

SIEMENS

SIMATIC

Logiciel standard pour S7-300 et S7-400 Fonctions standard 2^{ème} partie

Manuel de référence

Avant-propos, Sommaire

Fonctions de combinaison
de bits **1**

Fonctions de table **2**

Fonctions de décalage **3**

Fonction et bloc fonctionnel
de transfert **4**

Fonction et blocs fonction-
nels de temporisation **5**

Fonctions et bloc fonctionnel
de conversion **6**

Fonction arithmétique sur
nombres à virgule flottante **7**

Blocs fonctionnels de
comparaison **8**

Glossaire, Index

03/2000

3^{ème} édition

Informations relatives à la sécurité

Ce manuel donne des consignes que vous devez respecter pour votre propre sécurité ainsi que pour éviter des dommages matériels. Elles sont mises en évidence par un triangle d'avertissement et sont présentées, selon le risque encouru, de la façon suivante :



Danger

signifie que la non-application des mesures de sécurité appropriées **conduit** à la mort, à des lésions corporelles graves ou à un dommage matériel important.



Attention

signifie que la non-application des mesures de sécurité appropriées peut conduire à la mort, à des lésions corporelles graves ou à un dommage matériel important.



Avertissement

signifie que la non-application des mesures de sécurité appropriées peut conduire à des lésions corporelles légères ou à un dommage matériel.

Nota

doit vous rendre tout particulièrement attentif à des informations importantes sur le produit, aux manipulations à effectuer avec le produit ou à la partie de la documentation correspondante.

Personnel qualifié

La mise en service et l'utilisation du produit ne doivent être effectuées que conformément au manuel.

Seules des **personnes qualifiées** sont autorisées à effectuer des interventions sur l'équipement. Il s'agit de personnes qui ont l'autorisation de mettre en service, de mettre à la terre et de repérer des appareils, systèmes et circuits électriques conformément aux règles de sécurité en vigueur.

Utilisation conforme aux dispositions

Tenez compte des points suivants :



Attention

Le produit ne doit être utilisé que pour les applications spécifiées dans le catalogue ou dans la description technique, et exclusivement avec des périphériques et composants recommandés par Siemens.

Le transport, le stockage, le montage, la mise en service ainsi que l'utilisation et la maintenance adéquats du produit sont les conditions indispensables pour garantir un fonctionnement correct et sûr du produit.

Marque de fabrique

Siemens AG est un symbole de copyright de Siemens AG.

SIMATIC ®, SIMATIC NET ® et SIMATIC HMI ® sont des marques déposées de Siemens AG.

Les autres désignations figurant dans ce document peuvent être des marques dont l'utilisation par des tiers à leurs propres fins peut enfreindre les droits des propriétaires desdites marques.

Copyright © Siemens AG 1995 Tous droits réservés

Toute communication ou reproduction de ce support d'information, toute exploitation ou communication de son contenu sont interdites, sauf autorisation expresse. Tout manquement à cette règle est illicite et expose son auteur au versement de dommages et intérêts. Tous nos droits sont réservés, notamment pour le cas de la délivrance d'un brevet ou celui de l'enregistrement d'un modèle d'utilité.

Siemens AG
Bereich Automatisierungs- und Antriebstechnik
Geschäftsgebiet Industrie-Automatisierungssysteme
Postfach 4848, D- 90327 Nuernberg

Siemens Aktiengesellschaft

Exclusion de responsabilité

Nous avons vérifié la conformité du contenu du présent manuel avec le matériel et le logiciel qui y sont décrits. Or des divergences n'étant pas exclues, nous ne pouvons pas nous porter garants pour la conformité intégrale. Si l'usage de ce manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition. Veuillez nous faire part de vos suggestions.

© Siemens AG 1995
Sous réserve de modifications.

6ES7811-4AA0-0CX0



Avant-propos

Objet du manuel	Ce manuel décrit les fonctions et blocs fonctionnels S7 dans le langage de programmation schéma à contacts (CONT) et en fournit des exemples. Avec ces fonctions (FC) et blocs fonctionnels (FB), vous pouvez programmer l'automate programmable S7-300/S7-400 (AP). Ce manuel vous fournit les informations nécessaires relatives à chaque fonction et bloc fonctionnel.
Où se trouvent les fonctions S7 ?	Les fonctions et blocs fonctionnels décrits dans ce manuel sont enregistrés dans la bibliothèque standard de STEP 7. Le gestionnaire de fichiers de STEP 7 vous permet de copier les fonctions et blocs fonctionnels dont vous avez besoin dans le répertoire de votre programme. Assurez-vous tout d'abord que les FC ou FB que vous désirez copier de la bibliothèque ne portent pas le même numéro que ceux se trouvant dans votre programme. Si des fonctions ou blocs fonctionnels portant le même numéro s'y trouvent déjà, vous devez soit renommer ces derniers soit ceux que vous désirez copier.
A qui s'adresse ce manuel ?	Ce manuel est destiné aux ingénieurs, programmeurs et au personnel chargé de la maintenance possédant une connaissance générale des automates programmables.
Contenu du manuel	Chaque chapitre de ce manuel traite une famille de fonctions et blocs fonctionnels : <ul style="list-style-type: none">• Fonctions de combinaison de bits (Chapitre 1)• Fonctions de table (Chapitre 2)• Fonctions de décalage (Chapitre 3)• Fonction et bloc fonctionnel de transfert (Chapitre 4)• Fonction et blocs fonctionnels de temporisation (Chapitre 5)• Fonctions et bloc fonctionnel de conversion (Chapitre 6)• Fonction arithmétique sur nombres à virgule flottante (Chapitre 7)• Blocs fonctionnels de comparaison (Chapitre 8)• Le glossaire contient une liste alphabétique de termes indispensables pour la programmation à l'aide des schémas à contacts.

Chaque chapitre décrit les fonctions (FC) et blocs fonctionnels (FB) dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation. Chaque FC ou FB est désigné par son nom, son mnémonique et son numéro. Ils sont décrits à l'aide des informations suivantes :

- Description : une description du fonctionnement de base.
- Paramètres : un tableau fournit la déclaration, le type de données, les zones de mémoire valables et la description de chaque paramètre.
- Informations d'erreur : erreurs entravant l'exécution de la fonction ou du bloc fonctionnel.
- Exemple : une figure montre la représentation graphique de la fonction ou du bloc fonctionnel avec des exemples de paramètres et les résultats après l'exécution.

Synoptique de la documentation de STEP 7

Ce manuel est une partie de la documentation STEP 7 se composant des manuels suivants :

Manuel	Thème
STEP 7 Getting Started	Ce Getting Started constitue une introduction très simple à la méthodologie de configuration et de programmation d'un automate S7-300/S7-400. Il s'adresse tout particulièrement aux utilisateurs ne connaissant pas les automates programmables S7.
Programmer avec STEP 7 Manuel	Ce manuel présente les connaissances de base sur l'organisation du système d'exploitation et d'un programme utilisateur d'une CPU S7. Il est conseillé aux nouveaux utilisateurs des S7-300/S7-400 de l'utiliser pour avoir une vue d'ensemble de la méthodologie de programmation et pour concevoir, ensuite, leur programme utilisateur.
Logiciel système pour SIMATIC S7-300/400 Fonctions standard et fonctions système Manuel de référence	Les CPU S7 disposent de blocs d'organisation et de fonctions système intégrés dont vous pouvez vous servir lors de la programmation. Ce manuel présente une vue d'ensemble des fonctions système, blocs d'organisation et fonctions standard chargeables disponibles dans S7, ainsi que – comme informations de référence – des descriptions d'interface détaillées pour leur utilisation dans le programme utilisateur.
Configuration matérielle et communication dans STEP 7 Manuel	Ce manuel STEP 7 explique le principe d'utilisation et les fonctions du logiciel d'automatisation STEP 7. Que vous soyez un utilisateur débutant de STEP 7 ou que vous connaissiez bien STEP 5, il vous donne une vue d'ensemble sur la marche à suivre pour la configuration, la programmation et la mise en œuvre d'un automate S7-300/S7-400. Vous pouvez, lors de l'utilisation du logiciel, accéder de manière sélective à l'aide en ligne qui répondra à vos questions précises sur le logiciel.
STEP 7 Pour une transition facile de S5 à S7 Manuel	Vous aurez besoin de ce manuel si vous avez l'intention de convertir des programmes STEP 5 existants afin de les exécuter dans des CPU S7. Ce guide vous donne une vue d'ensemble du mode de fonctionnement et de l'utilisation du convertisseur ; vous trouverez des informations détaillées sur l'utilisation des fonctions du convertisseur dans l'aide en ligne. Cette dernière contient également la description d'interface des fonctions S7 converties disponibles.

Manuel	Thème
LIST, CONT, SCL¹ Manuels de référence	<p>Les manuels concernant les progiciels de langage LIST, CONT et SCL (Sequential Control Language) contiennent aussi bien des instructions pour l'utilisateur que la description du langage. Vous n'avez besoin, pour la programmation d'un S7-300/400, que de l'un de ces langages, mais pouvez les mélanger à l'intérieur d'un projet si besoin est. Il est conseillé, lors de la première utilisation des langages, de se familiariser avec la méthodologie de la création de programmes à l'aide du manuel.</p> <p>Dans le logiciel, vous pouvez appeler l'aide en ligne qui répondra à vos questions détaillées sur l'utilisation des éditeurs et compilateurs associés.</p>
S7-GRAPH¹, S7-HiGraph¹, CFC¹ Manuels	<p>Les langages S7-GRAPH, S7-HiGraph et CFC (Continuous Function Chart) offrent des possibilités supplémentaires pour la réalisation de commandes séquentielles, de graphes d'état ou d'interconnexions graphiques de blocs. Ces manuels contiennent aussi bien des instructions pour l'utilisateur que la description du langage. Il est conseillé, lors de la première utilisation de ces langages, de se familiariser avec la méthodologie de la création de programmes à l'aide du manuel.</p> <p>Dans le logiciel, vous pouvez appeler l'aide en ligne (excepté pour HiGraph) qui répondra à vos questions détaillées sur l'utilisation des éditeurs et compilateurs associés.</p>

¹ Progiciel optionnel pour le logiciel système de S7-300/S7-400

Autres manuels

Vous trouverez la description des différents CPU et modules S7-300 et S7-400 ainsi que des opérations des CPU dans les manuels suivants :

- Pour l'automate programmable S7-300 : Installation et configuration – Caractéristiques des CPU, Caractéristiques des modules et Liste des opérations
- Pour l'automate programmable S7-400 : Installation et configuration – Caractéristiques des CPU, Caractéristiques des modules et Liste des opérations

Vous trouverez des informations supplémentaires dans l'aide en ligne.

Assistance supplémentaire

N'hésitez pas à contacter votre agence Siemens si vous avez des questions restées sans réponse dans le manuel ou dans les autres manuels de STEP 7 ou si vous désirez des informations sur le reste de la documentation ou sur les offres de formation.

Liste des fonctions et blocs fonctionnels

Les fonctions et blocs fonctionnels suivants sont décrits dans ce manuel.

Fonction ou bloc fonctionnel	Numéro	Page
Temporisation sous forme de retard à la montée mémorisé (TONR)	FC80	5-2
Transfert indirect de blocs (IBLKMOV)	FC81	4-2
Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET)	FC82	1-2
Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET)	FC83	1-6
Ajouter valeur dans la table (ATT)	FC84	2-2
Première valeur entrée, première sortie (FIFO)	FC85	2-4
Recherche de valeur dans table (TBL_FIND)	FC86	2-6
Dernière valeur entrée, première sortie (LIFO)	FC87	2-9
Exécuter opération sur table (TBL)	FC88	2-11
Copier valeur de la table (TBL_WRD)	FC89	2-13
Déplacer mot vers registre à décalage (WSR)	FC90	3-2
Combiner valeur logiquement avec entrée de table et mémoriser (WRD_TBL)	FC91	2-15
Déplacer bit vers registre à décalage (SHRB)	FC92	3-4
Décodeur 7 segments (SEG)	FC93	6-2
Conversion ASCII-hexa (ATH)	FC94	6-4
Conversion hexa-ASCII (HTA)	FC95	6-6
Encoder position binaire (ENCO)	FC96	6-8
Décoder position binaire (DECO)	FC97	6-9
Complément à 10 (BCDCPL)	FC98	6-10
Compter bits à 1 (BITSUM)	FC99	6-11
Remettre à zéro plage de sorties directes (RSETI)	FC100	1-4
Mettre à un plage de sorties directes (SETI)	FC101	1-8
Ecart type (DEV)	FC102	7-2
Tables de données corrélées (CDT)	FC103	2-17
Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL)	FC104	2-19
Mise à l'échelle (SCALE)	FC105	6-12
Retour de mise à l'échelle (UNSCALE)	FC106	6-14
Algorithme d'avance et de retard de phase (LEAD_LAG)	FB80	6-16
Temporisation d'alarme avec commande tout ou rien (DCAT)	FB81	5-4
Temporisation d'alarme avec commande moteur (MCAT)	FB82	5-7
Comparaison de colonne de matrice (IMC)	FB83	8-2
Comparaison séquentielle de colonne de matrice (SMC)	FB84	8-6
Barillet d'événement avec masquage (DRUM)	FB85	5-10
Rassembler/répartir données de table (PACK)	FB86	4-4

Sommaire

1	Fonctions de combinaison de bits	1-1
1.1	Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET) : FC82	1-2
1.2	Remettre à zéro plage de sorties directes (RSETI) : FC100	1-4
1.3	Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET) : FC83	1-6
1.4	Mettre à un plage de sorties directes (SETI) : FC101	1-8
2	Fonctions de table	2-1
2.1	Ajouter valeur dans table (ATT) : FC84	2-2
2.2	Première valeur entrée, première sortie (FIFO) : FC85	2-4
2.3	Recherche de valeur dans table (TBL_FIND) : FC86	2-6
2.4	Dernière valeur entrée, première sortie (LIFO) : FC87	2-9
2.5	Exécuter opération sur table (TBL) : FC88	2-11
2.6	Copier valeur de la table (TBL_WRD) : FC89	2-13
2.7	Combiner valeur logiquement avec entrée de table et mémoriser (WRD_TBL) : FC91	2-15
2.8	Tables de données corrélées (CDT) : FC103	2-17
2.9	Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL) : FC104	2-19
3	Fonctions de décalage	3-1
3.1	Déplacer mot vers registre à décalage (WSR) : FC90	3-2
3.2	Déplacer bit vers registre à décalage (SHRB) : FC92	3-4
4	Fonction et bloc fonctionnel de transfert	4-1
4.1	Transfert indirect de blocs (IBLKMOV) : FC81	4-2
4.2	Rassembler/répartir données de table (Pack) : FB86	4-4
5	Fonction et blocs fonctionnels de temporisation	5-1
5.1	Temporisation sous forme de retard à la montée mémorisé (TONR) : FC80 ...	5-2
5.2	Temporisation d'alarme avec commande tout ou rien (DCAT) : FB81	5-4
5.3	Temporisation d'alarme avec commande moteur (MCAT) : FB82	5-7
5.4	Barillet d'événement avec masquage (DRUM) : FB85	5-10

6	Fonctions et bloc fonctionnel de conversion	6-1
6.1	Décodeur 7 segments (SEG) : FC93	6-2
6.2	Conversion ASCII-hexa (ATH) : FC94	6-4
6.3	Conversion hexa-ASCII (HTA) : FC95	6-6
6.4	Encoder position binaire (ENCO) : FC96	6-8
6.5	Décodeur position binaire (DECO) : FC97	6-9
6.6	Complément à 10 (BCDCPL) : FC98	6-10
6.7	Compter bits à 1 (BITSUM) : FC99	6-11
6.8	Mise à l'échelle (SCALE) : FC105	6-12
6.9	Annuler la mise à l'échelle (UNSCALE) : FC106	6-14
6.10	Algorithme d'avance et de retard de phase (LEAD_LAG) : FB80	6-16
7	Fonction arithmétique sur nombres à virgule flottante	7-1
7.1	Ecart type (DEV) : FC102	7-2
8	Blocs fonctionnels de comparaison	8-1
8.1	Comparaison de colonne de matrice (IMC) : FB83	8-2
8.2	Comparaison séquentielle de colonne de matrice (SMC) : FB84	8-6
	Glossaire	Glossaire-1
	Index	Index-1

1

Fonctions de combinaison de bits

Ce chapitre décrit les fonctions de combinaison de bits (FC) dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
1.1	Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET) : FC82	1-2
1.2	Remettre à zéro plage de sorties directes (RSETI) : FC100	1-4
1.3	Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET) : FC83	1-6
1.4	Mettre à un plage de sorties directes (SETI) : FC101	1-8

1.1 Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET) : FC82

Description La fonction Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET) remet à zéro l'état de signal des bits d'une zone donnée lorsque le bit MCR est à « 1 ». Si le bit MCR est à « 0 », l'état de signal des bits n'est pas modifié. Le nombre de bits de la zone devant être remis à zéro est indiqué par le paramètre N. Le paramètre S_BIT indique le début de la zone.

Paramètres Le tableau 1-1 décrit les paramètres de la fonction RSET.

Tableau 1-1 Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (FC82) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
S_BIT	Entrée	POINTER*	E, A, M, D	Pointe sur le premier bit de la zone.
N	Entrée	INT	E, A, M, D, L, P ou constante	Nombre de bits de la zone devant être remis à zéro.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Lorsque le pointeur S_BIT renvoie à la zone de mémoire de la périphérie externe (zone P), l'état de signal des bits de la zone n'est pas modifié et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 1-1 montre le mode de fonctionnement de l'opération RSET. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée) et si le bit MCR égale 1, la fonction RSET est exécutée. Dans cet exemple, le paramètre S_BIT désigne le premier bit à l'adresse M 0.0. Le paramètre N indique que 10 bits doivent être remis à zéro. Une fois l'opération effectuée, l'état de signal des bits de la zone de M 0.0 à M 1.1 est « 0 ».

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

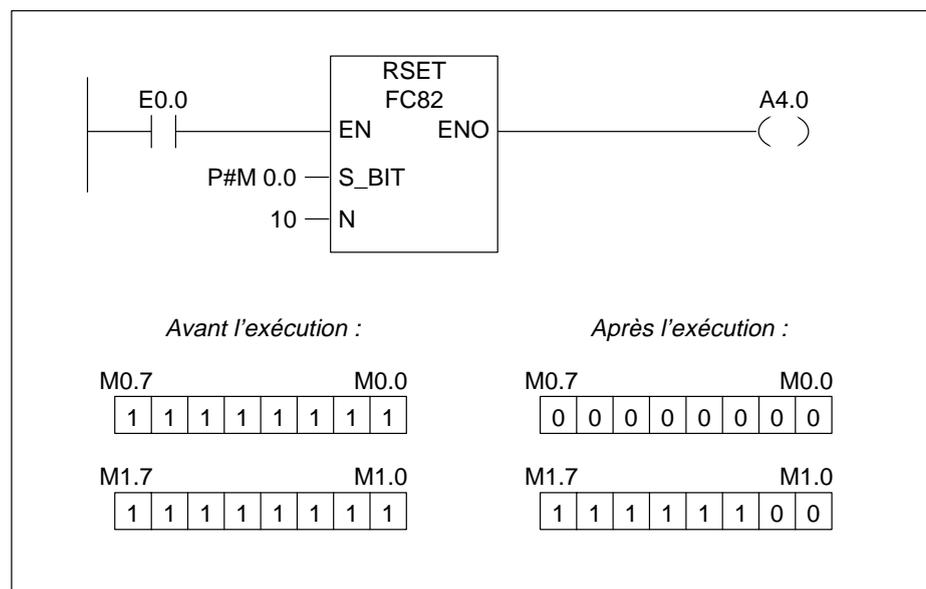


Figure 1-1 Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET)

1.2 Remettre à zéro plage de sorties directes (RSETI) : FC100

Description

La fonction Remettre à zéro plage de sorties directes (RSETI) remet à « 0 » l'état de signal des bits d'une plage d'octets donnée lorsque le bit MCR est à « 1 ». Si le bit MCR est à « 0 », l'état de signal des octets de la plage n'est pas modifié. Le paramètre S_BYTE désigne le premier octet de la plage. Le paramètre N indique la taille de la plage en précisant le nombre de bits de cette dernière. Si, par exemple, vous désirez définir une plage de 2 octets, entrez 16 (16 bits) comme valeur du paramètre N.

Nota

La valeur du paramètre N doit être un multiple de 8 (par exemple, 8, 16, 24, etc.).

Le pointeur S_BYTE doit renvoyer à la zone de mémoire de la périphérie externe (zone P). Comme l'accès à la zone de mémoire P se fait en format octet, mot ou double mot, le paramètre S_BYTE doit désigner une adresse alignée sur une limite d'octet, c'est-à-dire que le numéro de bit du pointeur doit être « 0 ».

Nota

L'état de signal des bits correspondants dans la mémoire image des sorties (zone de mémoire A) est également remis à « 0 ».

Paramètres

Le tableau 1-2 décrit les paramètres de la fonction RSETI.

Tableau 1-2 Remettre à zéro plage de sorties directes (FC100) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
S_BYTE	Entrée	POINTER*	P	Pointe sur le premier octet de la plage.
N	Entrée	INT	E, A, M, D, L, P ou constante	Taille de la plage d'octets devant être remise à « 0 », indiquée par le nombre de bits (multiples de 8, par exemple 8, 16, etc.).

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

L'état de signal des bits de la plage n'est pas modifié et l'état de signal de ENO est mis à « 0 » si l'une des situations suivantes se présente :

- Le pointeur S_BYTE désigne une autre zone de mémoire que celle de la périphérie externe (zone P).
- Le pointeur S_BYTE désigne une adresse qui n'est pas alignée sur une limite d'octet.
- La valeur du paramètre N n'est pas un multiple de 8.

Exemple

La figure 1-2 montre le mode de fonctionnement de l'opération RSETI. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée) et si le bit MCR égale 1, la fonction RSETI est exécutée. Dans cet exemple, le paramètre S_BYTE désigne le premier octet à l'adresse P 2.0. Le paramètre N indique que 16 bits (2 octets) doivent être remis à zéro. Une fois l'opération effectuée, l'état de signal des octets de la plage de P 2.0 à P 3.7 est « 0 ».

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

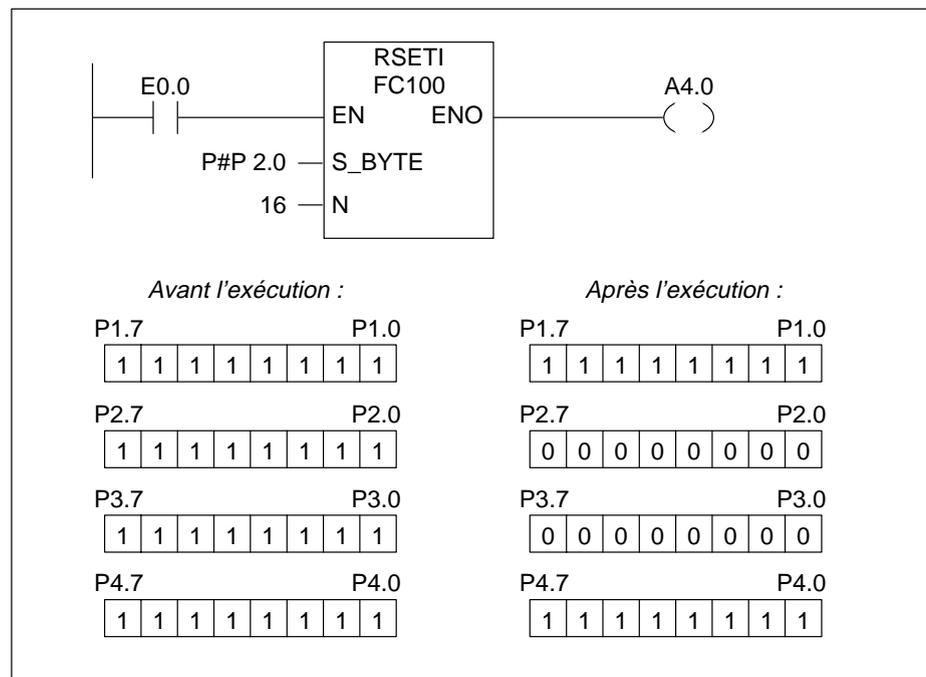


Figure 1-2 Remettre à zéro plage de sorties directes (RSETI)

1.3 Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET) : FC83

Description

La fonction Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET) met à « 1 » l'état de signal des bits d'une zone donnée lorsque le bit MCR est à « 1 ». Si le bit MCR est à « 0 », l'état de signal des bits de la zone n'est pas modifié. Le nombre de bits de la zone devant être mis à « 1 » est indiqué par le paramètre N. Le paramètre S_BIT indique le début de la zone.

