-plan			adresse + 2

Tous les paramètres sont des paramètres de sortie. Il n'est pas nécessaire d'entrer des valeurs dans le bloc paramètre pour programmer cette fonction. Les valeurs de sortie des trois fenêtres sont données en millisecondes.

Exemple

Dans l'exemple suivant, lorsque la sortie de validation %Q0102 est à "1", le système d'exploitation de l'API met les valeurs de temps courant des trois fenêtres dans le bloc paramètre en commençant à l'emplacement %R5010. Des exemples supplémentaires montrant la fonction Lire les valeurs de fenêtre sont inclus dans les trois descriptions de fonction SYS REQ suivantes.



SVCREQ #3 : Changer le mode fenêtre de communication de la console de programmation et la valeur du temporisateur

Utiliser la fonction SVCREQ #3 pour changer le mode gestion de la fenêtre de communication vers console de programmation et la valeur du temporisateur. Le changement prendra effet au cycle suivant celui pendans lequel la fonction est appelée.

Remarque

La requête de service 3 n'est supportée que par les UC 90-30, à partir de la Version 8.0.

La fonction SVCREQ #3 laissera passer le flux d'énergie vers la droite sauf si un mode autre que 0 (Limité), 1 (Constant), ou 2 (Exécution Complète) est sélectionné.

Le bloc paramètre a une longueur de un mot.

Pour désactiver la fenêtre de la console de programmation, entrer la fonction SVCREQ #3 avec ce bloc paramètre :

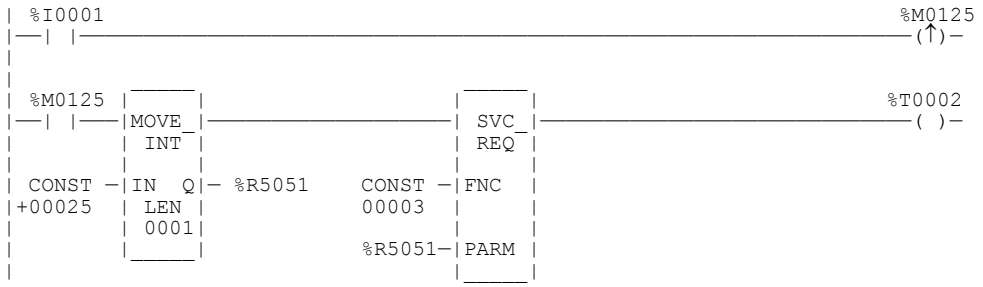
Octet haut	Octet bas	
0	0	adresse

Pour désactiver la fenêtre de la console de programmation, entrer la fonction SVCREQ #3 avec ce bloc paramètre :

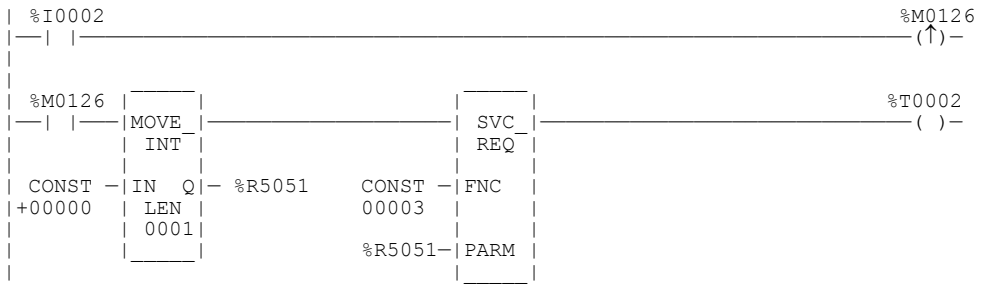
Octet haut	Octet bas	
Mode	Valeur de 1 à 255 ms	adresse

Exemple

Dans l'exemple suivant, lorsque %M0125 passe à "1", la fenêtre de communication de la console de programmation est validée et reçoit une valeur de 25 ms. Le bloc paramètre est dans l'emplacement mémoire %R5051.



Pour désactiver la fenêtre de communication vers la console de programmation, utiliser la requête de service 3 pour affecter une valeur de zéro (0). Dans cet exemple, lorsque %M0126 passe à "1", la fenêtre de communication de la console de programmation est validée et reçoit une valeur de 0 ms. Le bloc paramètre est dans l'emplacement mémoire %R5051.



SVCREQ #4 : Changer le mode fenêtre de communication du système et la valeur du temporisateur

Utiliser la fonction SVCREQ #4 pour changer le mode de fenêtre de communication du système et la valeur du temporisateur. Le changement se produira dans le cycle de l'UC suivant le cycle dans lequel la fonction est appelée.

Remarque

La requête de service 4 n'est supportée que par les UC 90-30, à partir de la Version 8.0.

La fonction SVCREQ #4 laissera passer le flux d'énergie vers la droite sauf si un mode autre que 0 (Limité), 1 (Constant), ou 2 (Exécution Complète) est sélectionné.

Le bloc paramètre a une longueur de un mot.

Pour désactiver la fenêtre de communication du système, entrer la fonction SVCREQ #4 avec le bloc paramètre :

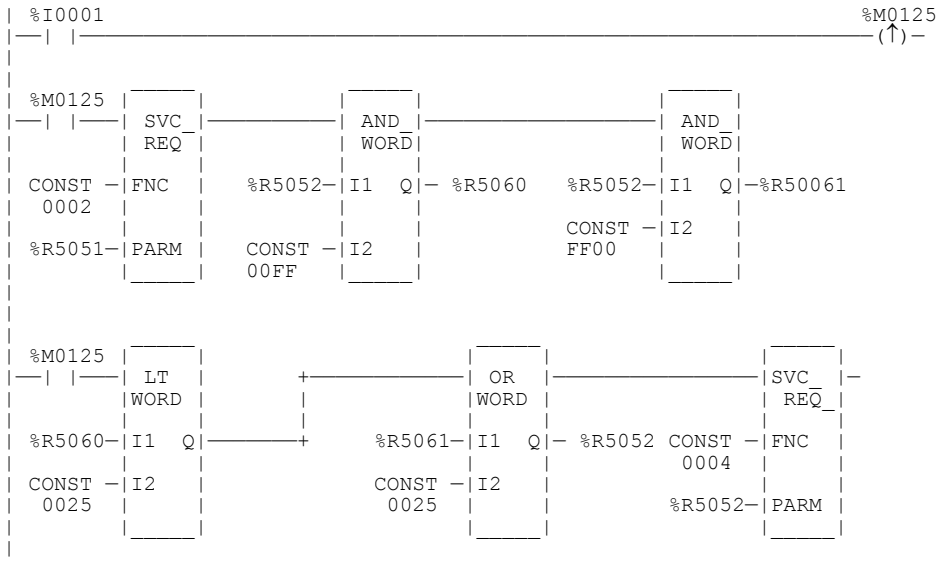
Octet haut	Octet bas	
0	0	adresse

Pour valider la fenêtre de communication du système, entrer la fonction SVCREQ #4 avec le bloc paramètre :

Octet haut	Octet bas	
Mode	Valeur de 1 à 255 ms	adresse

Exemple

Dans l'exemple suivant, lorsque la sortie de validation %M0125 passe à "1", le mode et la valeur du temporisateur de la fenêtre de communication du système sont lus. Si la valeur du temporisateur est supérieure ou égale à 25 ms, elle n'est pas changée. Si elle est inférieure à 25 ms, la valeur est mise à 25 ms. Dans les deux cas, la fenêtre est validée après exécution de cette logique. Le bloc paramètre, pour les trois fenêtres est à l'emplacement %R5051. Comme le mode et le temporisateur pour la fenêtre de communication du système sont la deuxième valeur du bloc renvoyé par la fonction Lire les valeurs de la fenêtre (fonction #2), l'emplacement du temps de la fenêtre existante est dans l'octet bas de %R5052.



SVCREQ #6 : Changer/Lire le nombre de mots à vérifier par totalisation

Utiliser la fonction SVCREQ # 6 pour :

- Lire le nombre de mots courant
- Régler un nouveau compte de mots.

L'exécution sera correcte sauf si un nombre autre que 0 ou 1 est entré comme opération demandée (voir ci-dessous).

Pour les fonctions de tâche de checksum, le bloc paramètre a une longueur de 2 mots.

Pour lire le nombre de mots courant :

Entrer la fonction SVCREQ # 6 avec ce bloc paramètre :

0	adresse
ignoré	adresse + 1

Après exécution de la fonction, le checksum courant est renvoyé dans le second mot du bloc paramètre. Aucune plage n'est spécifiée pour la fonction Lire ; la valeur retournée est le nombre de mots réellement vérifiés par totalisation.

0	adresse
nombre de mots courant	adresse + 1

Pour régler un nouveau nombre de mots :

Entrer la fonction SVCREQ # 6 avec ce bloc paramètre :

1	adresse
nouveau nombre de mots	adresse + 1

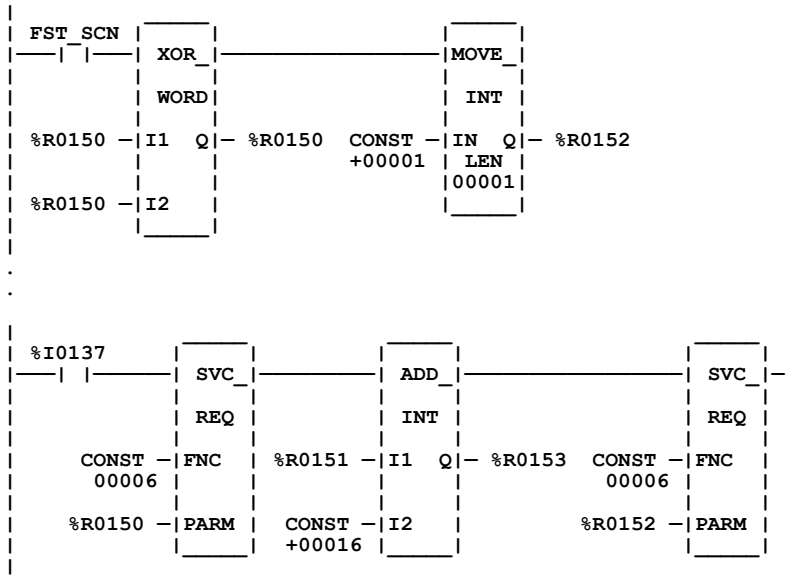
Lorsque 1 est entré, l'API ajuste le nombre de mots à vérifier par totalisation à la valeur donnée dans le deuxième mot du bloc paramètre. Pour les UC 331 ou 311, le nombre peut être 0 ou 32 ; dans l'UC 211, la valeur peut être 0 ou 4.

Remarque

Cette requête de service n'est pas disponible sur les API Micro.

Exemple

Dans l'exemple suivant, lorsque le contact de validation FST_SCN est à "1", les blocs paramètres de la fonction de tâche de checksum sont construits. Lorsque l'entrée %I0137 passe à "1", le nombre de mots vérifiés par totalisation est lu dans le système d'exploitation de l'API. Ce nombre est augmenté de 16 et les résultats de la fonction ADD_INT sont placés dans le paramètre “conserver le nouveau nombre pour l'initialisation”. La deuxième requête de service demande à l'API d'initialiser le nouveau nombre de mots.



Les blocs paramètres de l'exemple sont situés à l'adresse %R0150. Ils ont les contenus suivants :

0 = lire le nombre courant	%R0150
conserve le nombre courant	%R0151

1 = initialise le nombre courant	%R0152
conserve le nouveau nombre pour l'initialisation	%R0153

SVCREQ #7 : Changer/Lire l'horloge interne

Utiliser la fonction SVCREQ # 7 pour lire et régler l'horloge interne de l'API.

Remarque

Cette fonction n'est disponible qu'avec les UC 90-30 modèle 331 ou supérieure et sur les UC des API Série 90 Micro 28 points (c'est-à-dire, IC693UDR005, IC693UAA007 et IC693UDR010) et les UC des API Série 90 Micro 23 points (IC693UAL006).

L'exécution s'effectuera correctement sauf si :

1. Un nombre autre que 0 ou 1 est entré comme opération demandée (voir ci-dessous).
2. Un format de donnée invalide est spécifié.
3. les données fournies ne sont pas dans le format prévu.

Pour les fonctions date/heure, la longueur du bloc paramètre dépend du format des données. Le format BCD nécessite 6 mots ; le format ASCII compressé nécessite 12 mots.

0 = lire l'heure et la date 1 = régler l'heure et la date	adresse
1 = format BCD 3 = format ASCII compressé	adresse + 1
données	adresse + 2 à la fin

Dans le mot 1, spécifier si la fonction doit lire ou changer les valeurs.

```
0    =    lire
1    =    changer
```

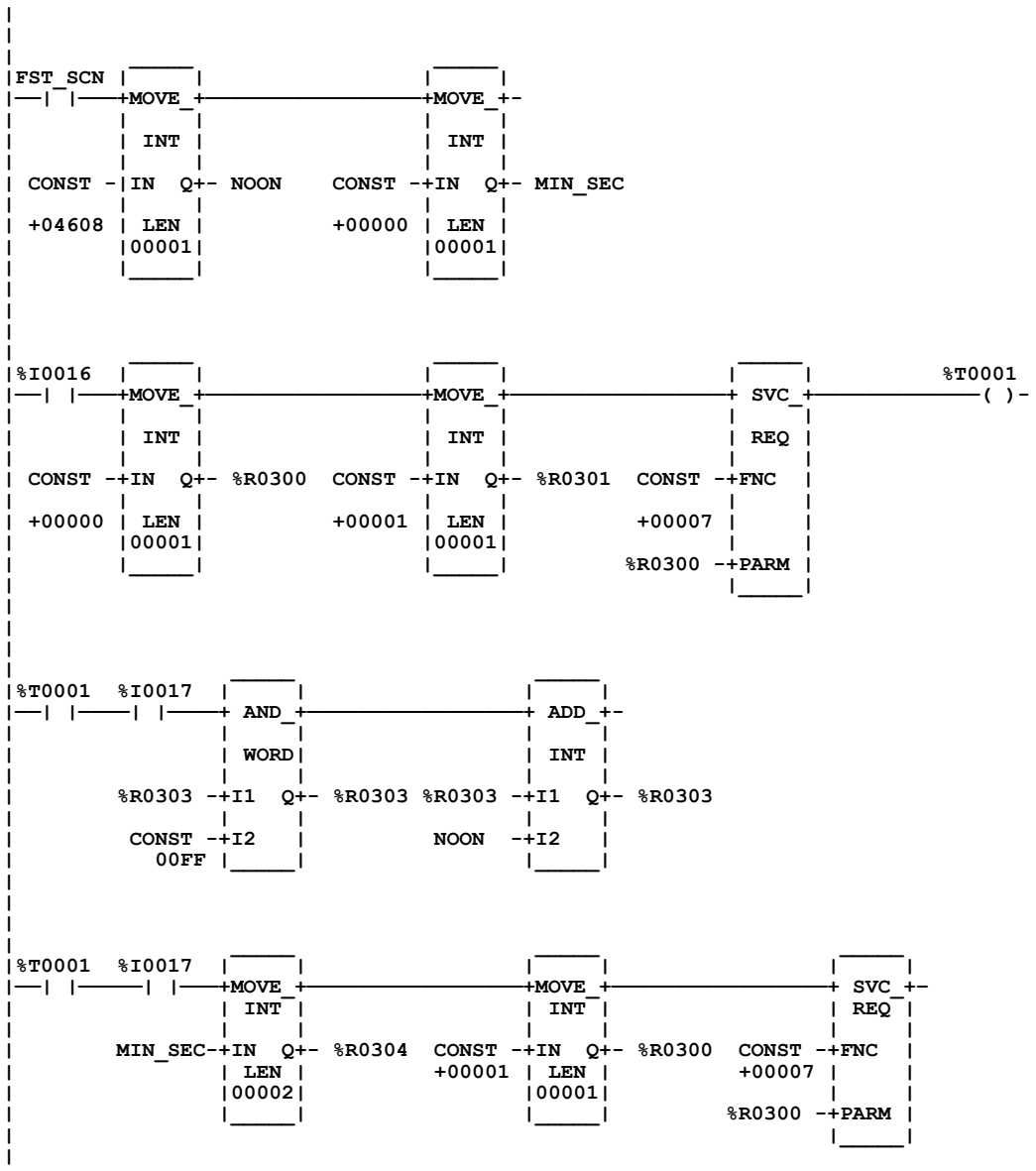
Dans le mot 2, spécifier un format de données :

```
1    =    BCD
3    =     ASCII compressé avec espaces et deux points imbriqués
```

Du mot 3 à la fin du bloc paramètre figurent les données renvoyées par une fonction lire ou les nouvelles valeurs à utiliser par une fonction Changer. 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 paramètre sont ignorés.

Exemple

Dans l'exemple suivant, un bloc paramètre est construit pour demander d'abord l'heure et la date courante, puis régler l'horloge à 12 heures en utilisant le format BCD. Le bloc paramètre est situé à l'adresse %R0300. NOON a été réglé ailleurs dans le programme pour contenir les valeurs 12, 0 et 0. Le format BCD nécessite six adresses mémoire contigües pour le bloc paramètre.



Contenu du bloc paramètre

Le contenu du bloc paramètre, pour les différents formats de données, figure sur les pages suivantes. Pour les deux formats de données :

- Les heures sont stockées dans un format 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 changer/lire la date et l'heure en utilisant le format BCD :

Dans le format BCD, chaque élément de date et d'heure occupe un seul octet. Ce format nécessite six mots. Le dernier octet du sixième mot n'est pas utilisé. Lors du réglage de la date et de l'heure, cet octet est ignoré ; lors de la lecture de la date et de l'heure, la fonction retourne un caractère nul (00).

Octet haut	Octet bas	
1 = changer ou	0 = lire	adresse
1		adresse + 1
mois	année	adresse + 2
heures	jour du mois	adresse + 3
secondes	minutes	adresse + 4
(nul)	jour de la semaine	adresse + 5

Exemple de bloc paramètre de sortie :
Lire la date et l'heure en format BCD
(Dim., 3 Juillet, 1988, à 14:45:30).

0	
1	
07	88
14	03
30	45
00	01

Pour changer/lire la date et l'heure en utilisant le format ASCII compressé avec deux points (:) imbriqués

Dans le format ASCII compressé, chaque chiffre des éléments de date et d'heure est un octet formaté ASCII. De plus, les espaces et les deux points (:) sont imbriqués dans les données pour permettre d'être transférés immédiatement vers un dispositif d'impression ou d'affichage. Ce format nécessite 12 mots.

Octet haut	Octet bas		Exemple de bloc paramètre de sortie : Lire la date et l'heure en format ASCII compressé (Lun, Oct. 2, 1989 à 23:13:00)	
1 = changer ou	0 = lire	adresse	0	
3		adresse + 1	3	
année	année	adresse + 2	39	38
mois	(espace)	adresse + 3	31	20
(espace)	mois	adresse + 4	20	30
jour du mois	jour du mois	adresse + 5	32	30
heures	(espace)	adresse + 6	32	20
:	heures	adresse + 7	3A	33
minutes	minutes	adresse + 8	33	31
secondes	:	adresse + 9	30	3A
(espace)	secondes	adresse + 10	20	30
jour de la semaine	jour de la semaine	adresse + 11	32	30

SVCREQ #8 : Réinitialisation du temporisateur chien de garde

Utiliser une fonction SVCREQ #8 pour réinitialiser le temporisateur chien de garde

Remarque

La requête de service 8 n'est supportée que par les UC 90-30, à partir de la Version 8.0.

Lorsque le temporisateur chien de garde a expiré, l'API se coupe sans avertissement. Cette fonction permet au temporisateur de continuer pendant un ceratin temps (par exemple, en attendant une réponse d'une ligne de communication).

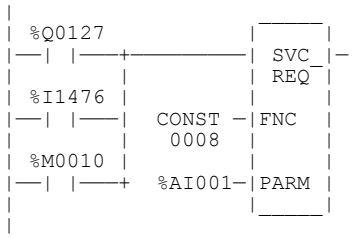
Précautions

S'assurer que le redémarrage du temporisateur chien de garde ne gêne pas le procédé contrôlé.

Cette fonction ne possède pas de bloc paramètre associé ; cependant, le logiciel de programmation nécessite une entrée pour PARM, mais elle ne sera pas utilisée.

Exemple

Dans l'exemple suivant, lorsque la sortie de validation %Q0127 ou l'entrée %I1476 ou la bobine interne %M0010 est à "1", le temporisateur chien de garde est réinitialisé.



SVCREQ #9 : Lire le temps écoulé depuis le début du cycle

Utiliser la fonction SVCREQ #9 pour lire le temps en millisecondes depuis le départ du cycle. Les données sont en format mot 16 bits.

Remarque

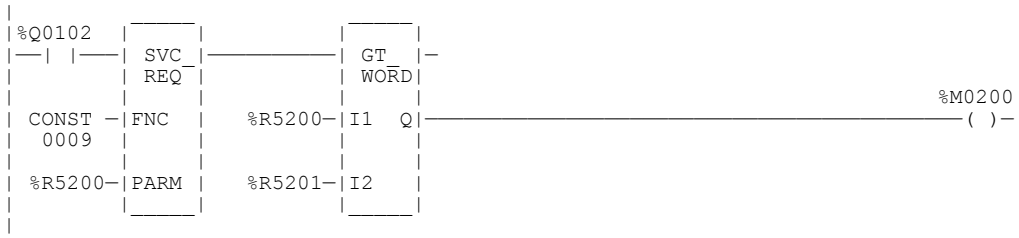
La requête de service 9 n'est supportée que par les UC 90-30, à partir de la Version 8.0.

Le bloc paramètre est un bloc paramètre de sortie seulement ; il a une longueur d'un mot.

temps depuis le départ du cycle	adresse

Exemple

Dans l'exemple suivant, le temps écoulé depuis le début du cycle est toujours lu dans l'emplacement %R5200. S'il est supérieur à la valeur de %R5201, la bobine interne %M0200 est mise à "1".



SVCREQ #10 : Lire le nom d'un dossier

Utiliser la fonction SVCREQ #10 pour lire le nom du dossier réellement en cours d'exécution.

Remarque

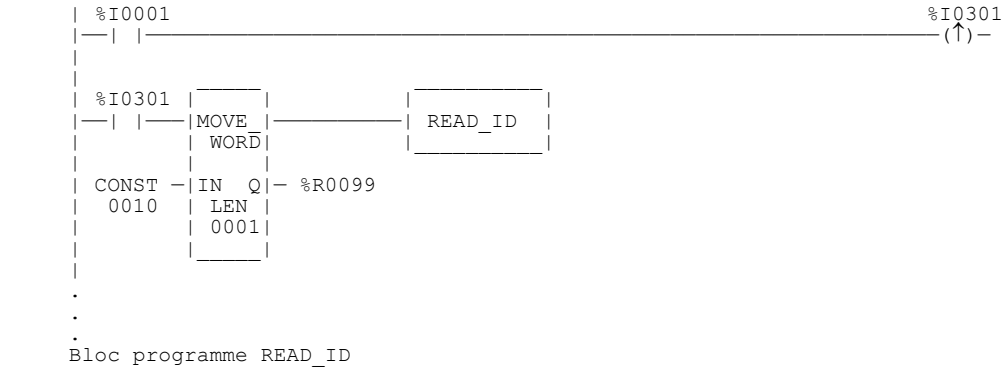
La requête de service 10 n'est supportée *que* par les UC 90-30, à partir de la Version 8.0.

Le bloc paramètre de sortie a une longueur de quatre mots. Il retourne huit caractères ASCII ; le dernier est un caractère nul (00h). Si le nom du programme a moins de sept caractères, des caractères nuls sont ajoutés à la fin.