Paramètres

Le tableau 1-3 décrit les paramètres de la fonction SET.

Tableau 1-3 Mettre à un zone de mémentos ou de périphérie dans la mémoire image (FC83) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
S_BIT	Entrée	POINTER*	E, A, M, D	Pointe sur le premier bit de la zone.
N	Entrée	INT	E, A, M, D, L, P ou constante	Nombre de bits de la zone devant être mis à 1.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Lorsque le pointeur S_BIT renvoie à la zone de mémoire de la périphérie externe (zone P), l'état de signal des bits de la zone n'est pas modifié et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 1-3 montre le mode de fonctionnement de l'opération SET. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée) et si le bit MCR égale 1, la fonction SET est exécutée. Dans cet exemple, le paramètre S_BIT désigne le premier bit à l'adresse M 0.0. Le paramètre N indique que 10 bits doivent être mis à 1. Une fois l'opération effectuée, l'état de signal des 10 bits de la zone de M 0.0 à M 1.1 est « 1 ».

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

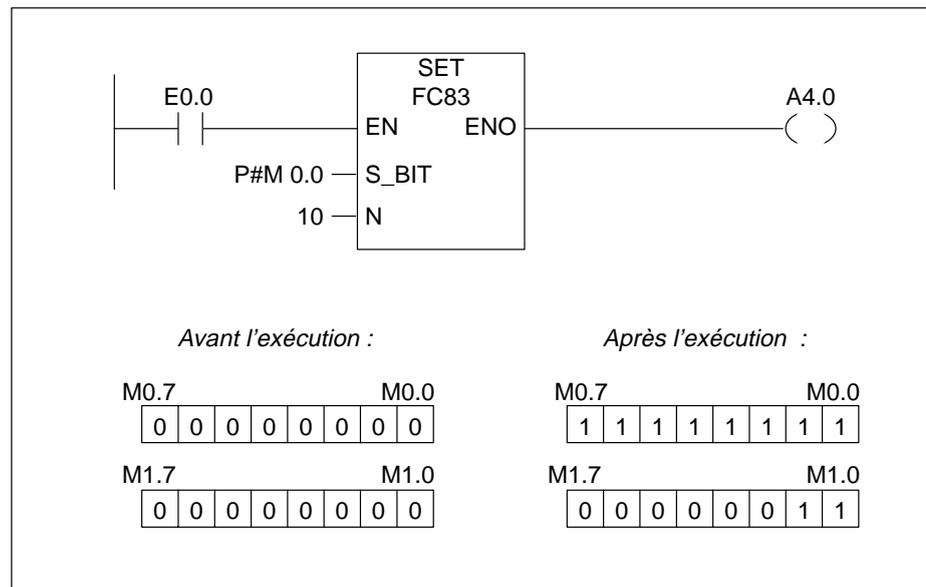


Figure 1-3 Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET)

1.4 Mettre à un plage de sorties directes (SETI) : FC101

Description

La fonction Mettre à un plage de sorties directes (SETI) met l'état de signal des bits d'une plage donnée d'octets à « 1 » lorsque le bit MCR est à « 1 ». Si le bit MCR est à « 0 », l'état de signal des octets n'est pas modifié. Le paramètre S_BYTE désigne le premier octet de la plage. Le paramètre N indique la taille de la plage en précisant le nombre de bits de cette dernière. Si, par exemple, vous désirez définir une plage de 2 octets, entrez 16 (16 bits) comme valeur du paramètre N.

Nota

La valeur du paramètre N doit être un multiple de 8 (par exemple, 8, 16, 24, etc.).

Le pointeur S_BYTE doit renvoyer à la zone de mémoire de la périphérie externe (zone P). Comme l'accès à la zone de mémoire P se fait en format octet, mot ou double mot, le paramètre S_BYTE doit désigner une adresse sur une limite d'octet, c'est-à-dire que le numéro de bit du pointeur doit être « 0 ».

Nota

L'état de signal des bits correspondants dans la mémoire image des sorties (zone de mémoire A) est également remis à « 0 ».

Paramètres

Le tableau 1-4 décrit les paramètres de la fonction SETI.

Tableau 1-4 Mettre à un plage de sorties directes (FC101) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
S_BYTE	Entrée	POINTER*	P	Pointe sur le premier octet de la plage.
N	Entrée	INT	E, A, M, D, L, P ou constante	Taille de la plage d'octets devant être mise à « 1 » indiquée par le nombre des bits (multiples de 8, par exemple 8, 16, etc.).

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

L'état de signal des bits de la plage n'est pas modifié et l'état de signal de ENO est mis à « 0 » si l'une des situations suivantes se présente :

- Le pointeur S_BYTE désigne une autre zone de mémoire que celle de la périphérie externe (zone P).
- Le pointeur S_BYTE désigne une adresse qui n'est pas alignée sur une limite d'octet.
- La valeur du paramètre N n'est pas un multiple de 8.

Exemple

La figure 1-4 montre le mode de fonctionnement de l'opération SETI. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée) et si le bit MCR égale 1, la fonction SETI est exécutée. Dans cet exemple, le paramètre S_BYTE désigne le premier octet à l'adresse P 2.0. Le paramètre N indique que 16 bits (2 octets) doivent être mis à 1. Une fois l'opération effectuée, l'état de signal des octets de la plage de P 2.0 à P 3.7 est « 1 ».

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

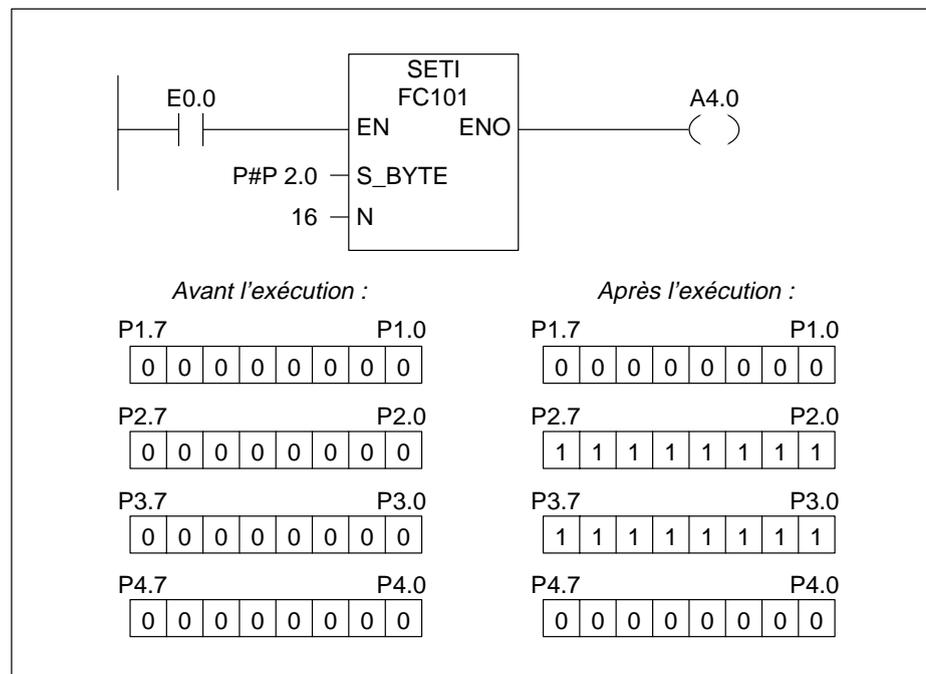


Figure 1-4 Mettre à un plage de sorties directes (SETI)

Fonctions de table

2

Ce chapitre décrit les fonctions de table dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
2.1	Ajouter valeur dans table (ATT) : FC84	2-2
2.2	Première valeur entrée, première sortie (FIFO) : FC85	2-4
2.3	Recherche de valeur dans table (TBL_FIND) : FC86	2-6
2.4	Dernière valeur entrée, première sortie (LIFO) : FC87	2-9
2.5	Exécuter opération sur table (TBL) : FC88	2-11
2.6	Copier valeur de la table (TBL_WRD) : FC89	2-13
2.7	Combiner valeur logiquement avec entrée de table et mémoriser (WRD_TBL) : FC91	2-15
2.8	Tables de données corrélées (CDT) : FC103	2-17
2.9	Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL) : FC104	2-19

2.1 Ajouter valeur dans table (ATT) : FC84

Description

La fonction Ajouter valeur dans table (ATT) ajoute le paramètre DATA comme entrée suivante dans une table et incrémente le nombre d'entrées d'une entrée. La table est composée de mots. Cette fonction vous permet d'ajouter des entrées de table qui sont utilisées par les fonctions FIFO et LIFO.

- La première entrée d'une table FIFO ou LIFO indique la longueur maximale de la table.
- La deuxième entrée d'une table indique le nombre d'entrées.
- La troisième entrée de la table contient le premier mot de données.

Nota

Vous devez initialiser les deux premières entrées lorsque vous créez une table.

Paramètres

Le tableau 2-1 décrit les paramètres de la fonction ATT.

Tableau 2-1 Ajouter valeur dans table (FC84) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
DATA	Entrée	WORD	E, A, M, D, L, P ou constante	Données devant être entrées dans la table.
TABLE	Entrée	POINTER*	E, A, M, D	Pointe sur l'adresse de début de la table FIFO ou LIFO.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Si le nombre des entrées est supérieur ou égal à la longueur de la table, les données ne sont pas ajoutées à la table et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 2-1 montre le mode de fonctionnement de l'opération ATT. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction ATT est effectuée. Dans cet exemple, le paramètre DATA est ajouté à la table comme cinquième entrée et le nombre d'entrées de la table augmente de 1 pour passer de 4 à 5.

Si la fonction est exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

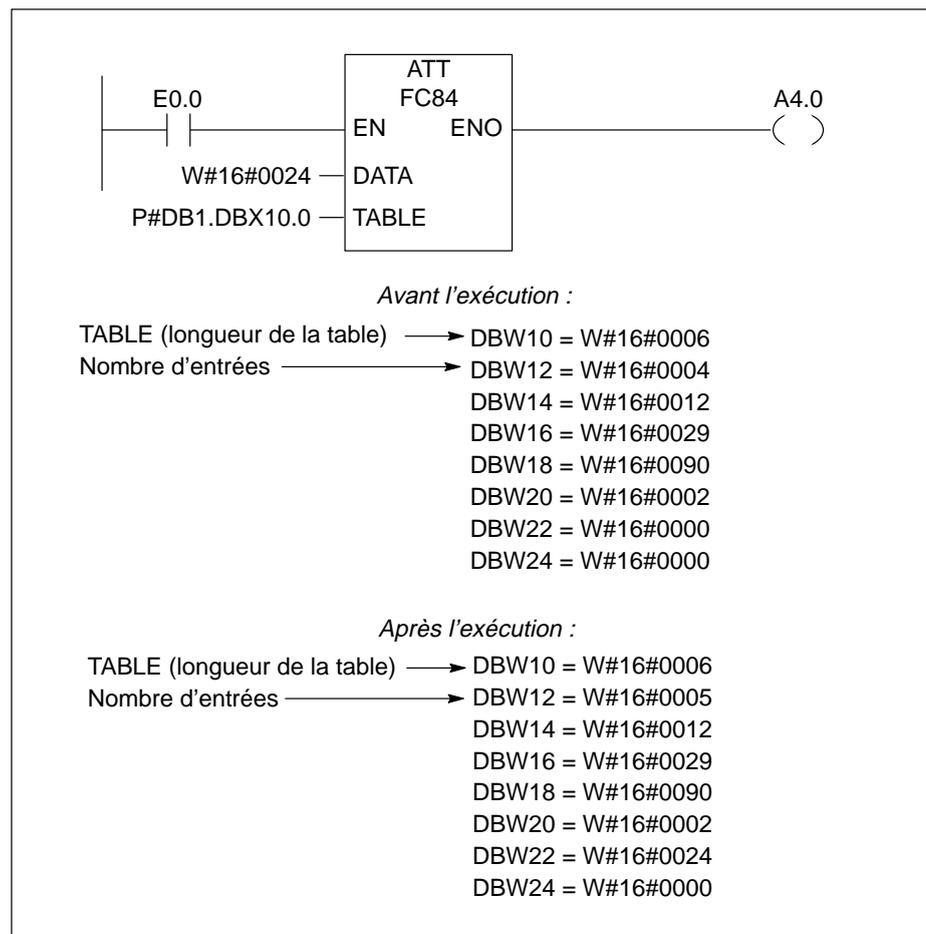


Figure 2-1 Ajouter valeur dans table (ATT)

2.2 Première valeur entrée, première sortie (FIFO) : FC85

Description

La fonction Première valeur entrée, première sortie (FIFO) renvoie la valeur la plus ancienne de la table FIFO comme valeur de fonction. Le nombre d'entrées diminue d'une entrée. Si des entrées se trouvent encore dans la table, celles-ci sont décalées vers le bas. La table FIFO est constituée de mots. Avec la fonction ATT, vous pouvez ajouter des valeurs à la table FIFO.

- La première entrée d'une table indique la longueur maximale de la table.
- La deuxième entrée de la table indique le nombre d'entrées.
- La troisième entrée de la table contient le premier mot de données.

Paramètres

Le tableau 2-2 décrit les paramètres de la fonction FIFO.

Tableau 2-2 Première valeur entrée, première sortie (FC85) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
TABLE	Entrée	POINTER*	E, A, M, D	Pointe sur l'adresse de début de la table FIFO.
RET_VAL	Sortie	WORD	E, A, M, D, L, P	L'entrée la plus ancienne de la table FIFO.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Si la table FIFO est vide (nombre d'entrées = 0), le paramètre RET_VAL n'est pas modifié et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 2-2 montre le mode de fonctionnement de l'opération FIFO. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction FIFO est exécutée. Dans cet exemple, l'entrée la plus ancienne de la table est renvoyée comme valeur de fonction (MW2). Le nombre d'entrées diminue de 1 pour passer de 5 à 4 et les entrées restantes sont décalées vers le bas de la table.

Si la fonction est exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

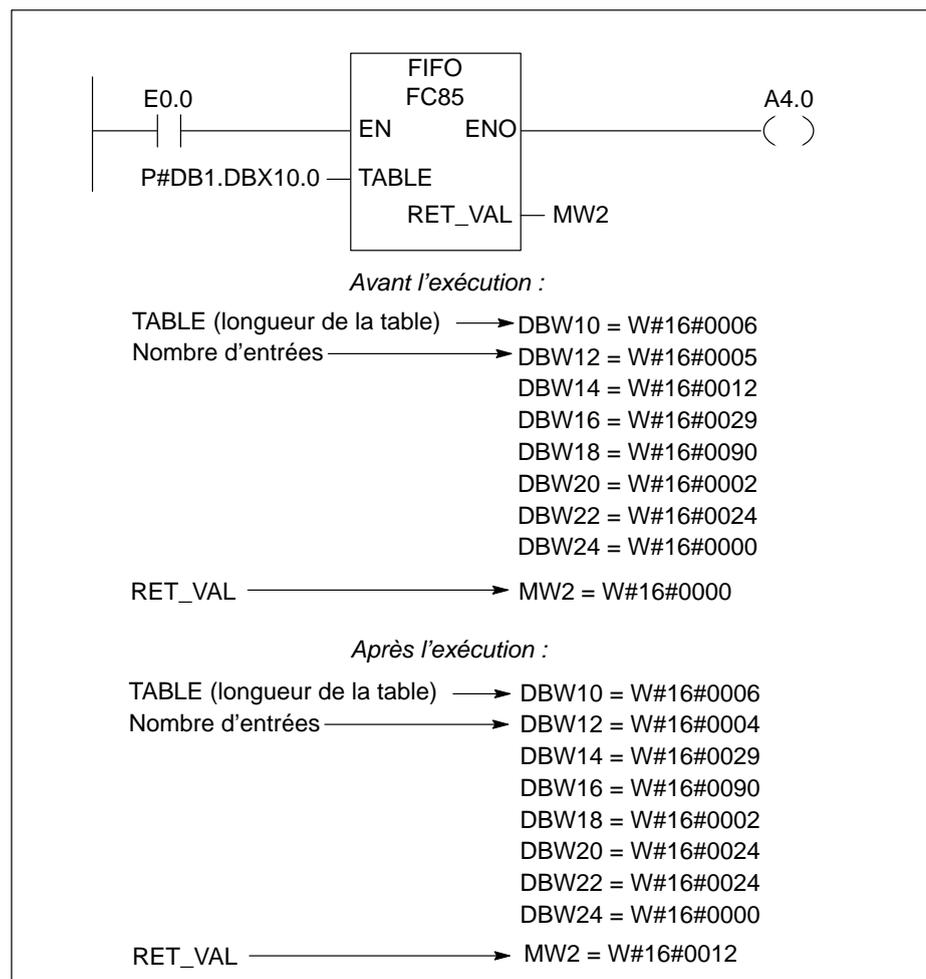


Figure 2-2 Première valeur entrée, première sortie (FIFO)

2.3 Recherche de valeur dans table (TBL_FIND) : FC86

Description

La fonction Recherche de valeur dans table (TBL_FIND) permet de rechercher dans la mémoire des profils particuliers ou des profils incompatibles. La fonction exécute une comparaison (CMD) entre le profil source (PATRN) et les entrées de la table source (SRC). Il s'agit de trouver la prochaine entrée (à partir de l'entrée indiquée par le paramètre INDX) de la table répondant aux critères de comparaison. Le numéro de cette entrée est placé dans le paramètre INDX. Si aucune entrée ne répond aux critères de comparaison, le paramètre INDX pointe au-delà de la table et la sortie de la fonction est désactivée.

- Si CMD égale 1, la fonction cherche la première valeur correspondant à la valeur PATRN.
- Si CMD égale 2, la fonction cherche la première valeur ne correspondant pas à la valeur PATRN.
- La première entrée dans la table indique la longueur maximale de la table.
- La deuxième entrée dans la table contient la première valeur de la table.

Nota

Vous devez initialiser la première entrée de la table (c'est-à-dire la longueur de la table).

Paramètres

Le tableau 2-3 décrit les paramètres de la fonction TBL_FIND.

Tableau 2-3 Recherche de valeur dans table (FC86) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
SRC	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table.
PATRN	Entrée	POINTER*	E, A, M, D	Pointe sur le profil devant être recherché.
CMD	Entrée	BYTE	E, A, M, D, L, P	Indique le type d'opération : B#16#01 = égal B#16#02 = différent
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées dans la table. Pour la fonction TBL_FIND, les types de données suivants sont admis : B#16#02 = BYTE B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
INDX	Entrée/sortie	WORD	E, A, M, D, L	Indice de la table fournissant les informations suivantes : Entrée : Entrée à partir de laquelle la recherche doit commencer Sortie : Numéro de l'entrée correspondant au profil défini

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Dans les situations décrites ci-dessous, les valeurs de la table ne sont pas modifiées. L'état de signal de ENO est mis à « 0 » et la valeur en retour est mise à l'une des valeurs suivantes (voir tableau 2-4) :

Tableau 2-4 Situations d'erreur pour FC86

RET_VAL	Explication
W#16#0008	Aucune entrée n'a répondu aux critères de recherche.
W#16#0009	Paramètre E_TYPE et/ou paramètre CMD incorrects.

Exemple

La figure 2-3 montre le mode de fonctionnement de l'opération TBL_FIND. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction TBL_FIND est exécutée. Dans cet exemple, les données de la table sont enregistrées dans des mots commençant à l'entrée désignée par le paramètre SRC, car le paramètre E_TYPE égale 4. Ces mots sont comparés au profil 5555, enregistré à l'adresse indiquée par le paramètre PATRN. Comme la valeur du paramètre CMD égale 1, la première valeur correspondant au profil est recherchée dans le paramètre SRC. Le paramètre INDX indique l'entrée à laquelle la recherche doit commencer. Une fois l'opération effectuée, le paramètre INDX indique l'entrée de la table dans laquelle une correspondance au profil a été trouvée.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

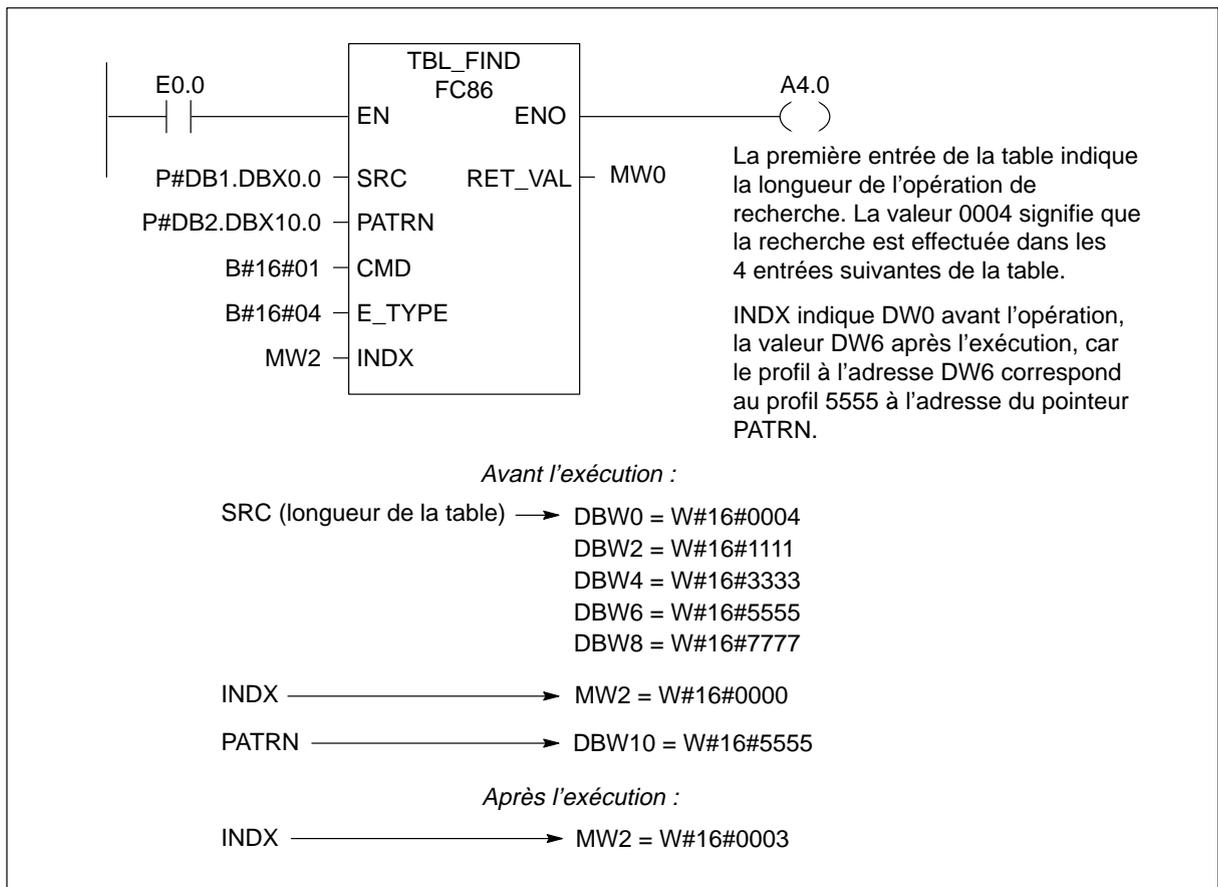


Figure 2-3 Recherche de valeur dans table (TBL_FIND)

2.4 Dernière valeur entrée, première sortie (LIFO) : FC87

Description La fonction Dernière valeur entrée, première sortie (LIFO) renvoie l'entrée la plus récente de la table LIFO comme valeur de fonction. Le nombre d'entrées diminue d'une entrée. La table LIFO se compose de mots. Avec la fonction ATT, vous pouvez entrer des valeurs dans la table LIFO.

- La première entrée d'une table indique la longueur maximale de la table.
- La deuxième entrée de la table indique le nombre d'entrées.
- La troisième entrée de la table contient le premier mot de données.

Paramètres Le tableau 2-5 décrit les paramètres de la fonction LIFO.