Octet bas	Octet haut	
caractère 1	caractère 2	adresse
caractère 3	caractère 4	adresse + 1
caractère 5	caractère 6	adresse + 2
caractère 7	00	adresse + 3

Exemple

Dans l'exemple suivant, lorsque l'entrée de validation %I0301 passe à "1", l'emplacement registre %R0099 est chargé avec la valeur 10, qui est le code de fonction de la fonction Lire le nom d'un dossier. Le bloc programme READ_ID est alors appelé pour retrouver le nom du dossier. Le bloc paramètre est situé à l'adresse %R0100. READ_ID est également utilisé dans l'exemple suivant.



Bloc programme READ_ID

SVCREQ #11 : Lire l'identificateur de l'API

Utiliser la fonction SVCREQ #11 pour lire le nom de l'API Série 90 exécutant le programme.

Remarque

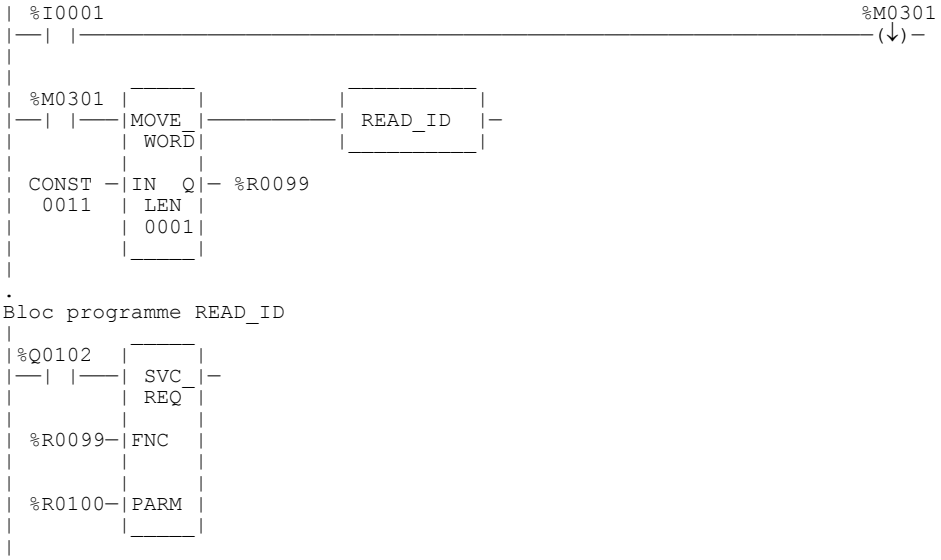
La requête de service 11 n'est supportée *que* par les UC 90-30, à partir de la Version 8.0.

Le bloc paramètre de sortie a une longueur de quatre mots. Il retourne huit caractères ASCII ; le dernier est un caractère nul (00h). Si l'identificateur de l'API a moins de sept caractères, des caractères nuls sont ajoutés à la fin.

Octet bas	Octet haut	
caractère 1	caractère 2	adresse
caractère 3	caractère 4	adresse + 1
caractère 5	caractère 6	adresse + 2
caractère 7	00	adresse + 3

Exemple

Dans l'exemple suivant, lorsque l'entrée de validation %I0001 passe à "0", l'emplacement registre %R0099 est chargé avec la valeur 11, qui est le code de fonction de la fonction Lire l'identificateur de l'API. Le bloc programme READ_ID est alors appelé pour retrouver l'identificateur. Le bloc paramètre est situé à l'adresse %R0100. Sauf pour le contact de validation et le numéro de la fonction, il s'agit du même code que dans l'exemple précédent.



SVCREQ #12 : Lire l'état d'exécution de l'API

Utiliser la fonction SVCREQ #12 pour lire l'état EXECUTION courant de l'UC de l'API.

Remarque

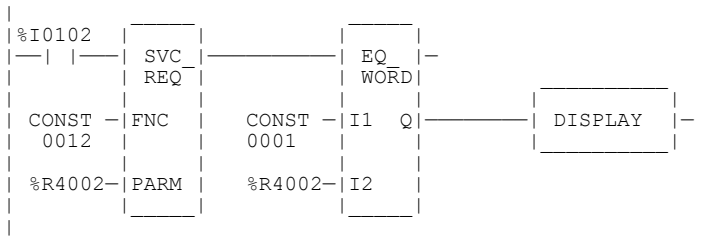
La requête de service 12 n'est supportée *que* par les UC 90-30, à partir de la Version 8.0.

Le bloc paramètre est un bloc paramètre de sortie seulement ; il a une longueur d'un mot.

1 = exécution/désactivée	adresse
2 = exécution/validée	

Exemple

Dans l'exemple suivant, l'état d'exécution de l'API est toujours lu dans l'emplacement %R4002. Si l'état est exécution/désactivée, la fonction CALL appelle le bloc programme DISPLAY.



SVCREQ #13 : Arrêt de l'API

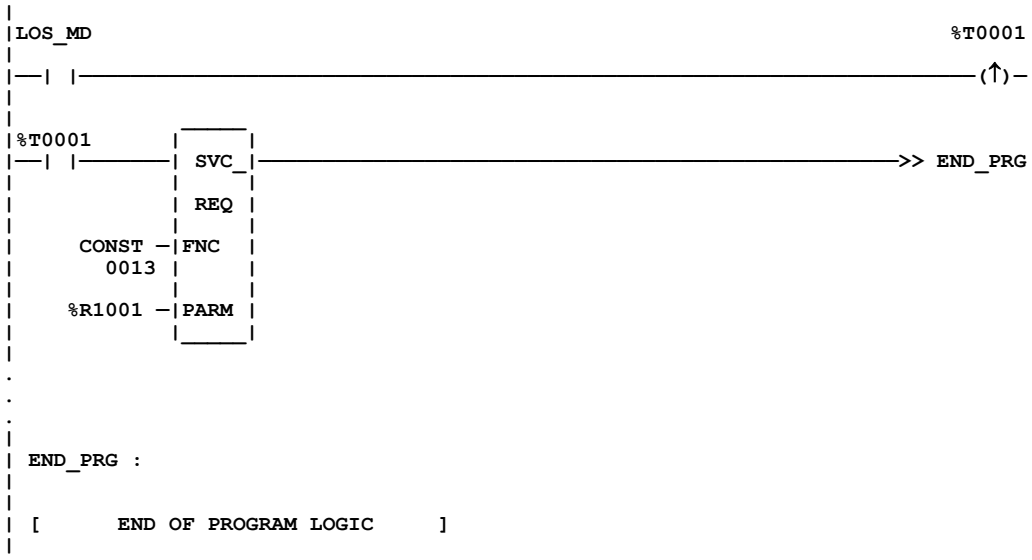
Utiliser la fonction SVCREQ #13 pour arrêter l'API ***à la fin du cycle suivant***. Toutes les sorties prendront leurs états par défaut au début du cycle suivant de l'API. Un défaut informationnel est placé dans la table des défauts de l'API, indiquant que le bloc fonctionnel “ARRET DE L'API” a été exécuté. La scrutation des E/S continuera selon la configuration.

Cette fonction ne possède pas de bloc paramètre associé ;

Exemple

Dans l'exemple suivant, lorsqu'un défaut de “Perte de module d'E/S” se produit, la fonction SVCREQ #13 s'exécute. Comme aucun bloc paramètre n'est nécessaire, l'entrée PARM n'est pas utilisée bien que le logiciel de programmation impose de saisir une adresse.

Cet exemple utilise un JUMP à la fin du programme pour forcer un arrêt si la fonction Arrêt de l'API s'exécute correctement. Ces JUMP et LABEL sont nécessaires car la transition vers le mode **ARRET** ne se produit pas avant la fin du cycle en cours.



Remarque

Pour s'assurer que le contact %S0002 LST_SCN fonctionnera correctement, l'API exécutera un cycle supplémentaire après le cycle dans lequel la fonction SVCREQ #13 était exécutée.

SVCREQ #14 : Effacer les tables de défauts

Utiliser la fonction SVCREQ #14 pour effacer la table des défauts Automate ou la table des défauts d' E/S. La sortie de SVCREQ est mise à "1" sauf si un chiffre autre que 0 ou 1 est saisi (voir ci-dessous).

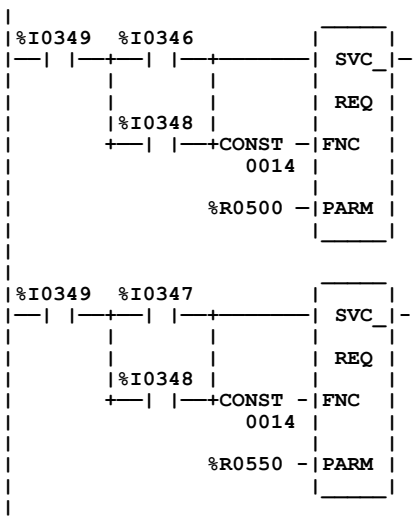
Pour cette fonction, le bloc paramètre a une longueur d'un mot. C'est un bloc paramètre d'entrée seulement.

0 = effacer la table des défauts de l'API. 1 = effacer la table des défauts des E/S.	adresse

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0346 est à "1" et que l'entrée %I0349 est à "1", la table des défauts automate est effacée. Lorsque l'entrée %I0347 est à "1" et que l'entrée %I0349 est à "1", la table des défauts d' E/S est effacée. Lorsque l'entrée %I0348 est à "1" et que l'entrée %I0349 est à "1", les deux tables sont effacées.

Le bloc paramètre de la table des défauts automate est situé à %R0500 ; pour la table des défauts d' E/S, le bloc paramètre est situé à %R0550. Les deux blocs paramètres sont initialisés ailleurs dans le programme.



SVCREQ #15 : Lire le dernier défaut enregistré dans la table des défauts

Utiliser la fonction SVCREQ #15 pour lire le dernier défaut enregistré dans la table des défauts automate et dans la table des défauts d' E/S. La sortie SVCREQ est mise à "1"sauf si une opération autre que 0 ou 1 est demandée (voir ci-dessous) ou si la table des défauts est vide. (Pour plus d'informations sur les entrées des tables des défauts, se référer au Chapitre 3, "Explications et correction des défauts").

Pour cette fonction, le bloc paramètre a une longueur de 22 mots. Le bloc paramètre d'entrée a ce format :

0 = Lire la table des défauts de l'API. 1 = Lire la table des défauts des E/S.	adresse

Le format du bloc paramètre de sortie varie selon que la fonction lit les données de la table des défauts automate ou de la table des défauts d' E/S.

Format de sortie de la table des défauts de l'API

Octet bas	Octet haut		
0			
court/long		adresse + 1	
réserve		adresse + 2	
adresse des défauts de l'API		adresse + 3	
		adresse + 4	
groupe de défauts et action		adresse + 5	
code d'erreur		adresse + 6	
		adresse + 7	
		adresse + 8	
		adresse + 9	
		adresse + 10	
		adresse + 11	
données spécifiques aux défauts		adresse + 12	
		adresse + 13	
		adresse + 14	
		adresse + 15	
		adresse + 16	
		adresse + 17	
		adresse + 18	
horodatage		adresse + 19	
		adresse + 20	
		adresse + 21	

Format de sortie de la table des défauts des E/S

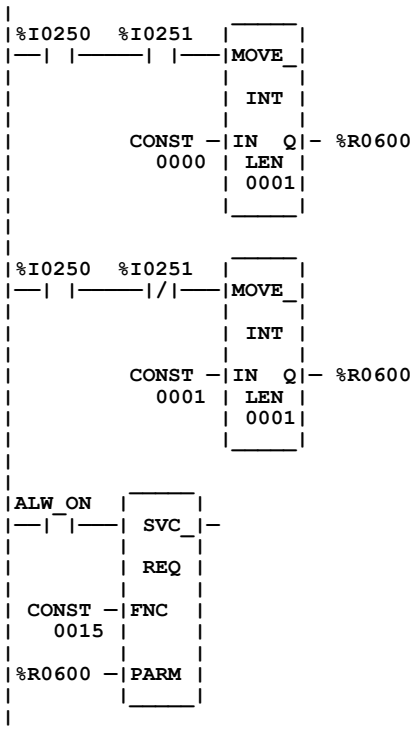
Octet bas	Octet haut
1	
court/long	
adresse de référence	
adresse des défauts d'E/S	
groupe de défauts et action	
catégorie de défaut	type de défaut
description du défaut	
données spécifiques aux défauts	
horodatage	

Dans le premier octet de l'adresse mot + 1, l'indicateur Long/Court définit la quantité de données spécifiques aux défauts présents. Elle peut être :

```
Table des défauts  00 = 8 octets (court)
de l'API
                   01 = 24 octets (long)
Table des défauts  02 = 5 octets (court)
des E/S
                   03 = 21 octets (long)
```

Exemple 1

Dans l'exemple suivant, lorsque l'entrée %I0251 est à "1" et que l'entrée %I0250 est à "1", la dernière entrée dans la table des défauts automateest lue dans le bloc paramètre. Lorsque l'entrée %I0251 est à "0" et que l'entrée %I0250 est à "1", la dernière entrée dans la table des défauts d' E/S est lue dans le bloc paramètre. Le bloc paramètre est situé à l'emplacement %R0600.



Exemple 2

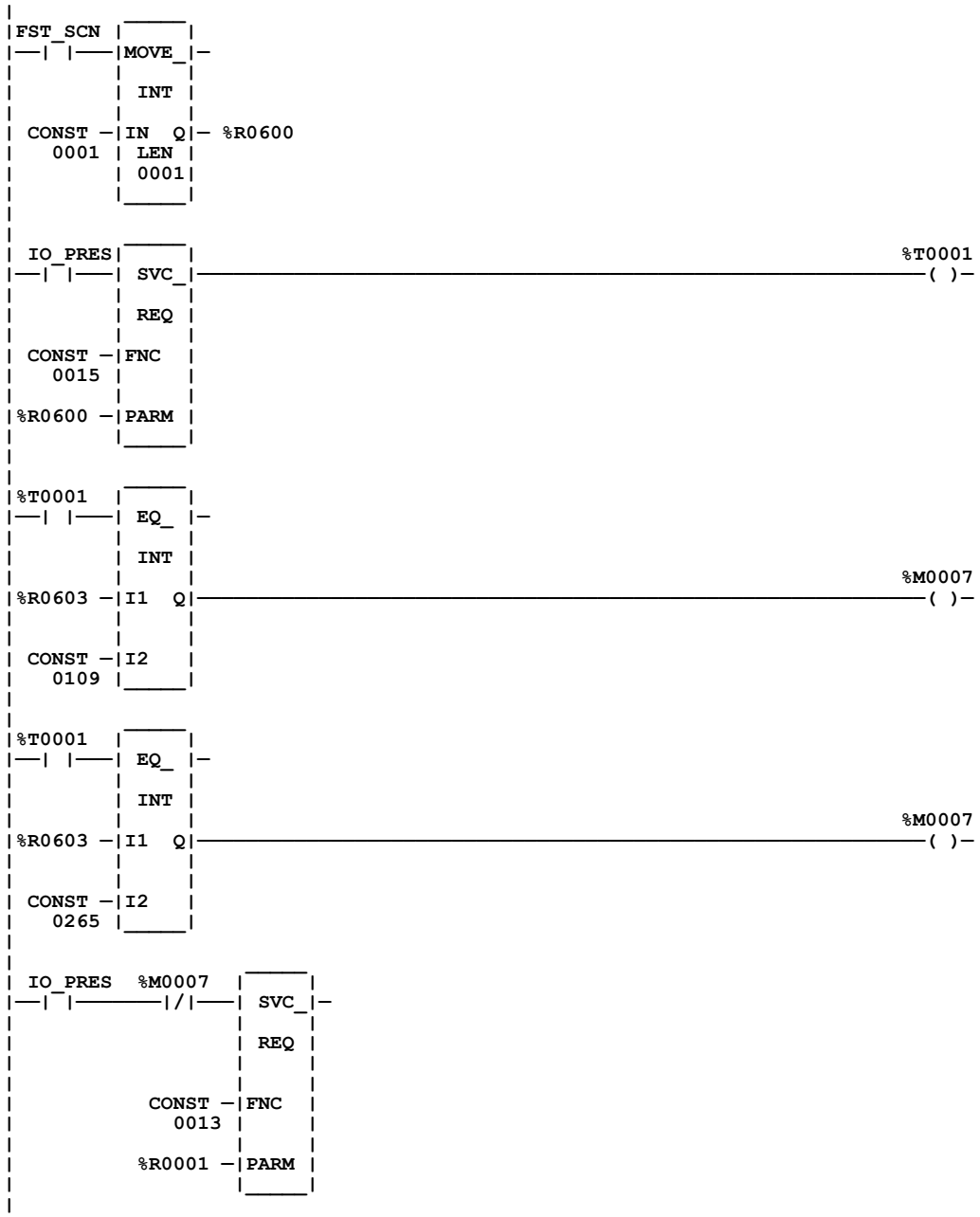
Dans l'exemple suivant, l'API est coupé lorsqu'un défaut se produit sur un module d'E/S sauf lorsque le défaut se produit sur des modules du châssis 0, emplacement 9 et du châssis 1, emplacement 9. Si un défaut se produit sur ces deux modules, le système continue à fonctionner. Le paramètre pour le "Type de table" est installé lors du premier cycle. Le contact IO_PRES, lorsqu'il est à "1", indique que la table des défauts des E/S contient une entrée. L'UC de l'API initialise le contact normalement ouvert dans le cycle après que la logique du défaut ait mis un défaut dans la table. Si les défauts sont mis dans la table en deux cycles consécutifs, le contact normalement ouvert est mis à "1" pendant deux cycles consécutifs.

L'exemple utilise un bloc paramètre situé à %R0600. Après l'exécution de la fonction SVCREQ, le quatrième, le cinquième et le sixième mot du bloc paramètre contiennent l'adresse du module d'E/S défectueux :

1		%R0600
court/long		%R0601
adresse de référence		%R0602
numéro de châssis	numéro d'emplacement	%R0603
N° de bus d'E/S	adresse du bus	%R0604
adresse du point		%R0605

Donnée de défaut

Dans le programme, les blocs EQ_INT comparent l'adresse châssis/emplacement de la table avec les constantes hexadécimales. La bobine interne %M0007 est mise à "1" lorsque le châssis/emplacement, où le défaut s'est produit, correspondent aux critères spécifiés ci-dessus. Si %M0007 est à "1", son contact normalement fermé est à "0", évitant l'arrêt. Réciproquement, si %M0007 est à "0" du fait que le défaut s'est produit sur un module différent, le contact normalement fermé est à "1" et l'arrêt se produit.


FST_SCN
MOVE_
INT
CONST
0001
IN
Q
%R0600
LEN
0001
IO_PRES
SVC_
%T0001
REQ
CONST
0015
FNC
%R0600
PARM
%T0001
EQ_
INT
%R0603
I1
Q
%M0007
CONST
0109
I2
%T0001
EQ_
INT
%R0603
I1
Q
%M0007
CONST
0265
I2
IO_PRES
%M0007
SVC_
REQ
CONST
0013
FNC
%R0001
PARM

SVCREQ #16 : Lire l'horloge de temps écoulé

Utiliser la fonction SVCREQ #16 pour lire la valeur de l'horloge de temps écoulé du système. Cette horloge suit le temps écoulé en secondes depuis la mise sous tension de l'API. Le temporisateur se bouclera environ une fois tous les 100 ans.

Cette fonction ne possède qu'un bloc paramètre de sortie. Le bloc paramètre a une longueur de 3 mots.

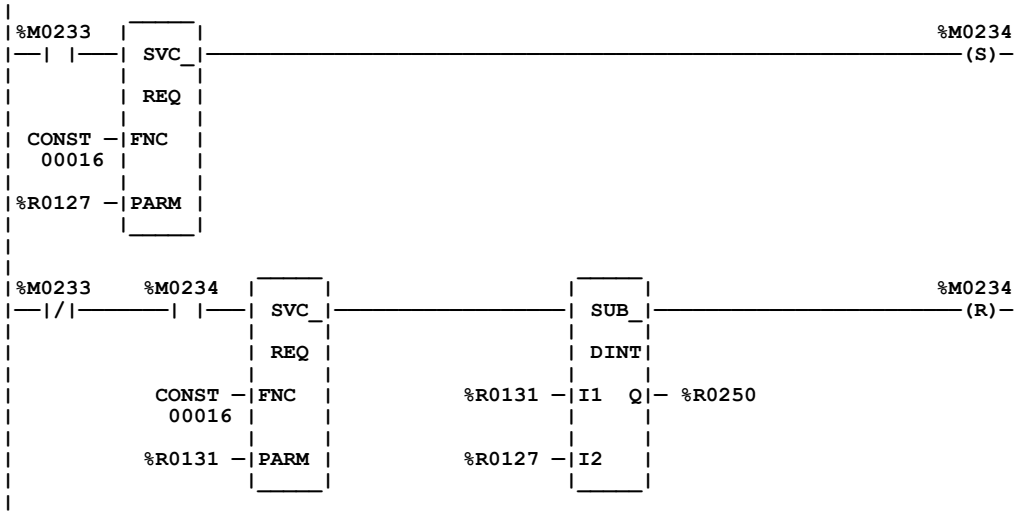
secondes depuis la mise sous tension (poids le plus faible)	adresse
secondes depuis la mise sous tension (poids le plus fort)	adresse + 1
impulsions de 100 microsecondes	adresse + 2

Les deux premiers mots représentent le temps écoulé en secondes. Le dernier mot est le nombre d'impulsions de 100 millisecondes dans la seconde courante.

Exemple

Dans l'exemple suivant, lorsque la bobine interne %M0233 est à "1", la valeur de l'horloge de temps écoulé est lue et la bobine %M0234 est mise à "1". Lorsqu'elle à "0", la valeur est lue à nouveau. La différence entre les valeurs est alors calculée et le résultat est stocké en mémoire registre à l'emplacement %R0250.

Le bloc paramètre pour la première lecture est à %R0127 ; pour la deuxième lecture, à %R0131. Le calcul ignore le nombre d'impulsions d'une centaine de millisecondes et le fait que le type DINT soit réellement une valeur signée. Le calcul est correct jusqu'à ce que le temps, depuis la mise sous tension, atteigne environ 50 ans.



SVCREQ #18 : Lire l'état de forçage des E/S

Utiliser la fonction SVCREQ #18 pour lire l'état courant des forçages dans l'UC.

Remarque

Cette fonction n'est disponible *que* pour les UC 331 ou supérieures.

Pour cette fonction, le bloc paramètre a une longueur d'un mot. C'est un bloc paramètre de sortie seulement.

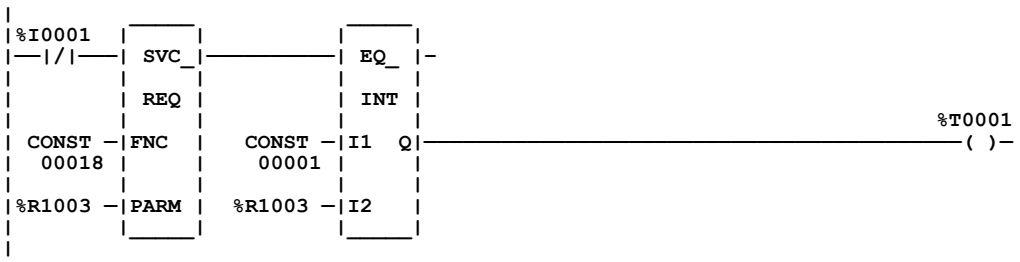
0 = Absence de forçage 1 = Présence de forçage	adresse

Remarque

SVCREQ #18 ne concerne que les forçages des références %I et %Q.

Exemple

Dans l'exemple suivant, l'état des forçages d'E/S est toujours lu dans l'emplacement %R1003. Si des forçages sont présents, la sortie %T0001 est mise à "1".