Tableau 2-5 Dernière valeur entrée, première sortie (FC87) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
TABLE	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table LIFO.
RET_VAL	Sortie	WORD	E, A, M, D, L, P	L'entrée la plus récente de la table LIFO.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Si la table LIFO est vide (nombre d'entrées = 0), le paramètre RET_VAL n'est pas modifié et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 2-4 montre le mode de fonctionnement de l'opération LIFO. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction LIFO est exécutée. Dans cet exemple, l'entrée la plus récente de la table LIFO est renvoyée comme valeur de fonction (MW2). Le nombre d'entrées diminue de 1 pour passer de 5 à 4.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

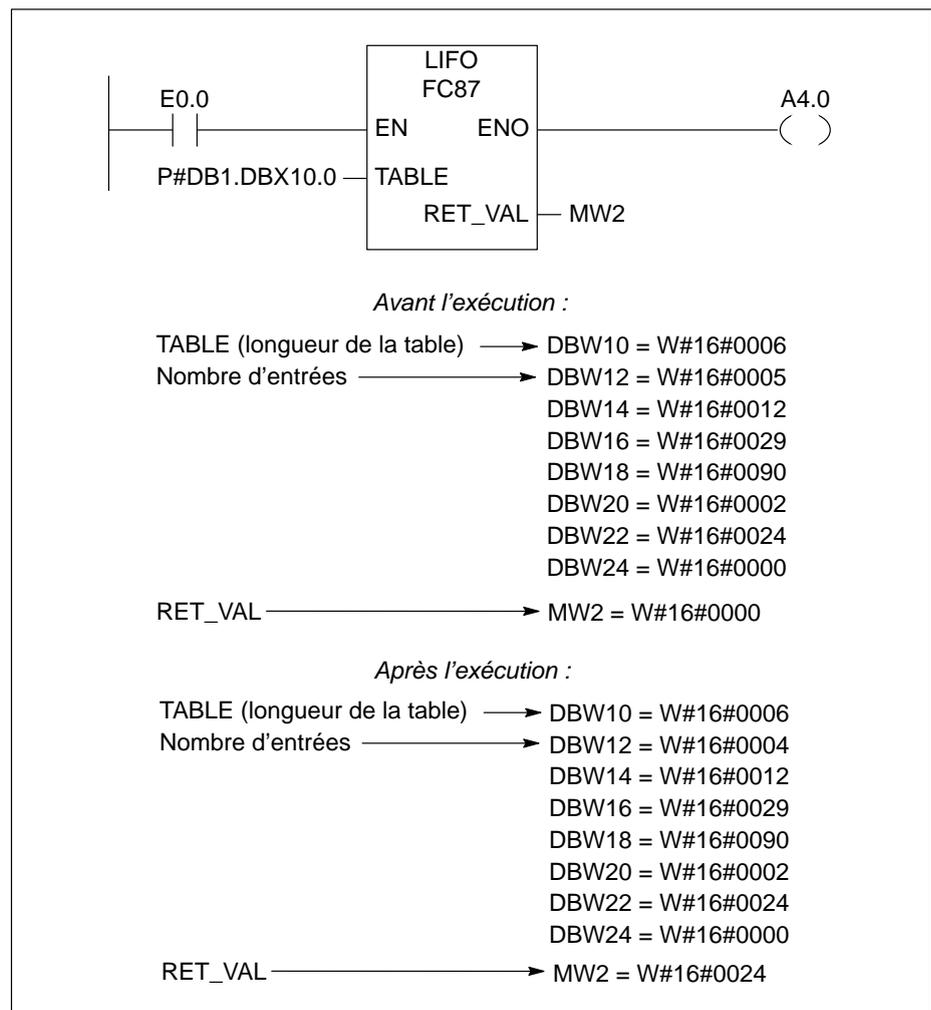


Figure 2-4 Dernière valeur entrée, première sortie (LIFO)

2.5 Exécuter opération sur table (TBL) : FC88

Description

La fonction Exécuter opération sur table (TBL) exécute l'opération (CMD) indiquée sur la table source et écrit le résultat dans la même entrée de la table.

- La première entrée dans la table indique la longueur maximale de la table.
- La deuxième entrée dans la table contient la première valeur de la table.
- Si le paramètre E_TYPE a la valeur REAL, la valeur de CMD correspondant au complément à 1 n'est pas autorisée.

Nota

Vous devez initialiser la première entrée lorsque vous créez la table.

Paramètres

Le tableau 2-6 décrit les paramètres de la fonction TBL.

Tableau 2-6 Exécuter opération sur table (FC88) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
SRC	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table.
CMD	Entrée	BYTE	E, A, M, D, L, P	Indique le type d'opération devant être effectuée. Les opérations et valeurs suivantes sont admises : B#16#03 = Complément à 1 B#16#04 = Effacer B#16#05 = NON B#16#06 = Racine carrée
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Pour la fonction TBL, les types de données suivants sont admis : B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Si CMD et E_TYPE sont incompatibles ou incorrects, les valeurs de la table restent inchangées. L'état de signal de ENO est mis à « 0 » et RET_VAL est mis à la valeur W#16#0008.

Exemple

La figure 2-5 montre le mode de fonctionnement de l'opération TBL. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction TBL est exécutée. Dans cet exemple, le paramètre SRC désigne les adresses du bloc de données qui seront traitées par l'opération. Comme le paramètre E_TYPE égale 4, les données de la table sont enregistrées dans les mots commençant à l'entrée indiquée par le paramètre SRC. Comme la valeur de CMD est 4 (Effacer), tous les mots de la table sont effacés (mis à « 0 ») lorsque l'opération TBL est exécutée. La longueur indiquée dans la première entrée de la table étant 5, les cinq entrées suivantes de la table sont effacées.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

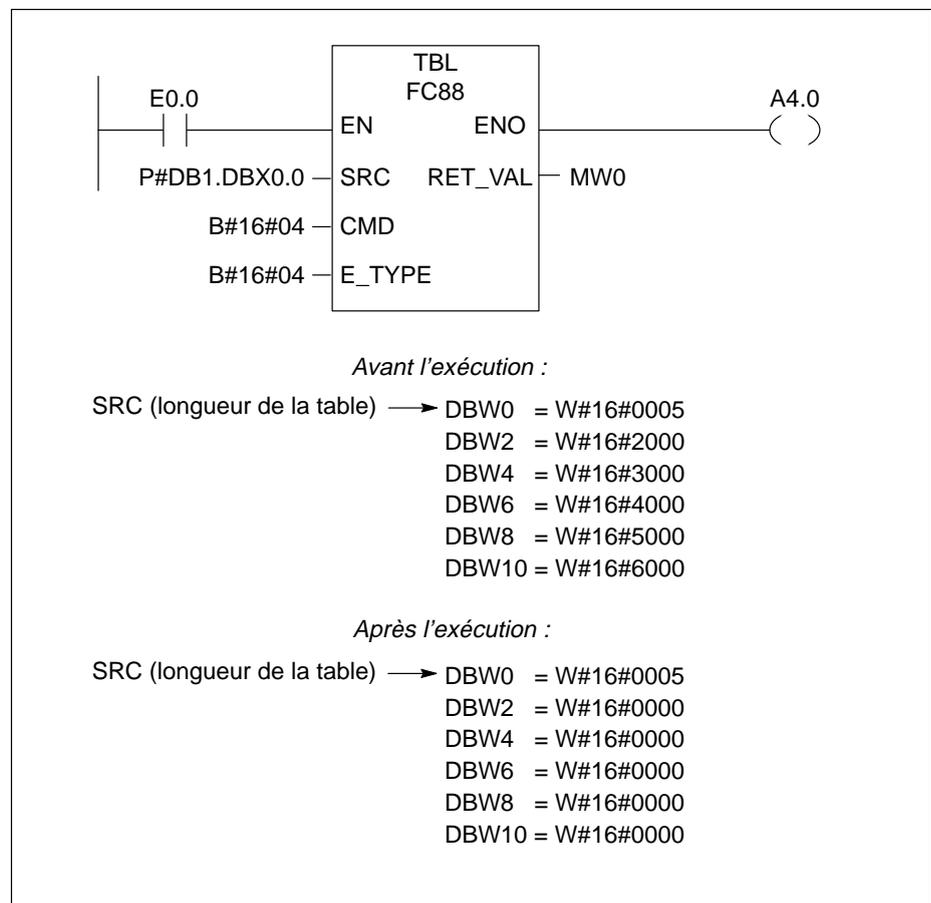


Figure 2-5 Exécuter opération sur table (TBL)

2.6 Copier valeur de la table (TBL_WRD) : FC89

Description

La fonction Copier valeur de la table (TBL_WRD) copie l'entrée indiquée par le paramètre INDX de la table SRC à l'emplacement indiqué par le paramètre DEST et incrémente la valeur de INDX, dans la mesure où la valeur de celui-ci est inférieure à la longueur indiquée dans le premier mot de la table SRC[0]. Si le paramètre INDX indique la dernière entrée de la table lorsque l'opération est appelée, le bit de sortie Q est mis à « 0 » après l'exécution de l'opération.

- La première entrée dans la table indique la longueur maximale de la table.
- La deuxième entrée dans la table contient la première valeur de la table.

Nota

Vous devez initialiser la première entrée lorsque vous créez la table.

Paramètres

Le tableau 2-7 décrit les paramètres de la fonction TBL_WRD.

Tableau 2-7 Copier valeur de la table (FC89) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
SRC	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table.
DEST	Entrée	POINTER*	E, A, M, D	Pointe sur la destination.
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Pour la fonction TBL_WRD, les types de données suivants sont admis : B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
Q	Sortie	BOOL	A, M, D, L	Donne la valeur « 0 » en retour lorsque la variable INDX contient la dernière entrée de la table à l'appel de la fonction.
INDX	Entrée/sortie	WORD	E, A, M, L	Numéro de l'entrée devant être copiée.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Dans les situations décrites au tableau 2-8, la fonction n'est pas exécutée. L'état de signal de ENO est mis à « 0 » et la valeur en retour est mise à l'une des valeurs suivantes :

Tableau 2-8 Situations d'erreur pour FC89

RET_VAL	Explication
W#16#0007	Le paramètre INDX est égal à 0.
W#16#0008	Le paramètre E_TYPE est incorrect.
W#16#0009	Le paramètre INDX pointe au-delà de la fin de la table.

Exemple

La figure 2-6 montre le mode de fonctionnement de l'opération TBL_WRD. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction TBL_WRD est exécutée. Comme le paramètre E_TYPE égale 4, les données (mots) rangées dans la table commençant à l'entrée désignée par SRC sont copiées dans l'entrée désignée par DEST. La valeur de INDX désigne l'entrée de table à copier. Une fois l'opération exécutée sans erreur, la valeur de INDX est automatiquement incrémentée d'une entrée après l'entrée copiée. Dans cet exemple, à l'appel de la fonction, la valeur de INDX ne contient pas la dernière entrée de la table ; par conséquent, le paramètre Q est mis à « 1 » après l'opération.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

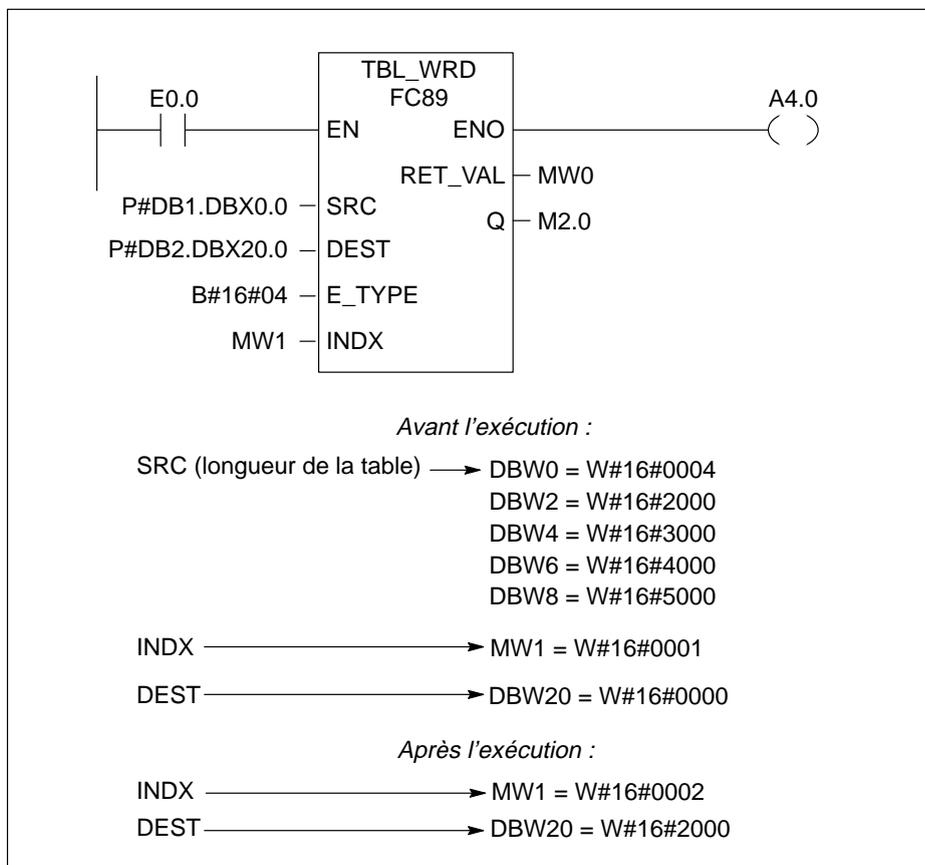


Figure 2-6 Copier valeur de la table (TBL_WRD)

2.7 Combiner valeur logiquement avec entrée de table et mémoriser (WRD_TBL) : FC91

Description

La fonction Combiner valeur logiquement avec entrée de table et mémoriser (WRD_TBL) exécute la commande indiquée (CMD) entre les données source (indiquées par SRC) et l'entrée de la table au décalage indiqué par le paramètre INDX. La fonction incrémente alors la valeur de INDX, dans la mesure où la valeur de celui-ci est inférieure à la longueur de la table.

- La première entrée dans la table indique la longueur maximale de la table.
- La deuxième entrée dans la table contient la première valeur de la table.
- Si le paramètre E_TYPE a la valeur REAL, CMD ne peut avoir que la valeur correspondant à « Copier ».

Nota

Vous devez initialiser la première entrée lorsque vous créez la table.

Paramètres

Le tableau 2-9 décrit les paramètres de la fonction WRD_TBL.

Tableau 2-9 Combiner valeur logiquement avec entrée de table et mémoriser (FC91) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
SRC	Entrée	POINTER*	E, A, M, D	Indique les données source.
TABLE	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table.
CMD	Entrée	BYTE	E, A, M, D, L, P	Indique le type d'opération devant être effectuée. Les opérations et valeurs suivantes sont admises : B#16#0E = Copier B#16#07 = Combinaison ET B#16#08 = Combinaison OU B#16#09 = Combinaison OU exclusif
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Pour la fonction WRD_TBL, les types de données suivants sont admis : B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
Q	Sortie	BOOL	A, M, D, L	Donne la valeur « 0 » en retour lorsque INDX contient le numéro de la dernière entrée de la table.
INDX	Entrée/sortie	WORD	E, A, M, D, L	Numéro de l'entrée sur laquelle doit porter l'opération.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Dans les situations décrites au tableau 2-10, la fonction n'est pas exécutée. L'état de signal de ENO est mis à « 0 » et la valeur en retour est mise à l'une des valeurs suivantes :

Tableau 2-10 Situations d'erreur pour FC91

RET_VAL	Explication
W#16#0007	Le paramètre INDX est égal à 0.
W#16#0008	CMD et E_TYPE sont incompatibles ou incorrects.
W#16#0009	Le paramètre INDX pointe au-delà de la fin de la table.

Exemple

La figure 2-7 montre le mode de fonctionnement de l'opération WRD_TBL. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction WRD_TBL est exécutée. Comme le paramètre E_TYPE égale 6, les données (doubles mots) sont rangées dans la table commençant à l'adresse de mémoire désignée par le paramètre TABLE. Le premier mot de la table indique que la table contient trois doubles mots. La valeur de INDX indique l'entrée de la table devant être traitée par l'opération. Comme la valeur de CMD égale 8, une combinaison OU est effectuée sur la valeur désignée par le paramètre INDX. Comme la valeur de INDX est 2, le deuxième double mot (66665544) est combiné selon OU à la valeur désignée par SRC (11111111). Après l'exécution de l'opération, le résultat de la combinaison OU (77775555) est réécrit dans la table et la valeur de INDX est automatiquement incrémentée d'une entrée. Si le paramètre INDX pointe sur la dernière entrée de la table à l'appel de l'opération, le bit de sortie Q est mis à « 0 » après l'exécution. Dans cet exemple, la valeur de INDX ne contient pas la dernière entrée de cette table ; par conséquent, le paramètre Q est mis à « 1 » après l'opération.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

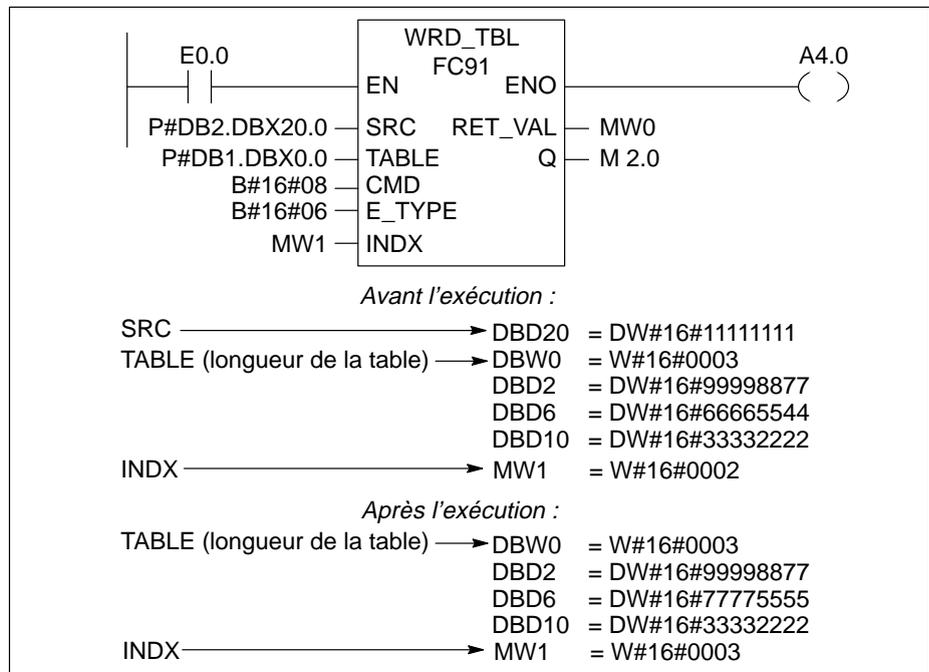


Figure 2-7 Combiner valeur logiquement avec entrée de table et mémoriser (WRD_TBL)

2.8 Tables de données corrélées (CDT) : FC103

Description

La fonction Tables de données corrélées (CDT) compare une valeur d'entrée (IN) à une table d'entrée préexistante (IN_TBL) et localise la première entrée de cette table supérieure ou égale à la valeur d'entrée. Dans ce cas, l'indice de l'entrée localisée est utilisé pour copier, dans la valeur de sortie (OUT), la valeur correspondante de la table de sortie (OUT_TBL).

- Les valeurs de la table d'entrée doivent être en ordre croissant : la première entrée de la table contient la plus petite valeur et la dernière entrée la plus grande valeur.
- La taille de la valeur d'entrée, des valeurs de la table et de la valeur de sortie est déterminée à partir de E_TYPE.
- La première entrée dans la table indique la longueur maximale de la table.
- La deuxième entrée dans la table contient la première valeur de la table.
- Le nombre d'entrées des deux tables doit être supérieur ou égal à zéro.

Nota

Vous devez initialiser la première entrée lorsque vous créez chaque table.

Paramètres

Le tableau 2-11 décrit les paramètres de la fonction CDT.

Tableau 2-11 Tables de données corrélées (FC103) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
IN_TBL	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table d'entrée.
OUT_TBL	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table de sortie.
IN	Entrée	POINTER*	E, A, M, D	Pointe sur la valeur d'entrée.
OUT	Entrée	POINTER*	E, A, M, D	Pointe sur la valeur de sortie.
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Pour la fonction CDT, les types de données suivants sont admis : B#16#05 = INT B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Dans les situations décrites au tableau 2-12, la fonction n'est pas exécutée. L'état de signal de ENO est mis à « 0 » et la valeur en retour est mise à l'une des valeurs suivantes :

Tableau 2-12 Situations d'erreur pour FC103

RET_VAL	Explication
W#16#0001	Indication d'un type de mémoire incorrect pour un paramètre
W#16#0002	E_TYPE incorrect
W#16#0003	La longueur de la table d'entrée et celle de la table de sortie ne correspondent pas.
W#16#0004	La longueur de la table est zéro.
W#16#0007	Aucune valeur de IN_TBL n'est supérieure ou égale à la valeur d'entrée.

Exemple

La figure 2-8 montre le mode de fonctionnement de l'opération CDT. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction CDT est exécutée. Dans cet exemple, les deux tables IN_TBL et OUT_TBL contiennent cinq entrées comme indiqué par le premier mot de chaque table. Le paramètre E_TYPE précise que le type de données des valeurs des tables est INTEGER et la valeur de IN est 22. La valeur de IN_TBL qui est supérieure ou égale à 22 est 64 qui a l'indice 5. La valeur corrélée dans OUT_TBL est 25 ; la valeur 25 est donc écrite dans OUT.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

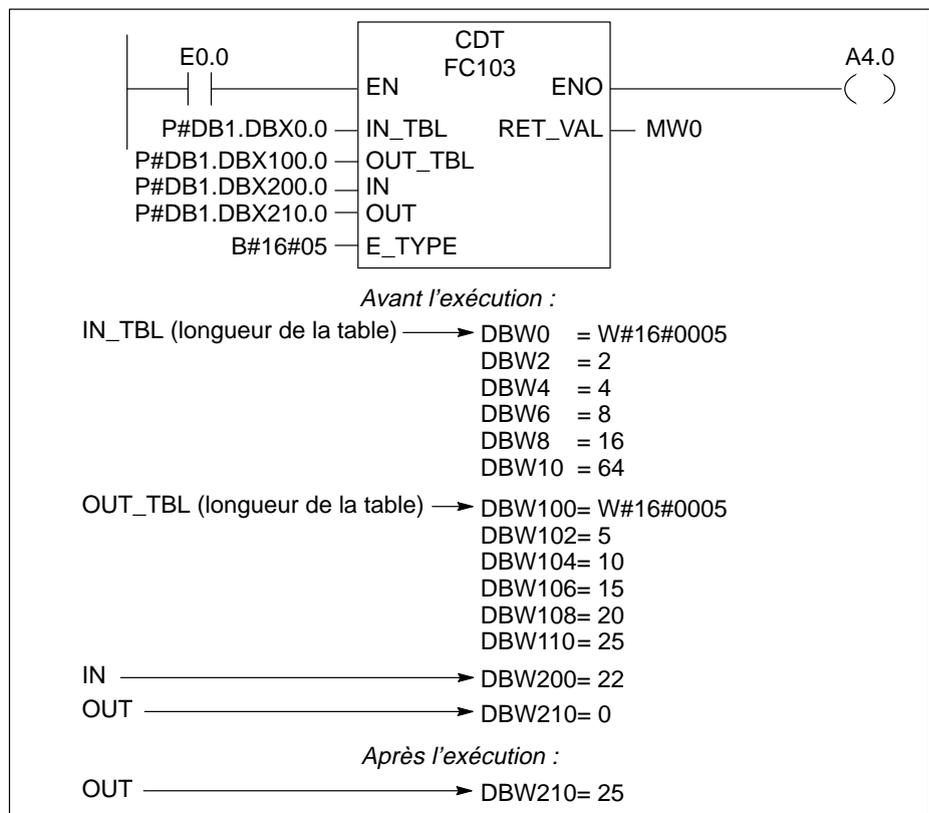


Figure 2-8 Tables de données corrélées (CDT)

2.9 Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL) : FC104

Description

La fonction Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL) exécute la commande indiquée (CMD) sur les entrées correspondantes des deux tables source (TBL1 et TBL2) et écrit le résultat dans les entrées correspondantes de la table de destination (DEST_TBL).

- Les types de données INT, DINT et REAL ne sont valables que pour les opérations arithmétiques.
- La première entrée dans la table indique la longueur maximale de la table.
- Le nombre d'entrées dans toutes les tables doit être identique et doit être supérieur à zéro.

Nota

Vous devez initialiser la première entrée lorsque vous créez chaque table.

Paramètres

Le tableau 2-13 décrit les paramètres de la fonction TBL_TBL.

Tableau 2-13 Exécuter opération sur tables et mémoriser dans table cible (FC104) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction est exécutée sans erreur.
TBL1	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la première table source.
TBL2	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la seconde table source.
DEST_TBL	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table de destination.
CMD	Entrée	BYTE	E, A, M, D, L, P	Indique le type d'opération devant être effectuée. Les opérations et valeurs suivantes sont admises : B#16#07 = Combinaison ET B#16#08 = Combinaison OU B#16#09 = Combinaison OU exclusif B#16#0a = Addition B#16#0b = Soustraction B#16#0c = Multiplication B#16#0d = Division
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Pour la fonction TBL_TBL, les types de données suivants sont admis : B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Dans les situations décrites au tableau 2-14, la fonction n'est pas exécutée. L'état de signal de ENO est mis à « 0 » et la valeur en retour est mise à l'une des valeurs suivantes :

Tableau 2-14 Situations d'erreur pour FC104

RET_VAL	Explication
W#16#0001	Indication d'un type de mémoire incorrect pour un paramètre
W#16#0002	E_TYPE incorrect
W#16#0003	La longueur des tables d'entrée et celle de la table de sortie ne correspondent pas.
W#16#0004	La longueur de la table est zéro.
W#16#0005	Les paramètres E_TYPE et CMD ne sont pas compatibles.
W#16#0006	CMD incorrect

Exemple

La figure 2-9 montre le mode de fonctionnement de l'opération TBL_TBL. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction TBL_TBL est exécutée. Dans cet exemple, toutes les tables contiennent trois entrées comme indiqué par le premier mot de chaque table. Le paramètre E_TYPE précise que le type de données des valeurs des tables est WORD et le paramètre CMD que la commande à exécuter sur TBL1 et TBL2 est la combinaison ET.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

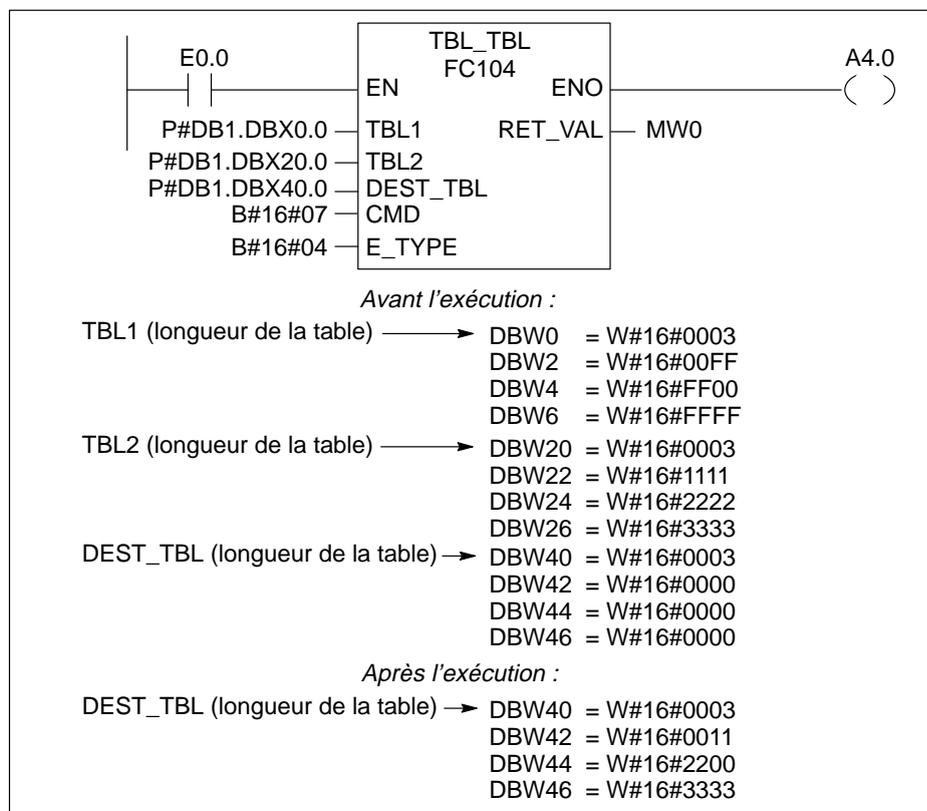


Figure 2-9 Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL)

Fonctions de décalage

3

Ce chapitre décrit les fonctions de décalage dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
3.1	Déplacer mot vers registre à décalage (WSR) : FC90	3-2
3.2	Déplacer bit vers registre à décalage (SHRB) : FC92	3-4

3.1 Déplacer mot vers registre à décalage (WSR) : FC90

Description

La fonction Déplacer mot vers registre à décalage (WSR) déplace des données en provenance de la source indiquée vers un registre à décalage. Les valeurs sont déplacées vers l'adresse suivante. Le paramètre LENGTH indique le nombre d'adresses devant être déplacées. Les données contenues dans la dernière adresse du registre à décalage sont perdues à l'issue de l'opération. De nouvelles données sont lues à partir de la source (S_DATA) à chaque fois que l'opération est exécutée. Ces données sont déplacées vers l'adresse de début (START) du registre à décalage lorsque l'entrée RESET est mise à « 0 ». Si l'entrée RESET est mise à « 1 », les adresses du registre sont mises à « 0 » lors de l'exécution de l'opération. La sortie Q est activée lorsque le registre à décalage est vide ou qu'il est effacé (c'est-à-dire après une remise à zéro ou lorsque le registre ne contient que des zéros).

Paramètres

Le tableau 3-1 décrit les paramètres de la fonction WSR.

Tableau 3-1 Déplacer mot vers registre à décalage (FC90) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
RESET	Entrée	BOOL	E, A, M, D, L	Si mis à « 1 », le registre à décalage est remis à zéro.
S_DATA	Entrée	POINTER*	E, A, M, D	Pointe sur les données source devant être insérées dans la table.
START	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table.
LENGTH	Entrée	WORD	E, A, M, D, L, P	Nombre d'éléments devant être déplacés.
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Pour la fonction WSR, les types de données suivants sont admis : B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
Q	Sortie	BOOL	A, M, D, L	Indique « 0 » lorsque le paramètre RESET est actif (à 1) ou que tous les éléments devant être déplacés ont la valeur « 0 ».

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Si le paramètre E_TYPE est incorrect, la fonction n'est pas exécutée et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 3-1 montre le mode de fonctionnement de l'opération WSR. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction WSR est exécutée. Comme le paramètre E_TYPE égale 4, des mots sont enregistrés dans la table commençant à l'adresse de mémoire désignée par START. Le paramètre LENGTH indique « 4 », signifiant que 4 mots doivent être déplacés, le premier mot étant indiqué par le pointeur START. Après que la première valeur de la table a été déplacée vers l'adresse suivante, la première adresse reçoit les données désignées par le pointeur S_DATA. La dernière valeur de la table est perdue. Lorsque l'entrée RESET est mise à « 1 », les adresses de la table sont mises à « 0 » et ne sont pas déplacées.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

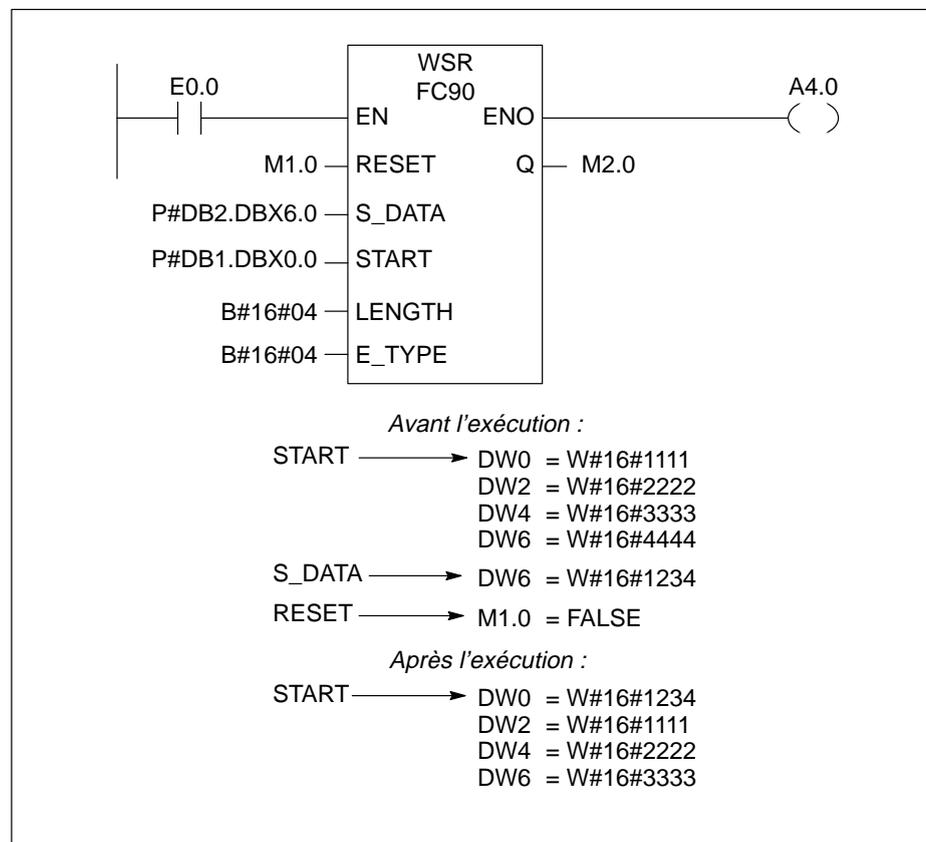


Figure 3-1 Déplacer mot vers registre à décalage (WSR)

3.2 Déplacer bit vers registre à décalage (SHRB) : FC92

Description

La fonction Déplacer bit vers registre à décalage (SHRB) déplace un bit de la source indiquée (DATA) vers un registre à décalage. De nouvelles données sont lues à partir de la source à chaque fois que l'opération est exécutée. Ces données sont déplacées vers l'adresse de début (S_BIT) du registre à décalage lorsque l'entrée RESET est à « 0 ». Tous les autres bits suivants sont repoussés d'un bit. Le bit dans la dernière adresse (S_BIT + N) est perdu après le déplacement. Lorsque l'entrée RESET est mise à « 1 », les adresses de la table sont mises à « 0 » et ne sont pas déplacées.

Paramètres

Le tableau 3-2 décrit les paramètres de la fonction SHRB.

Tableau 3-2 Déplacer bit vers registre à décalage (FC92) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
DATA	Entrée	BOOL	E, A, M, D, L	Bit source
RESET	Entrée	BOOL	E, A, M, D, L	Si mis à « 1 », le registre à décalage est remis à zéro.
S_BIT	Entrée	POINTER*	E, A, M, D	Pointe sur le bit de début dans le registre à décalage.
N	Entrée	WORD	E, A, M, D, L, P	Longueur du registre à décalage (nombre de bits devant être déplacés).

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Cette fonction ne reconnaît aucune erreur.

Exemple

La figure 3-2 montre le mode de fonctionnement de l'opération SHR. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction SHR est exécutée. Dans cet exemple, le paramètre N est égal à « 14 » (E en notation hexadécimale), indiquant que 14 bits doivent être déplacés, en commençant par le premier bit à l'adresse de pointeur S_BIT. Une fois les bits déplacés, la première adresse reçoit les données indiquées par l'entrée DATA. La toute dernière valeur binaire est perdue.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

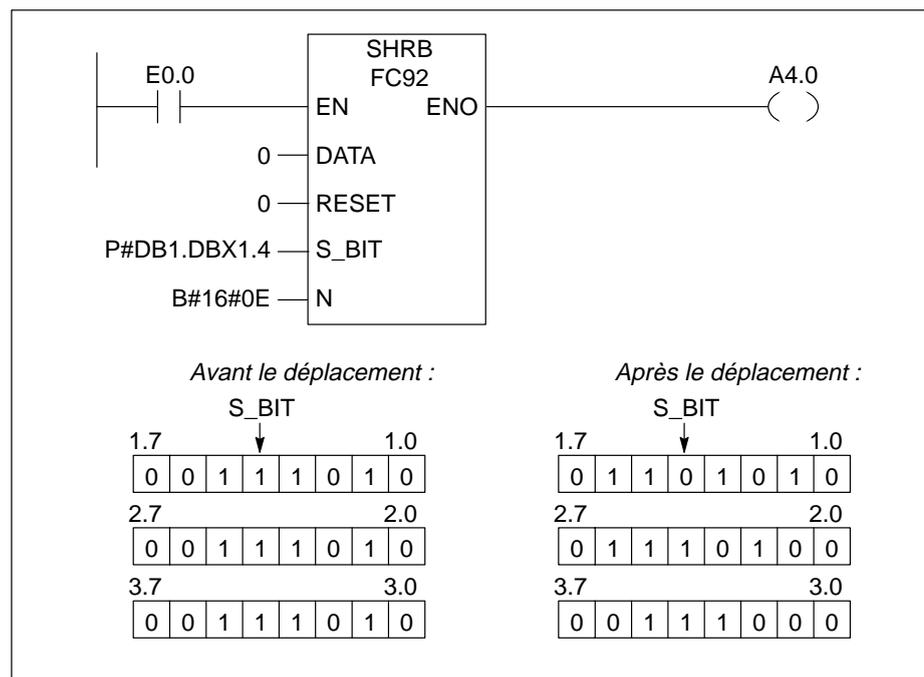


Figure 3-2 Déplacer bit vers registre à décalage (SHRB)

Fonction et bloc fonctionnel de transfert

4

Ce chapitre décrit la fonction (FC) et le bloc fonctionnel (FB) de transfert dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
4.1	Transfert indirect de blocs (IBLKMOV) : FC81	4-2
4.2	Rassembler/répartir données de table (PACK) : FB86	4-4

4.1 Transfert indirect de blocs (IBLKMOV) : FC81

Description

Avec la fonction Transfert indirect de blocs (IBLKMOV), vous pouvez transférer un bloc de données constitué soit d'octets, de mots, de nombres entiers de 16 bits, de doubles mots ou de nombres entiers de 32 bits d'un bloc source à un bloc de destination. Le nombre d'éléments devant être transféré est indiqué par le paramètre LENGTH. La taille des éléments est indiquée par le paramètre E_TYPE. Les pointeurs S_DATA et D_DATA indiquent l'adresse des pointeurs identifiant l'adresse de début des données source et celle des données de destination. Comme les données devant être transférées sont désignées de cette façon indirecte, cette fonction est appelée fonction de transfert indirect.

Paramètres

Le tableau 4-1 décrit les paramètres de la fonction IBLKMOV.

Tableau 4-1 Transfert indirect de blocs (FC81) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
S_DATA	Entrée	POINTER*	E, A, M, D	Indique un pointeur identifiant l'adresse de début des données source.
LENGTH	Entrée	POINTER*	E, A, M, D	Indique la longueur du bloc de données devant être transféré.
D_DATA	Entrée	POINTER*	E, A, M, D	Indique un pointeur identifiant l'adresse de début des données de destination.
E_TYPE	Entrée	BYTE	E, A, M, D, L	Indique le type de données. Pour la fonction IBLKMOV, les types de données suivants sont admis : B#16#02 = BYTE B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Si le paramètre E_TYPE est incorrect, la fonction n'est pas exécutée et l'état de signal de ENO est mis à « 0 ».

Exemple

La figure 4-1 montre le mode de fonctionnement de l'opération IBLKMOV. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), la fonction est exécutée. Le paramètre S_DATA pointe sur DB1.DBX0.0 qui contient le pointeur DB1.DBX50.0 (adresse de début des données source). Le paramètre D_DATA pointe sur DB1.DBX20.0 qui contient le pointeur DB2.DBX10.0 (adresse de début des données de destination). Après l'exécution de la fonction, un bloc de deux mots est transféré.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

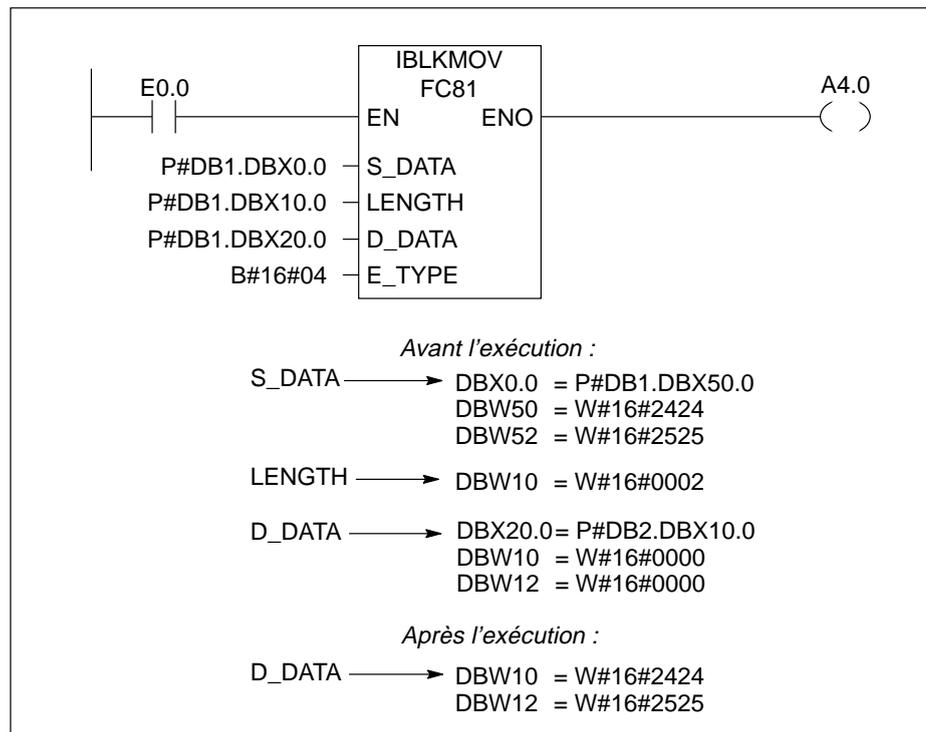


Figure 4-1 Transfert indirect de blocs (IBLKMOV)

4.2 Rassembler/répartir données de table (Pack) : FB86

Description

Le bloc fonctionnel Rassembler/répartir données de table (PACK) transfère des données entre des adresses individuelles et une table. Le paramètre DIR précise le sens du transfert. Chaque opération PACK traite jusqu'à cinq paquets de données : P_DATA1 à P_DATA5. Si DIR indique « vers », le bloc fonctionnel FB86 rassemble les données de ces adresses dans la table précisée. En revanche, si DIR indique « à partir de », les données sont réparties de la table vers les différentes adresses.

Voici les règles pour « rassembler » des données dans une table :

- Les bits individuels (BOOL) sont transférés dans le bit disponible suivant de la table.
- Les types de données de huit bits sont transférés dans l'octet disponible suivant de la table. Lorsqu'un octet est écrit dans la table, des zéros sont reportés dans les bits non utilisés de l'octet précédent.
- Les types de données de 16 et de 32 bits sont transférés dans le mot disponible suivant de la table. Lorsqu'un mot est écrit dans la table, des zéros sont reportés dans les bits non utilisés du mot précédent.

Voici les règles pour « répartir » des données à partir d'une table :

- Il est interdit de sauter des sections d'une table.
- Tous les bits BOOL indiqués sont transférés à partir de la table.
- Les types de données de huit bits sont transférés à partir du premier octet disponible de la table. Ainsi, les bits non utilisés dans l'octet précédent de la table ne sont pas inclus dans un octet transféré à partir de la table.
- Les types de données de 16 et de 32 bits sont transférés à partir du premier mot disponible de la table. Ainsi, les bits non utilisés du mot précédent de la table ne sont pas inclus dans un mot transféré à partir de la table.

Le bloc fonctionnel PACK autorise les types de données suivants pour le pointeur ANY :

- BOOL
- WORD
- INT
- BYTE
- DINT
- REAL
- CHAR
- DWORD

Paramètres

Le tableau 4-2 décrit les paramètres du bloc fonctionnel PACK.

Tableau 4-2 Rassembler/répartir données de table (FB86) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
TABLE	Entrée	POINTER*	E, A, M, D	Pointe sur le début de la table.
P_DATA1	Entrée	ANY	E, A, M, D	Désigne le début d'un paquet de données à transférer.
P_DATA2	Entrée	ANY	E, A, M, D	Désigne le début d'un paquet de données à transférer.
P_DATA3	Entrée	ANY	E, A, M, D	Désigne le début d'un paquet de données à transférer.
P_DATA4	Entrée	ANY	E, A, M, D	Désigne le début d'un paquet de données à transférer.
P_DATA5	Entrée	ANY	E, A, M, D	Désigne le début d'un paquet de données à transférer.
ERR_CODE	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
DIR	statique	BOOL	E, A, M, D, L	Sens du transfert. L'état de signal « 0 » signifie « vers » et l'état de signal « 1 » signifie « à partir de ».

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Dans les situations décrites au tableau 4-3, le bloc fonctionnel n'est pas exécuté. L'état de signal de ENO est mis à « 0 » et ERR_CODE prend l'une des valeurs suivantes :

Tableau 4-3 Situations d'erreur pour FB86

ERR_CODE	Explication
W#16#0001	Indication d'un type de mémoire incorrect pour un paramètre
W#16#0002	E_TYPE incorrect

Exemple

La figure 4-2 montre le mode de fonctionnement de l'opération PACK. Si l'état de signal à l'entrée E 0.0 égale 1 (entrée activée), le bloc fonctionnel PACK est exécuté. Dans cet exemple, quatre paquets de données sont « rassemblés » dans la table.

Si le bloc fonctionnel a été exécuté sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et ERR_CODE est mis à la valeur W#16#0000.

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

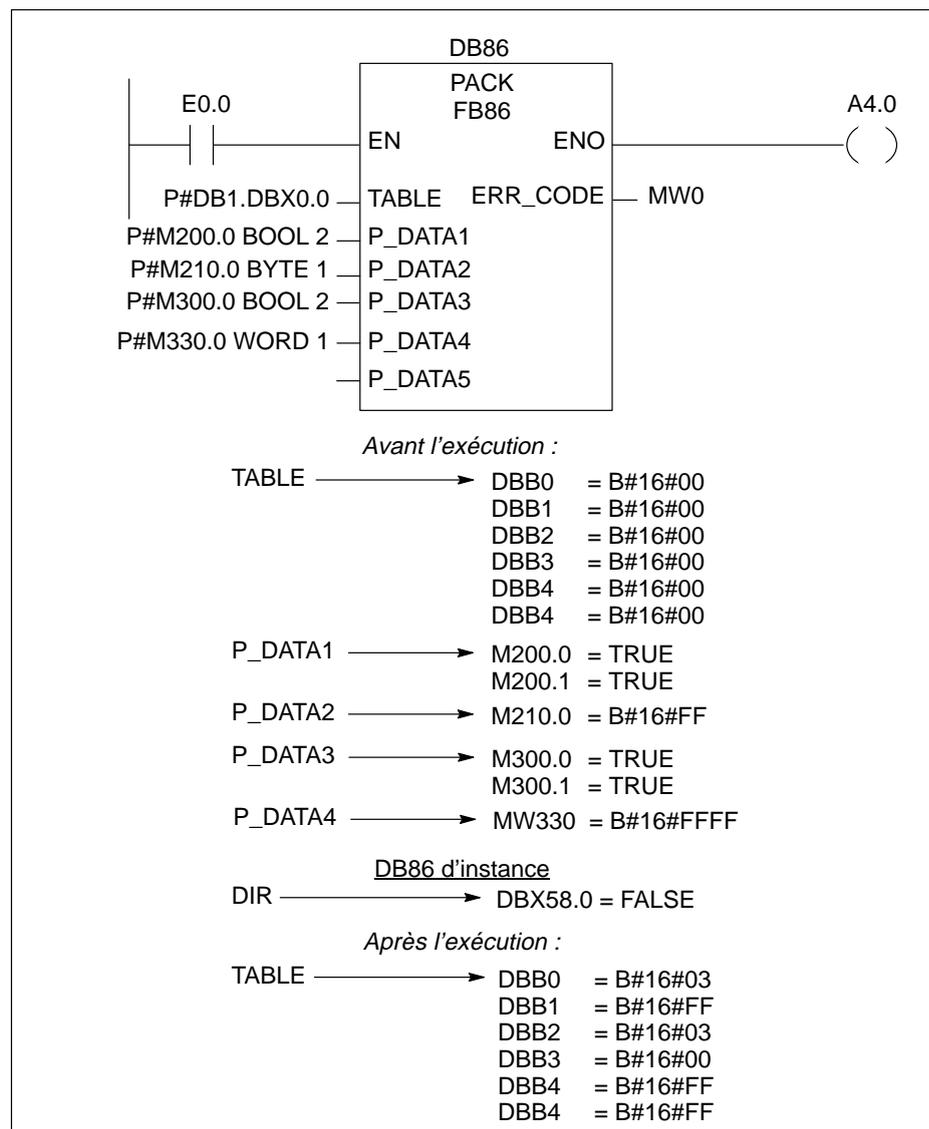


Figure 4-2 Rassembler/répartir données de table (PACK)

Fonction et blocs fonctionnels de temporisation

5

Ce chapitre décrit la fonction (FC) et les blocs fonctionnels (FB) de temporisation dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
5.1	Temporisation sous forme de retard à la montée mémorisé (TONR) : FC80	5-2
5.2	Temporisation d'alarme avec commande tout ou rien (DCAT) : FB81	5-4
5.3	Temporisation d'alarme avec commande moteur (MCAT) : FB82	5-7
5.4	Barillet d'événement avec masquage (DRUM) : FB85	5-10

5.1 Temporisation sous forme de retard à la montée mémorisé (TONR) : FC80

Description

La fonction Temporisation sous forme de retard à la montée mémorisé (TONR) mémorise la durée jusqu'à ce que la valeur en cours du temps écoulé (ET) soit supérieure ou égale à la valeur de temps prédéfinie (PV). Comme la fonction TONR se base sur le temps d'exécution du dernier cycle du bloc d'organisation dans lequel elle s'exécute pour mémoriser la durée écoulée, vous ne devez utiliser cette fonction que pour les blocs d'organisation répétitifs tels que l'OB1 et les blocs d'organisation cycliques.