SVCREQ #23 : Lire le checksum maître

Utiliser la fonction SVCREQ #23 pour lire les checksum maîtres pour le programme et la configuration utilisateur. La sortie SVCREQ est toujours mise à "1" si la fonction est validée et le bloc d'informations de sortie (voir ci-dessous) commence à l'adresse donnée dans le paramètre 3 (PARM) de la fonction SVCREQ.

Lorsqu'un **MODE EXECUTION STOCKAGE** est actif, les checksums du programme peuvent ne être valides tant que le stockage n'est pas terminé. Par conséquent, deux indicateurs sont fournis au début du bloc paramètre de sortie pour indiquer quand les checksums du programme et de la configuration sont valides.

Pour cette fonction, le bloc paramètre de sortie a une longueur de 12 mots avec ce format :

Checksum programme maître valide (0 = invalide, 1 = valide)	adresse
Checksum configuration maître valide (0 = invalide, 1 = valide)	adresse + 1
Nombre de blocs programme (incluant _MAIN)	adresse + 2
Taille du programme utilisateur en octets (type de donnée DWORD)	adresse + 3
Checksum additive programme	adresse + 5
Checksum CRC programme (type de donnée DWORD)	adresse + 6
taille des données de configuration en octets	adresse + 8
Checksum additive configuration	adresse + 9
Checksum CRC configuration (type de données DWORD)	adresse + 10

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0251 est à "1", les informations du checksum maître sont placées dans le bloc paramètre et la bobine d sortie (%Q0001) est mise à "1". Le bloc paramètre est situé à l'emplacement %R0050.

SVCREQ #26/30 : Interroger les E/S

Utiliser la fonction SVCREQ #26 (ou #30—elles sont identiques) pour interroger les modules réellement présents et les comparer avec la configuration du châssis/emplacement, en générant des alarmes d'addition, de perte et de non-correspondance, comme après un téléchargement de configuration. Cette fonction génèrera des défauts dans les tables automate et d'E/S, en fonction du défaut.

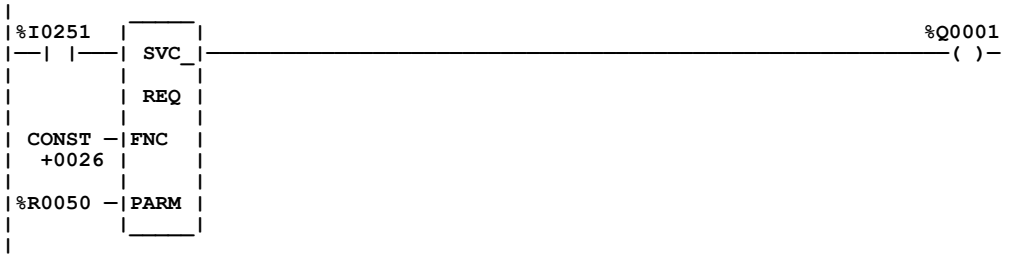
Cette fonction n'a pas de bloc paramètre et sort toujours le flux d'énergie.

Remarque

Le temps mis par cette SVCREQ pour s'exécuter dépend du nombre de défauts existants. Par conséquent, le temps d'exécution de cette SVCREQ sera supérieur dans les cas où plusieurs modules sont défectueux.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0251 est à "1", les modules présents sont interrogés et comparés avec la configuration châssis/emplacement. La sortie %Q0001 est mise à "1" après exécution du SVCREQ .



Remarque

Cette requête de service n'est pas disponible sur les API Micro.

SVCREQ #29 : Lire le temps d'arrêt écoulé

Utiliser la fonction SVCREQ #29 pour lire la durée écoulée entre la dernière coupure et la dernière mise sous tension. La sortie SVCREQ est toujours mise à "1" et le bloc de sortie d'informations (voir ci-dessous) commence à l'adresse donnée dans le paramètre 3 (PARM) de la fonction SVCREQ.

Remarque

Cette fonction n'est disponible que pour les UC 331 ou supérieures.

Cette fonction ne possède qu'un bloc paramètre de sortie. Le bloc paramètre a une longueur de 3 mots.

Secondes de coupure écoulées (poids faible)	adresse
Secondes de coupure écoulées (poids fort)	adresse + 1
impulsions de 100 microsecondes	adresse + 2

Les deux premiers mots représentent le temps de coupure écoulé en secondes. Le dernier mot est le temps de coupure écoulé restant en impulsions de 100 microsecondes. Lorsque l'API ne peut pas calculer correctement le temps écoulé de coupure, le temps sera mis à "0". Ceci se produira lorsque l'API est mis en route avec CLR M/T appuyé sur le HHP. Ceci se produira également si le temporisateur chien de garde expire avant la coupure.

Exemple

Dans l'exemple suivant, lorsque l'entrée %I0251 est à "1", le temps de coupure écoulé est placé dans le bloc paramètre et la bobine de sortie (%Q0001) est mise à "1". Le bloc paramètre est situé à l'emplacement %R0050.

SVCREQ #45 : Sauter la scrutation des entrées/sorties suivante

(Suspendre les E/S) Utiliser la fonction SVCREQ #45 pour sauter les scrutations des entrées/sorties suivantes. Tous les changements dans les tables de références de sortie, pendant le cycle dans lequel SVCREQ #45 a été exécutée, ne se reflèteront pas sur les sorties physiques des modules correspondants. Tous les changements dans les données des entrées physiques des modules ne se reflèteront pas dans les références d'entrée correspondantes pendant le cycle suivant celui dans lequel SVCREQ #45 a été exécutée.

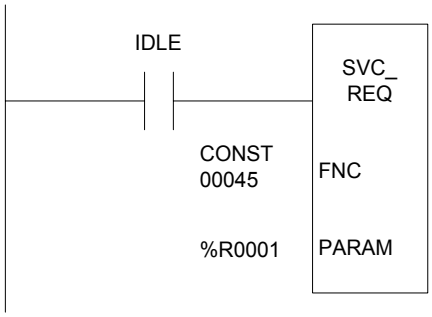
Cette fonction ne possède pas de bloc paramètre associé ;

Remarque

Le bloc fonctionnel DOIO n'est pas affecté par l'utilisation de SVCREQ #45. Il rafraîchira les E/S lorsqu'il sera utilisé dans le même programme logique que SVCREQ #45.

Exemple

Dans l'exemple suivant, le contact "libre" laisse passer le flux d'énergie, la scrutation des entrées/sorties suivante est sautée.



SVCREQ #46 : Accès rapide à l'état du fond de panier

Utiliser la fonction SVCREQ #46 pour effectuer l'une des fonctions d'accès rapide au fond de panier suivantes :

1. Lire un mot de données d'état supplémentaires à partir d'un ou plusieurs modules intelligents spécifiés.
2. Ecrire un mot de données d'état supplémentaires à partir d'un ou plusieurs modules intelligents spécifiés.
3. Lire/Ecrire : Lire un mot de données d'état supplémentaires à partir d'un ou plusieurs modules intelligents spécifiés et écrire la valeur des données entre 0 et 15 dans le même module en une seule opération.

Remarque

Cette requête de service ne peut être utilisée qu'avec les modules qui la supportent. En fait, le seul module conçu pour supporter cette fonction est le DSM (Module Servo Digital) version 312, qui n'est pas disponible au moment de la publication de ce manuel. Il est cependant prévu pour sortir bientôt.

Un bloc fonctionnel COMM_REQ ou DOIO ne doit pas être effectué avec le(s) module(s) spécifié(s) dans le même cycle que celui pendant lequel l'une des fonctions d'écriture de données est effectuée, car il pourrait provoquer la perte des données écrites.

Deux fonctions qui écrivent à un module (Ecrire ou Lire/Ecrire) ne doivent pas être effectuées pendant le même cycle car elles pourraient provoquer la perte des premières données écrites.

Cette requête de service a une longueur variable comme décrit ci-dessous. Le premier mot du bloc paramètre détermine quelle fonction sera utilisée et il a le format suivant :

1 = Lire données supplémentaires 2 = Ecrire données supplémentaires 3 = Lire/Ecrire données supplémentaires	adresse

Lire les données d'état supplémentaires (Fonction #1)

Cette fonction permet de lire un mot de données d'état supplémentaires pour chacun des modules spécifiés par une liste du bloc paramètre. Le bloc paramètre nécessite (N + 4) adresses mot, où N est le nombre de modules dans lesquels les données seront écrites.

Utiliser la table de la page suivante pour interpréter les valeurs de sortie.

Table 12-5. Valeurs de sortie pour la fonction Lire les données supplémentaires

Emplacement	Champ	Signification
Adresse	Fonction	1 = lire données d'état supplémentaires
Adresse + 1	Code d'erreur :	Un code d'erreur est placé ici si la fonction est défaillante à cause d'un module absent, inapproprié ou défectueux. Pour les détails, voir “Codes d'erreur” à la page 12-69.
Adresse + 2	Erreur de châssis et d'emplacement	Le numéro du châssis et de l'emplacement dans lequel l'erreur s'est produite
Adresse + 3	Premier châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du 1er module dans lequel les données seront lues
Adresse + 4	Lire les données dans le premier module	Les données lues dans le premier module seront placées ici
Adresse + 5	Deuxième châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du 2ème module dans lequel les données seront lues
Adresse + 6	Lire les données dans le deuxième module	Les données lues dans le deuxième module seront placées ici
Adresse + (I * 2) + 1	Nième	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du nième module dans lequel les données seront lues
Adresse + (I * 2) + 2	Lire les données dans le nième module	Les données lues dans le nième module seront placées ici
Adresse + (N * 2) + 1	Dernier châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du dernier module dans lequel les données seront lues
Adresse + (N * 2) + 2	Lire les données dans le dernier module	Les données lues dans le dernier module seront placées ici
Adresse + (N * 2) + 3	Indicateur de fin de liste	Un zéro dans ce mot indique la fin de la liste des modules

Ecrire donnée (Fonction #2)

Cette fonction permet d'écrire une valeur de donnée entre 0 et 15 à partir du bloc paramètre dans un ou plusieurs modules spécifiés par une liste du bloc paramètre. Le bloc paramètre nécessite (N + 4) adresses mot, où N est le nombre de modules dans lesquels les données seront écrites.

Table 12-6. Valeurs de sortie pour la fonction Ecrire donnée

Emplacement	Champ	Signification
Adresse	Fonction	2 = Ecrire données
Adresse + 1	Code d'erreur :	Un code d'erreur est placé ici si la fonction est défaillante à cause d'un module absent, inapproprié ou défectueux. Aucun code d'erreur n'est mis si la fonction s'exécute mais l'un des modules ne reçoit pas la donnée écrite correctement. Pour les détails, voir "Codes d'erreur" à la page 12-69.
Adresse + 2	Erreur de châssis et d'emplacement	Le numéro du châssis et de l'emplacement dans lequel l'erreur s'est produite
Adresse + 3	Premier châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du 1er module vers lequel les données seront envoyées
Adresse + 4	Ecrire les données pour le premier module	Cette valeur de donnée sera écrite dans le premier module
Adresse + 5	Deuxième châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du 2ème module vers lequel les données seront envoyées
Adresse + 6	Ecrire les données pour le deuxième module	Cette valeur de donnée sera écrite dans le deuxième module
Adresse + (I * 2) + 1	Nième châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du nième module vers lequel les données seront envoyées
Adresse + (I * 2) + 2	Ecrire les données pour le nième module	Cette valeur de donnée sera écrite dans le nième module
Adresse + (N * 2) + 1	Dernier châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du dernier module vers lequel les données seront envoyées
Adresse + (N * 2) + 2	Ecrire les données pour le dernier module	Cette valeur de donnée sera écrite dans le dernier module
Adresse + (N * 2) + 3	Indicateur de fin de liste	Un zéro dans ce mot indique la fin de la liste des modules

Lire/Ecrire donnée (Fonction #3)

Cette fonction permet de lire un mot de données d'état supplémentaire dans un module spécifié dans le bloc paramètre, puis écrit une valeur de donnée entre 0 et 15 du bloc paramètre dans ce module. Ce procédé de lecture/écriture se répète pour chaque module d'une liste du bloc paramètre. Le bloc paramètre nécessite (N * 3) + 3 adresses mot, où N est le nombre de modules avec lesquels les données seront échangées.

Table 12-7. Valeurs de sortie pour la fonction Lire/Ecrire des données supplémentaires

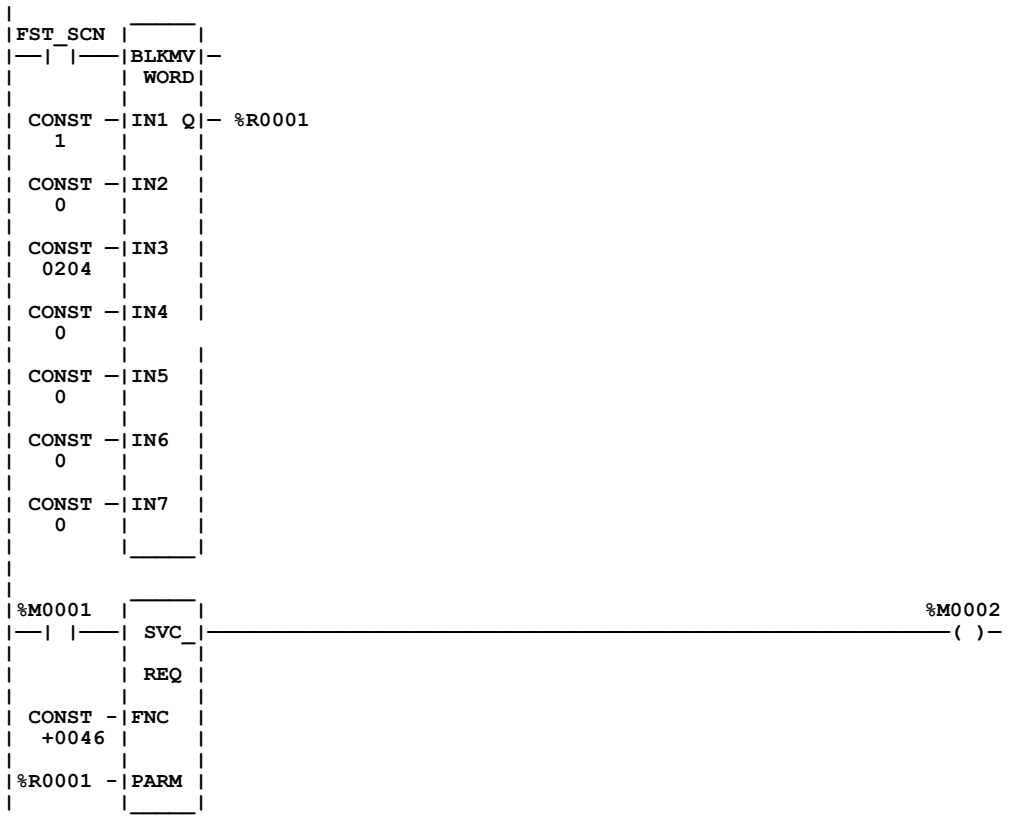
Emplacement	Champ	Signification
Adresse	Fonction	3 = lire/écrire données
Adresse + 1	Code d'erreur :	Un code d'erreur est placé ici si la fonction est défaillante à cause d'un module absent, inapproprié ou défectueux. Aucun code d'erreur n'est mis si la fonction s'exécute mais l'un des modules ne reçoit pas la donnée écrite correctement. Pour les détails, voir “Codes d'erreur” à la page 12-69.
Adresse + 2	Erreur de châssis et d'emplacement	Le numéro du châssis et de l'emplacement dans lequel l'erreur s'est produite
Adresse + 3	Premier châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du 1er module avec lequel les données seront échangées
Adresse + 4	Lire les données dans le premier module	Les données lues dans le premier module seront placées ici
Adresse + 5	Ecrire les données pour le premier module	Cette valeur de donnée sera écrite dans le premier module
Adresse + 6	Deuxième châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du 2ème module avec lequel les données seront échangées
Adresse + 7	Lire les données dans le deuxième module	Les données lues dans le deuxième module seront placées ici
Adresse + 8	Ecrire les données pour le deuxième module	Cette valeur de donnée sera écrite dans le deuxième module
Adresse + ((I-1) * 3) + 3	Nième châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du nième module avec lequel les données seront échangées
Adresse + ((I-1) * 3) + 4	Lire les données dans le nième module	Les données lues dans le nième module seront placées ici
Adresse + ((I-1) * 3) + 5	Ecrire les données pour le nième module	Cette valeur de donnée sera écrite dans le nième module
Adresse + ((N-1) * 3) + 3	Dernier châssis et emplacement	Numéro de châssis et d'emplacement (dans la forme CCEE en hexadécimal, où CC est le numéro du châssis et EE est le numéro de l'emplacement) du dernier module avec lequel les données seront échangées
Adresse + ((N-1) * 3) + 4	Lire les données dans le dernier module	Les données lues dans le dernier module seront placées ici
Adresse + ((N-1) * 3) + 5	Ecrire les données pour le dernier module	Cette valeur de donnée sera écrite dans le dernier module
Adresse + (N * 3) + 3	Indicateur de fin de liste	Un zéro dans ce mot indique la fin de la liste des modules

Codes d'erreur

Valeur	Description
1	Correct — la fonction s'est exécutée normalement.
-1	Le module n'est pas présent dans l'emplacement spécifié.
-2	Module inapproprié — le module de l'emplacement spécifié n'est pas un module intelligent ou ne supporte pas cette fonctionnalité.
-3	Module défectueux — le module de l'emplacement spécifié ne communique pas correctement avec l'UC.
-4	Erreur de parité de lecture de données — une erreur de parité s'est produite pendant une opération de lecture dans un châssis d'extension ou déporté.
-5	Fonction invalide spécifiée dans le bloc de commande.

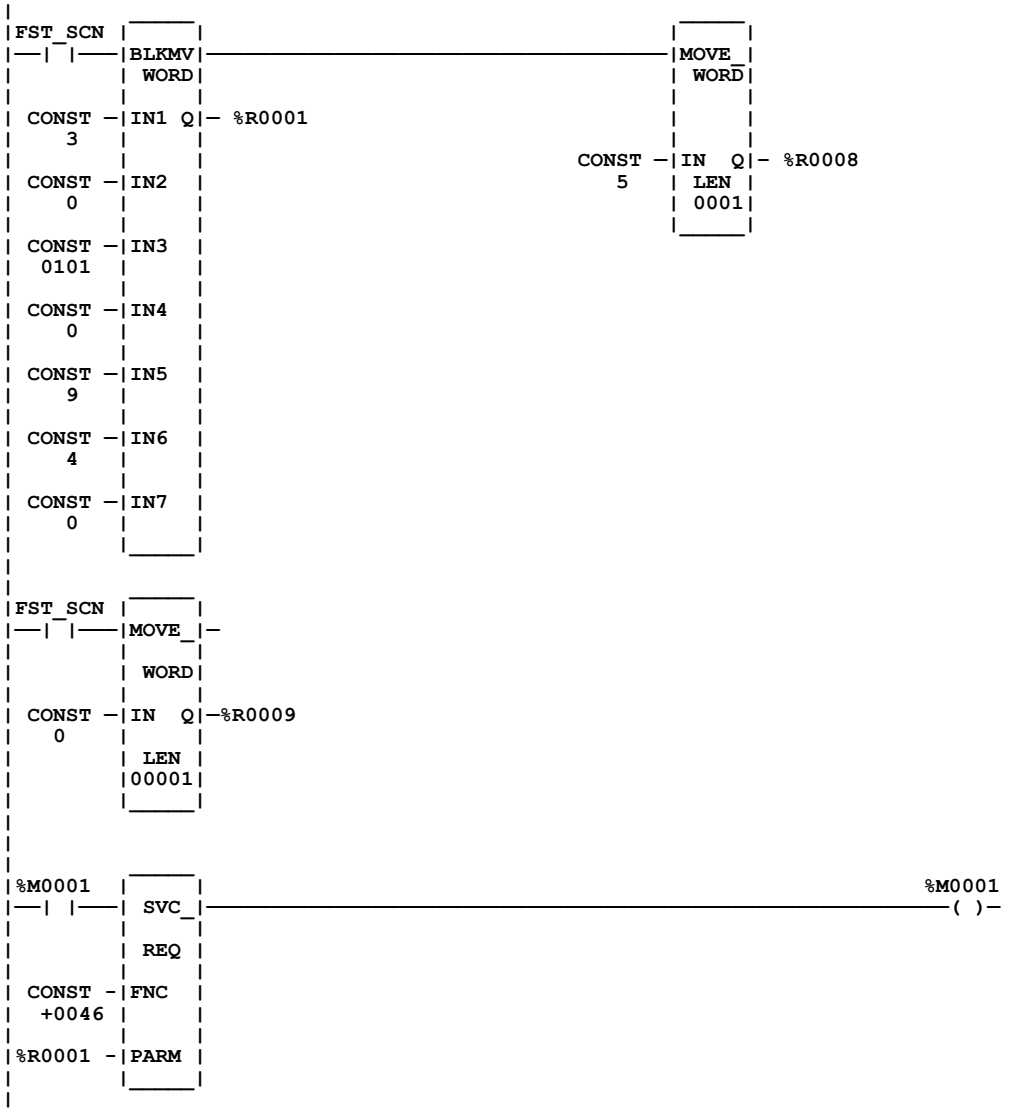
Exemple 1

L'exemple suivant montre la lecture d'un seul module dans le châssis 2, emplacement 4. IN4 et IN5 doivent être mis à zéro (0). IN6 et IN7 ne sont pas importants dans cet exemple. Si la fonction se termine correctement, les données se trouveront dans %R0004.



Exemple 2

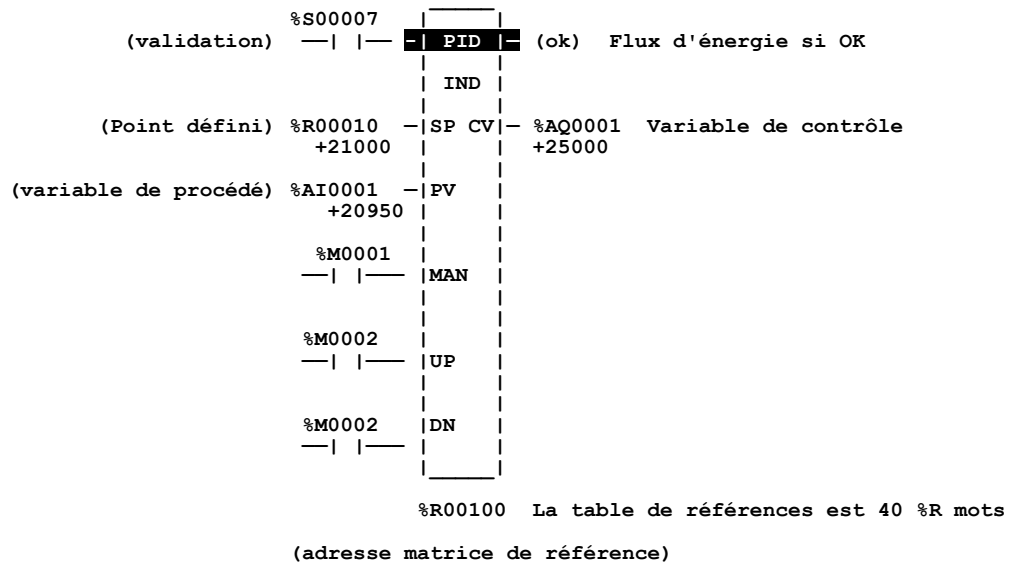
Cet exemple lit les données d'état supplémentaires dans le module du châssis 0, emplacement 4 et dans le module du châssis 1, emplacement 1. Il écrit un 5 dans le premier module et un 9 dans le deuxième. Noter que les modules n'ont pas besoin d'être listés dans l'ordre des numéros d'emplacement. Les données lues dans le module du châssis 0, emplacement 4 seront mises dans %R0007. Les données lues dans le module du châssis 1, emplacement 1 seront mises dans %R0004.