Nota

Vous devez transférer le temps de cycle du bloc d'organisation des variables locales de démarrage de la table de déclaration des variables du bloc d'organisation vers la variable globale DELTA_T.

Tant que l'état de signal du paramètre RESET égale 0, que l'état de signal du paramètre TMR_EN égale 1 et que ET est inférieure à PV, la fonction TONR ajoute la valeur de DELTA_T à la valeur de ET. Si l'état de signal du paramètre TMR_EN n'est pas « 1 », aucune valeur de temps n'est ajoutée à la valeur ET. Lorsque la valeur ET est supérieure ou égale à la valeur PV, l'état de signal de la sortie Q est mis à « 1 ». Une fois la sortie Q activée, elle le reste et la valeur de ET n'est plus modifiée jusqu'à la remise à zéro. La fonction remet la valeur ET à « 0 » et désactive la sortie Q lorsque l'état de signal du paramètre RESET est « 1 ».

Paramètres

Le tableau 5-1 décrit les paramètres de la fonction TONR.

Tableau 5-1 Temporisation sous forme de retard à la montée mémorisé (FC80) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
TMR_EN	Entrée	BOOL	E, A, M, D, L	Active la temporisation de mémorisation de la durée.
RESET	Entrée	BOOL	E, A, M, D, L	Si RESET = 1, la temporisation est remise à « 0 ».
PV	Entrée	DINT	E, A, M, D, L, P ou constante	Valeur prédéfinie
DELTA_T	Entrée	INT	E, A, M, D, L ou constante	Temps d'exécution de l'OB lors du cycle précédent.
Q	Sortie	BOOL	A, M, D, L	Est mis à « 1 » lorsque ET est supérieur ou égal à PV.
ET	Entrée/sortie	DINT	E, A, M, D, L	Valeur en cours du temps écoulé.

Informations d'erreur

Cette fonction ne reconnaît aucune erreur.

Exemple

La figure 5-1 montre le mode de fonctionnement de l'opération TONR. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction TONR est exécutée. Si l'état de signal de l'entrée E 0.1 égale 1, si l'état de signal de E 0.2 égale 0 et si ET est inférieur à PV, la valeur DELTA_T est ajoutée à la valeur ET (100 + 50 = 150). Si la valeur de ET est inférieure à PV, l'état de signal de A 1.1 restera à 0.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

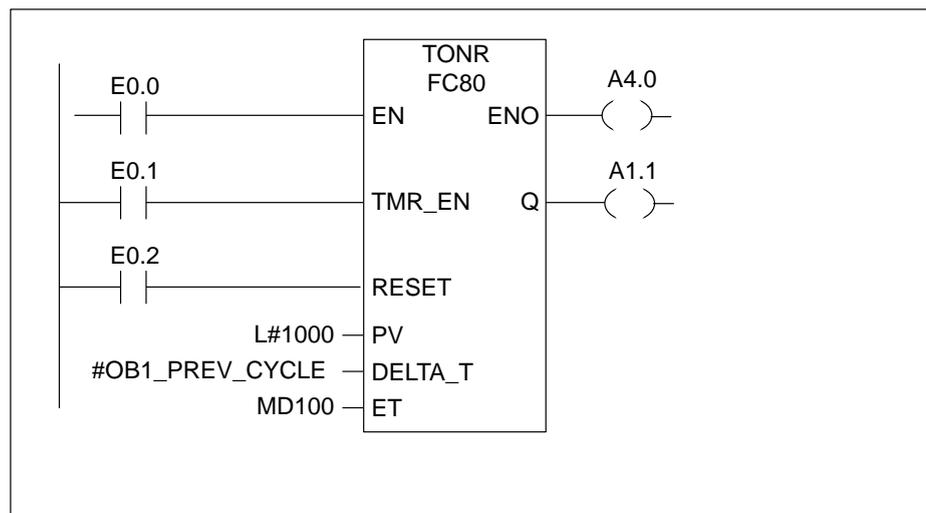


Figure 5-1 Temporisation sous forme de retard à la montée mémorisé (TONR)

5.2 Temporisation d'alarme avec commande tout ou rien (DCAT) : FB81

Description

Le bloc fonctionnel Temporisation d'alarme avec commande tout ou rien (DCAT) mémorise la durée à partir de la transition de l'entrée de commande (CMD) vers l'ouverture – ou vers la fermeture – soit jusqu'à ce que le temps prédéfini PT soit dépassé, soit jusqu'à ce que l'entrée en retour (O_FB ou O_FC) signale que l'appareil s'est ouvert – ou fermé – pendant l'intervalle de temps prescrit. Si le temps prédéfini expire avant réception du signal en retour, l'alarme correspondante est activée. Si la commande d'entrée change d'état avant le temps prédéfini, le temps est redémarré.

- Lorsque l'état de signal de l'entrée CMD passe de « 0 » à « 1 », l'état de signal de Q est mis à « 1 », ET est mis à « 0 », l'état de signal des deux sorties d'alarme (OA et CA) est mis à « 0 » et celui de CMD_HIS est mis à « 1 ».
- Lorsque l'état de signal de l'entrée CMD passe de « 1 » à « 0 », l'état de signal de Q est mis à « 0 », ET est mis à « 0 », l'état de signal des deux sorties d'alarme (OA et CA) est mis à « 0 » et celui de CMD_HIS est mis à « 0 ».
- Lorsque l'état de signal des deux paramètres CMD et CMD_HIS est « 1 » et que l'état de signal de O_FB est « 0 », la différence de temps (ms) depuis la dernière exécution du bloc fonctionnel est ajoutée à ET. Si ET dépasse PT, l'état de signal de OA est mis à « 1 » ; sinon, il est mis à « 0 ». L'état de signal de CMD_HIS est posé égal à celui de CMD.
- Lorsque l'état de signal des deux paramètres CMD et CMD_HIS est « 1 », que celui de O_FB est « 1 » et que celui de C_FB est « 0 », l'état de signal de OA est mis à « 0 ». ET est posé égal à PT afin que si, par la suite, l'état de signal de O_FB est mis à « 0 », l'alarme soit activée lors du prochain appel du bloc fonctionnel. L'état de signal de CMD_HIS est posé égal à celui de CMD.
- Lorsque l'état de signal des deux paramètres CMD et CMD_HIS est « 0 » et que l'état de signal de C_FB est « 0 », la différence de temps (ms) depuis la dernière exécution du bloc fonctionnel est ajoutée à ET. Si ET dépasse PT, l'état de signal de CA est mis à « 1 » ; sinon, il est mis à « 0 ». L'état de signal de CMD_HIS est posé égal à celui de CMD.
- Lorsque l'état de signal des deux paramètres CMD et CMD_HIS est « 0 », que celui de O_FB est « 0 » et que celui de C_FB est « 1 », l'état de signal de CA est mis à « 0 ». ET est posé égal à PT afin que si, par la suite, l'état de signal de C_FB est mis à « 0 », l'alarme soit activée lors de la prochaine exécution du bloc fonctionnel. L'état de signal de CMD_HIS est posé égal à celui de CMD.
- Si l'état de signal de O_FB et celui de C_FB sont à « 1 » simultanément, il s'agit d'une situation d'erreur et l'état de signal des deux sorties d'alarme est mis à « 1 ».

Paramètres

Le tableau 5-2 décrit les paramètres du bloc fonctionnel DCAT.

Tableau 5-2 Temporisation d'alarme avec commande tout ou rien (FB81) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
CMD	Entrée	BOOL	E, A, M, D, L	L'état de signal « 0 » correspond à une commande de fermeture et l'état de signal « 1 » à une commande d'ouverture.
O_FB	Entrée	BOOL	E, A, M, D, L	Entrée de retour d'ouverture
C_FB	Entrée	BOOL	E, A, M, D, L	Entrée de retour de fermeture
Q	Sortie	BOOL	E, A, M, D, L	Suit l'entrée CMD.
OA	Sortie	BOOL	E, A, M, D, L	Sortie d'alarme d'ouverture
CA	Sortie	BOOL	E, A, M, D, L	Sortie d'alarme de fermeture
ET	statique	DINT	E, A, M, D, L	Décompte en cours du temps écoulé avec 1 décompte = 1 ms
PT	statique	DINT	E, A, M, D, L	Décompte prédéfini de la temporisation avec 1 décompte = 1 ms
PREV_TIME	statique	DWORD	E, A, M, D, L	Temps système précédent
CMD_HIS	statique	BOOL	E, A, M, D, L	Bit d'historique de CMD

Informations d'erreur

Ce bloc fonctionnel ne reconnaît aucune erreur.

Exemple

La figure 5-2 montre le mode de fonctionnement de l'opération DCAT. Si l'état de signal de E0.0 égale 1 (entrée activée), le bloc fonctionnel DCAT est exécuté. Dans cet exemple, l'entrée CMD passe de l'état de signal « 0 » à l'état de signal « 1 » comme indiqué par CMD_HIS et CMD. En fonction de cela, Q et CMD_HIS sont mis à « 1 », ET est mis à « 0 » et les deux sorties d'alarme OA et CA sont mises à « 0 ».

Si le bloc fonctionnel est exécuté sans erreur, l'état de signal de ENO et de A4.0 est mis à « 1 ».

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

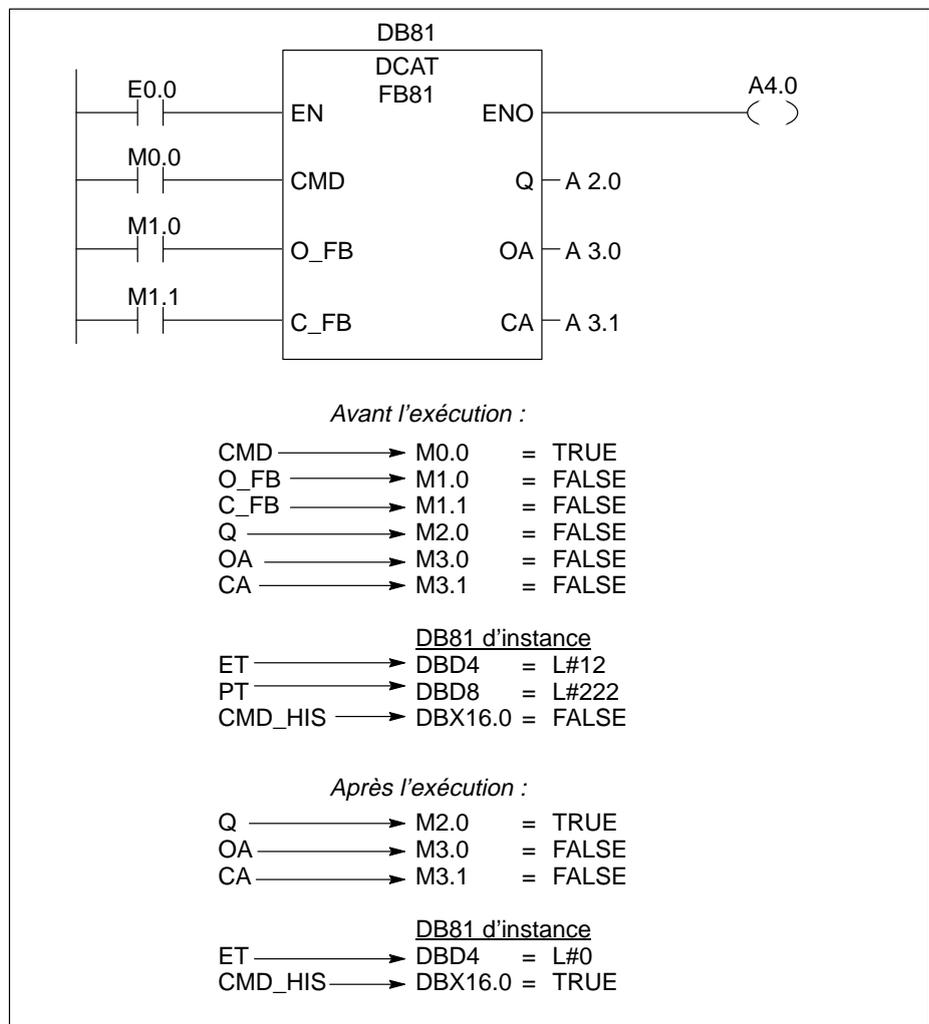


Figure 5-2 Temporisation d'alarme avec commande tout ou rien (DCAT)

5.3 Temporisation d'alarme avec commande moteur (MCAT) : FB82

Description

Le bloc fonctionnel Temporisation d'alarme avec commande moteur (MCAT) mémorise la durée à partir de la transition ON de l'une des entrées de commande – ouverture ou fermeture – soit jusqu'à ce que le temps prédéfini PT soit dépassé, soit jusqu'à ce que l'entrée en retour correspondante signale que l'appareil a achevé l'opération commandée pendant l'intervalle de temps prescrit. Si le temps prédéfini expire avant réception du signal en retour, l'alarme correspondante est activée. Les descriptions de la réaction de MCAT aux différentes conditions d'entrée sont résumées dans la table de vérité MCAT (tableau 5-3).

Tableau 5-3 Table de vérité pour MCAT

ENTREES								SORTIES								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	ETAT
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarme
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarme
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Arrêt
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Arrêt
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Commencer ouverture
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Ouverture en cours
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Ouvert
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Alarme d'ouverture
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Commencer fermeture
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Fermeture en cours
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Fermé
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Alarme de fermeture
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Arrêté

Avec :

- INC = Ajouter à ET la différence de temps (ms) depuis la dernière exécution du bloc fonctionnel
- PT = PT est posé égal à ET
- X = Sans objet
- <PT = ET < PT
- >= PT = ET >= PT

Paramètres

Le tableau 5-4 décrit les paramètres du bloc fonctionnel MCAT.

Tableau 5-4 Temporisation d'alarme avec commande moteur (FB82) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
O_CMD	Entrée	BOOL	E, A, M, D, L	Entrée de commande d'ouverture
C_CMD	Entrée	BOOL	E, A, M, D, L	Entrée de commande de fermeture
S_CMD	Entrée	BOOL	E, A, M, D, L	Entrée de commande d'arrêt
O_FB	Entrée	BOOL	E, A, M, D, L	Entrée de retour d'ouverture
C_FB	Entrée	BOOL	E, A, M, D, L	Entrée de retour de fermeture
OO	Sortie	BOOL	E, A, M, D, L	Sortie d'ouverture
CO	Sortie	BOOL	E, A, M, D, L	Sortie de fermeture
OA	Sortie	BOOL	E, A, M, D, L	Sortie d'alarme d'ouverture
CA	Sortie	BOOL	E, A, M, D, L	Sortie d'alarme de fermeture
Q	Sortie	BOOL	E, A, M, D, L	Un état de signal « 1 » indique une condition d'alarme.
ET	statique	DINT	E, A, M, D, L	Décompte en cours du temps écoulé avec 1 décompte = 1 ms
PT	statique	DINT	E, A, M, D, L	Décompte prédéfini de la temporisation avec 1 décompte = 1 ms
PREV_TIME	statique	DWORD	E, A, M, D, L	Temps système précédent
O_HIS	statique	BOOL	E, A, M, D, L	Bit d'historique d'ouverture
C_HIS	statique	BOOL	E, A, M, D, L	Bit d'historique de fermeture

Informations d'erreur

Ce bloc fonctionnel ne reconnaît aucune erreur.

Exemple

La figure 5-3 montre le mode de fonctionnement de l'opération MCAT. Si l'état de signal de E0.0 égale 1 (entrée activée), le bloc fonctionnel MCAT est exécuté. Dans cet exemple, en fonction de l'état des entrées, MCAT est dans l'état OUVERTURE EN COURS et les sorties sont définies en conséquence.

Si le bloc fonctionnel est exécuté sans erreur, l'état de signal de ENO et de A4.0 est mis à « 1 ».

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

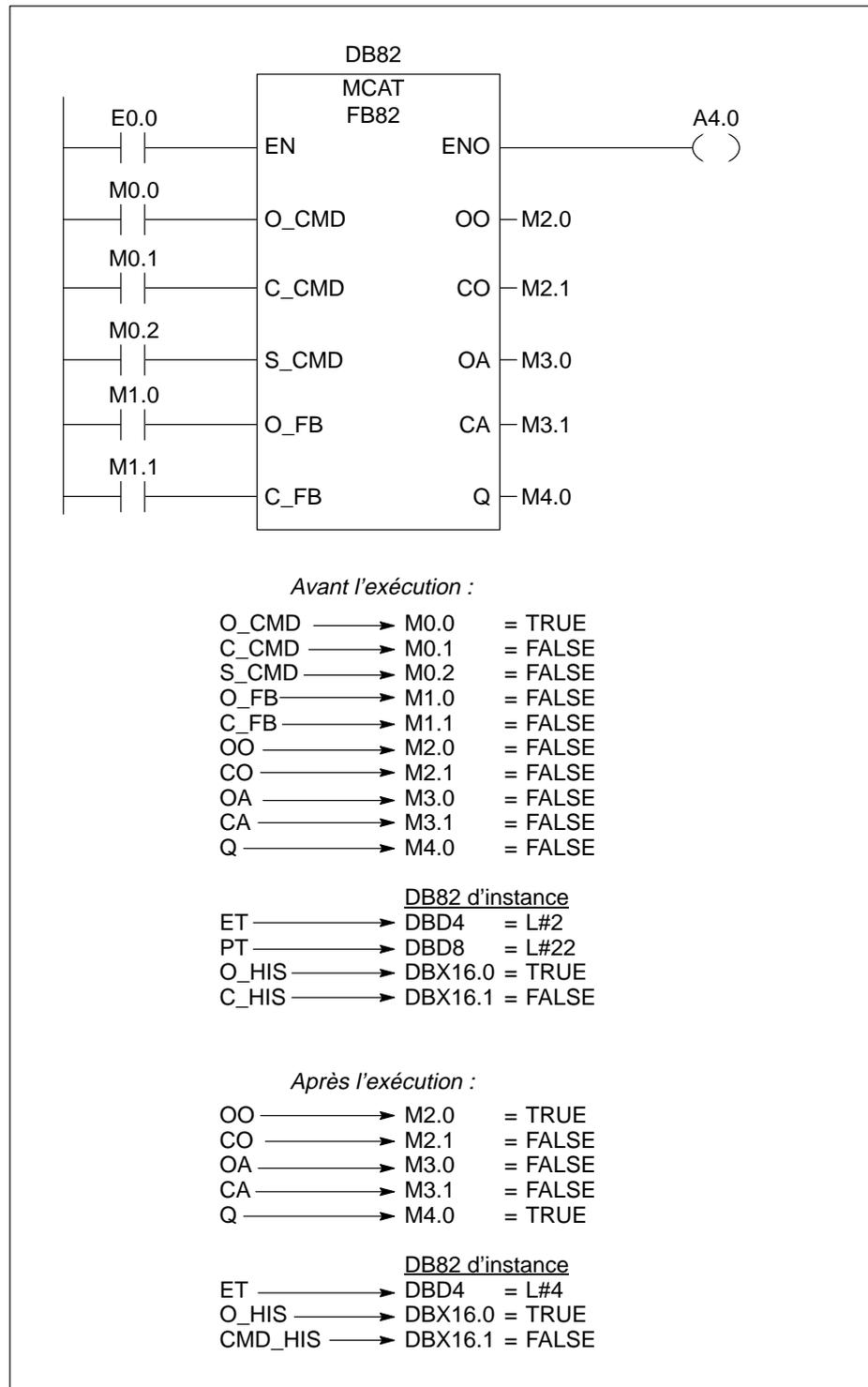


Figure 5-3 Temporisation d'alarme avec commande moteur (MCAT)

5.4 Barillet d'événement avec masquage (DRUM) : FB85

Description

Le bloc fonctionnel Barillet d'événement avec masquage (DRUM) écrit les valeurs programmées (OUT_VAL) de l'étape appropriée dans les bits de sortie programmés (OUT1 à OUT16) et le mot de sortie OUT_WORD en tenant compte des valeurs du masque de validation (S_MASK) pour cette étape. Les valeurs de sortie restent inchangées tant que le barillet demeure dans l'étape considérée. Le barillet progressera à l'étape suivante soit lorsque l'événement pour cette étape est vrai et que le temps programmé pour l'étape en cours a expiré, soit lorsque l'état de signal de l'entrée de progression (JOG) passe de « 0 » à « 1 ». Lorsque l'état de signal de RESET égale « 1 », le barillet est remis à zéro, l'étape en cours étant alors posée égale à l'étape prédéfinie (DSP).

La durée passée sur une étape est déterminée par le produit de la base de temps prédéfinie du barillet (DTBP) et des valeurs de décompte prédéfinies (S_PRESET) correspondant à chaque étape. Au début de chaque nouvelle étape, cette valeur calculée est chargée dans DCC qui contient le temps restant pour l'étape en cours. Si, par exemple, DTBP égale 2 et que la valeur prédéfinie pour l'étape 1 soit égale à 100 (100 ms), DCC sera égal à 200 (200 ms).

Il est possible de programmer une étape avec une valeur de temps ou un événement ou avec les deux. Les étapes avec un bit d'événement et une valeur de temps de zéro progressent à l'étape suivante dès que l'état de signal du bit d'événement est égal à « 1 ». Pour les étapes avec uniquement une valeur de temps, le temps commence à s'écouler dès l'entrée dans cette étape. Pour les étapes avec un bit d'événement et une valeur de temps supérieure à zéro, le temps commence à s'écouler lorsque l'état de signal du bit d'événement égale « 1 ». Les bits d'événement sont initialisés à l'état de signal « 1 ».

Lorsque le pointeur d'étapes est sur la dernière étape programmée (LST_STEP) et que le temps pour cette étape a expiré, l'état de signal de la sortie Q est mis à « 1 » ; sinon il est mis à « 0 ». Une fois Q à « 1 », le barillet demeure dans cette étape jusqu'à la remise à zéro (RESET).

Le masque configurable S_MASK permet de sélectionner les bits individuels du mot de sortie (OUT_WORD) et les bits de sortie (OUT1 à OUT16) devant être mis à « 1 » ou à « 0 » par les valeurs de sortie (OUT_VAL). Lorsqu'un bit du masque configurable est à « 1 », la valeur OUT_VAL met à « 1 » ou à « 0 » le bit correspondant. Lorsqu'un bit du masque configurable est à « 0 », le bit correspondant reste inchangé. Chacun des bits du masque configurable pour les 16 étapes est configuré à l'état de signal « 1 ».

Le bit de sortie OUT1 correspond au bit de poids faible et le bit de sortie OUT16 au bit de poids fort de la sortie (mot) OUT_WORD.

Paramètres

Le tableau 5-5 décrit les paramètres du bloc fonctionnel DRUM.