PID

La fonction de commande Proportionnelle , Intégrale et Dérivée (PID) est le plus connu des algorithmes pour le contrôle de processus en boucle fermée. Le bloc fonctionnel PID Série 90 compare le signal de retour d'une variable de procédé (PV) avec un point de consigne (SP) et rafraîchit une variable de commande (CV) basée sur l'erreur.

Le bloc utilise les gains de boucle PID et d'autres paramètres stockés dans une matrice de 40 mots de 16 bits (décrite à la page 12-73) pour résoudre l'algorithme PID dans l'intervalle de temps souhaité. Tous les paramètres sont des mots de 16 bits par soucis de compatibilité avec les variables de procédé analogiques de 16 bits. Ceci permet d'utiliser la mémoire %AI pour les variables de procédé et %AQ pour les variables de commande. L'exemple montré ci-dessous inclut des entrées typiques.



De nombreux paramètres doivent être définis soit en comptes ou unités PV, soit en comptes ou unités CV. Par exemple, l'entrée SP doit être convertie dans la même grandeur que PV car le bloc PID calcule l'erreur à partir de la différence entre ces deux entrées. Les comptes PV et CV peuvent être de –32000 à 32000 ou 0 à 32000, correspondant à la conversion analogique de 0 à 10000, pour afficher des variables comme 0.00% à 100.00%. Si les comptes PV et CV n'ont pas la même conversion il faudra un facteur d'échelle dans les gains PID.

Remarque

Une fonction PID ne s'exécutera pas plus qu'une fois toutes les 10 millisecondes. Ceci pourrait modifier vos résultats si vous l'utilisez en vue de l'exécuter à chaque cycle et que le cycle est inférieur à 10 millisecondes. Dans ce cas, la fonction PID ne sera excécutée qu'après un cumul de temps supérieur à 10 millisecondes. Par exemple, si le temps de cycle est de 9 millisecondes, la fonction PID s'exécutera après un autre cycle, soit un temps écoulé de 18 millisecondes à chaque fois.

Paramètres

Paramètre	Description
validation	Lorsqu'elle est validée par un contact, la fonction PID est exécutée.
SP	SP point de consigne du procédé. Etabli en utilisant les comptes PV, PID ajuste la sortie CV de façon à ce que PV corresponde à SP (zéro erreur).
PV	Mesure variable du procédé contrôlé, souvent une entrée %AI.
MAN	Lorsqu'il est à "1" (par un contact), le bloc PID est en mode **MANUEL** . Si ce paramètre n'est pas excité (0), le bloc PID est en mode automatique.
UP	S'il est excité avec MAN, il incrémente CV de 1 par résolution,.*
DN	S'il est excité avec MAN, il diminue CV de 1 par résolution.*
Adresse Table de références	L'adresse est l'emplacement de l'information du bloc PID (paramètres utilisateur et internes). Utilise les 40 mots %R qui ne peuvent pas être partagés.
ok	La sortie ok est excitée lorsque la fonction est exécutée sans erreur. Elle est à "0" s'il existe une(des) erreur(s).
CV	CV est la variable de commande vers le procédé, souvent une sortie analogique %AQ est utilisée.

*****Incrémenté (paramètre UP) ou décrémenté (paramètre DN) de 1 par accès de la fonction PID.

Types de mémoire valides

Paramètre	flux	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	aucun
validation	•											
SP		•	•	•	•		•	•	•	•	•	
PV		•	•	•	•		•	•	•	•		
MAN	•											
UP	•											
DN	•											
address								•				
ok	•											•
CV		•	•	•	•		•	•	•	•		

• Référence valide ou emplacement où l'énergie peut passer à travers la fonction.

Bloc paramètre PID

Avec les 2 mots d'entrée et les 3 contacts de contrôle manuels, le bloc PID utilise 13 paramètres de la table de références. Ces paramètres doivent être fixés avant d'appeler le bloc. Les autres paramètres sont utilisés par l'API et ne sont pas configurables. Le %Ref de la table ci-dessous correspond à la même adresse de références du bloc PID. Le nombre suivant le signe plus indique le décalage de la table. Par exemple, si la table de références commence à %R100, %R113 contiendra la commande manuelle utilisée pour régler la variable de contrôle et l'intégrateur en mode manuel.

Table 12-8. Vue d'ensemble des paramètres PID

Registre	Paramètre	Unités bit de poids faible	Plage de valeurs
%Ref+0000	Nombre de boucles	Entier	0 à 255 (pour l'affichage utilisateur seulement)
%Ref+0001	Algorithme	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0002	Période d'échantillonnage	10 millisecondes	0 (chaque cycle) à 65535 (10,9 Min). Utiliser au moins 10 pour les API 90-30 (voir la note à la page 12-71).
%Ref+0003	Plage morte	Comptes PV	0 à 32000 (jamais négatif)
%Ref+0004	Plage morte	Comptes PV	32000 à 0 (jamais positif)
%Ref+0005	Gain proportionnelle –Kp	0,01 CV%/PV%	0 à 327,67 %/%
%Ref+0006	Gain dérivée –Kd	0,01 seconde	0 à 327,67 sec
%Ref+0007	Vitesse intégrale –Ki	Répétitions/1000 sec	0 à 32,767 répétitions/sec
%Ref+0008	Polarisation CV/décalage sortie	Comptes CV	–32000 à 32000 (ajouter à la sortie intégrateur)
%Ref+0009	Blocage supérieur	Comptes CV	–32000 à 32000 (>%Ref+10) limite sortie
%Ref+0010	Blocage inférieur	Comptes CV	–32000 à 32000 (%Ref+09) limite sortie
%Ref+0011	Temps de parcours de course minimal	Seconde/Course totale	0 (aucune) à 32000 sec pour déplacer 32000 CV
%Ref+0012	Mot de configuration	5 bits de faible poids utilisés	Bits 0 à 2 pour erreur +/–, OutPolarity, Dériv.
%Ref+0013	Commande manuelle	Comptes CV	Suit CV en Auto ou règle CV en manuel
%Ref+0014	Mot de commande	Maintenu par l'API, ***sauf*** si le bit 1 est à "1".	Maintenu par l'API sauf si réglé autrement : Le bit de faible poids met le forçage si 1 (voir la description dans la table “Détails des paramètres PID” à la page 12-76)
%Ref+0015	SP interne	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0016	CV interne	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0017	PV interne	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0018	Sortie	Non disponible ; réglé et conservé par l'API	Non configurable

Table 12-8. Vue d'ensemble des paramètres PID - Suite

Registre	Paramètre	Unités bit de poids faible	Plage de valeurs
%Ref+0019	Stockage de l'expression différentielle	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0020 et %Ref+0021	Stockage de l'expression intégrale	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0022	Stockage de l'expression parcours de course	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0023	Horloge	Non disponible ; réglé et	Non configurable
%Ref+0024		conservé par	
%Ref+0025	(temps de la dernière exécution)	l'API	
%Ref+0026	Stockage du reste Y	Non disponible ; réglé et conservé par l'API	Non configurable
%Ref+0027	Plage inférieure pour SP, PV	Comptes PV	–32000 à 32000 (>%Ref+28) pour l'affichage
%Ref+0028	Plage supérieure pour SP, PV	Comptes PV	–32000 à 32000 (%Ref+27) pour l'affichage
%Ref+0029 +• %Ref+0034	Réservé pour usage interne	Non disponible	Non configurable
%Ref+0035 • %Ref+0039	Réservé pour usage externe	Non disponible	Non configurable

La table de références doit correspondre à des registres %R dans l'API 90-30. Noter que chaque bloc PID utilise une table de 40 mots différents même si seuls les 13 paramètres utilisateursont configurables, les autres mots étant utilisés pour le stockage de données interne de PID. S'assurer que la matrice ne s'étend pas au-delà de la limite mémoire.

Pour configurer les paramètres d'opération, sélectionner la fonction PID et appuyer sur **F10** pour zoomer dans un écran affichant les paramètres utilisateur ; ensuite, utiliser les touches flèches pour sélectionner les champs et taper les valeurs souhaitées. Vous pouvez utiliser 0 pour la plupart des valeurs par défaut, sauf pour le blocage supérieur CV qui doit être supérieur au blocage inférieur CV pour que le bloc PID fonctionne. Noter que le bloc PID ne laisse **pas** passer l'énergie s'il y a une erreur dans les paramètres utilisateur, aussi est-it nécessaire de surveiller sa bonne exécution avec une bobine temporaire pendant son paramétrage.

Une fois que les paramètres PID ont été fixés, ils doivent être déclarés comme des constantes par une fonction BLKMOV afin qu'elles puissent être utilisées pour recharger les paramètres utilisateur PID par défaut si nécessaire.

Fonctionnement de l'instruction PID

L'opération automatique normale consiste à appeler le bloc PID à chaque cycle avec le flux d'énergie sur validation et aucun flux d'énergie sur les contacts d'entrée manuelle. Le bloc compare l'horloge de temps écoulé courant de l'API avec le dernier temps de résolution stocké dans la table de références. Si la différence de temps est supérieure à la période d'échantillonnage définie dans le troisième mot (%Ref+2) de la table de références, l'algorithme PID est résolu en utilisant la différence de temps et le temps de la dernière résolution et la sortie de la variable de contrôle sont mis à jour. En mode automatique, la variable de contrôle de sortie est placée dans le paramètre de commande manuelle %Ref+13.

Si le flux d'énergie est fourni aux contacts validation et entrée manuelle, le bloc PID est mis en mode manuel et la variable de contrôle de sortie est mise à "1" par le paramètre de commande manuelle %Ref+13. Si l'une des deux entrées UP ou DN a le flux d'énergie, le mot de commande manuelle est incrémenté ou décrémenté d'un compte CV à chaque résolution PID. Pour un changement manuel plus rapide de la variable de contrôle de sortie, il est également possible d'ajouter ou de soustraire une valeur de compte CV directement vers le/à partir du mot de commande manuelle.

Le bloc PID utilise les paramètres de blocage supérieur et inférieur CV pour limiter la sortie CV. Si un temps de parcours de course minimal positif est défini, il est utilisé pour limiter la vitesse de changement de la sortie CV. Si la limite d'amplitude ou de vitesse est dépassée, la valeur stockée dans l'intégrateur est ajustée afin que CV soit à la limite. Cette fonction d'enroulement anti-réinitialisation (définie à la page 12-78) signifie que même si l'erreur a essayé de piloter CV au-dessus (ou en dessous) des blocages pendant une longue période, la sortie CV se dégagera du blocage dès que l'expression d'erreur changera de signe.

Cette opération, avec la commande manuelle suivant CV en mode automatique et mettant CV en mode manuel, permet un transfert sans heurt entre les modes automatique et manuel. Les blocages supérieur et inférieur CV et le temps de parcours de course minimal s'appliquent toujours à la sortie CV en mode manuel et la valeur interne stockée dans l'intégrateur est rafraîchie. Ceci signifie que si vous vouliez exécuter la commande manuelle en pas-à-pas en mode manuel, la sortie CV ne changerait pas plus vite que la limite de vitesse du temps de parcours de course minimal (inverse) et n'irait pas au-dessus ou en dessous des limites de blocage supérieure et inférieure CV.

Remarque

Une fonction PID particulière ne doit pas être appelée plus d'une fois par cycle.

La table suivante fournit davantage de détails concernant les paramètres abordés brièvement dans la Table 12-3. Le nombre entre parenthèses, après le nom de chaque paramètre, est le décalage dans la table de références.

Table 12-9. Détails des paramètres PID

Eléments 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 signé 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 l'algorithme 2.
Période d'échantillonnage (02)	Temps le plus court en incréments de 10 millisecondes, entre les résolutions de l'algorithme PID. Utilisez 10, par exemple, pour une période d'échantillonnage de 100 millisecondes. S'il est à zéro, l'algorithme est résolu à chaque fois que le bloc est appelé (voir la section ci-dessous concernant 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 résolution PID plus cette période d'échantillonnage. N'oubliez pas que le 90-30 n'utilise pas de temps de résolution inférieur à 10 millisecondes (voir la note à la page 12-71) ; ainsi des cycles seront sautés pour les temps de cycle plus courts. Cette fonction compense le temps réellement écoulé depuis la dernière exécution dans les 100 microsecondes. Si cette valeur est mise à "0", la fonction est exécutée à chaque fois qu'elle est validée ; cependant, elle est limitée à un minimum de 10 millisecondes comme noté ci-dessus.
Plage morte (+/—) (03/04)	Valeurs INT définissant les limites supérieures (+) et inférieure (–) de plage morte dans les comptes CV. Si aucune plage morte n'est nécessaire, ces valeurs doivent être de 0. Si l'erreur PID (SP – PV) ou (PV – SP) est supérieure à la valeur (–) et inférieure à la valeur (+), les calcules de PID sont résolus avec une erreur de 0. Si celle-ci est différente de 0, la valeur (+) doit être supérieure à 0 et la valeur (–) inférieure à 0 ou le bloc PID ne fonctionnera pas. *Vous devez les laisser à zéro jusqu'à ce que les gains de boucle PID soient définis ou ajustés*. Ensuite, vous pourriez souhaiter ajouter une plage morte pour éviter les petits changements de sortie CV dus à de petites variations de l'erreur, peut-être pour réduire l'usure mécanique.
Gain proportionnel–Kp (05)	Ce nombre INT, appelé gains du contrôleur, Kc, dans la version ISA, détermine le changement dans les comptes CV pour un changement de 100 comptes dans l'expression de l'erreur. Il est affiché sous la forme 0.00 %/% avec deux décimales implicites. Par exemple, un Kp entré de 450 sera affiché sous la forme 4.50 et résultera en une contribution Kp*Error/100 ou 450*Error/100 sur la sortie PID. Kp est généralement le premier gain établi lors du réglage d'une boucle PID.
Gain dérivée–Kd (06)	Ce nombre INT détermine le changement dans CV de comptes CV si l'erreur ou PV change 1 compte PV toutes les 10 millisecondes. Entré sous forme d'heure avec le bit de poids le plus faible indiquant 10 millisecondes, il est affiché 0,00 seconde avec 2 décimales implicites. Par exemple, un Kd entré de 120 sera affiché 1,20 seconde et résultera en une contribution Kd * delta Error/delta time ou 120*4/3 à la sortie PID si l'erreur a changé de 4 comptes PV toutes les 30 millisecondes. Kd peut être utilisé pour accélérer une réponse de boucle lente, mais il est très sensible aux parasites d'entrée.
Gain de vitesse intégrale–Ki (07)	Ce nombre INT détermine le changement de comptes CV dans CV si l'erreur était constamment de 1 compte PV. Il est affiché sous la forme 0,000 répétition/sec avec trois décimales implicites. Par exemple, un Ki entré de 1400 sera affiché 1,400 Répétitions/Sec et résultera en une contribution Ki * Error *dt ou 1400 * 20 * 50/1000 à la sortie PID pour une erreur de 20 comptes PV et un temps de cycle de l'API de 50 millisecondes (période d'échantillonnage de 0). Ki est habituellement le deuxième gain réglé après Kp.
Polarisation CV/décalage sortie	Valeur INT des comptes CV ajoutés à la sortie PID avant les blocages de vitesse et d'amplitude. Elle peut être utilisée pour établir des valeurs CV différentes de zéro si seuls les gains proportionnels Kp sont utilisés ou pour le contrôle anticipé de cette sortie de boucle PID à partir d'une autre boucle de contrôle.

Table 12-9. Détails des paramètres PID - Suite

Eléments de données	Description
Blocages CV supérieur et inférieur (09/10)	Valeurs INT de comptes CV qui définit les valeurs supérieures et inférieures pour CV. Ces valeurs sont nécessaires et le blocage supérieur doit avoir une valeur positive plus élevée que celle du blocage inférieur, sinon, le bloc PID ne fonctionnera pas. Elles sont habituellement utilisées pour définir des limites basées sur des limites physiques pour une sortie CV. Elles sont également utilisées pour convertir l'affichage de l'histogramme pour CV, pour l'affichage LM90. Le bloc possède un bouclage anti-réinitialisation pour modifier la valeur de l'intégrateur lorsqu'un blocage de CV est atteint.
Temps de parcours de course minimal (11)	Une valeur positive pour définir le nombre minimal de secondes pour qu'une sortie se déplace de 0 à la pleine course de 100% ou 32000 comptes CV. Il s'agit d'une limite de vitesse inverse sur la rapidité à laquelle la sortie CV peut être changée. Si elle est positive, CV ne peut pas changer de plus de 32000 comptes CV x temps Delta (secondes) divisé par le temps de parcours de course minimal. Par exemple, si la période d'échantillonnage était de 2,5 secondes et que le temps de parcours de course minimal est de 500 secondes, CV ne peut pas changer de plus de 32000*2.5/500 ou 160 comptes CV par résolution PID. Comme avec les blocages CV, il y a une fonction anti-saturation qui ajuste la valeur de l'intégrateur si la limite de vitesse de CV est dépassée. Si le temps de parcours de course minimal est de 0, il n'y a pas de limite de vitesse pour CVt. Vérifiez que vous réglez le temps de parcours de course minimal à 0 pendant que vous ajustez les gains de boucle PID.
Mot de configuration	Les 5 bits de poids faible de ce mot sont utilisés pour modifier les trois réglages standards de PID. Les autres bits doivent être mis à "0". Mettre le premier bit de poids faible à "1" pour modifier l'expression de l'erreur PID standard de (SP – PV) normal à (PV – SP), inversant le signe de l'expression signal de retour. Ceci concerne les contrôles d'action inverse où CV doit diminuer lorsque PV augmente. Mettre le deuxième bit à "1" pour inverser la polarité de la sortie de façon à ce que CV soit le négatif de la sortie PID plutôt que la valeur positive normale. Mettre le quatrième bit à "1" pour modifier l'action de la dérivée qui utilise le changement normal de l'expression de l'erreur, pour qu'elle utilise le changement de l'expression signal de retour de PV. Les 5 bits de poids faible du mot de configuration sont définis en détail ci-dessous : **Bit 0** = Expression de l'erreur. Lorsque ce bit est mis à "0", l'expression de l'erreur est SP — PV. Lorsque ce bit est mis à "1", l'expression de l'erreur est PV — SP. **Bit 1** = Polarité de sortie. Lorsque ce bit est mis à "0", la sortie CV représente la sortie du calcul de PID. Lorsqu'il est mis à "1", la sortie CV représente le négatif de la sortie du calcul de PID. **Bit 2** = Action dérivée sur PV. Lorsque ce bit est mis à "0", l'action dérivée est appliquée à l'expression de l'erreur. Lorsqu'il est mis à "1", l'action dérivée est appliquée à PV. Tous les bits restants doivent être à "0". **Bit 3** = Action plage morte. Lorsque le bit action plage morte est mis à "0", aucune action de plage morte n'est choisie. Si l'erreur se trouve dans les limites de la plage morte, l'erreur est forcée à zéro. Autrement, l'erreur n'est pas affectée par les limites de la plage morte. Si le bit action de plage morte est mis à "1", l'action plage morte est choisie. Si l'erreur se trouve dans les limites de la plage morte, l'erreur est forcée à zéro. Cependant, si l'erreur se trouve à l'extérieur des limites de plage morte, l'erreur est alors réduite par la limite de la plage morte (erreur = limite – de la plage morte). **Bit 4** = Action bouclage anti-réinitialisation. Lorsque ce bit est mis à "0", l'action de bouclage anti-réinitialisation utilise un calcul de réinitialisation. Lorsque la sortie est bloquée, ceci remplace la valeur du reste accumulé Y (défini à la page 12-78) avec la valeur nécessaire pour produire la sortie bloquée exactement. Lorsque le bit est mis à "1", ceci remplace l'expression Y accumulé par la valeur de l'expression Y au début du calcul. De cette façon, la valeur de pré-blocage Y est maintenue aussi longtemps que la sortie est bloquée. **NOTE :** Le bit d'action de bouclage anti-réinitialisation n'est disponible qu'avec les versions 6.50 ou supérieures des UC 90-30. N'oubliez pas que les bits sont définis en puissance de 2. Par exemple, pour mettre Config Word à "0" pour la configuration de PID par défaut, vous ajouteriez 1 pour changer l'expression de l'erreur de SP–PV à PV–SP ou ajouteriez 2 pour changer la polarité de sortie de CV = sortie PID en CV = – sortie PID ou ajouteriez 4 pour changer l'action dérivée de la vitesse d'erreur de changement en vitesse PV de changement, etc.

Table 12-9. Détails des paramètres PID - Suite

Eléments de données	Description
Commande manuelle (13)	Ceci est une valeur INT réglée sur la sortie CV courante pendant que le bloc PID est en mode automatique. Lorsque le bloc est commuté en mode **`Manuel`**, cette valeur est utilisée pour définir la sortie CV et la valeur interne de l'intégrateur dans les limites de blocage supérieur et inférieur et de temps de parcours de course.
Mot de contrôle (14)	Ceci est un paramètre interne qui est normalement laissé à "0". Si le bit de poids faible Forçage est mis à "1", ce mot et les autres paramètres internes SP, PV et CV doivent être utilisés pour le fonctionnement déporté de ce bloc PID (voir ci-dessous). Ceci permet à l'interface opérateur déporté, tel qu'un ordinateur, de prendre le contrôle à la place du programme de l'API. Précautions : si vous ne souhaitez pas que cela se produise, assurez-vous que le mot de contrôle est mis à "0". Si le bit de poids faible est à "0", les 4 bits suivants peuvent être lus pour suivre l'état des contacts d'entrée de PID tant que le contact validation de PID reçoit l'énergie. Une structure de données logiques avec les cinq premiers bits se positionne dans le format suivant : **Bit : Valeur du mot : Fonction : Etat ou action extérieure si le bit de forçage est à "1" :** 0 1 Forçage S'il est à "0", surveille les contacts du bloc ci-dessous. S'il est à "1", il les règle extérieurement. 1 2 Manuel/Auto S'il est à "1", le bloc est en mode **`Manuel`** ; en cas d'autres nombres, il est en mode **`Automatique`**. 2 4 Validation Doit normalement être à "1" ; autrement, le bloc n'est jamais appelé. 3 8 UP/Elever S'il est à "1" et que Manuel (Bit 1) est à "1", CV est incrémenté à chaque résolution. 4 16 DN/Abaisser S'il est à "1" et que Manuel (Bit 1) est à "1", CV est incrémenté à chaque résolution.
SP (15)	(Non configurable–défini et géré par l'API) Suit l'entrée SP ; doit être défini extérieurement si Forçage = 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 PV in ; doit être défini extérieurement si Forçage = 1.
Sortie (18)	(Non configurable–défini et géré par l'API) Ceci est une valeur mot signée représentant la sortie du bloc fonctionnel avant l'application de l'inversion optionnelle. Si aucune inversion de sortie n'est configurée et que le bit de polarité de la sortie du mot de contrôle est mis à "0", cette valeur sera égale à la sortie CV. Si l'inversion est sélectionnée et que le bit de polarité de la sortie est mis à "1", cette valeur sera égale au négatif de la sortie CV.
Stockage de l'expression différentielle (19)	Utilisé de façon interne pour le stockage de valeurs intermédiaires. *Ne pas écrire dans cette adresse.*
Stockage de l'expression intégrale (20/21)	Utilisé de façon interne pour le stockage de valeurs intermédiaires. *Ne pas écrire dans cette adresse.*
Stockage de l'expression parcours de la course (22)	Utilisé de façon interne pour le stockage de valeurs intermédiaires. *Ne pas écrire dans cette adresse.*
Horloge (23–25)	Stockage du temps écoulé interne (durée après la dernière exécution de PID). *Ne pas écrire dans ces adresses.*
Reste Y (26)	Conserve le reste pour la conversion de la division de l'intégrateur pour aucune une erreur d'état stable.
Plage inférieure et supérieure (27/28)	Valeurs INT optionnelles de comptes PV qui définissent les valeurs d'affichage la plus haute et la plus basse pour l'histogramme horizontal de la touche **`Zoom`** SP et PV du Logicmaster
Réservé (29–34 et 35–39)	29–34 sont réservés à l'usage interne ; 35–39 sont réservés à l'usage externe. Ils sont réservés pour l'utilisation par GE Fanuc et ne peuvent pas être utilisés à d'autres fins.