Tableau 5-5 Barillet d'événement avec masquage (FB85) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
RESET	Entrée	BOOL	E, A, M, D, L	L'état de signal « 1 » indique une condition de remise à zéro.
JOG	Entrée	BOOL	E, A, M, D, L	Une transition de l'état de signal de « 0 » à « 1 » fait progresser le barillet à l'étape suivante.
DRUM_EN	Entrée	BOOL	E, A, M, D, L	L'état de signal « 1 » permet au barillet de progresser selon les critères d'événement et de temps.
LST_STEP	Entrée	BYTE	E, A, M, D, L ou constante	Numéro de la dernière étape programmée
EVENT1	Entrée	BOOL	E, A, M, D, L	Bit d'événement 1 ; état de signal initial = 1
EVENT2	Entrée	BOOL	E, A, M, D, L	Bit d'événement 2 ; état de signal initial = 1
EVENT3	Entrée	BOOL	E, A, M, D, L	Bit d'événement 3 ; état de signal initial = 1
EVENT4	Entrée	BOOL	E, A, M, D, L	Bit d'événement 4 ; état de signal initial = 1
EVENT5	Entrée	BOOL	E, A, M, D, L	Bit d'événement 5 ; état de signal initial = 1
EVENT6	Entrée	BOOL	E, A, M, D, L	Bit d'événement 6 ; état de signal initial = 1
EVENT7	Entrée	BOOL	E, A, M, D, L	Bit d'événement 7 ; état de signal initial = 1
EVENT8	Entrée	BOOL	E, A, M, D, L	Bit d'événement 8 ; état de signal initial = 1
EVENT9	Entrée	BOOL	E, A, M, D, L	Bit d'événement 9 ; état de signal initial = 1
EVENT10	Entrée	BOOL	E, A, M, D, L	Bit d'événement 10 ; état de signal initial = 1
EVENT11	Entrée	BOOL	E, A, M, D, L	Bit d'événement 11 ; état de signal initial = 1
EVENT12	Entrée	BOOL	E, A, M, D, L	Bit d'événement 12 ; état de signal initial = 1
EVENT13	Entrée	BOOL	E, A, M, D, L	Bit d'événement 13 ; état de signal initial = 1
EVENT14	Entrée	BOOL	E, A, M, D, L	Bit d'événement 14 ; état de signal initial = 1
EVENT15	Entrée	BOOL	E, A, M, D, L	Bit d'événement 15 ; état de signal initial = 1
EVENT16	Entrée	BOOL	E, A, M, D, L	Bit d'événement 16 ; état de signal initial = 1
OUT1	Sortie	BOOL	E, A, M, D, L	Bit de sortie 1
OUT2	Sortie	BOOL	E, A, M, D, L	Bit de sortie 2
OUT3	Sortie	BOOL	E, A, M, D, L	Bit de sortie 3
OUT4	Sortie	BOOL	E, A, M, D, L	Bit de sortie 4
OUT5	Sortie	BOOL	E, A, M, D, L	Bit de sortie 5
OUT6	Sortie	BOOL	E, A, M, D, L	Bit de sortie 6
OUT7	Sortie	BOOL	E, A, M, D, L	Bit de sortie 7
OUT8	Sortie	BOOL	E, A, M, D, L	Bit de sortie 8
OUT9	Sortie	BOOL	E, A, M, D, L	Bit de sortie 9
OUT10	Sortie	BOOL	E, A, M, D, L	Bit de sortie 10
OUT11	Sortie	BOOL	E, A, M, D, L	Bit de sortie 11

Tableau 5-5 Barillet d'événement avec masquage (FB85) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
OUT12	Sortie	BOOL	E, A, M, D, L	Bit de sortie 12
OUT13	Sortie	BOOL	E, A, M, D, L	Bit de sortie 13
OUT14	Sortie	BOOL	E, A, M, D, L	Bit de sortie 14
OUT15	Sortie	BOOL	E, A, M, D, L	Bit de sortie 15
OUT16	Sortie	BOOL	E, A, M, D, L	Bit de sortie 16
Q	Sortie	BOOL	E, A, M, D, L	L'état de signal « 1 » indique que la durée de la dernière étape a expiré.
OUT_WORD	Sortie	WORD	E, A, M, D, L, P	Mot dans lequel le barillet écrit les valeurs de sortie.
ERR_CODE	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
JOG_HIS	statique	BOOL	E, A, M, D, L	Bit d'historique de progression
EOD	statique	BOOL	E, A, M, D, L	L'état de signal « 1 » indique que la durée de la dernière étape a expiré.
DSP	statique	BYTE	E, A, M, D, L, P	Étape prédéfinie pour le barillet
DSC	statique	BYTE	E, A, M, D, L, P	Étape de barillet en cours
DCC	statique	DWORD	E, A, M, D, L, P	Décompte d'étape en cours
DTBP	statique	WORD	E, A, M, D, L, P	Base de temps prédéfinie du barillet
PREV_TIME	statique	DWORD	E, A, M, D, L	Temps système précédent
S_PRESET	statique	ARRAY OF WORD	E, A, M, D, L	Décompte prédéfini pour chaque étape [1 à 16] avec 1 décompte = 1 ms
OUT_VAL	statique	ARRAY OF BOOL	E, A, M, D, L	Valeurs de sortie pour chaque étape [1 à 16, 0 à 15]
S_MASK	statique	ARRAY OF BOOL	E, A, M, D, L	Masque configurable pour chaque étape [1 à 16, 0 à 15] Etats de signal initiaux = 1.

Informations d'erreur

Le barillet reste sur l'étape en cours si l'une des situations décrites au tableau 5-6 se présente. L'état de signal de ENO est mis à « 0 » et ERR_CODE prend l'une des valeurs suivantes :

Tableau 5-6 Situations d'erreur pour FB85

ERR_CODE	Explication
W#16#000B	Valeur LST_STEP inférieure à 1 ou supérieure à 16
W#16#000C	Valeur DSC inférieure à 1 ou supérieure à LST_STEP
W#16#000D	Valeur DSP inférieure à 1 ou supérieure à LST_STEP

Exemple

La figure 5-4 montre le mode de fonctionnement de l'opération DRUM. Si l'état de signal de E0.0 égale 1 (entrée activée), le bloc fonctionnel DRUM est exécuté. Dans cet exemple, le barillet progresse de l'étape 1 à l'étape 2. Les bits de sortie (OUT1, etc.) et OUT_WORD sont mis à « 1 » en fonction du masque configuré pour l'étape 2 et des bits OUT_VAL pour l'étape 2.

Si le bloc fonctionnel est exécuté sans erreur, l'état de signal de ENO et de A4.0 est mis à « 1 » et ERR_CODE prend la valeur W#16#0000.

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

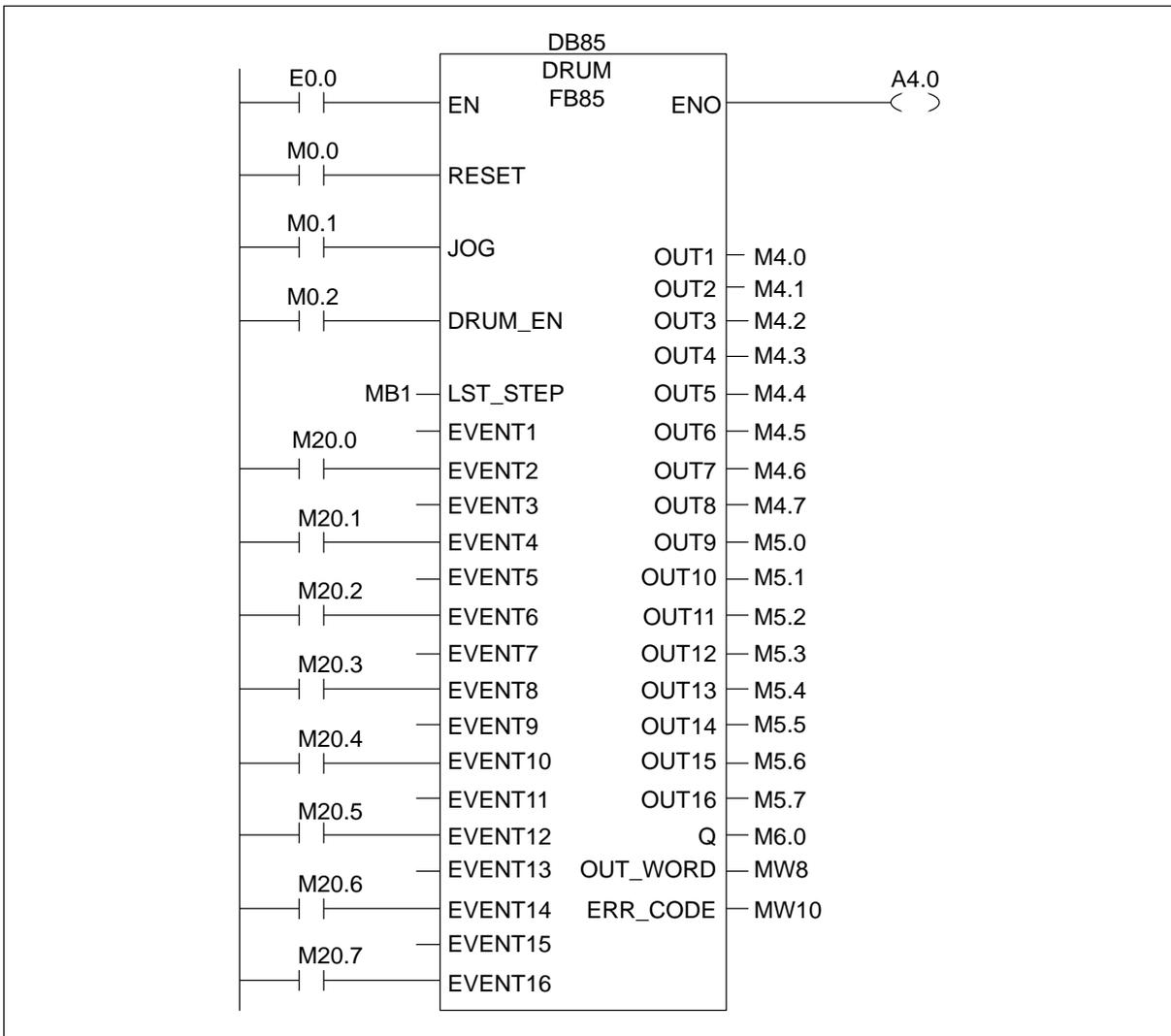


Figure 5-4 Barillet d'événement avec masquage (DRUM)

Fonctions et bloc fonctionnel de conversion

6

Ce chapitre décrit les fonctions (FC) et le bloc fonctionnel (FB) de conversion dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
6.1	Décodeur 7 segments (SEG) : FC93	6-2
6.2	Conversion ASCII-hexa (ATH) : FC94	6-4
6.3	Conversion hexa-ASCII (HTA) : FC95	6-6
6.4	Encoder position binaire (ENCO) : FC96	6-8
6.5	Décodeur position binaire (DECO) : FC97	6-9
6.6	Complément à 10 (BCDCPL) : FC98	6-10
6.7	Compter bits à 1 (BITSUM) : FC99	6-11
6.8	Mise à l'échelle (SCALE) : FC105	6-12
6.9	Annuler la mise à l'échelle (UNSCALE) : FC106	6-14
6.10	Algorithme d'avance et de retard de phase (LEAD_LAG) : FB80	6-16

6.1 Décodeur 7 segments (SEG) : FC93

Description

La fonction Décodeur 7 segments (SEG) convertit chacun des quatre chiffres hexadécimaux du mot source indiqué (IN) en quatre codes équivalents pour un affichage à 7 segments et les écrit dans le double mot de sortie (OUT).

La figure 6-1 montre la relation entre les chiffres hexadécimaux d'entrée et les profils binaires de sortie.

Chiffre	- g f e d c b a	Affichage
0 0 0 0	0 0 1 1 1 1 1 1	0
0 0 0 1	0 0 0 0 0 1 1 0	1
0 0 1 0	0 1 0 1 1 0 1 1	2
0 0 1 1	0 1 0 0 1 1 1 1	3
0 1 0 0	0 1 1 0 0 1 1 0	4
0 1 0 1	0 1 1 0 1 1 0 1	5
0 1 1 0	0 1 1 1 1 1 0 1	6
0 1 1 1	0 0 0 0 0 1 1 1	7
1 0 0 0	0 1 1 1 1 1 1 1	8
1 0 0 1	0 1 1 0 0 1 1 1	9
1 0 1 0	0 1 1 1 0 1 1 1	A
1 0 1 1	0 1 1 1 1 1 1 0	b
1 1 0 0	0 0 1 1 1 1 0 0 1	C
1 1 0 1	0 1 0 1 1 1 1 0	d
1 1 1 0	0 1 1 1 1 1 0 0 1	E
1 1 1 1	0 1 1 1 1 0 0 0 1	F

Affichage à sept segments

Figure 6-1 Profils binaires de sortie pour décodeur 7 segments

Paramètres

Le tableau 6-1 décrit les paramètres de la fonction SEG.

Tableau 6-1 Décodeur 7 segments (FC93) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	WORD	E, M, D, P ou constante	Mot de données source à quatre chiffres hexadécimaux.
OUT	Sortie	DWORD	A, M, D, L, P	Profil binaire destination sur quatre octets.

**Informations
d'erreur**

Cette fonction ne reconnaît aucune erreur.

Exemple

La figure 6-2 montre le mode de fonctionnement de l'opération SEG. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction SEG est exécutée.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

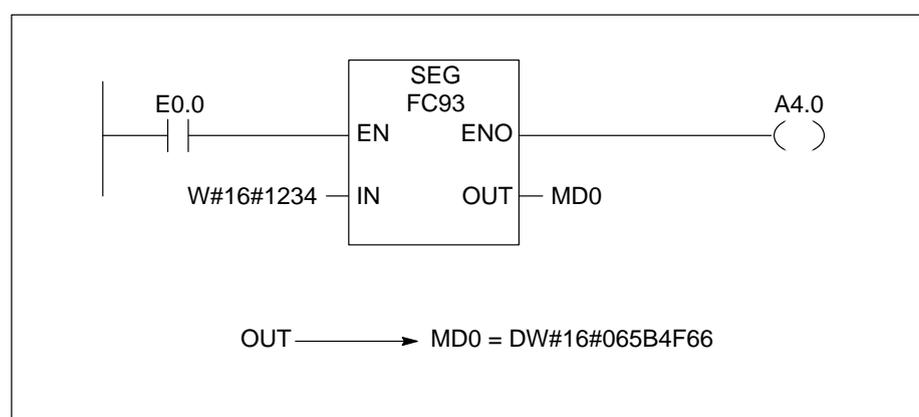


Figure 6-2 Décodeur 7 segments (SEG)

6.2 Conversion ASCII-hexa (ATH) : FC94

Description La fonction Conversion ASCII-hexa (ATH) convertit la chaîne de caractères ASCII désignée par le paramètre IN en chiffres hexadécimaux qu'elle range dans la table de destination désignée par le paramètre OUT. Comme un caractère ASCII nécessite 8 bits et un chiffre hexadécimal seulement 4 bits, la longueur du mot de sortie est inférieure de moitié à celle du mot d'entrée. Après la conversion, les caractères ASCII sont rangés dans la sortie hexadécimale dans le même ordre qu'à leur lecture. Si le nombre de caractères ASCII est impair, le chiffre hexadécimal du quartet de droite du chiffre hexadécimal converti en dernier est complété par des zéros.

Paramètres Le tableau 6-2 décrit les paramètres de la fonction ATH.

Tableau 6-2 Conversion ASCII-hexa (FC94) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	POINTER*	E, A, M, D, L	Pointe sur l'adresse de début d'une chaîne de caractères ASCII.
N	Entrée	INT	E, A, M, L, P	Nombre de caractères d'entrée ASCII devant être convertis.
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
OUT	Sortie	POINTER*	A, M, D, L	Pointe sur l'adresse de début de la table.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur Si un caractère ASCII incorrect est décelé, il est converti en « 0 ». L'état de signal de ENO est mis à « 0 » et la valeur en retour RET_VAL est égale à W#16#0007.

Exemple La figure 6-3 montre le mode de fonctionnement de l'opération ATH. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction ATH est exécutée. Le paramètre d'entrée N égal à 5 indique que 5 caractères ASCII doivent être convertis. Les caractères ASCII sont enregistrés dans le bloc de données 1 commençant à l'adresse indiquée par le pointeur IN : DB1.DBX10.0. La chaîne de sortie sera rangée à l'adresse indiquée par le pointeur OUT commençant à DB2.DBX0.0 (bloc de données 2). Comme le nombre de caractères ASCII est impair, le dernier chiffre hexadécimal ne contient que des zéros dans le quartet de droite, la valeur hexadécimale étant alors 0xC0 (pour la correspondance hexadécimale de chaque caractère ASCII, reportez-vous à la figure 6-4).

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et la valeur en retour RET_VAL est égale à W#16#000.

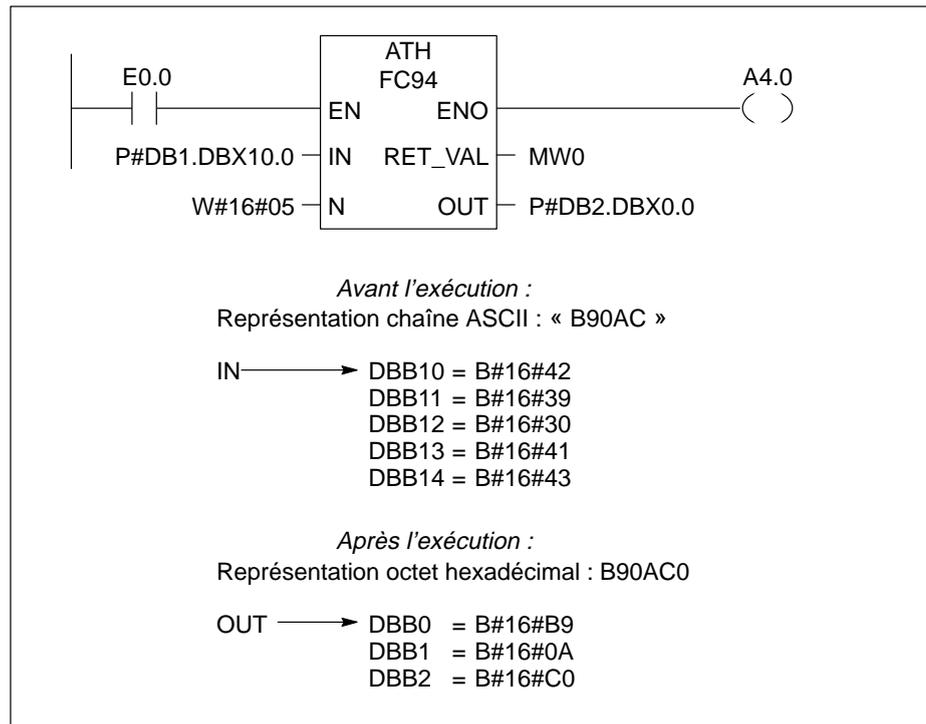


Figure 6-3 Conversion ASCII-hexa (ATH)

Caractères ASCII	Valeur hexadécimale ASCII	Chiffre hexadécimal converti
0	30	0
1	31	1
2	32	2
3	33	3
4	34	4
5	35	5
6	36	6
7	37	7
8	38	8
9	39	9
A	41	A
B	42	B
C	43	C
D	44	D
E	45	E
F	46	F

Figure 6-4 Caractères ASCII et valeurs hexadécimales correspondantes

6.3 Conversion hexa-ASCII (HTA) : FC95

Description

La fonction Conversion hexa-ASCII (HTA) convertit les chiffres hexadécimaux indiqués par le pointeur IN et les enregistre dans la chaîne de destination désignée par le paramètre OUT. Comme un caractère ASCII nécessite 8 bits et un chiffre hexadécimal seulement 4 bits, la longueur du mot de sortie est le double de celle du mot d'entrée. Chaque quartet du chiffre hexadécimal est converti en un caractère, et ce dans le même ordre qu'à la lecture : le quartet de gauche d'un chiffre hexadécimal est converti en premier, suivi par le quartet de droite du même chiffre).

Paramètres

Le tableau 6-3 décrit les paramètres de la fonction HTA.

Tableau 6-3 Conversion hexa-ASCII (FC95) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	POINTER*	E, A, M, D	Pointe sur l'adresse de début des chiffres hexadécimaux.
N	Entrée	WORD	E, A, M, L, P	Nombre des octets d'entrée hexadécimaux devant être convertis.
OUT	Sortie	POINTER*	A, M, D, L	Indique l'adresse de début de la table de destination.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

Cette fonction ne reconnaît aucune erreur.

Exemple

La figure 6-5 montre le mode de fonctionnement de l'opération HTA. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction HTA est exécutée. Le paramètre d'entrée N égal à 3 indique que trois chiffres hexadécimaux doivent être convertis. Les octets hexadécimaux sont enregistrés dans le bloc de données 1 commençant à l'adresse indiquée par le pointeur IN : DB1.DBX10.0. La chaîne de sortie sera rangée à l'adresse indiquée par le pointeur OUT commençant à DB2.DBX0.0 (bloc de données 2). Pour le caractère ASCII correspondant à chaque valeur hexadécimale, reportez-vous à la figure 6-6.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

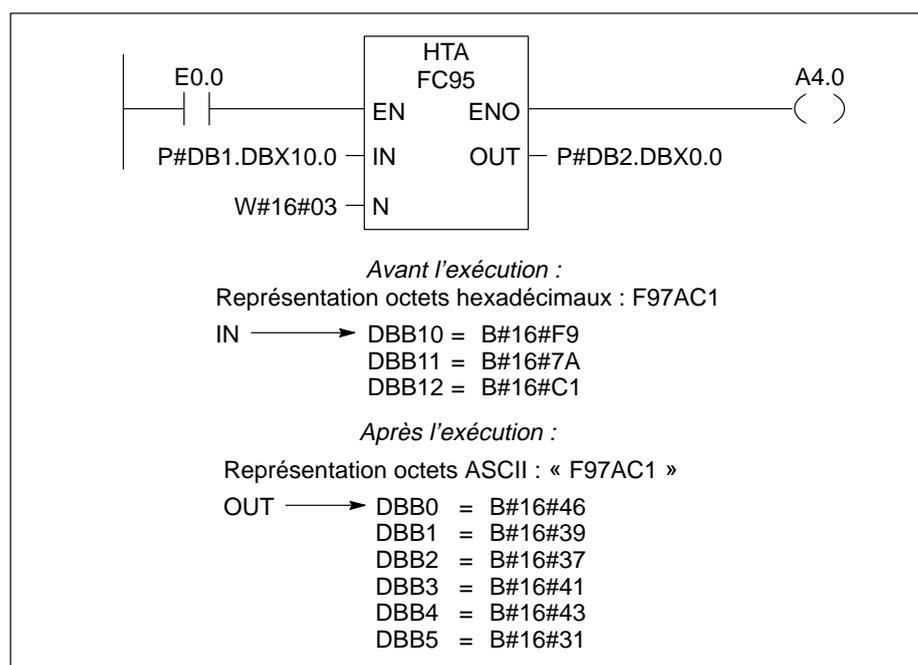


Figure 6-5 Conversion hexa-ASCII (HTA)

Chiffres hexadécimaux	Valeur hexadécimale ASCII	Caractère ASCII converti
0	30	0
1	31	1
2	32	2
3	33	3
4	34	4
5	35	5
6	36	6
7	37	7
8	38	8
9	39	9
A	41	A
B	42	B
C	43	C
D	44	D
E	45	E
F	46	F

Figure 6-6 Chiffres hexadécimaux et valeurs hexadécimales ASCII correspondantes

6.4 Encoder position binaire (ENCO) : FC96

Description La fonction Encoder position binaire (ENCO) convertit le contenu du paramètre IN au nombre binaire de 5 bits correspondant à la position du bit mis à 1 le plus à droite dans le paramètre IN et renvoie le résultat comme valeur de la fonction. Si le paramètre IN est égal à 0000 0001 ou à 0000 0000, la valeur en retour est « 0 ».

Paramètres Le tableau 6-4 décrit les paramètres de la fonction ENCO.

Tableau 6-4 Encoder position binaire (FC96) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	DWORD	E, M, D, L, P ou constante	Valeur devant être codée.
RET_VAL	Sortie	INT	A, M, D, L, P	Valeur en retour (contient un nombre binaire de 5 bits).

Informations d'erreur La fonction ne reconnaît aucune erreur.