Paramètres internes de la table de références

Comme décrit dans la Table 12-3 dans les pages précédentes, le bloc PID lit 13 paramètres utilisateur et utilise le reste des 40 mots de la table de références pour un stockage interne de PID. Normalement, vous ne devriez pas avoir à modifier l'une de ces valeurs. Si vous appelez le bloc PID en mode Auto après un long moment, vous pourriez vouloir utiliser SVC_REQ #16 pour charger l'horloge de temps écoulé courant de l'API dans %Ref+23 pour rafraîchir la dernière durée de la résolution PID afin d'éviter un changement de pas sur l'intégrateur. Si vous avez mis le bit de poids faible Forçage du mot de contrôle (%Ref+14) à "1", les quatre bits suivants du mot de contrôle doivent être définis pour contrôler les contacts d'entrée du bloc PID (comme décrit dans la Table 12-3 sur les pages précédentes) et les SP et PV internes doivent être définis puisque vous avez éloigné le contrôle du bloc PID, de la logique en échelle.

Sélection d'algorithme PID (PIDISA ou PIDIND) et gains

Le bloc PID peut être programmé en sélectionnant soit l'expression indépendante (PID_IND), soit les versions standards ISA (PID_ISA) de l'algorithme PID. La seule différence entre les algorithmes est la façon dont les gains de l'intégrale et de la dérivée sont définis. Voici une explication de cette différence :

Les deux types de PID calculent l'expression de l'erreur comme SP – PV, ce qui peut être modifié en mode action inversée PV – SP si l'expression de l'erreur (bit de poids le plus faible 0 dans le mot de configuration %Ref+12 est mise à "1". Le mode action inversée peut être utilisé si vous souhaitez déplacer la sortie CV dans le sens opposé à partir des changements d'entrée PV (CV en bas pour PV en haut) plutôt que le normal CV en haut pour un PV en haut.

Erreur = (SP – PV) ou (PV – SP) si le bit de faible poids du mot de configuration est mis à "1"

La dérivée est normalement basée sur le changement de l'expression de l'erreur depuis la dernière résolution PID, ce qui peut provoquer un grand changement de la sortie si la valeur SP est modifiée. Si cela n'est pas souhaitable, le troisième bit du mot de configuration peut être mis à "1" pour calculer la dérivée à partir du changement de PV. Le dt (ou temps Delta) est déterminé en soustrayant le dernier temps d'horloge de la résolution PID, pour ce bloc, du temps écoulé courant de l'API.

dt = Horloge de temps écoulé courant de l'API – Horloge du temps écoulé de l'API lors de la dernière résolution PID.

Dérivée = (Erreur – erreur précédente)/dt ou (PV – PV précédent)/dt si le troisième bit du mot de configuration est mis à "1" 1

L'algorithme PID de l'expression indépendante (PID_IND) calcule la sortie comme étant :

Sortie PID = Kp * Erreur + Ki * Erreur * dt + Kd * Dérivée + polarisation CV

L'algorithme ISA standard (PID_ISA) a une forme différente :

Sortie PID = Kc * (Erreur + Erreur * dt/Ti + Td * Dérivée) + polarisation CV

où Kc est le gain du contrôleur et Ti est le temps de l'intégrale et Td est le temps de la dérivée. L'avantage de ISA est que l'ajustement de Kc change la contribution des expressions intégrale et dérivée ainsi que l'expression proportionnelle, ce qui peut faciliter l'ajustement de la boucle. Si vous avez des gains PID en expressions ou Ti et Td, utilisez

Kp = Kc Ki = Kc/Ti et Kd = Kc/Td

pour les convertir afin de les utiliser comme entrées de paramètre utilisateur PID.

L'expression de polarisation CV ci-dessus est une expression supplémentaire séparée des composants de PID. Il peut être nécessaire si vous utilisez seulement le gain Kp proportionnel et si vous souhaitez que CV soit une valeur différente de zéro lorsque PV est égal à SP et que l'erreur est de 0. Dans ce cas, mettre la polarisation CV au CV souhaité lorsque PV est à SP. La polarisation CV peut également être utilisée pour le contrôle anticipé où une autre boucle PID ou un autre algorithme de contrôle est utilisé pour ajuster la sortie CV de cette boucle PID.

Si un gain d'intégrale Ki est utilisé, la polarisation CV serait normalement de 0 puisque l'intégrateur agit comme une polarisation automatique. Mettre en route simplement en mode manuel et utiliser le mot de commande manuelle (%Ref+13) pour mettre l'intégrateur au CV souhaité, puis passer en mode automatique. Ceci fonctionne également si Ki est de 0, sauf que l'intégrateur ne sera pas ajusté par rapport à l'erreur après être passé en mode automatique.

Le schéma suivant montre comment fonctionnent les algorithmes PID :

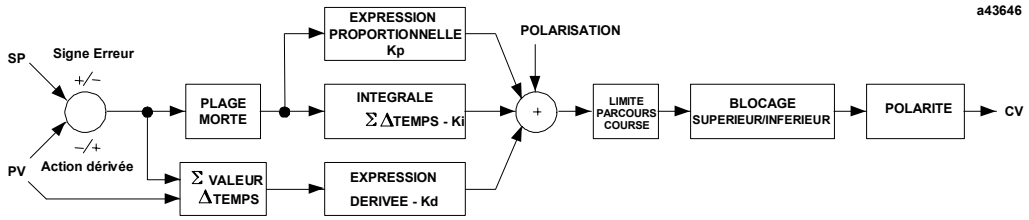


Figure 12-4. Algorithme de l'expression indépendante (PIDIND)

L'algorithme ISA (PIDISA) est similaire sauf que le gain Kp est factorisé de Ki et Kd de façon à ce que le gain de l'intégrale soit Kp * Ki et que le gain de la dérivée soit Kp * Kd. Le signe d'erreur, l'action dérivée et la polarisation sont définis par des bits du paramètre utilisateur Mot de configuration.

Amplitude CV et limites de vitesse

Le bloc n'envoie pas la sortie calculée de PID directement à CV. Les deux algorithmes peuvent imposer l'amplitude et la vitesse des limites de changement sur la variable de contrôle de sortie. La vitesse maximale de changement est déterminée en divisant la valeur CV maximale de 100% (32000) par le temps de parcours de course minimal, s'il est spécifié comme étant supérieur à 0. Par exemple, si le temps de parcours de course minimal est de 100 secondes, la limite de vitesse sera de 320 comptes CV par seconde. Si le temps de la résolution dt était de 50 millisecondes, la nouvelle sortie CV ne peut pas changer de plus de 320*50/1000 ou 16 comptes CV de la sortie CV précédente.

La sortie CV est alors comparée aux valeurs de blocage CV supérieure et inférieure. Si l'une des limites est dépassée, la sortie CV est mise à la valeur bloquée. Si l'une des limites de vitesse ou d'amplitude est dépassée en modifiant CV, la valeur interne de l'intégrateur est ajustée pour correspondre à la valeur limitée pour éviter un bouclage de réinitialisation.

Finalement, le bloc vérifie la polarisation de la sortie (2ème bit du mot de configuration %Ref+12) et change le signe de la sortie si le bit est à "1".

CV = Sortie PID bloquée ou – Sortie PID bloquée si le bit de polarisation de sortie est mis à "1"

Si le bloc est en mode automatique, le CV final est placé dans la commande manuelle %Ref+13. Si le bloc est en mode manuel, l'équation PID est sautée puisque CV est défini par la commande manuelle, mais toutes les limites de vitesse et d'amplitude restent vérifiées. ceci signifie que la commande manuelle ne peut pas changer la sortie au-dessus du blocage supérieur CV ou en dessous du blocage inférieur CV et que la sortie ne peut pas changer plus vite que le temps de parcours de course minimal autorisé.

Période d'échantillonnage et planification du bloc PID

Le bloc PID est une mise en œuvre numérique d'une fonction de commande analogique, ainsi, le temps d'échantillonnage dt de l'équation de sortie PID n'est pas le temps d'échantillonnage infiniment petit disponible avec des commandes analogiques. La majorité des procédés contrôlés peuvent être évalués approximativement comme un gain avec une erreur de poursuite de premier ou de second ordre, avec un retard pur possible. Le bloc PID définit une sortie CV au procédé et utilise le PV de retour pour déterminer une erreur pour ajuster la sortie CV suivante. Paramètre de procédé clé, la constante de temps total est le temps de réponse de PV lorsque CV est changé. Comme décrit dans la section Réglages des gains de boucle ci-dessous, la constante de temps total, Tp+Tc, pour un système de premier ordre est le temps requis pour PV pour atteindre 63% de sa valeur finale lorsque CV est echantillonné. Le bloc PID ne sera pas capable de contrôler un procédé si son temps d'échantillonnage n'est pas largement en dessous de la moitié de la constante de temps total. Des périodes d'échantillonnage plus grandes le rendront instable.

La période d'échantillonnage ne doit pas être supérieure à la constante de temps total divisée par 10 (ou au pire, par 5). Par exemple, si PV semble atteindre environ les 2/3 de sa valeur finale en 2 secondes, la période d'échantillonnage doit être inférieure à 0,2 seconde ou 0,4 seconde dans le pire des cas. D'autre part, la période d'échantillonnage ne doit pas être trop petite, c'est-à-dire, inférieure à la constante de temps total divisée par 1000, sinon, l'expression Ki * Error * dt pour l'intégrateur PID s'arrondira à 0. Par exemple, un procédé très lent prenant 10 heures ou 36000 secondes pour atteindre le niveau de 63% doit avoir une période d'échantillonnage de 40 secondes ou plus.

Sauf si le procédé est très rapide, il n'est pas habituellement nécessaire d'utiliser une période d'échantillonnage de 0 pour résoudre l'algorithme PID à chaque cycle PID. Si de nombreuses boucles PID sont utilisées avec un temps d'échantillonnage supérieur au temps de cycle, il pourrait y avoir de grandes variations dans le temps de cycle de l'API si de nombreuses boucles résolvent leur algorithme en même temps. La solution simple consiste à séquencer un ou plusieurs bits dans une matrice de bits mis à "0" utilisés pour valider le flux d'énergie dans chaque bloc PID.

Détermination des caractéristiques du procédé

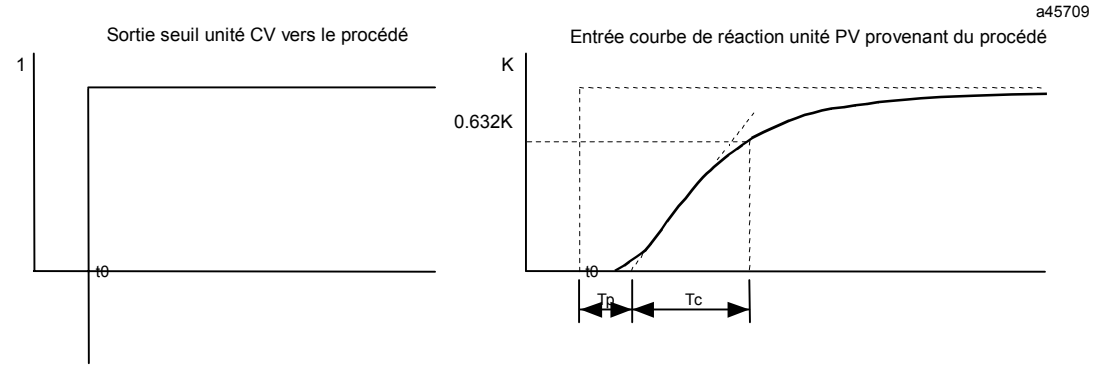
Les gains de boucle PID, Kp, Ki et Kd, sont déterminés par les caractéristiques du procédé contrôlé. Les deux principales questions qui se posent lors du réglage d'une boucle PID sont :

1. Quelle est l'ampleur du changement dans PV lorsque CV change d'une valeur fixe, ou quel est le gain en boucle ouverte ?
2. A quelle vitesse le système répond-il ou à quelle vitesse PV change-t-il après qu'une sortie CV est echantillonnée ?

De nombreux procédés peuvent être évalués approximativement par un gain de procédé, d'un retard de premier ou second ordre et d'une temporisation pure. Dans le domaine de la fréquence, la fonction de transfert pour un système à retard de premier ordre avec une temporisation pure est :

$$PV(s)/CV(s) = G(s) = K * e^{**(-Tp\ s)/(1 + Tc\ s)}$$

Le traçage d'une réponse echantillonnée à l'instant t0 dans le temps fournit une courbe de réaction en boucle ouverte :



Les paramètres du modèle de procédé suivants peuvent être déterminés à partir de la courbe de réaction de PV :

K	Gain de boucle ouverte du procédé = changement final dans PV/changement en CV à l'instant t0 (Noter l'absence d'indice pour K)
Tp	Temporisation du procédé ou de l'exécution en cascade, ou temps mort après t0 avant que la sortie PV du procédé commence le déplacement
Tc	Constante de temps de premier ordre du procédé, temps nécessaire après Tp pour que PV atteigne 63,2% du PV final

Habituellement, le moyen le plus rapide pour mesurer ces paramètres consiste à mettre le bloc PID en mode manuel et à effectuer un petit pas dans la sortie CV en changeant la commande manuelle %Ref+13 et en traçant la réponse de PV par rapport au temps. Pour les procédés lents, ceci peut se faire manuellement, mais pour les procédés plus rapides, il est préférable d'effectuer un enregistrement des données graphiques sur un enregistreur de diagrammes ou un ordinateur. La taille du seuil de CV doit être suffisamment grande pour provoquer un changement visible de PV, mais pas trop grande pour ne pas altérer le procédé mesuré. La bonne taille serait de 2 à 10% de la différence entre les valeurs de blocage supérieure et inférieure de CV .

Réglage des paramètres utilisateur incluant le réglage des gains de boucle

Comme tous les paramètres PID sont totalement dépendants du procédé à contrôler, il n'y a pas de valeurs prédéterminées qui pourraient fonctionner ; cependant, il existe une procédure simple et itérative pour trouver un gain de boucle acceptable.

1. Mettre tous les paramètres du bloc fonctionnel à "0", puis régler les blocages supérieur et inférieur de CV aux CV le plus haut et le plus bas souhaité. Régler la période d'échantillonnage à la constante de temps estimée du procédé.

2. Mettre le bloc en mode manuel et régler la commande manuelle (%Ref+13) à des valeurs différentes pour vérifier que CV peut être déplacé vers les blocages supérieur et inférieur. Enregistrer la valeur PV sur un point quelconque de CV et la charger dans SP.

3. Régler un petit gain, comme 100 * CV maximal/PV maximal, dans Kp et couper le mode manuel. Avancer pas-à-pas SP de 2 à 10% de la plage maximale de PV et observer la réponse de PV. Augmenter Kp si la réponse de PV est trop lente ou réduire Kp si PV dépasse et oscille sans atteindre une valeur stable.

4. Lorsqu'un Kp est trouvé, commencer à augmenter Ki pour que le dépassement s'amortisse vers une valeur stable en 2 à 3 cycles. Ceci peut nécessiter la réduction de Kp. Essayer également différentes valeurs de seuils et de points d'opération de CV.

5. Lorsque des gains Kp et Ki corrects sont trouvés, essayer d'ajouter Kd pour obtenir des réponses plus rapides pour les changements d'entrée, dans la mesure où cela ne provoque pas d'oscillations. Kd est souvent inutile et ne fonctionnera pas avec un PV parasité.

6. Vérifier les gains sur différents points de fonctionnement SP et ajouter la plage morte et le temps de parcours de course minimal si nécessaire. Certains procédés à action inversée peuvent nécessiter le réglage du signe de l'erreur dans le mot de configuration ou des bits de polarisation.

Réglage des gains de boucle—Approche de réglage de Ziegler et Nichols

Lorsque les trois paramètres de modèle de procédé, K, Tp et Tc, sont déterminés, ils peuvent être utilisés pour estimer les gains initiaux de boucle PID. L'approche suivante, développée par Ziegler et Nichols dans les années 40, est conçue pour fournir une bonne réponse aux perturbations du système avec des gains produisant un rapport d'amplitude de 1/4. Le rapport d'amplitude est le rapport de la seconde crête par rapport à la première crête dans la réponse d'une boucle fermée.

1. Calcul du taux de réaction :

   R = K/Tc

2. Pour le contrôle proportionnel seulement, calculer Kp comme

   Kp = 1/(R * Tp) = Tc/(K * Tp)

3. Pour le contrôle proportionnel et intégral, utiliser

   Kp = 0.9/(R * Tp) = 0.9 * Tc/(K * Tp)
   Ki = 0.3 * Kp/Tp

4. Pour le contrôle proportionnel, intégral et dérivé, utiliser

   Kp = G/(R * Tp) où G est de 1,2 à 2.0
   Ki = 0.5 * Kp/Tp
   Kd = 0.5 * Kp * Tp

5. vérifier que le temps d'échantillonnage se trouve dans la plage (Tp + Tc)/10 à (Tp + Tc)/1000

Une autre approche, la procédure de "Réglage idéal", est conçue pour fournir la meilleure réponse aux changements de SP, retardée seulement par la temporisation de procédé Tp ou le temps mort.

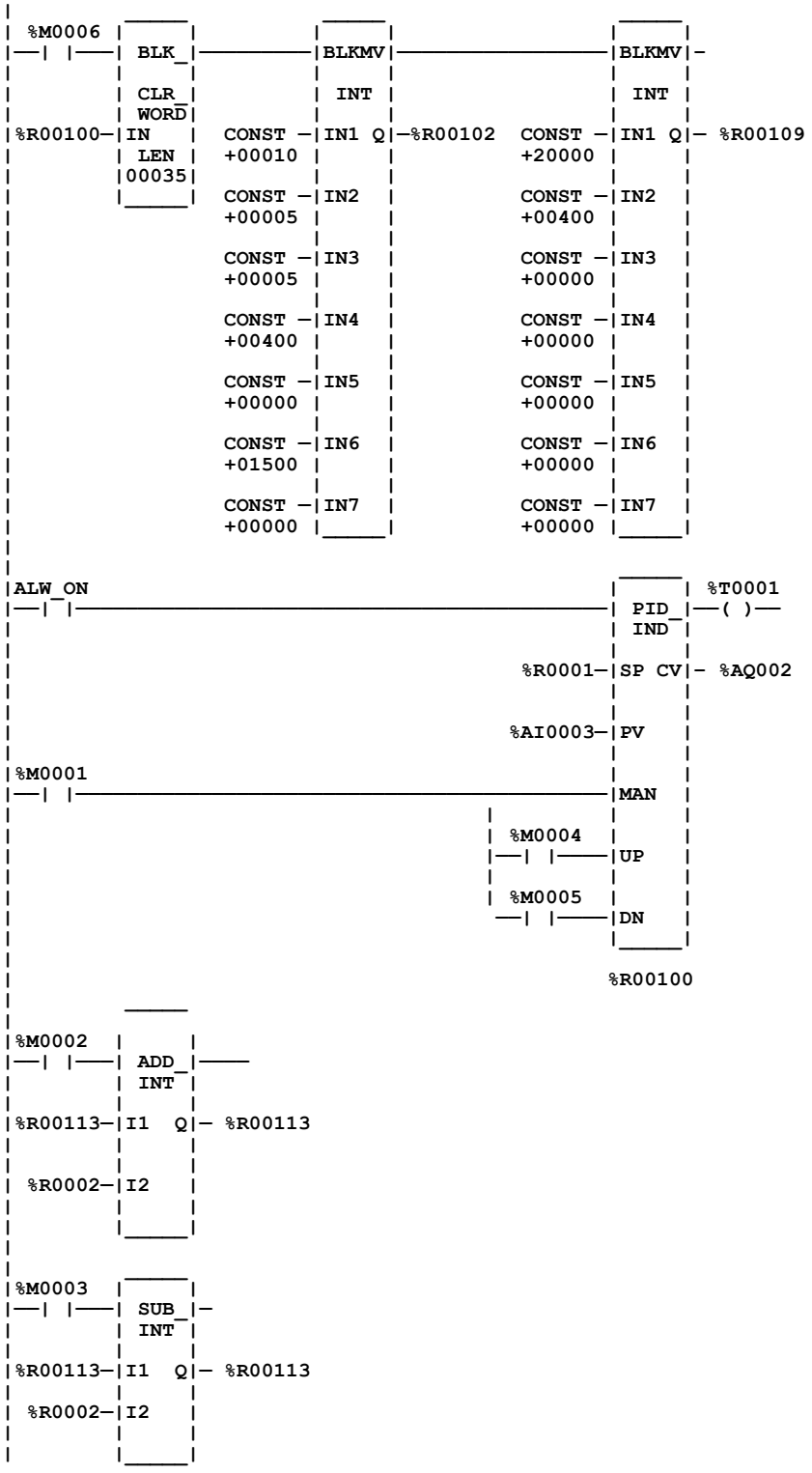
Kp = 2 * Tc/(3 * K * Tp)
Ki = Tc
Kd = Ki/4 si l'expression dérivée est utilisée

Lorsque les gains initiaux sont déterminés, ils doivent être convertis en entiers pour les paramètres utilisateur. Pour éviter les problèmes de conversion, le gain du procédé, K, doit être calculé comme étant un changement en comptes d'entrée de PV divisé par le changement de seuil de la sortie en comptes CV et non en unités techniques PV ou CV du procédé. Tous les temps doivent également être exprimés en secondes. Lorsque Kp, Ki et Kd sont déterminés, Kp et Kd peuvent être multipliés par 100 et entrés comme entiers alors que Ki peut être multiplié par 1000 et entré dans le paramètre utilisateur de la table de références.

Exemple d'appel PID

L'exemple suivant utilise une période d'échantillonnage de 100 millisecondes, un gain Kp de 4,00 et un gain Ki de 1,500. Le point défini est stocké dans %R1 avec la sortie de variable de contrôle dans %AQ2 et la variable de procédé dans %AI3. Les blocages supérieur et inférieur de CV doivent être définis, dans ce cas à 20000 et 400, et une petite plage morte optionnelle de +5 et –5 a été incluse. La table de références de 40 mots commence dans %R100. Normalement, les paramètres utilisateur sont définis dans la table de références avec la touche Zoom de PID **F10**, mais %M6 peut être défini pour initialiser les 14 mots commençant à %R102 (%Ref+2) à partir des constantes stockées dans la logique.

Le bloc peut être basculé en mode manuel avec %M1 de façon à ce que la commande manuelle, %R113, puisse être ajustée. Les bits %M4 ou %M5 peuvent être utilisés pour augmenter ou diminuer %R113 et le CV PID et l'intégrateur de 1 à chaque résolution de 100. Pour une opération manuelle plus rapide, les bits %M2 et %M3 peuvent être utilisés pour ajouter ou soustraire la valeur de %R2 à/de %R113 à chaque cycle de l'API. La sortie %T1 est à "1" lorsque PID est OK.