Exemple La figure 6-7 montre le mode de fonctionnement de l'opération ENCO. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction ENCO est exécutée.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

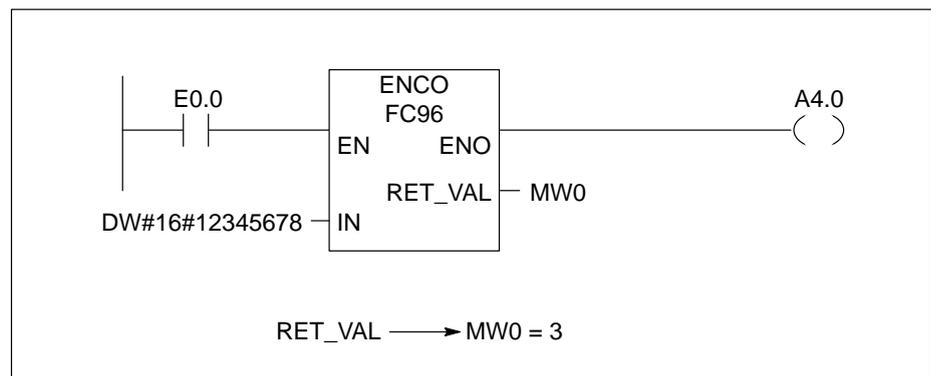


Figure 6-7 Encoder position binaire (ENCO)

6.5 Décoder position binaire (DECO) : FC97

Description La fonction Décoder position binaire (DECO) convertit un nombre binaire de 5 bits (0 à 31) de l'entrée IN en une valeur en mettant à 1 la position binaire correspondante dans la valeur en retour de la fonction. Si le paramètre IN est supérieur à 31, une opération modulo 32 est exécutée de façon à obtenir un nombre binaire de 5 bits.

Paramètres Le tableau 6-5 décrit les paramètres de la fonction DECO.

Tableau 6-5 Décoder position binaire (FC97) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	WORD	E, M, D, L, P ou constante	Variable devant être décodée.
RET_VAL	Sortie	DWORD	A, M, D, L, P	Valeur en retour

Informations d'erreur La fonction ne reconnaît aucune erreur.

Exemple La figure 6-8 montre le mode de fonctionnement de l'opération DECO. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction DECO est exécutée. Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

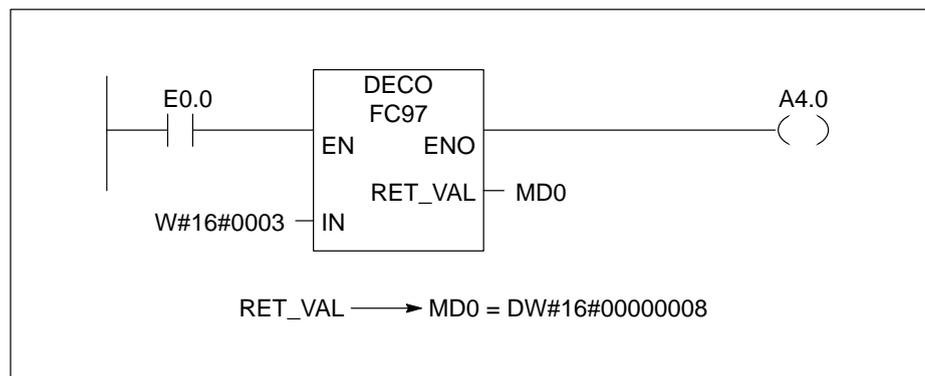


Figure 6-8 Décoder position binaire (DECO)

6.6 Complément à 10 (BCDCPL) : FC98

Description La fonction Complément à 10 (BCDCPL) renvoie le complément à 10 du nombre DCB à sept chiffres indiqué par le paramètre IN. Cette opération s'effectue selon la formule mathématique suivante :

$$\begin{aligned} & 10000000 \text{ (DCB)} \\ & - \text{valeur DCB à 7 chiffres} \\ & = \text{complément à 10 (DCB)} \end{aligned}$$

Paramètres Le tableau 6-6 décrit les paramètres de la fonction BCDCPL.

Tableau 6-6 Complément à 10 (FC98) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	DWORD	E, M, D, L, P ou constante	Nombre DCB à 7 chiffres
RET_VAL	Sortie	DWORD	A, M, D, L, P	Valeur en retour

Informations d'erreur La fonction ne reconnaît aucune erreur.

Exemple La figure 6-9 montre le mode de fonctionnement de l'opération BCDCPL. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction BCDCPL est exécutée.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

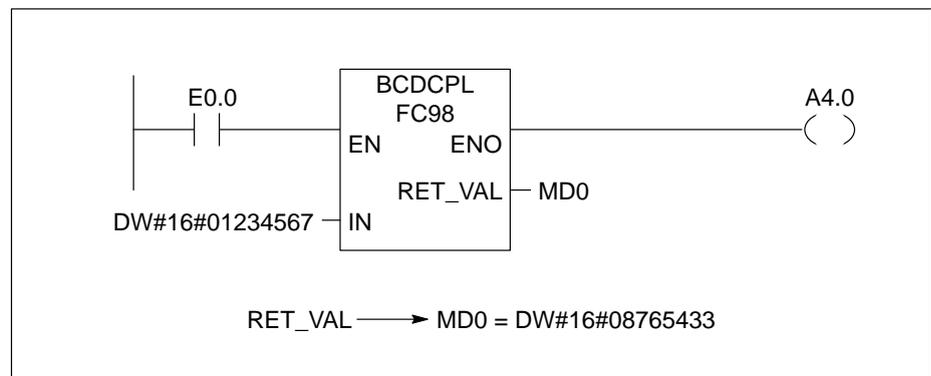


Figure 6-9 Complément à 10 (BCDCPL)

6.7 Compter bits à 1 (BITSUM) : FC99

Description La fonction Compter bits à 1 (BITSUM) compte le nombre de bits mis à « 1 » dans l'entrée IN et renvoie cette valeur comme valeur de la fonction.

Paramètres Le tableau 6-7 décrit les paramètres de la fonction BITSUM.

Tableau 6-7 Compter bits à 1 (FC99) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	DWORD	E, M, D, L, P ou constante	Variable dans laquelle les bits doivent être comptés.
RET_VAL	Sortie	INT	A, M, D, L, P	Valeur en retour

Informations d'erreur La fonction ne reconnaît aucune erreur.

Exemple La figure 6-10 montre le mode de fonctionnement de l'opération BITSUM. Si l'état de signal de l'entrée E 0.0 égale 1 (entrée activée), la fonction BITSUM est exécutée. Dans cet exemple, la valeur en retour dans MW0 est 13 (D en notation hexadécimale) : il s'agit de la somme des bits mis à « 1 » dans le double mot DW#16#12345678 (valeur hexadécimale).

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

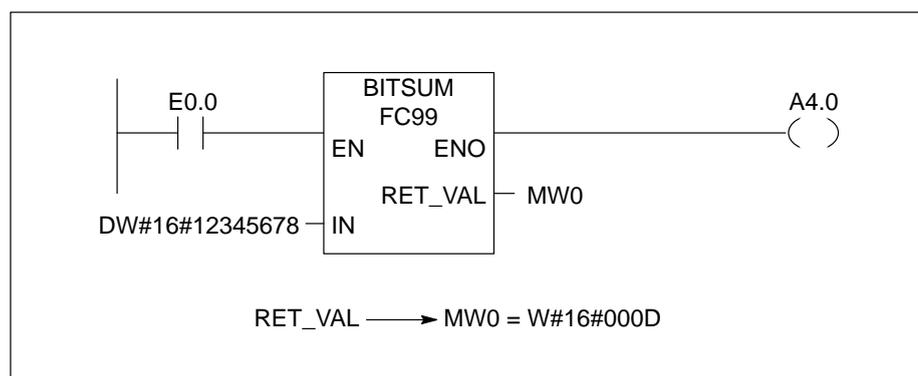


Figure 6-10 Compter bits à 1 (BITSUM)

6.8 Mise à l'échelle (SCALE) : FC105

Description

La fonction Mise à l'échelle (SCALE) prend une valeur entière (IN) et la convertit selon l'équation ci-après en une valeur réelle exprimée en unités physiques, comprises entre une limite inférieure (LO_LIM) et une limite supérieure (HI_LIM) :

$$\text{OUT} = [((\text{FLOAT}(\text{IN}) - \text{K1}) / (\text{K2} - \text{K1})) * (\text{HI_LIM} - \text{LO_LIM})] + \text{LO_LIM}$$

Le résultat est écrit dans OUT.

Les constantes K1 et K2 sont définies selon que la valeur d'entrée est bipolaire ou unipolaire.

- **Bipolaire :** La valeur entière d'entrée est supposée être comprise entre -27648 et 27648, donc :
K1 = -27648.0 et K2 = +27648.0
- **Unipolaire :** La valeur entière d'entrée est supposée être comprise entre 0 et 27648, donc :
K1 = 0.0 et K2 = +27648.0

Si la valeur entière d'entrée est supérieure à K2, la sortie (OUT) est saturée à la valeur la plus proche de la limite supérieure (HI_LIM) et une erreur est signalée. Si la valeur entière d'entrée est inférieure à K1, la sortie est saturée à la valeur la plus proche de la limite inférieure (LO_LIM) et une erreur est signalée.

Une mise à l'échelle inversée peut être obtenue en programmant une limite inférieure supérieure à la limite supérieure (LO_LIM > HI_LIM). Dans ce cas, la valeur de la sortie diminue quand la valeur de l'entrée augmente.

Paramètres

Le tableau 6-8 décrit les paramètres de la fonction SCALE.

Tableau 6-8 Mise à l'échelle (FC105) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	INT	E, A, M, D, L, P ou constante	Valeur d'entrée à convertir selon l'échelle en valeur réelle exprimée en unités physiques
HI_LIM	Entrée	REAL	E, A, M, D, L, P ou constante	Limite supérieure en unités physiques
LO_LIM	Entrée	REAL	E, A, M, D, L, P ou constante	Limite inférieure en unités physiques
BIPOLAR	Entrée	BOOL	E, A, M, D, L	L'état de signal « 1 » signifie que la valeur d'entrée est bipolaire et l'état de signal « 0 » qu'elle est unipolaire.
OUT	Sortie	REAL	E, A, M, D, L, P	Résultat de la conversion d'échelle
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.

Informations d'erreur

Si la valeur entière d'entrée est supérieure à K2, la sortie (OUT) est saturée à la valeur la plus proche de la limite supérieure (HI_LIM) et une erreur est signalée. Si la valeur entière d'entrée est inférieure à K1, la sortie est saturée à la valeur la plus proche de la limite inférieure (LO_LIM) et une erreur est signalée. L'état de signal de ENO est mis à « 0 » et RET_VAL prend la valeur W#16#0008.

Exemple

La figure 6-11 montre le mode de fonctionnement de l'opération SCALE. Si l'état de signal de E 0.0 égale 1 (entrée activée), la fonction SCALE est exécutée. Dans cet exemple, la valeur entière 22 sera convertie en une valeur réelle échelonnée entre 0.0 et 100.0 et écrite dans le paramètre de sortie OUT. La valeur d'entrée est bipolaire comme indiqué par l'état de signal de E 2.0.

Si la fonction est exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

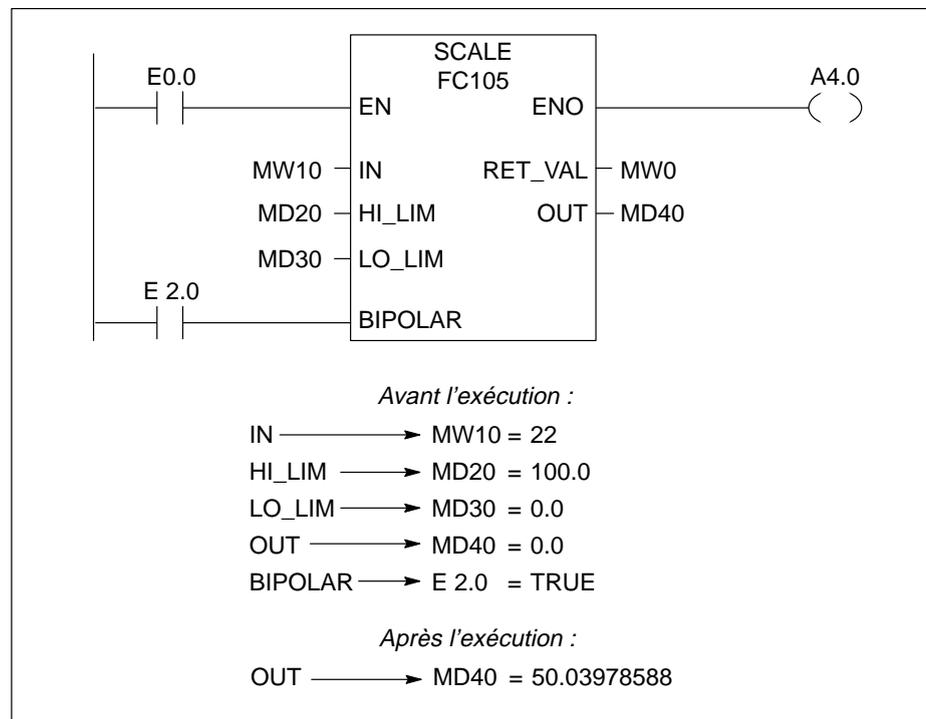


Figure 6-11 Mise à l'échelle (SCALE)

6.9 Annuler la mise à l'échelle (UNSCALE) : FC106

Description

La fonction Annuler la mise à l'échelle (UNSCALE) prend une valeur d'entrée réelle (IN) exprimée en unités physiques comprises entre une limite inférieure (LO_LIM) et une limite supérieure (HI_LIM) et la convertit selon l'équation ci-après en une valeur entière :

$$\text{OUT} = [((\text{IN}-\text{LO_LIM})/(\text{HI_LIM}-\text{LO_LIM})) * (\text{K2}-\text{K1})] + \text{K1}$$

Le résultat est écrit dans OUT.

Les constantes K1 et K2 sont définies selon que la valeur d'entrée est bipolaire ou unipolaire.

- **Bipolaire :** La valeur entière de sortie est supposée être comprise entre -27648 et 27648, donc :
K1 = -27648.0 et K2 = +27648.0
- **Unipolaire :** La valeur entière de sortie est supposée être comprise entre 0 et 27648, donc :
K1 = 0.0 et K2 = +27648.0

Si la valeur entière d'entrée se situe en dehors de la plage définie par les limites inférieure (LI_LIM) et supérieure (HI_LIM), la sortie (OUT) est saturée à la valeur la plus proche de la limite inférieure ou supérieure de la plage indiquée pour son type (bipolaire ou unipolaire) et une erreur est signalée.

Paramètres

Le tableau 6-9 décrit les paramètres de la fonction UNSCALE.

Tableau 6-9 Annuler la mise à l'échelle (FC106) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
IN	Entrée	REAL	E, A, M, D, L, P ou constante	Valeur d'entrée réelle à restaurer en valeur entière
HI_LIM	Entrée	REAL	E, A, M, D, L, P ou constante	Limite supérieure en unités physiques
LO_LIM	Entrée	REAL	E, A, M, D, L, P ou constante	Limite inférieure en unités physiques
BIPOLAR	Entrée	BOOL	E, A, M, D, L	L'état de signal « 1 » signifie que la valeur d'entrée est bipolaire et l'état de signal « 0 » qu'elle est unipolaire.
OUT	Sortie	INT	E, A, M, D, L, P	Résultat de l'annulation de la mise à l'échelle
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.

Informations d'erreur

Si la valeur réelle d'entrée se situe en dehors de la plage définie par les limites inférieure (LO_LIM) et supérieure (HI_LIM), la sortie (OUT) est saturée à la valeur la plus proche de la limite inférieure ou supérieure de la plage indiquée pour son type (bipolaire ou unipolaire) et une erreur est signalée. L'état de signal de ENO est mis à « 0 » et RET_VAL prend la valeur W#16#0008.

Exemple

La figure 6-12 montre le mode de fonctionnement de l'opération UNSCALE. Si l'état de signal de E 0.0 égale 1 (entrée activée), la fonction UNSCALE est exécutée. Dans cet exemple, la valeur réelle 50.03978588 qui avait été mise à l'échelle entre 0.0 et 100.0 (unités physiques) sera restaurée à sa valeur entière, puis écrite dans le paramètre de sortie OUT. La valeur d'entrée est bipolaire comme indiqué par l'état de signal de E 2.0.

Si la fonction est exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et RET_VAL est mis à la valeur W#16#0000.

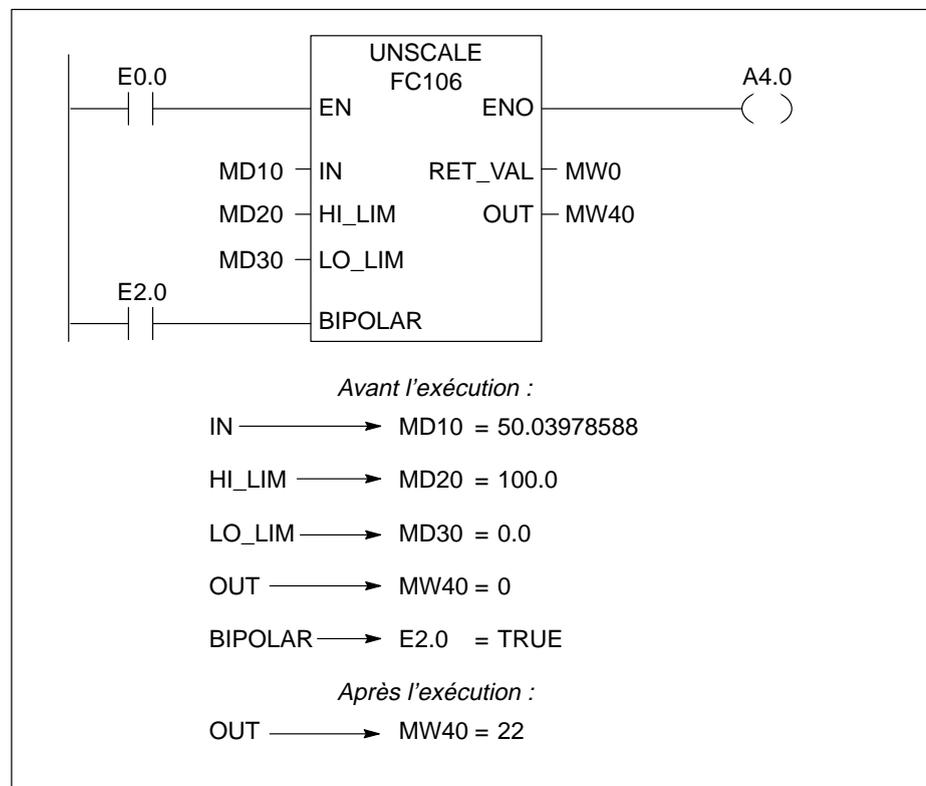


Figure 6-12 Annulation de la mise à l'échelle (UNSCALE)

6.10 Algorithme d'avance et de retard de phase (LEAD_LAG) : FB80

Description

Le bloc fonctionnel Algorithme d'avance et de retard de phase (LEAD_LAG) permet d'effectuer un traitement du signal sur une variable analogique. La sortie OUT est calculée selon l'entrée IN et les valeurs de gain GAIN, d'avance LD_TIME et de retard LG_TIME spécifiées. La valeur de gain doit être supérieure à zéro. L'algorithme LEAD_LAG utilise l'équation suivante :

$$\text{OUT} = \left[\frac{\text{LG_TIME}}{\text{LG_TIME} + \text{SAMPLE_T}} \right] \text{PREV_OUT} + \text{GAIN} \left[\frac{\text{LD_TIME} + \text{SAMPLE_T}}{\text{LG_TIME} + \text{SAMPLE_T}} \right] \text{IN} - \text{GAIN} \left[\frac{\text{LD_TIME}}{\text{LG_TIME} + \text{SAMPLE_T}} \right] \text{PREV_IN}$$

En général, LEAD_LAG est utilisé comme compensateur pour des boucles de commande dynamique. LEAD_LAG comporte deux parties. L'avance de phase décale et avance la phase de la sortie du bloc fonctionnel par rapport à la phase de l'entrée ; le retard de phase décale et retarde la phase de la sortie par rapport à la phase de l'entrée. Comme le retard de phase équivaut à une intégration, elle peut servir de suppresseur de bruit ou de filtre passe-bas. Quant à l'avance de phase, elle équivaut à une différenciation et constitue donc un filtre passe-haut. La combinaison d'avance et de retard de phase permet de retarder la phase de la sortie par rapport à la phase de l'entrée à basse fréquence et de l'avancer à haute fréquence, ce qui permet de réaliser un filtre passe-bande.

Paramètres

Le tableau 6-10 décrit les paramètres du bloc fonctionnel LEAD_LAG.

Tableau 6-10 Algorithme d'avance et de retard de phase (FB80) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
IN	Entrée	REAL	E, A, M, D, L, P ou constante	Valeur d'entrée à traiter pour la période d'échantillonnage en cours
SAMPLE_T	Entrée	INT	E, A, M, D, L, P ou constante	Instant d'échantillonnage
OUT	Sortie	REAL	E, A, M, D, L, P ou constante	Résultat de l'opération LEAD_LAG
ERR_CODE	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
LD_TIME	statique	REAL	E, A, M, D, L, P ou constante	Avance de phase en minutes
LG_TIME	statique	REAL	E, A, M, D, L, P ou constante	Retard de phase en minutes
GAIN	statique	REAL	E, A, M, D, L, P ou constante	Gain en rapport de pourcentages (rapport de la variation en sortie à la variation en entrée en régime établi)
PREV_IN	statique	REAL	E, A, M, D, L, P ou constante	Entrée précédente
PREV_OUT	statique	REAL	E, A, M, D, L, P ou constante	Sortie précédente

Informations d'erreur

Le bloc fonctionnel n'est pas exécuté si le gain est inférieur ou égal à zéro. L'état de signal de ENO est mis à « 0 » et ERR_CODE prend la valeur W#16#0009.

Exemple

La figure 6-13 montre le mode de fonctionnement de l'opération LEAD_LAG. Si l'état de signal de E 0.0 égale 1 (entrée activée), le bloc fonctionnel LEAD_LAG est exécuté. Dans cet exemple, la valeur d'entrée IN (2.0) est traitée à l'aide de l'algorithme LEAD_LAG qui fournit la sortie OUT.

Si le bloc fonctionnel est exécuté sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et ERR_CODE est mis à la valeur W#16#0000.

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

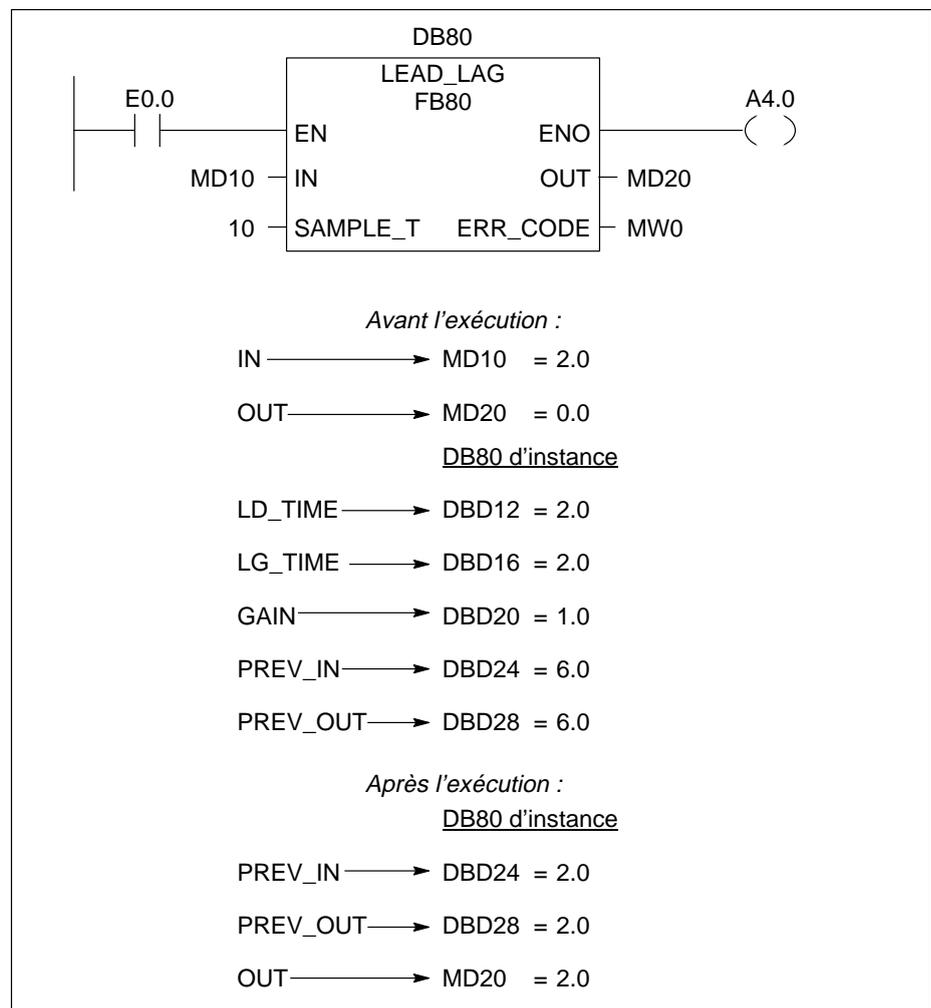


Figure 6-13 Algorithme d'avance et de retard de phase (LEAD_LAG)

Fonction arithmétique sur nombres à virgule flottante

7

Ce chapitre décrit la fonction arithmétique sur nombres à virgule flottante (FC) dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
7.1	Ecart type (DEV) : FC102	7-2

7.1 Ecart type (DEV) : FC102

Description

La fonction Ecart type (DEV) calcule l'écart type d'un ensemble de valeurs figurant dans la table TBL et range le résultat dans OUT. Le calcul se fait selon la formule suivante :

$$\text{Ecart type} = \sqrt{\frac{(N * \text{SommeCarrés}) - \text{Somme}^2}{N * (N - 1)}}$$

avec : Somme = Somme des valeurs dans TBL
 N = Nombre de valeurs dans TBL
 SommeCarrés = Somme des carrés de toutes les valeurs dans TBL

Des valeurs à virgule flottante IEEE sont utilisées pour tous les calculs, toutes les conversions de type nécessaires étant automatiquement effectuées par l'appel de fonction.