%M0006
BLK_
CLR_
WORD
%R00100
IN
LEN
00035
CONST
+00010
IN
BLKMV
INT
IN1
Q
%R00102
+00005
IN2
+00400
IN3
+00000
IN4
+00000
IN5
+00000
IN6
+00000
IN7
+00000
+20000
%R00109
ALW_ON
PID_
IND
%T0001
%R0001
SP
CV
%AQ002
%AI0003
PV
%M0001
MAN
%M0004
UP
%M0005
DN
%R00100
%M0002
ADD_
INT
%R00113
I1
I2
%R0002
%M0003
SUB_

Annexe
A

Temps d'exécution des instructions

Les API Série 9030, 9020 et Micro supportent de nombreuses fonctions et blocs fonctionnels. Cette annexe contient des tables montrant la taille des mémoires en octets et le temps d'exécution en microsecondes pour chaque fonction. La taille mémoire correspond au nombre d'octets pris par la fonction dans un programme d'application en schéma à relais.

Deux temps d'exécution sont disponibles pour chaque fonction :

Temps d'exécution	Description
Validé	Temps nécessaire pour exécuter la fonction ou le bloc fonctionnel lorsque le flux d'énergie entre et sort de la fonction. Typiquement, les temps dans le meilleur des cas sont lorsque les données utilisées par le bloc sont contenues dans la RAM utilisateur (mémoire orientée mot) et non dans la mémoire logique.
Désactivé	Temps nécessaire pour exécuter la fonction lorsque le flux d'énergie passe dans la fonction ou le bloc fonctionnel ; cependant, il est dans un état inactif comme lorsqu'un temporisateur est maintenu dans l'état de réinitialisation.

Remarque

Les temporisateurs et les compteurs sont rafraîchis chaque fois qu'ils sont rencontrés dans la logique

Remarque

Pour les UC des API 350, 351, 352 et 360, les temps sont identiques sauf pour l'instruction MOVE, qui est différente pour l'UC 350 — se référer à la remarque en bas de la table de la page A-6.

Table A-1. Temps d'exécution des instructions, Modèles standards

Groupe de fonctions	Fonction	Activée				Désactivée				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
Temporisateurs	Temporisateur de retard à l'activation	146	81	80	42	105	39	38	21	–	–	–	–	15
	Temporisateur de retard à la désactivation	98	47	44	23	116	63	58	32	–	–	–	–	9
	Temporisateur	122	76	75	40	103	54	53	30	–	–	–	–	15
Compteurs	Compteur	137	70	69	36	130	63	62	33	–	–	–	–	11
	Décompteur	136	70	69	37	127	61	61	31	–	–	–	–	11
Mathématiques	Addition (INT)	76	47	46	24	41	0	1	0	–	–	–	–	13
	Addition (DINT)	90	60	60	34	41	1	0	0	–	–	–	–	13
	Soustraction (INT)	75	46	45	25	41	0	1	0	–	–	–	–	13
	Soustraction (DINT)	92	62	62	34	41	1	0	0	–	–	–	–	13
	Multiplication (INT)	79	49	50	28	41	0	1	0	–	–	–	–	13
	Multiplication (DINT)	108	80	101	43	41	1	0	0	–	–	–	–	13
	Division (INT)	79	51	50	27	41	0	1	0	–	–	–	–	13
	Division (DINT)	375	346	348	175	41	1	0	0	–	–	–	–	13
	Division modulo	78	51	49	27	41	0	1	0	–	–	–	–	13
	Division modulo (DINT)	134	103	107	54	41	1	0	0	–	–	–	–	13
	Racine carrée (INT)	153	124	123	65	42	0	1	0	–	–	–	–	9
	Racine carrée (DINT)	268	239	241	120	42	0	0	1	–	–	–	–	9
Relationnelle	Egal à (INT)	66	35	36	19	41	1	1	0	–	–	–	–	9
	Egal à (DINT)	86	56	54	29	41	1	0	0	–	–	–	–	9
	Différent de (INT)	67	39	35	22	41	1	1	0	–	–	–	–	9
	Différent de (DINT)	81	51	51	28	41	1	0	0	–	–	–	–	9
	Supérieur à (INT)	64	33	35	20	41	1	1	0	–	–	–	–	9
	Supérieur à (DINT)	89	59	58	32	41	1	0	0	–	–	–	–	9
	Supérieur ou égal à (INT)	64	36	34	19	41	1	1	0	–	–	–	–	9
	Supérieur ou égal à (DINT)	87	58	57	30	41	1	0	0	–	–	–	–	9
	Inférieur à (INT)	66	35		19	41	1	1	0	–	–	–	–	9
	Inférieur à (DINT)	87	57		30	41	1	1	0	–	–	–	–	9
	Inférieur ou égal à (INT)	66	36	34	21	41	1	1	0	–	–	–	–	9
	Inférieur ou égal à (DINT)	86	57	56	31	41	1	1	0	–	–	–	–	9
	Plage (INT)	92	58	54	29	46	1	0	1	–	–	–	–	15
	Plage (DINT)	106	75	57	37	45	0	0	0	–	–	–	–	15
	Plage (WORD)	93	60	54	29	0	0	0	0	–	–	–	–	15

Remarques :
1. Le temps (en microsecondes) est basé sur la Version 5.01 du logiciel Logicmaster 9030/20 pour les UC Modèles 311, 313, 340 et 341 (Version 7 pour le 331).
2. Pour les fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les fonctions d'opération sur bit, en microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.

Table A-1. Temps d'exécution des instructions, Modèles standards - Suite

Groupe de fonctions	Fonction	Activée				Désactivée				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
Opérations logiques	ET logique	67	37	37	22	42	0	0	1	–	–	–	–	13
	OU logique	68	38	38	21	42	0	0	1	–	–	–	–	13
	OU logique exclusif	66	38	37	20	42	0	1	1	–	–	–	–	13
	NON logique inversé	62	32	31	17	42	0	1	1	–	–	–	–	9
	Décalage de bits à gauche	139	89	90	47	74	26	23	13	11.61	11.61	12.04	6.29	15
	Décalage de bits à droite	135	87	85	45	75	26	24	13	11.63	11.62	12.02	6.33	15
	Rotation de bits à gauche	156	127	126	65	42	1	1	0	11.70	11.78	12.17	6.33	15
	Rotation de bits à droite	146	116	116	62	42	1	1	0	11.74	11.74	12.13	6.27	15
	Position de bit	102	72	49	38	42	1	0	0	–	–	–	–	13
	Effacement de bit	68	38	35	21	42	1	1	1	–	–	–	–	13
	Test de bit	79	49	51	28	41	0	0	1	–	–	–	–	13
	Etablissement de bit	67	37	37	20	42	0	0	0	–	–	–	–	13
	Comparaison masquée (WORD)	217	154	141	74	107	44	39	21	–	–	–	–	25
	Comparaison masquée (DWORD)	232	169	156	83	108	44	39	22	–	–	–	–	25
Déplacement de donnée	Déplacement (INT)	68	37	39	20	43	0	0	0	1.62	1.62	5.25	1.31	13
	Déplacement (BIT)	94	62	64	35	42	0	0	0	12.61	12.64	12.59	6.33	13
	Déplacement (WORD)	67	37	40	20	41	0	0	0	1.62	1.63	5.25	1.31	13
	Déplacement de bloc (INT)	76	48	50	28	59	30	30	16	–	–	–	–	27
	Déplacement de bloc (WORD)	76	48	49	29	59	29	28	15	–	–	–	–	27
	Effacement de bloc	56	28	27	14	43	0	0	0	1.35	1.29	1.40	0.78	9
	Registre de décalage (BIT)	201	153	153	79	85	36	34	18	0.69	0.68	0.71	0.37	15
	Registre de décalage (WORD)	103	53	52	29	73	25	23	12	1.62	1.62	2.03	1.31	15
	Séquenceur de bits	165	101	99	53	96	31	29	16	0.07	0.07	0.08	0.05	15
	COMM_REQ	1317	1272	1489	884	41	2	0	0	–	–	–	–	13
Table	Déplacement de matrice													
	INT	230	201	177	104	72	41	40	20	1.29	1.15	10.56	2.06	21
	DINT	231	202	181	105	74	44	42	23	3.24	3.24	10.53	2.61	21
	BIT	290	261	229	135	74	43	42	23	–.03	–.03	-0.01	0.79	21
	BYTE	228	198	176	104	74	42	42	23	0.81	0.82	8.51	1.25	21
	WORD	230	201	177	104	72	41	40	20	1.29	1.15	10.56	2.06	21
	Rechercher égal à													
	INT	197	158	123	82	78	39	37	20	1.93	1.97	2.55	1.55	19
	DINT	206	166	135	87	79	38	36	21	4.33	4.34	4.55	2.44	19
	BYTE	179	141	117	74	78	38	36	21	1.53	1.49	1.83	1.03	19
	WORD	197	158	123	82	78	39	37	20	1.93	1.97	2.55	1.55	19

Remarque :
1. Le temps (en microsecondes) est basé sur la Version 5.01 du logiciel Logicmaster 9030/20 pour les UC Modèles 311, 313, 340 et 341 (Version 7 pour le 331).
2. Pour les fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les opérations logiques, microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.
7. Pour les instructions qui ont une valeur d'incrément, multiplier l'incrément par (Longueur –1) et ajouter cette valeur au temps de base.

Table A-1. Temps d'exécution des instructions, Modèles standards - Suite

Groupe de fonctions	Fonction	Activée				Désactivée				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
	Recherche différent de													
	INT	198	159	124	83	79	39	36	21	1.93	1.93	2.48	1.52	19
	DINT	201	163	132	84	79	37	35	21	6.49	6.47	6.88	3.82	19
	BYTE	179	141	117	73	79	38	36	19	1.54	1.51	1.85	1.05	19
	WORD	198	159	124	83	79	39	36	21	1.93	1.93	2.48	1.52	19
	Recherche supérieur à													
	INT	198	160	125	82	79	37	38	19	3.83	3.83	4.41	2.59	19
	DINT	206	167	135	88	78	38	36	20	8.61	8.61	9.03	4.88	19
	BYTE	181	143	118	73	79	37	36	19	3.44	3.44	3.75	2.03	19
	WORD	198	160	125	82	79	37	38	19	3.83	3.83	4.41	2.59	19
	Recherche supérieur ou égal à													
	INT	197	160	124	83	77	38	36	20	3.86	3.83	4.45	2.52	19
	DINT	205	167	136	87	80	39	36	21	8.62	8.61	9.02	4.87	19
	BYTE	180	142	118	75	79	37	37	20	3.47	3.44	3.73	2.00	19
	WORD	197	160	124	83	77	38	36	20	3.86	3.83	4.45	2.52	19
	Recherche inférieur à													
	INT	199	159	124	84	78	38	36	20	3.83	3.86	4.48	2.48	19
	DINT	206	168	135	87	79	38	38	19	8.62	8.60	-1.36	4.88	19
	BYTE	181	143	119	75	80	38	37	20	3.44	3.44	3.75	2.00	19
	WORD	199	159	124	84	78	38	36	20	3.83	3.86	4.45	2.48	19
	Recherche inférieur ou égal à													
	INT	200	158	124	82	79	38	37	21	3.79	3.90	4.45	2.55	19
	DINT	207	167	137	88	78	39	37	19	8.60	8.61	9.01	4.86	19
	BYTE	180	143	119	74	78	40	37	19	3.46	3.44	3.73	2.02	19
	WORD	200	158	124	82	79	38	37	21	3.79	3.90	4.45	2.55	19
Conversion	Conversion en INT	74	46	39	25	42	1	1	1	–	–	–	–	9
	Conversion en BCD 4 chiffres	77	50	34	25	42	1	1	1	–	–	–	–	9

Remarque :
1. Le temps (en microsecondes) est basé sur la Version 5.01 du logiciel Logicmaster 9030/20 pour les UC Modèles 311, 313, 340 et 341 (Version 7 pour le 331).
2. Pour les fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les opérations logiques, microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.
7. Pour les instructions qui ont une valeur d'incrément, multiplier l'incrément par (Longueur –1) et ajouter cette valeur au temps de base.

Table A-1. Temps d'exécution des instructions, Modèles standards - Suite

Groupe de fonctions	Fonction	Activée				Désactivée				Incrément				Taille
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
Contrôle	Appel de sous-programme	155	93	192	85	41	0	0	0	–	–	–	–	7
	Faire E/S	309	278	323	177	38	1	0	0	–	–	–	–	12
	Algorithme PID – ISA	1870	1827	1812	929	91	56	82	30	–	–	–	–	15
	Algorithme PID – IND	2047	2007	2002	1017	91	56	82	30	–	–	–	–	15
	Instruction Fin	–	–	–	–	–	–	–	–	–	–	–	–	–
	Requête de service													
	# 6	93	54	63	45	41	2	0	0	–	–	–	–	9
	# 7 (Lire)	–	37	309	161	–	2	0	0	–	–	–	–	9
	# 7 (Etablir)	–	37	309	161	–	2	0	0	–	–	–	–	9
	#14	447	418	483	244	41	2	0	0	–	–	–	–	9
	#15	281	243	165	139	41	2	0	0	–	–	–	–	9
	#16	131	104	115	69	41	2	0	0	–	–	–	–	9
	#18	–	56	300	180	–	2	0	0	–	–	–	–	9
	#23	1689	1663	1591	939	43	1	0	0	–	–	–	–	9
	#26//30*	1268	1354	6680	3538	42	0	0	0	–	–	–	–	9
	#29	–	–	55	41	–	–	1	0	–	–	–	–	9
	MCR/ENDMCR combinés imbriqués	135	73	68	39	75	25	21	12	–	–	–	–	8

***** La requête de service #26/30 a été mesurée en utilisant un compteur rapide, une sortie de 16 points, dans un châssis de cinq emplacements.

Remarques :
1. Le temps (en microsecondes) est basé sur la Version 5.01 du logiciel Logicmaster 9030/20 pour les UC Modèles 311, 313, 340 et 341 (Version 7 pour le 331).
2. Pour les fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les fonctions d'opérations logiques, microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.
7. Pour les instructions qui ont une valeur d'incrément, multiplier l'incrément par (Longueur –1) et ajouter cette valeur au temps de base.

Table A-2. Temps d'exécution des instructions, Modèles de hautes performances

Groupe de fonctions	Fonction	Activée 350/351/36x	Désactivée 350/351/36x	Incrément 350/351/36x	Activée 352	Désactivée 352	Incrément 352	Taille
Temporisateurs	Temporisateur de retard à l'activationOn-Delay Timer	4	6	–	4	5	–	15
	Temporisateur	3	3	–	2	2	–	15
	Temporisateur de retard à la désactivation	3	3	–	3	2	–	15
Compteurs	Compteur	1	3	–	2	2	–	13
	Décompteur	3	3	–	1	2	–	13
Mathématique	Addition (INT)	2	0	–	1	0	–	13
	Addition (DINT)	2	0	–	2	0	–	19
	Addition (REAL)	52	0	–	33	0	–	17
	Soustraction (INT)	2	0	–	1	0	–	13
	Soustraction (DINT)	2	0	–	2	0	–	19
	Soustraction (REAL)	53	0	–	34	0	–	17
	Multiplication (INT)	21	0	–	21	0	–	13
	Multiplication (DINT)	24	0	–	24	0	–	19
	Multiplication (REAL)	68	1	–	38	1	–	17
	Division (INT)	22	0	–	22	0	–	13
	Division (DINT),	25	0	–	25	0	–	19
	Division (REAL)	82	2	–	36	2	–	17
	Division modulo (INT)	21	0	–	21	0	–	13
	Division modulo (DINT)	25	0	–	25	0	–	19
	Racine carrée (INT)	42	1	–	41	1	–	10
	Racine carrée (DINT)	70	0	–	70	0	–	13
	Racine carrée (REAL)	137	0	–	35	0	–	11
Trigonométrique	SIN (REAL)	360	0	–	32	0	–	11
	COS (REAL)	319	0	–	29	0	–	11
	TAN (REAL)	510	1	–	32	1	–	11
	ASIN (REAL)	440	0	–	45	0	–	11
	ACOS (REAL)	683	0	–	63	0	–	11
	ATAN (REAL)	264	1	–	33	1	–	11
Logarithmique	LOG (REAL)	469	0	–	32	0	–	11
	LN (REAL)	437	0	–	32	0	–	11
Exponentielle	EXP	639	0	–	42	0	–	11
	EXPT	89	1	–	54	1	–	17
Conversion radian	Conversion RAD en DEG	65	1	–	32	1	–	11
	Conversion DEG en RAD	59	0		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 fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les fonctions d'opérations logiques, microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.

Table A-2. Temps d'exécution des instructions, Modèles de hautes performances - Suite

Groupe de fonctions	Fonction	Activée 350/351/36x	Désactivée 350/351/36x	Incrément 350/351/36x	Activée 352	Désactivée 352	Incrément 352	Taille
Relationnelle	Egale à (INT)	1	0	–	1	0	–	10
	Egal à (DINT)	2	0	–	2	0	–	16
	Egale à (REAL)	57	0	–	28	0	–	14
	Différent de (INT)	1	0	–	1	0	–	10
	Différent de (DINT)	1	0	–	1	0	–	16
	Différent de (REAL)	62	0	–	31	0	–	14
	Supérieur à (INT)	1	0	–	1	0	–	10
	Supérieur à (DINT)	1	0	–	1	0	–	16
	Supérieur à (REAL)	57	0	–	32	0	–	14
	Supérieur ou égal à (INT)	1	0	–	1	0	–	10
	Supérieur ou égal à (DINT)	1	0	–	1	0	–	10
	Supérieur ou égal à (REAL)	57	1	–	31	1	–	14
	Inférieur à (INT)	1	0	–	1	0	–	10
	Inférieur à (DINT)	1	0	–	1	0	–	16
	Inférieur à (REAL)	58	1	–	36	1	–	14
	Inférieur ou égal à (INT)	1	0	–	1	0	–	10
	Inférieur ou égal à (DINT)	3	0	–	3	0	–	16
	Inférieur ou égal à (REAL)	37	0	–	37	0	–	14
	Plage (INT)	2	1	–	2	1	–	13
	Plage (DINT)	2	1	–	2	1	–	22
	Plage (WORD)	1	0	–	1	0	–	13
Opérations logiques	ET logique	2	0	–	2	0	–	13
	OU logique	2	0	–	2	0	–	13
	OU logique exclusif	1	0	–	1	0	–	13
	NON logique inversé	1	0	–	1	0	–	10
	Décalage de bits à gauche	31	1	1.37	31	1	1.37	16
	Décalage de bits à droite	28	0	3.03	28	0	3.03	16
	Rotation de bits à gauche	25	0	3.12	25	0	3.12	16
	Rotation de bits à droite	25	0	4.14	25	0	4.14	16
	Position de bit	20	1	–	20	1	–	13
	Effacement de bit	20	0	–	20	0	–	13
	Test de bit	20	0	–	20	0	–	13
	Etablissement de bit	19	1	–	19	1	–	13
	Comparaison masquée (WORD)	52	0	–	52	0	–	25
	Comparaison masquée (DWORD)	50	0	–	49	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 fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les fonctions d'opérations logiques, microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.
7. Pour les instructions qui ont une valeur d'incrément, multiplier l'incrément par (Longueur –1) et ajouter cette valeur au temps de base.

Table A-2. Temps d'exécution des instructions, Modèles de hautes performances - Suite

Groupe de fonctions	Fonction	Activée 350/351/36X	Désactivée 350/351/36X	Incrément 350/351/36X	Activée 352	Désactivée 352	Incrément 352	Taille
Déplacement de donnée	Déplacement (INT)	2	0	0.41	2	0	0.41	10
	Déplacement (BIT)	28	0	4.98	28	0	4.98	13
	Déplacement (WORD)	2	0	0.41	2	0	0.41	10
	Déplacement (REAL)	24	1	0.82	24	1	0.82	13
	Déplacement de bloc (INT)	2	0	–	2	0	–	28
	Déplacement de bloc (WORD)	4	4	–	3	0	–	28
	Déplacement de bloc (REAL)	41	0	–	41	0	–	13
	Effacement de bloc	1	0	0.24	1	0	0.24	11
	Registre de décalage (BIT)	49	0	0.23	46	0	0.23	16
	Registre de décalage (WORD)	27	0	0.41	27	0	0.41	16
	Séquenceur de bits	38	22	0.02	38	22	0.02	16
	COMM_REQ	765	0	–	765	0	–	13
Table	Déplacement de matrice							
	INT	54	0	0.97	54	0	0.97	22
	DINT	54	0	0.81	54	0	0.81	22
	BIT	69	0	0.36	69	0	0.36	22
	BYTE	54	1	0.64	54	1	0.64	22
	WORD	54	0	0.97	54	0	0.97	22
	Recherche égal à							
	INT	37	0	0.62	37	0	0.62	19
	DINT	41	1	1.38	41	1	1.38	22
	BYTE	35	0	0.46	35	0	0.46	19
	WORD	37	0	0.62	37	0	0.62	19
	Recherche différent de							
	INT	37	0	0.62	37	0	0.62	19
	DINT	38	0	2.14	38	0	2.14	22
	BYTE	37	0	0.47	37	0	0.47	19
	WORD	37	0	0.62	37	0	0.62	19
	Recherche supérieur à							
	INT	37	0	1.52	37	0	1.52	19
	DINT	39	0	2.26	39	0	2.26	22
	BYTE	36	1	1.24	36	1	1.24	19
	WORD	37	0	1.52	37	0	1.52	19
	Recherche supérieur ou égal à							
	INT	37	0	1.48	37	0	1.48	19
	DINT	39	0	2.33	39	0	2.33	22
	BYTE	37	1	1.34	37	1	1.34	19
	WORD	37	0	1.48	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 350 et 360.
2. Pour les fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les fonctions d'opérations logiques, microsecondes/bit. ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.
7. Pour les instructions qui ont une valeur d'incrément, multiplier l'incrément par (Longueur –1) et ajouter cette valeur au temps de base.

Table A-2. Temps d'exécution des instructions, Modèles de hautes performances - Suite

Groupe de fonctions	Fonction	Activée 350/351/36x	Désactivée 350/351/36x	Incrément 350/351/36x	Activée 352	Désactivée 352	Incrément 352	Taille
	Recherche inférieur à							
	INT	37	0	1.52	37	0	1.52	19
	DINT	41	1	2.27	41	1	2.27	22
	BYTE	37	0	1.41	37	0	1.41	19
	WORD	37	0	1.52	37	0	1.52	19
	Recherche inférieur ou égal à							
	INT	38	0	1.48	38	0	1.48	19
	DINT	40	1	2.30	40	1	2.30	22
	BYTE	37	0	1.24	37	0	1.24	19
	WORD	38	0	1.48	38	0	1.48	19
Conversion	Conversion en INT	19	1	–	19	1	–	10
	Conversion en BCD-4	21	1	–	21	1	–	10
	Conversion en REAL	27	0	–	21	0	–	8
	Conversion en WORD	28	1	–	30	1	–	11
	Tronquer en INT	32	0	–	32	0	–	11
	Tronquer en DINT	63	0	–	31	0	–	11
Contrôle	Appel de sous-programme	72	1	–	73	1	–	7
	Faire E/S	114	1	–	115	1	–	13
	Algorithme PID – ISA	162	34	–	162	34	–	16
	Algorithme PID – IND	146	34	–	146	34	–	16
	Instruction Fin	–	–	–	–	–	–	–
	Requête de service							
	#6	22	1	–	22	1	–	10
	# 7 (Lire)	75	1	–	75	1	–	10
	# 7 (Etablir)	75	1	–	75	1	–	10
	#14	121	1	–	121	1	–	10
	#15	46	1	–	46	1	–	10
	#16	36	1	–	36	1	–	10
	#18	261	1	–	261	1	–	10
	#23	426	0	–	426	0	–	10
	#26//30**	2260	1	–	2260	1	–	10
	#29	20	0	–	20	0	–	10
	#43							
	MCR/ENDMCR imbriquées Combinées	1	1	–	1	1	–	4
	Enregistreur d'Evénement Séquentiel (SER)	Voir la Table A-3	26.50	Voir la Table A-3				

*Les temps PID montrés ci-dessus sont basés sur la Version 6,5 de l'UC 351.

La requête de service #26/30 a été mesurée en utilisant un compteur rapide, une sortie de 16 points, dans un châssis de cinq emplacements.

Remarques :

1. Le temps (en microsecondes) est basé sur la Version 7 du logiciel Logicmaster 90-30/20/Micro pour les UC Modèles 350 et 360..
2. Pour les fonctions sur tableau, l'incrément est dans les unités de longueur spécifiées ; pour les fonctions d'opération sur bit, en microsecondes/bit ; pour les fonctions de déplacement de données, en microsecondes/nombre de bits ou mots.
3. Temps de validation pour des unités de longueur simple du type %R, %AI et %AQ.
4. Le temps COMMREQ a été mesuré entre l'UC et HSC.
5. DOIO est le temps mis pour sortir des valeurs vers un module de sorties logiques.
6. Lorsqu'il y a plus d'un cas possible, le temps indiqué ci-dessus représente le cas le plus défavorable possible.
7. Pour les instructions qui ont une valeur d'incrément, multiplier l'incrément par (Longueur –1) et ajouter cette valeur au temps de base.

Table A-3. Temps d'exécution du bloc fonctionnel SER

Configuration	Exemple	Temps (µsec)
Sans flux d'énergie (désactivé)	—	26.50
Contiguës		
8 voies	%I1—8	79.94
16 voies	%I1—16	80.58
24 voies	%I1—24	81.56
32 voies	%I1—32	81.73
8 + 8 voies contiguës	%I1—8 et %Q1—8	111.03
8 + 8 + 8 voies contiguës	%I1—8, %Q1—8 et %M1—8	143.38
8 + 8 + 8 + 8 voies contiguës	%I1—8, %Q1—8 et %M1—8 and %T1—8	175.79
Non contiguës	%I1, %M10, %Q3, etc.	
8 voies		299.64
16 voies		552.83
24 voies		806.35
32 voies		1059.85
Réinitialisation		
avec 8 voies	—	162.63
avec 16 voies	—	267.51
avec 24 voies	—	372.73
avec 32 voies	—	477.95

Remarques :

Lorsqu'un emplacement comportant un module d'entrées est spécifié, ajouter 46 µsecondes supplémentaires à chaque temps d'exécution **Contiguës** et **Non contiguës**.

Lorsque le déclenchement se produit, ajouter 29 µsecondes supplémentaires si le format BCD est utilisé ou 148 µsecondes si le format Posix est utilisé.

Les temps montrés pour la réinitialisation sont pour la taille maximale de tampon de 1024 échantillons. (La réinitialisation efface tous les échantillons en cours de mémorisation).

Tailles des instructions pour les UC de hautes performances

La taille mémoire est le nombre d'octets nécessités par l'instruction dans un programme d'application en schéma en échelle. Les UC Modèles 351 et 352 nécessitent trois octets pour la plupart des fonctions Booléennes ; voir la Table A3.

Table A-4. Taille des instructions pour les UC 350—352, 360, 363 et 364

Fonction	Taille
Pas d'opération	1
Dépiler la pile et ET en haut	1
Dépiler la pile et OU en haut	1
Dupliquer le haut de la pile	1
Dépiler la pile	1
Pile initiale	1
Etiquette	5
Saut	5
Toutes les autres instructions	3
Blocs fonctionnels ; voir la Table A-2	–

Temps d'exécution des éléments Booléens

La table ci-dessous liste les temps d'exécution des bobines et des contacts pour les modules d'UC Série 90-30.

Table A-5. Temps d'exécution des éléments Booléens

Modèle d'UC	Temps d'exécution par 1 000 contacts/bobines Booléens
Modèles 350 et 360	0,22 milliseconde
Modèles 340/341	0,3 milliseconde
Modèle 331	0,4 milliseconde
Modèles 313/323	0,6 milliseconde
Modèle 311	18,0 millisecondes

Annexe
B

Interprétation des Tables de défauts

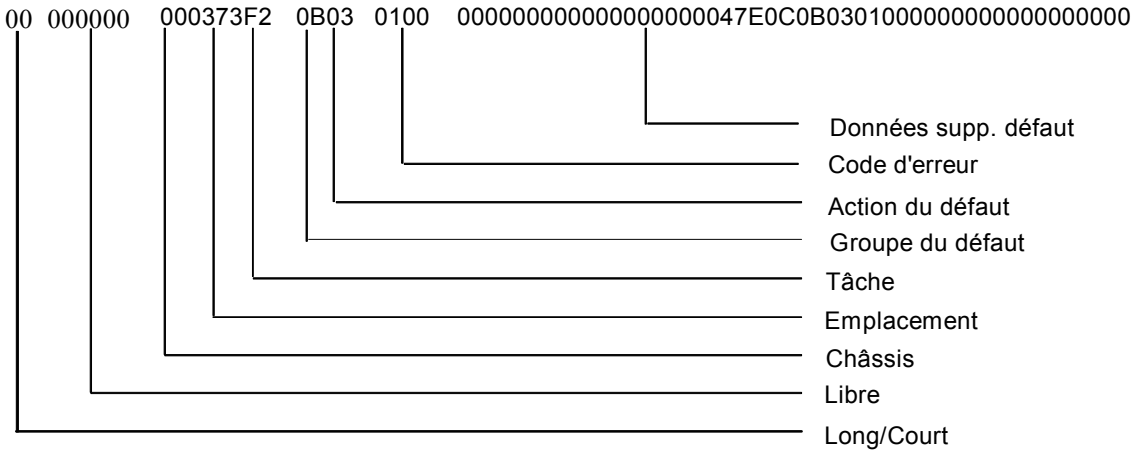
Les UC Série 90-30, Série 90-20 et Série 90 Micro maintiennent à jour deux tables de défauts, la table de défauts d'E/S pour les défauts générés par les modules d'E/S (y compris les contrôleurs d'E/S) et la table de défauts automate pour les défauts internes à l'automate. Les informations figurant dans cette annexe vous permettront d'interpréter les structures de messages en lisant ces tables de défauts. Les deux tables contiennent des informations similaires.

- La table de défauts automate contient :
  - L'emplacement du défaut.
  - La description du défaut.
  - La date et l'heure du défaut.
- La table de défauts des E/S contient :
  - L'emplacement du défaut.
  - L'adresse de référence.
  - La catégorie de défaut.
  - Le type de défaut.
  - La date et l'heure du défaut.

Table de défauts automate

Accéder à la table de défauts automate par le logiciel de programmation. Pour des informations concernant l'accession aux tables de défauts, se référer à l'assistance en ligne, *Manuel de l'utilisateur du logiciel de programmation Logicmaster 90 Série 90-30/20/Micro*, GFK-0466.

Les codes de défaut corespondant à une incohérence de configuration du système sont les suivants (Toutes les données sont en hexadécimal).



Le tableau suivant identifie chaque champ du code correspondant au défaut d'incohérence de la configuration du système affiché ci-dessus :

Champ	Valeur	Description
Long/Court	00	Ce défaut contient 8 octets de données supplémentaires du défaut
Châssis	00	Châssis principal (châssis 0)
Emplacement	03	Emplacement 3
Tâche	44	
Groupe de défauts	0B	Défaut d'incohérence de configuration du système
Action du défaut	03	Défaut FATAL
Code d'erreur	01	

Les paragraphes suivants décrivent chaque champ d'entrée du défaut, avec des tables indiquant la plage de valeurs que chaque champ peut prendre.

Indicateur long/court

Cet octet indique si le défaut contient 8 octets ou 24 octets de données supplémentaires concernant le défaut.

Type	Code	Données supplémentaires sur le défaut
Court	00	8 octets
Long	01	24 octets

Réserve

Ces six octets sont des octets de remplissage, utilisés pour mettre l'entrée de la table de défauts automate à la même longueur que l'entrée de la table de défauts des E/S.

Châssis

Le numéro de châssis va de 1 à 7. Zéro correspond au châssis principal, contenant l'UC. Les châssis 1 à 7 sont des châssis d'extension connectés au châssis principal par un câble d'extension.

Emplacement

Le numéro d'emplacement va de 0 à 9. L'UC automate occupe toujours l'emplacement 1 du châssis principal (châssis 0).

Tâche

Le numéro de tâche va de 1 à +65.535. Parfois, ce champ donne des indications supplémentaires sur le défaut, mais il peut souvent être ignoré.

Groupe de défauts automate

Le groupe de défauts est la plus haute classification d'un défaut. Il identifie la catégorie générale du défaut. Le texte de description du défaut affiché par le logiciel Logicmaster 90-30/20/Micro est basé sur le groupe de défauts et sur les codes d'erreurs.

La Table B-1 liste les groupes de défauts possibles dans la table de défauts automate.

Le dernier groupe de défauts non-masquable, ***Codes de défauts automate supplémentaires*** , englobe les défauts système dont l'automate ignore les codes. Tous les codes de défaut non reconnus appartiennent à ce groupe.

Table B-1. Groupes de défauts automate

Numéro de groupe		Nom de groupe	Action du défaut
Décimal	Hexadécimal		
1	1	Perte de châssis ou châssis manquant	Fatal
4	4	Perte de module optionnel ou module optionnel manquant	Diagnostic
5	5	Addition de châssis ou châssis en trop	Diagnostic
8	8	Addition de module ou module en trop	Diagnostic
11	B	Incohérence de configuration du système	Fatal
12	C	Erreur de bus système	Diagnostic
13	D	Défaillance matérielle de l'UC automate	Fatal
14	E	Défaillance matérielle de module non-fatale	Diagnostic
16	10	Défaillance du logiciel du module optionnel	Diagnostic
17	11	Défaillance de checksum du bloc programme	Fatal
18	12	Signal de pile faible	Diagnostic
19	13	Temps de balayage constant dépassé	Diagnostic
20	14	Table de défauts système automate pleine	Diagnostic
21	15	Table de défauts d'E/S pleine	Diagnostic
22	16	Défaut de l'application utilisateur	Diagnostic
–	–	Codes de défauts automate supplémentaires	Comme spécifié
128	80	Défaillance du bus système	Fatal
129	81	Pas de programme utilisateur à la mise sous tension	Informationnel
130	82	RAM utilisateur corrompue détectée	Fatal
132	84	défaillance d'accès par mot de passe	Informationnel
135	87	Défaillance du logiciel de l'UC automate	Fatal
137	89	Défaillance stockage-séquence automate	Fatal

Actions associées aux défauts

Chaque défaut produit l'une des trois actions ci-dessous. Ces actions sont fixes sur l'API Série 90-30 et ne peuvent être modifiées par l'utilisateur.

Table B-2. Actions des défauts automate

Catégorie du défaut	Action prise par l'UC	Code
Informationnel	Enregistre le défaut dans la table de défauts	1
Diagnostic	Enregistre le défaut dans la table de défauts Etablit les références de défauts	2
Fatal	Enregistre le défaut dans la table de défauts Etablit les références de défauts Passe en mode STOP	3

Code d'erreur

Le code d'erreur décrit plus précisément le défaut. Chaque groupe de défauts possède son propre jeu de codes d'erreur. La Table B-3 montre les codes d'erreur pour le groupe d'erreur Logiciel automate (Groupe 87H).

Table B-3. Codes d'erreur d'alarme pour les défauts de logiciel de l'UC automate

Décimal	Hexadécimal	Nom
20	14	Mémoire programme automate corrompue
39	27	Mémoire programme automate corrompue
82	52	Défaillance de la communication de fond de panier
90	5A	Coupure demandée par l'utilisateur
Tous les autres		Erreur système interne de l'UC automate

La Table B-4 montre les codes d'erreur pour tous les autres groupes de défauts.

Table B-4. Codes d'erreur pour les défauts automate

Décimal	Hexadécimal	Nom
Codes d'erreur automate pourle groupe perte de modules optionnels (4)		
44	2C	Défaillance de la réinitialisation du module optionnel
45	2D	Défaillance de la réinitialisation du module optionnel
255	FF	Défaillance de la communication du module optionnel
79	4F	Perte de carte fille
Codes d'erreur pour le groupe réinitialisation, addition ou dépassement de modules optionnels (8)		
2	2	Redémarrage du module terminé
04	4	Addition d'une carte fille
05	5	Réinitialisation d'une carte fille
	Tous les autres	Réinitialisation, addition d'un module optionnel ou module optionnel en trop
Codes d'erreur pour le groupe défaillance du logiciel de module optionnel (10 hex)		
1	1	Type de carte non supporté
2	2	COMREQ – boîte à lettres de messages sortants pleine
3	3	COMREQ – boîte à lettres de réponse pleine
5	5	Communication du fond de panier avec l'API ; Requête perdue
11	B	Erreur de ressource (allocation, débordement de table, etc.)
13	D	Erreur de programme utilisateur
401	191	Logiciel de module corrompu ; Rechargement nécessaire
Codes d'erreur pour le groupe incohérence de configuration du système (B hex)		
8	8	Incohérence extension analogique
10	A	Fonction non supportée
23	17	Le programme dépasse les limites mémoire
58	3A	Incohérence de carte fille
Codes d'erreur pour le groupe erreur de bus système (C hex)		
Tous les autres		Erreur de bus système
Codes d'erreur pour le groupe checksum du bloc programme (11 hex)		
3	3	Défaillance de checksum du programme ou du bloc programme
Code d'erreur signal pile faible		
0	0	Pile faible sur l'UC automate ou sur un autre module
1	1	Pile faible sur l'UC automate ou sur un autre module
Codes d'erreur pour le groupe défaut d'application utilisateur (16 hex)		
2	2	Expiration du temporisateur chien de garde automate
5	5	COMREQ – Mode ATTENTE indisponible pour cette commande
6	6	COMREQ – Identifieur de tâche incorrect
7	7	Débordement de la pile application
Codes d'erreur pour le groupe défaillance du bus système (80 hex)		
1	1	Système d'exploitation
Codes d'erreur pour le groupe RAM utilisateur corrompue à la mise sous tension (82 hex)		
1	1	RAM utilisateur corrompue à la mise sous tension
2	2	Instruction Booléenne illégale détectée
3	3	PLC_ISCP_PC_OVERFLOW
4	4	PRG_SYNTAX_ERR
Codes d'erreur pour les défauts matériels de l'UC automate (D hex)		
Tous les codes		Défaillance matérielle de l'UC automate

Données supplémentaires sur les défauts

Ce champ contient des détails sur les défauts, dont voici un exemple :

Groupe RAM utilisateur corrompue :

Quatre des codes d'erreur du groupe incohérence de configuration système donnent des indications supplémentaires sur les défauts :

Table B-5. Données de défauts automate - Instruction Booléenne illégale détectée

Données supplémentaires sur le défaut	Incohérence avec le numéro du modèle
[0]	Contenu du registre de défauts
[1]	Instruction incorrecte
[2,3]	Compteur programme ISCP
[4,5]	Numéro de fonction

Pour une défaillance de RAM dans l'UC automate (un des défauts rapporté comme défaillance matérielle de l'UC automate), l'adresse de la défaillance est stockée dans les quatre premiers octets du champ.

Défaillance matérielle UC API (défaillance RAM) :

Horodatage des défauts automate

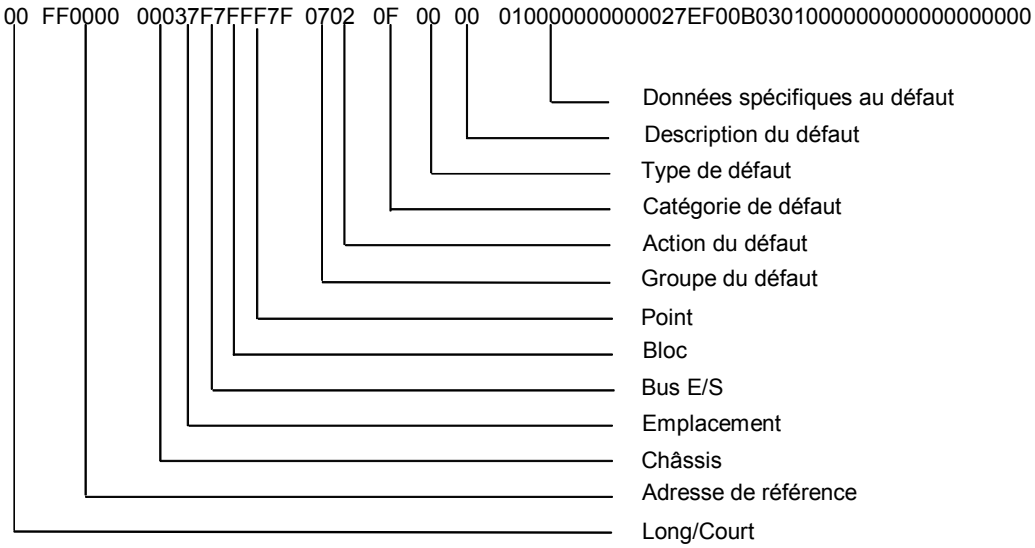
L'horodatage indique la valeur de l'horloge du système sur six octets lorsque le défaut a été enregistré par l'UC automate. (Les valeurs sont codées en format BCD).

Table B-6. Horodatage des défauts automate

Numéro d'octet	Description
1	Secondes
2	Minutes
3	Heures
4	Jour du mois
5	Mois
6	Année

Table de défauts d’ E/S

Les codes de défaut en hexadécimal correspondant à une entrée de défaut d’E/S sont :



Les paragraphes suivants décrivent chaque champ de la table des défauts d'E/S, avec des tables indiquant la plage de valeurs que chaque champ peut prendre.

indicateur long/court

Cet octet indique si le défaut contient 5 octets ou 21 octets de données spécifiques sur le défaut.

Table B-7. Octet indicateur du format de la table des défauts des E/S

Type	Code	Données spécifiques aux défauts
Court	02	5 octets
Long	03	21 octets

Adresse de référence

L'adresse de référence est une adresse de trois octets contenant le type de mémoire d'E/S et l'adresse (ou le décalage) dans cette mémoire indiquant le point en défaut. Lorsqu'un défaut de bloc Genius ou de module analogique se produit, l'adresse de référence indique le premier point du bloc où le défaut s'est produit.

Table B-8. Adresse de référence des E/S

Octet	Description	Plage
0	Type de mémoire	0 – FF
1–2	Décalage	0 – 7FF

L'octet type de mémoire prend l'une des valeurs suivantes.

Table B-9. Type de référence mémoire d'E/S

Nom	Valeur (Hexadécimale)
Entrée analogique	0A
Sortie analogique	0C
Analogiques groupés	0D
Entrée logique	10 or 46
Sortie logique	12 or 48
Logiques groupés	1F

Adresse de défauts d'E/S

L'adresse du défaut d'E/S est composée de six octets contenant l'adresse des châssis, de l'emplacement, du bus, du bloc et du point d'E/S qui a généré le défaut. L'adresse du point est un mot ; toutes les autres adresses font un octet chacune. Les cinq adresses ne sont pas forcément présentes pour un défaut.

Lorsqu'une adresse de défaut d'E/S ne contient pas les cinq adresses, une valeur 7F hexadécimal apparaît dans l'adresse pour indiquer la fin de l'adresse. Par exemple, si 7F apparaît dans l'octet

bus, le défaut est un défaut de moduleet seules les adresses du châssis et de l'emplacement sont significatives.

Châssis

Le numéro de châssis va de 1 à 7. Zéro correspond au châssis principal, c'est-à-dire, celui qui contient l'UC. Les châssis 1 à 7 sont des châssis d'extension.

Emplacement

Le numéro d'emplacement va de 0 à 9. L'UC automate occupe toujours l'emplacement 1 du châssis principal
(châssis 0).

Point

Les points vont de 1 à 1024 (décimaux). Il indique quel point du bloc est en défaut lorsque le défaut est de type point.

Groupe de défauts d'E/S

Le groupe de défauts constitue la plus haute classification d'un défaut. Il identifie la catégorie générale du défaut. Le message de défaut affiché par le logiciel Logicmaster 90-30/20/Micro est basé sur le groupe de défauts et sur les codes d'erreurs.

La Table B-10 liste les groupes de défauts possibles dans la table de défauts d'E/S. Les numéros de groupe inférieurs à 80 (Hex) sont des défauts masquables.

Le dernier groupe de défauts non-masquable, ***Codes de défauts supplémentaires des E/S***, englobe les défauts d'E/S dont l'automate ignore les codes. Tous les codes de défaut de type E/S non reconnus appartiennent à ce groupe.

Table B-10. Groupes de défauts d'E/S

Numéro de groupe	Nom de groupe	Action du défaut
3	Perte ou absence de module d'E/S	Diagnostic
7	Addition ou excès de module d'E/S	Diagnostic
9	Défaut de bus d'IOC ou d'E/S	Diagnostic
A	Défaut de module d'E/S	Diagnostic
–	Codes de défauts d'E/S supplémentaires	Comme spécifié

Actions associées aux défauts d'E/S

Ces actions indiquent le comportement de l'UC automate lorsqu'un défaut se produit. La Table B-11 liste les actions de défauts possibles.

Table B-11. Table des défauts des E/S

Catégorie du défaut	Action prise par l'UC	Code
Informationnel	Enregistre le défaut dans la table des défauts	1
Diagnostic	Enregistre le défaut dans la table de défauts Etablit les références de défauts	2
Fatal	Enregistre le défaut dans la table de défauts Etablit les références de défauts Passe en mode STOP	3

Données spécifiques aux défauts d'E/S

Une table des défauts d'E/S peut contenir jusqu'à 5 octets de données spécifiques aux défauts d'E/S.

Données spécifiques aux défauts symboliques

La Table B-12 liste les données nécessaires à la configuration du circuit de bloc.

Table B-12. Données spécifiques aux défauts d'E/S

Nombre décimal	Code hexadécimal	Description
Configuration du circuit		
	1 2 3	Le circuit est une entrée à trois états Le circuit est une entrée Le circuit est une sortie

Actions des défauts pour des défauts spécifiques

Les défauts des circuits forcés/non forcés sont rapportés comme défauts informationnels. Tous les autres sont de type diagnostic ou fatals.

Les défauts d'incohérence de modèle, de type d'E/S et d'absence de module d'E/S sont rapportés dans la table des défauts automate dans le groupe incohérence de configuration du système. Ils ne sont pas rapportés dans la table des défauts des E/S.

Horodatage des défauts d'E/S

L'horodatage de six octets est la valeur de l'horloge système lorsque le défaut a été enregistré par l'UC automate. Les valeurs sont codées en format BCD.

Table B-13. Horodatage des défauts d'E/S

Numéro d'octet	Description
1	Secondes
2	Minutes
3	Heures
4	Jour du mois
5	Mois
6	Année

Annexe C

Mnémoniques des instructions

En mode Affichage/Edition de programme, vous pouvez entrer ou rechercher rapidement une instruction de programmation en tapant le caractère "et" commercial (&) suivi du mnémonique de l'instruction. Pour certaines instructions, vous pouvez également spécifier une adresse de référence ou un symbole, une étiquette ou une adresse de référence d'emplacement.

Cette annexe liste les mnémoniques des instructions de programmation pour le logiciel de programmation Logicmaster 90-30/20/Micro. L'ensemble des mnémoniques est montré dans la colonne 3 de cette table et l'entrée la plus courte que vous pouvez faire pour chaque instruction est listée dans la colonne 4.

Pendant la programmation, vous pouvez afficher à tout moment un écran d'assistance avec ces mnémoniques en appuyant simultanément sur les touches ALT et I.

Groupe de fonctions	Instruction	Mnémonique						
		Tous	INT	DINT	BIT	OCTET	MOT	REEL
Contacts	Tout contact	&CON	&CON					
	Contact normalement ouvert	&NOCON	&NOCON					
	Contact normalement fermé	&NCCON	&NCCON					
	Contact de continuité	&CONC	&CONC					
Bobines	Toute bobine	&COI	&COI					
	Bobine normalement ouverte	&NOCOI	&NOCOI					
	Bobine inversée	&NCCOI	&NCCOI					
	Bobine de transition positive	&PCOI	&PCOI					
	Bobine de transition négative	&NCOI	&NCOI					
	Bobine SET	&SL	&SL					
	Bobine RESET	&RL	&RL					
	Bobine SET rémanente	&SM	&SM					
	Bobine RESET rémanente	&RM	&RM					
	Bobine rémanente	&NOM	&NOM					
	Bobine rémanente inversée	&NCM	&NCM					
	Bobine de continuité	&COILC	&COILC					
Liaisons	Liaison horizontale	&HO	&HO					
	Liaison verticale	&VE	&VE					
Temporisateurs	Temporisateur de retard à l'activation	&ON	&ON					
	Temporisateur de temps écoulé	&TM	&TM					
	Temporisateur de retard à la désactivation	&OF	&OF					
Compteurs	Compteur	&UP	&UP					
	Décompteur	&DN	&DN					

Groupe de fonctions	Instruction	Mnémonique							
		Tous	**BCD 4 chiffres**	**INT**	**DINT**	**BIT**	**OCTET**	**MOT**	**REEL**
Mathématique	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 Base 10	&LOG							
	Logarithme néperien	&LN							
	Puissance e								
	Puissance X	&EXP							
		&EXPT							
Relationnelle	Egal à	&EQ		&EQ_I	&EQ_DI				&EQ_R
	Différent de	&NE		&NE_I	&NE_DI				&NE_R
	Supérieur à	&GT		&GT_I	&GT_DI				&GT_R
	Supérieur ou égal à	&GE		&GE_I	&GE_DI				&GE_R
	Inférieur à	&LT		&LT_I	&LT_DI				&LT_R
	Inférieur ou égal à	&LE		&LE_I	&LE_DI				&LE_R
Opérations logiques	ET	&AN						&AN_W	
	OU	&OR						&OR_W	
	OU exclusif	&XO						&XO_W	
	NON	&NOT						&NOT_W	
	Décalage de bits à gauche	&SHL						&SHL_W	
	Décalage de bits à droite	&SHR						&SHR_W	
	Rotation de bits à gauche	&ROL						&ROL_W	
	Rotation de bits à droite	&ROR						&ROR_W	
	Test de bit	&BT						&BT_W	
	Etablissement de bit	&BS						&BS_W	
	Effacement de bit	&BCL						&BCL_W	
	Position de bit	&BP						&BP_W	
	Comparaison masquée	&MCMP						&MCM_W	
Conversion	Conversion en entier	&TO_INT	&TO_INT_BCD 4						
	Conversion en entier double	&TO_DINT							
	Conversion en BCD 4 chiffres	&BCD4			&TO_REAL_DI				&BCD4_R
	Conversion en REEL	&TO_REAL						&TO_REAL_W	
	Conversion en MOT	&TO_W							
	Tronquage en entier	&TRINT							
	Tronquage en entier double	&TRDINT							

Groupe de fonctions	Instruction	Mnémonique						
		Tous	INT	DINT	BIT	OCTET	MOT	REEL
Transfertde donnée	Transfert Transfertde bloc Effacement de bloc Registre à décalage Séquenceur de bits Requête de communication	&MOV &BLKM &BLKC &SHF &BI &COMMR	&MOV_I &BLKM_I		&MOV_BI &SHF_BI		&MOV_W &BLKM_W &AR_W	&MOV_R &BLKM_R
Table	Transfert de table Recherche égal à Recherche différent de Recherche supérieur à Recherche supérieur ou égal à Recherche inférieur à Recherche inférieur ou égal à	&AR &SRCHE &SRCHN &SRCHGT &SRCHGE &SRCHLT &SRCHLE	&AR_I &SRCHE_I &SRCHN_I &SRCHGT_I &SRCHGE_I &SRCHLT_I &SRCHLE_I	&AR_DI &SRCHE_DI &SRCHN_DI &SRCHGT_DI &SRCHGE_DI &SRCHLT_DI &SRCHLE_DI	&AR_BI	&AR_BY &SRCHE_BY &SRCHN_BY &SRCHGT_BY &SRCHGE_BY &SRCHLT_BY &SRCHLE_BY	&AR_W &SRCHE_W &SRCHN_W &SRCHGT_W &SRCHGE_W &SRCHLT_W &SRCHLE_W_	
Contrôle	Appel de sous-programme Scruter E/S SER Algorithme PID – ISA Algorithme PID – IND Réinitialisation SFC Fin Commentaire Requête de service système Relais de contrôle maître Fin de relais de contrôle maître Relais de contrôle maître imbriqué Fin de contrôle maître imbriqué Saut Saut imbriqué Etiquette Etiquette imbriquée	&CA &DO &SER &PIDIS &PIDIN &SFCR &END &COMME &SV &MCR &ENDMCR &MCRN &ENDMCRN &JUMP &JUMPN &LABEL &LABELN						

Annexe D

Les raccourcis clavier

Cette annexe liste les raccourcis clavier disponibles dans l'environnement logiciel. Pour afficher ces informations sur l'écran de la console de programmation, appuyer sur ALT-K

Séquence des touches	Description	Séquence des touches	Description
Touches disponibles dans le progiciel			
ALT-A	Abandonner.	CTRL-Break	Sortir du progiciel.
ALT-C	Effacer le champ.	Esc	Zoom arrière.
ALT-M	Changer le mode de la console de programmation.	CTRL-Home	Rappel de la commande précédente
ALT-R	Changer l'état Exécution/Arrêt de l'API.	CTRL-End	Rappel de la commande suivante.
ALT-E	Inverser la zone d'état.	CTRL- ←	Curseur à gauche dans le champ.
ALT-J	Inverser la ligne de commande.	CTRL-→	Curseur à droite dans le champ.
ALT-L	Lister les fichiers du répertoire.	CTRL-D	Décrémenter l'adresse de référence.
ALT-P	Imprimer l'écran.	CTRL-U	Incrémenter l'adresse de référence.
ALT-H	Aide.	Tab	Changer/incrémenter le contenu du champ.
ALT-K	Touche Aide.	Shift-Tab	Changer/décrémenter le contenu du champ.
ALT-I	Aide sur les mnémoniques des instructions.	Enter	Accepter le contenu du champ.
ALT-N	Inverser les options d'affichage.	CTRL-E	Afficher la dernière erreur système.
ALT-T	Démarrer le mode apprentissage.	F12 ou touche - du clavier numérique	Changer l'état d'une référence logique.
ALT-Q	Arrêter le mode apprentissage.	F11 ou touche * du clavier numérique	Forcer une référence logique.
ALT-n	Fichier n de reproduction (n = 0 à 9).		
Touches disponibles avec l'éditeur de programme seulement			
ALT-B	Inverser la sonnerie de l'éditeur de texte.	+ du clavier numérique	Valider un circuit
ALT-D	Effacer un élément du circuit/Effacer un circuit.	Enter	Valider un circuit
ALT-S	Stocker un bloc vers l'API et le disque.	CTRL-PgUp	Circuit précédent.
ALT-X	Afficher le niveau de zoom.	CTRL-PgDn	Circuit suivant.
ALT-U	Mettre le disque à jour.	~	Lien horizontal.
ALT-V	Fenêtre table des variables.	|	Lien vertical.
ALT-F2	Aller à la table de références des opérandes.	Tab	Aller au champ d'opérande suivant.
Touches spéciales			
ALT-O	Forçage du mot de passe. Disponible seulement sur l'écran Mot de passe dans le logiciel de configuration.		

La carte Aide de la page suivante contient une liste des aides sur les raccourcis clavier et également une aide sur les mnémoniques des instructions pour le logiciel Logicmaster 90-30/20/Micro. Cette carte est imprimée en trois exemplaires et elle est perforée pour être facilement ôtée du manuel.

Annexe E

Utilisation des nombres à virgule flottante

Les notions à connaître relatives à l'utilisation des nombres à virgule flottante sont peu nombreuses. La première section décrit les notions générales. Se référer à la page E-5 et aux suivantes pour les instructions sur l'introduction et l'affichage des nombres à virgule flottante.

Remarque

Les possibilités de virgule flottante ne sont supportées *que* par les UC des modèles 35x et 36x, Version 9 ou supérieures et par toutes les versions d'UC 352.

Nombres à virgule flottante

Le logiciel de programmation donne la possibilité d'éditer, d'afficher, de stocker et de retrouver des nombres avec des valeurs réelles. Certaines fonctions utilisent des nombres à virgule flottante. Cependant, pour utiliser des nombres à virgule flottante avec le logiciel de programmation, vous devez avoir une UC série 35x ou 36x (voir la remarque ci-dessus). Les nombres à virgule flottante sont représentés en notation scientifique décimale, avec l'affichage de six chiffres significatifs.

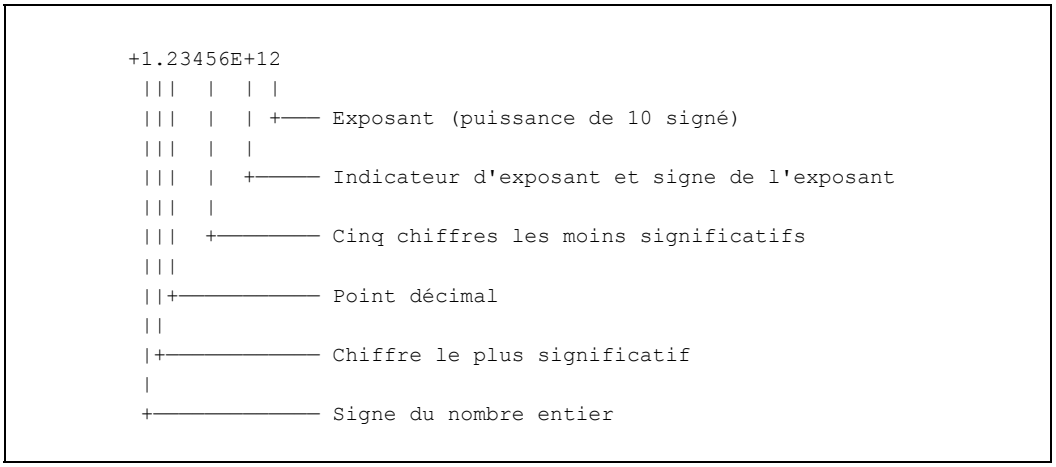
Remarque

Dans ce manuel, les termes "virgule flottante" et "réel" sont utilisés de façon interchangeable pour décrire la fonction affichage/introduction des nombres à virgule flottante du logiciel de programmation.

Le format suivant est utilisé. Pour les nombres de 9999999 à .0001, l'affichage n'a pas d'exposant et jusqu'à six ou sept chiffres significatifs. Par exemple :

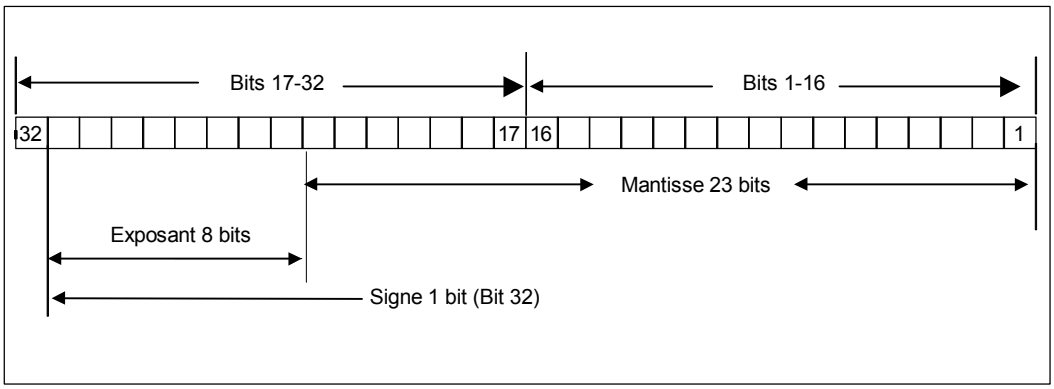
Entré	Affiché	Description
.000123456789	+.0001234567	Dix chiffres, six ou sept chiffres significatifs.
–12.345e-2	–.1234500	Sept chiffres, six ou sept chiffres significatifs.
1234	+1234.000	Sept chiffres, six ou sept chiffres significatifs.

Hors de la plage citée ci-dessus, seuls six chiffres significatifs sont affichés et l'affichage prend la forme :

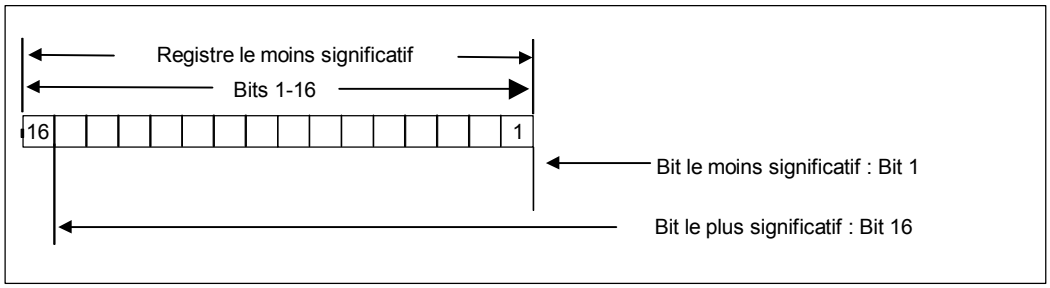


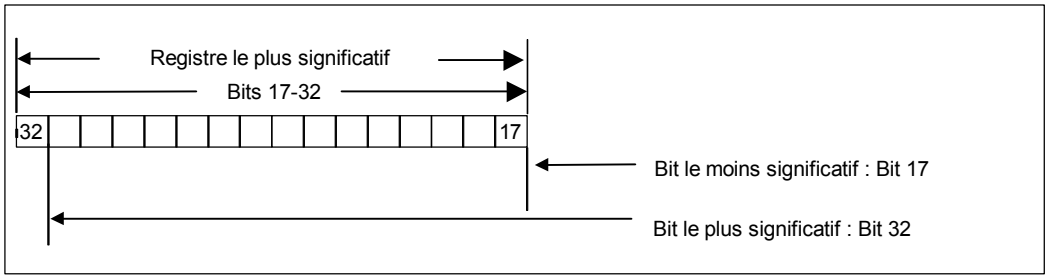
Format interne des nombres à virgule flottante

Les nombres à virgule flottante sont stockés en format IEEE standard à simple précision. Ce format nécessite 32 bits, qui se composent de deux registres automate adjacents de 16 bits. Le codage des bits est indiqué ci-dessous.



L'utilisation du registre par un simple nombre à virgule flottante est indiquée ci-dessous. Dans ce schéma, si le nombre à virgule flottante occupe les registres R5 et R6, par exemple, R5 est le registre le moins significatif et R6 est le registre le plus significatif.





Valeurs des nombres à virgule flottante

Utiliser la table suivante pour calculer la valeur du nombre à virgule flottante à partir du nombre binaire stocké dans deux registres.

Exposant (e)	Mantisse (f)	Valeur du nombre à virgule flottante
255	Différent de zéro	Pas un nombre valide (NaN).
255	0	$-1^s * \infty$
0 < e < 255	Toute valeur	$-1^s * 2^{e-127} * 1.f$
0	Différent 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 tel que E+127 est la puissance de 2 par laquelle la mantisse doit être multipliée pour former une valeur à virgule flottante.
s = le bit signe.
* = l'opérateur multiplication.

Par exemple, considérez le nombre à virgule flottante 12.5. La représentation binaire à virgule flottante IEEE du nombre est :

```
01000001 01001000 00000000 00000000
```

ou 41480000 hex. Le bit le plus significatif (le bit signe) est zéro (s=0). Les huit bits les plus significatifs suivants sont 10000010 ou 130 en décimal (e=130).

La mantisse est stockée comme nombre binaire décimal avec le point décimal précédant le plus significatif des 23 bits. Ainsi, le bit le plus significatif de la mantisse est un multiple de 2^{-1}, le bit le plus significatif suivant est un multiple de 2^{-2}, et ainsi de suite jusqu'au bit le moins significatif qui est un multiple de 2^{-23}. Les 23 bits finaux (la mantisse) sont :

```
1001000 00000000 00000000
```

La valeur de la mantisse est alors .5625 (c'est-à-dire, $2^{-1} + 2^{-4}$).

Comme e > 0 et e < 255, nous utilisons la troisième formule de la table ci-dessus :

$$\begin{aligned} \text{nombre} &= -1^s * 2^{e-127} * 1.f \\ &= -1^0 * 2^{130-127} * 1.5625 \\ &= 1 * 2^3 * 1.5625 \\ &= 8 * 1.5625 \\ &= 12.5 \end{aligned}$$

Ainsi, vous pouvez voir que la représentation binaire ci-dessus est correcte.

La plage de nombres qui peuvent être stockés dans ce format est de ± 1.401298E–45 à ± 3.402823E+38 et le nombre zéro.

Introduction et affichage des nombres à virgule flottante

Dans la mantisse, jusqu'à six ou sept chiffres significatifs de précision peuvent être entrés et stockés ; cependant, le logiciel de programmation n'affichera que les six premiers chiffres. La mantisse peut être précédée d'un signe positif ou négatif. Si aucun signe n'est entré, le nombre à virgule flottante est présumé positif.

Si un exposant est entré, il doit être précédé d'une lettre **E** ou **e**, et la mantisse doit contenir un point décimal afin d'éviter de le confondre avec un nombre hexadécimal. L'exposant peut être précédé d'un signe ; mais, si aucun n'est fourni, il est présumé positif. Si aucun exposant n'est entré, il est présumé zéro. Aucun espace n'est autorisé dans un nombre à virgule flottante.

Pour une utilisation plus aisée, plusieurs formats sont acceptés dans la ligne de commande et dans le champ d'entrée de donnée. Ces formats incluent un entier, un nombre décimal ou un nombre décimal suivi d'un exposant. Ces nombres sont convertis en forme standard pour l'affichage lorsque l'utilisateur a entré les données et appuyé sur la touche **Enter**.

Des exemples d'entrées de nombres à virgule flottante valides et leur affichage normalisé sont montrés ci-dessous.

Entré	Affiché
250	+250,0000
+4	+4.000000
–2383019	–2383019.
34.	+34.00000
–.0036209	–.003620900
12.E+9	+1.20000E+10
–.0004E–11	–4.00000E–15
731.0388	+731.0388
99.20003e–29	+9.92000E–28

Des exemples d'entrées de nombres à virgule flottante invalides sont montrés ci-dessous.

Entrée invalide	Explication
–433E23	Point décimal manquant
10e-19	Point décimal manquant
10.e19	La mantisse ne peut pas contenir d'espaces entre les chiffres ou les caractères. Ceci est accepté en tant que 10.e0 et un message d'erreur est affiché.
4.1e19	L'exposant ne peut pas contenir d'espaces entre les chiffres ou les caractères. Ceci est accepté en tant que 4.1.e0 et un message d'erreur est affiché.

Erreurs avec les nombres et les opérations à virgule flottante

Sur une UC 352, un débordement se produit lorsqu'un nombre supérieur à 3.402823E+38 ou inférieur à -3.402823E+38 est généré par une fonction REAL . Sur tous les autres modèles 90-30 qui supportent les opérations à virgule flottante, la plage est supérieure à 2^{16} ou inférieure à $-2^{16.}$ Lorsque votre nombre dépasse la plage, la sortie ok de la fonction est mise à "0" ; le résultat est mis en infinité positive (pour un nombre supérieur à 3.402823E+38 sur une UC 352 ou 2^{16} sur tous les autres modèles) ou négative (pour un nombre inférieur à –3.402823E+38 ou -2^{16} sur tous les autres modèles). Vous pouvez déterminer si ceci se produit en testant le sens de la sortie ok.

POS_INF	= 7F800000h	– Représentation IEEE de l'infinité positive en hexadécimal.
NEG_INF	= FF800000h	– Représentation IEEE de l'infinité négative en hexadécimal.

Remarque

Si vous utilisez des nombres à virgule flottante logiciels (tous les modèles capables d'effectuer des opérations à virgule flottante, sauf l'UC 352), les nombres sont arrondis de zéro (0) à ±1.175494E–38.

Si les valeurs infinies produites par débordement sont utilisées comme opérandes pour d'autres fonctions REAL, elles peuvent provoquer des résultats indéfinis. Ce résultat indéfini est appelé NaN (Pas un Nombre). Par exemple, le résultat de l'addition d'une infinité positive avec une infinité négative est indéfini. Lorsque la fonction ADD_REAL est invoquée avec une infinité positive et une infinité négative comme opérandes, elles produisent un NaN comme résultat.

Sur une UC 352, chaque fonction REAL capable de produire un NaN produit un NaN spécialisé qui identifie la fonction :

NaN_SW	= FFFFFFFFh	– NaN à virgule flottante logiciel
NaN_ADD.	= 7F81FFFFh	– Valeur de l'erreur d'addition de réels en hexadécimal.
NaN_SUB	= 7F81FFFFh	– Valeur de l'erreur de soustraction de réels en hexadécimal.
NaN_MUL	= 7F82FFFFh	– Valeur de l'erreur de multiplication de réels en hexadécimal.
NaN_DIV	= 7F83FFFFh	– Valeur de l'erreur de division de réels en hexadécimal.
NaN_SQRT	= 7F84FFFFh	– Valeur de l'erreur de racine carrée de réel en hexadécimal.
NaN_LOG	= 7F85FFFFh	– Valeur de l'erreur de logarithme de réel en hexadécimal.
NaN_POW0	= 7F86FFFFh	– Valeur de l'erreur d'exposant de réel en hexadécimal.
NaN_SIN	= 7F87FFFFh	– Valeur de l'erreur de sinus de réel en hexadécimal.
NaN_COS	= 7F88FFFFh	– Valeur de l'erreur de cosinus de réel en hexadécimal.
NaN_TAN	= 7F89FFFFh	– Valeur de l'erreur de tangente de réel en hexadécimal.
NaN_ASIN	= 7F8AFFFFh	– Valeur de l'erreur de sinus inverse de réel en hexadécimal.
NaN_ACOS	= 7F8BFFFFh	– Valeur de l'erreur de cosinus inverse de réel en hexadécimal.
NaN_BCD	= 7F8CFFFFh	– Erreur de BCD 4 chiffres en réel.
REAL_INDEF	= FFC00000h	– Réel indéfini, erreur de division de 0 par 0.

Toutes les autres UC qui supportent les opérations à virgule flottante produiront une sortie NaN : FFFF FFFF.

Lorsqu'un résultat NaN est transmis dans une autre fonction, il passe dans le résultat. Par exemple, si un NaN_ADD est le premier opérande de la fonction SUB_REAL, le résultat de SUB_REAL st NaN_ADD. Si les deux opérandes d'une fonction sont des NaNs, le premier opérande passera au travers. A cause de cette caractéristique de propagation de NaNs dans les fonctions, vous pouvez identifier la fonction d'où vient le NaN.

Remarque

Pour NaN, la sortie ok est à "0" (désexcitée).

La table suivante montre le passage du flux d'énergie lors de manipulation de nombres vus comme des infinités pour des opérations comme Additionner, Multiplier, etc. Comme montré précédemment, les sorties qui dépassent les limites positive ou négative sont vues comme étant POS_INF ou NEG_INF respectivement.

Table E-1. Cas général du flux d'énergie pour les opérations à virgule flottante

Opération	Entrée 1	Entrée 2	Sortie	Flux d'énergie
Toutes	Nombre	Nombre	Infinité positive ou négative	Non
Toutes sauf la division	Infinité	Nombre	Infinité	Oui
Toutes	Nombre	Infinité	Infinité	Oui
Division	Infinité	Nombre	Infinité	Non
Toutes	Nombre	Nombre	NaN	Non

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Index