- La première entrée dans la table indique la longueur maximale de la table.
- La deuxième entrée dans la table contient la première valeur de la table.
- La taille des entrées de la table et de la valeur calculée (OUT) est déterminée par E_TYPE.

Paramètres

Le tableau 7-1 décrit les paramètres de la fonction DEV.

Tableau 7-1 Ecart type (FC102) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque la fonction a été exécutée sans erreur.
TBL	Entrée	POINTER*	E, A, M, D	Désigne l'adresse de début d'une table de valeurs.
OUT	Entrée	POINTER*	E, A, M, D	Désigne l'adresse de la valeur d'écart type calculée.
E_TYPE	Entrée	BYTE	E, A, M, D, L, P	Indique le type de données des entrées de la table. Les types de données admis pour la fonction DEV sont les suivants : B#16#05 = INT B#16#07 = DINT B#16#08 = REAL
RET_VAL	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.

* Pointeur en format double mot pour l'adressage indirect interzone par registre

Informations d'erreur

La fonction n'est pas exécutée si l'une des situations décrites au tableau 7-2 se présente. L'état de signal de ENO est mis à « 0 » et la valeur en retour prend l'une des valeurs suivantes :

Tableau 7-2 Situations d'erreur pour FC102

RET_VAL	Explication
W#16#0001	Indication d'un type de mémoire incorrect pour un paramètre
W#16#0002	Paramètre E_TYPE incorrect
W#16#0004	La longueur de la table est égale à zéro.

Exemple

La figure 7-1 montre le mode de fonctionnement de l'opération DEV. Si l'état de signal de E 0.0 égale 1 (entrée activée), la fonction DEV est exécutée. Dans cet exemple, il y a cinq valeurs dans la table comme indiqué par le premier mot de la table. Le paramètre E_TYPE précise que les valeurs dans la table sont de type de données REAL.

Si la fonction a été exécutée sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et la valeur en retour RET_VAL est égale à W#16#0000.

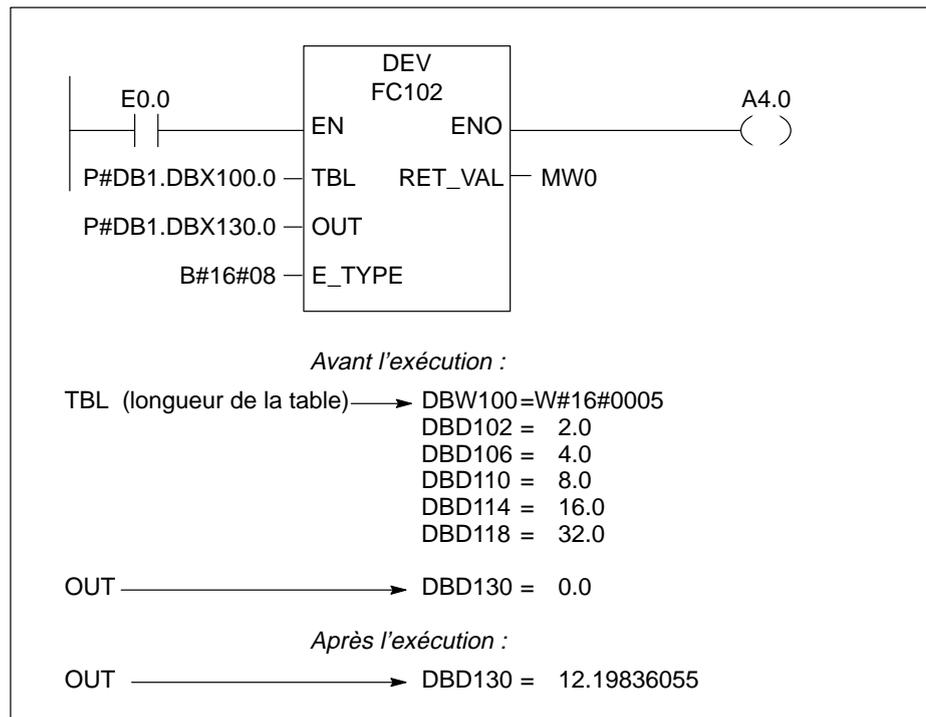


Figure 7-1 Ecart type (DEV)

Blocs fonctionnels de comparaison

8

Ce chapitre décrit les blocs fonctionnels de comparaison (FB) dont vous disposez en plus des opérations standard, vous offrant ainsi une plus grande souplesse lors de la programmation.

Paragraphe	Thème	Page
8.1	Comparaison de colonne de matrice (IMC) : FB83	8-2
8.2	Comparaison séquentielle de colonne de matrice (SMC) : FB84	8-6

8.1 Comparaison de colonne de matrice (IMC) : FB83

Description

Le bloc fonctionnel Comparaison de colonne de matrice (IMC) compare l'état de signal de 16 bits d'entrée programmés IN_BIT0 à IN_BIT15 (au maximum) aux bits de même indice d'un masque de comparaison. Cela constitue une étape de comparaison ; il est possible de programmer jusqu'à 16 étapes de comparaison avec masques. La comparaison se fait comme suit : IN_BIT0 est comparé à CMP_VAL [x,0], x étant le numéro d'étape, IN_BIT1 est comparé à CMP_VAL [x,1], etc. Le paramètre CMP_STEP précise le numéro d'étape du masque avec lequel doit se faire la comparaison. Les bits d'entrée non programmés ou les bits non programmés du masque prennent par défaut l'état de signal FALSE. S'il y a correspondance pour une étape donnée, l'état de signal de la sortie OUT est mis à « 1 » ; sinon, il est mis à « 0 ».

Paramètres

Le tableau 8-1 décrit les paramètres du bloc fonctionnel IMC.

Tableau 8-1 Comparaison de colonne de matrice (FB83) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
IN_BIT0	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 0 à comparer au bit 0 du masque
IN_BIT1	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 1 à comparer au bit 1 du masque
IN_BIT2	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 2 à comparer au bit 2 du masque
IN_BIT3	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 3 à comparer au bit 3 du masque
IN_BIT4	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 4 à comparer au bit 4 du masque
IN_BIT5	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 5 à comparer au bit 5 du masque
IN_BIT6	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 6 à comparer au bit 6 du masque
IN_BIT7	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 7 à comparer au bit 7 du masque
IN_BIT8	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 8 à comparer au bit 8 du masque
IN_BIT9	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 9 à comparer au bit 9 du masque
IN_BIT10	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 10 à comparer au bit 10 du masque
IN_BIT11	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 11 à comparer au bit 11 du masque
IN_BIT12	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 12 à comparer au bit 12 du masque
IN_BIT13	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 13 à comparer au bit 13 du masque
IN_BIT14	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 14 à comparer au bit 14 du masque
IN_BIT15	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 15 à comparer au bit 15 du masque
CMP_STEP	Entrée	BYTE	E, A, M, D, L, P	Numéro d'étape du masque avec lequel effectuer la comparaison
OUT	Sortie	BOOL	E, A, M, D, L	L'état de signal « 1 » signifie qu'il y a correspondance et l'état de signal « 0 » qu'aucune correspondance n'a été trouvée.
ERR_CODE	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
CMP_VAL	statique	ARRAY OF BOOL	E, A, M, D, L	Masques de comparaison [0 à 15, 0 à 15], le premier indice correspondant au numéro d'étape et le second au numéro de bit du masque.

**Informations
d'erreur**

Si la valeur de CMP_STEP est supérieure à 15, le bloc fonctionnel n'est pas exécuté. L'état de signal de ENO est mis à « 0 » et ERR_CODE prend la valeur W#16#000A.

Exemple

La figure 8-1 montre le mode de fonctionnement de l'opération IMC. Si l'état de signal de E 0.0 égale 1 (entrée activée), le bloc fonctionnel IMC est exécuté. Dans cet exemple, la totalité des 16 bits d'entrée est comparée au masque pour l'étape 2 (comme précisé par le paramètre CMP_STEP). L'état de signal de OUT prend la valeur TRUE, car les bits d'entrée correspondent à ceux du masque de cette étape.

Si le bloc fonctionnel est exécuté sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 » et ERR_CODE est mis à la valeur W#16#0000.

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

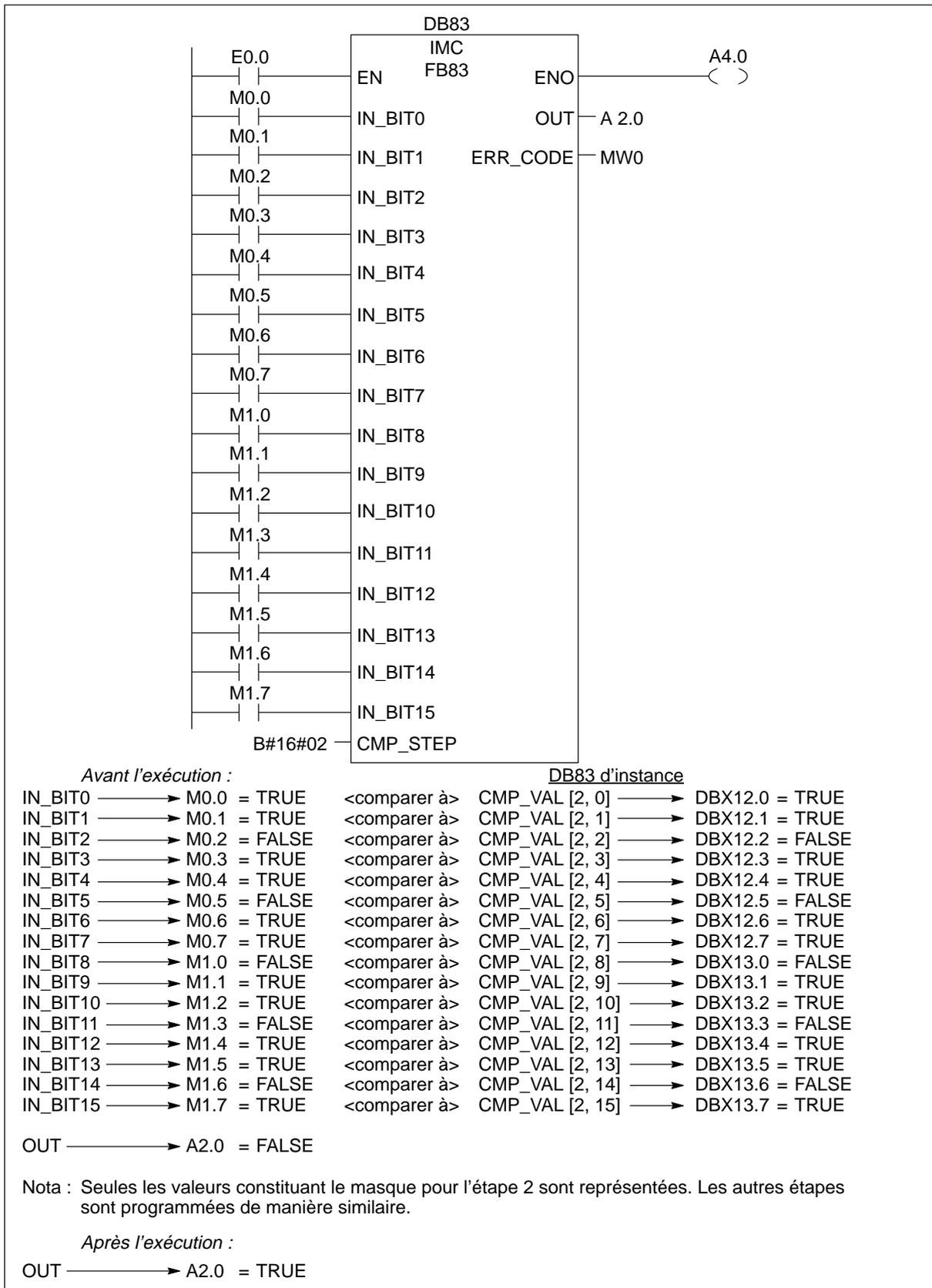


Figure 8-1 Comparaison de colonne de matrice (IMC)

8.2 Comparaison séquentielle de colonne de matrice (SMC) : FB84

Description

Le bloc fonctionnel Comparaison séquentielle de colonne de matrice (SMC) compare l'état de signal de 16 bits d'entrée programmés IN_BIT0 à IN_BIT15 (au maximum) aux bits de même indice du masque de comparaison de chaque étape, en commençant par la première étape et en progressant jusqu'à la dernière étape programmée (LAST) ou jusqu'à ce qu'une correspondance soit trouvée. La comparaison se fait de la manière suivante : IN_BIT0 est comparé à CMP_VAL [x,0], x étant le numéro d'étape, IN_BIT1 est comparé à CMP_VAL [x,1], etc. Lorsqu'une correspondance est trouvée, l'état de signal de la sortie OUT est mis à « 1 » et le numéro d'étape dont le masque correspond est écrit dans le paramètre OUT_STEP. Les bits d'entrée non programmés ou les bits non programmés des masques prennent par défaut l'état de signal FALSE. Par définition du bloc fonctionnel, si plus d'une étape comporte un masque qui correspond, seule la première est détectée et mémorisée dans OUT_STEP. Si aucune correspondance n'a été détectée, l'état de signal du paramètre de sortie OUT est mis à « 0 » et OUT_STEP prend la valeur LAST + 1.

Paramètres

Le tableau 8-2 décrit les paramètres du bloc fonctionnel SMC.

Tableau 8-2 Comparaison séquentielle de colonne de matrice (FB84) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
EN	Entrée	BOOL	E, A, M, D, L	Un état de signal « 1 » à l'entrée de validation active le cadre de fonction.
ENO	Sortie	BOOL	E, A, M, D, L	La sortie de validation a l'état de signal « 1 » lorsque le bloc fonctionnel a été exécuté sans erreur.
IN_BIT0	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 0 à comparer au bit 0 du masque
IN_BIT1	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 1 à comparer au bit 1 du masque
IN_BIT2	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 2 à comparer au bit 2 du masque
IN_BIT3	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 3 à comparer au bit 3 du masque
IN_BIT4	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 4 à comparer au bit 4 du masque
IN_BIT5	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 5 à comparer au bit 5 du masque
IN_BIT6	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 6 à comparer au bit 6 du masque
IN_BIT7	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 7 à comparer au bit 7 du masque
IN_BIT8	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 8 à comparer au bit 8 du masque
IN_BIT9	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 9 à comparer au bit 9 du masque
IN_BIT10	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 10 à comparer au bit 10 du masque
IN_BIT11	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 11 à comparer au bit 11 du masque
IN_BIT12	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 12 à comparer au bit 12 du masque
IN_BIT13	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 13 à comparer au bit 13 du masque
IN_BIT14	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 14 à comparer au bit 14 du masque
IN_BIT15	Entrée	BOOL	E, A, M, D, L	Bit d'entrée 15 à comparer au bit 15 du masque
OUT	Sortie	BOOL	E, A, M, D, L	L'état de signal « 1 » signifie qu'une correspondance a été trouvée et l'état de signal « 0 » qu'aucune correspondance n'a été trouvée.

Tableau 8-2 Comparaison séquentielle de colonne de matrice (FB84) : paramètres

Paramètres	Déclaration	Type de données	Zone de mémoire	Description
ERR_CODE	Sortie	WORD	E, A, M, D, L, P	Donne la valeur W#16#0000 en retour lorsque l'opération a été effectuée sans erreur. Pour toute valeur en retour autre que W#16#0000, reportez-vous aux informations d'erreur.
OUT_STEP	Sortie	BOOL	E, A, M, D, L, P	Contient le numéro d'étape dont le masque correspond ou la valeur LAST + 1 si aucune correspondance n'a été trouvée.
LAST	statique	BYTE	E, A, M, D, L, P	Précise le numéro de la dernière étape à examiner dans la séquence de recherche.
CMP_VAL	statique	ARRAY OF BOOL	E, A, M, D, L	Masques de comparaison [0 à 15, 0 à 15], le premier indice correspondant au numéro d'étape et le second au numéro de bit du masque.

Informations d'erreur

Le bloc fonctionnel n'est pas exécuté si la valeur de LAST est supérieure à 15. L'état de signal de ENO est mis à « 0 » et ERR_CODE prend la valeur W#16#000E.

Exemple

La figure 8-2 montre le mode de fonctionnement de l'opération SMC. Si l'état de signal de E 0.0 égale 1 (entrée activée), le bloc fonctionnel SMC est exécuté. Dans cet exemple, la totalité des 16 bits d'entrée est comparée aux masques pour les étapes 0 à 5 (comme précisé par le paramètre LAST) jusqu'à ce qu'une correspondance soit trouvée. Comme le masque pour l'étape 2 correspond aux bits d'entrée, seuls les masques des étapes 0 à 2 sont examinés.

Si le bloc fonctionnel est exécuté sans erreur, l'état de signal de ENO et de A 4.0 est mis à « 1 ».

Nota

Il est possible d'initialiser les paramètres statiques à l'aide de l'éditeur de bloc de données.

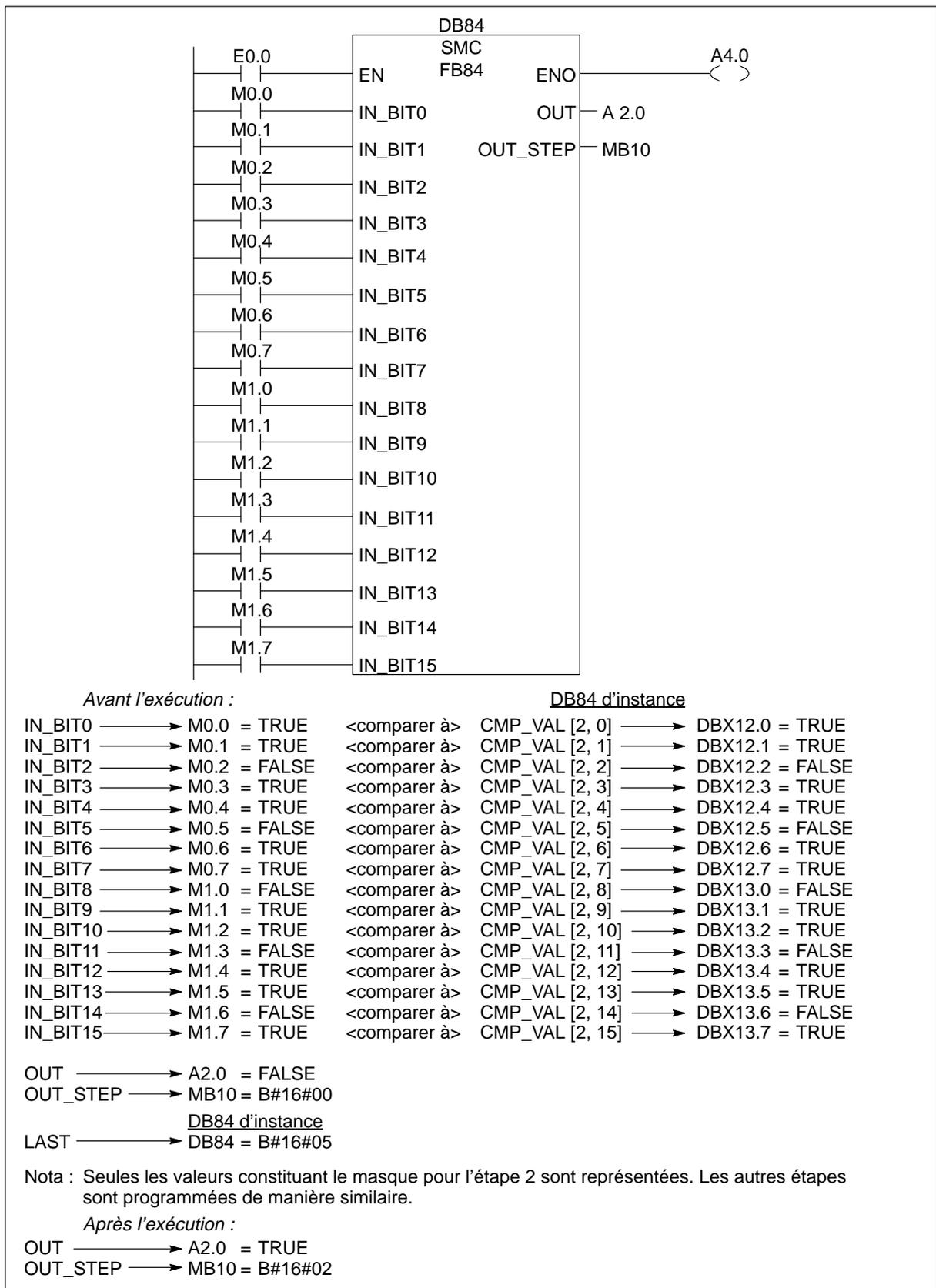


Figure 8-2 Comparaison séquentielle de colonne de matrice (SMC)

Glossaire

A

Abréviations utilisées

Les noms des opérandes et des opérations sont représentés par des abréviations mnémotechniques dans le programme, par exemple « E » pour « entrée » et « U » pour l'opération « ET ». STEP 7 prend en charge les abréviations internationales (sur la base de la langue anglaise) et les abréviations SIMATIC (sur la base de la représentation allemande des opérations et des conventions d'adressage SIMATIC).

Adressage absolu

L'adressage absolu indique l'adresse effective d'une unité de données particulière dans la mémoire d'une CPU. L'adressage absolu vous permet de référencer une E/S, par exemple, en utilisant une adresse indiquant le type de signal (E pour entrée, A pour sortie), le numéro du module de périphérie et le signal correspondant. Exemple : A 4.0. L'automate programmable interprète les adresses absolues sans l'aide d'une table de mnémoniques. Voir « Adressage symbolique ».

Adressage direct

Pour l'adressage direct, l'opérande d'une opération désigne directement l'adresse de la valeur sur laquelle l'opération doit porter. Voir « Adressage immédiat ».

Adressage immédiat

Dans l'adressage immédiat, la valeur effective sur laquelle l'opération doit porter est indiquée comme paramètre d'entrée. Cette valeur est l'opérande de l'opération. Voir « Adressage direct ».

Adressage symbolique

Dans la CPU, chaque élément a une adresse absolue (par exemple E 0.0). Mais, vous pouvez également créer des mnémoniques ou noms symboliques que vous pouvez utiliser pour l'adressage. Par exemple, vous pouvez attribuer à l'entrée E 1.3 le mnémonique « Retour_Pompe_2 ». Vous définissez les mnémoniques dans une table de mnémoniques que vous créez avec l'éditeur de mnémoniques.

B

Bit de résultat binaire

Le bit 8 du mot d'état est appelé bit de résultat binaire (bit RB). Ce bit constitue une liaison entre le traitement de bits et le traitement de mots. Avec ce bit, votre programme peut interpréter le résultat d'une opération sur mots comme résultat binaire et intégrer ce résultat dans une chaîne de combinaison binaire.

Le bit RB vous permet, par exemple, d'écrire un bloc fonctionnel (FB) ou une fonction (FC) en LIST (liste d'instructions ; voir le manuel *Langage LIST pour SIMATIC S7-300/400, Programmation de blocs*), puis d'appeler ce FB ou cette FC en CONT (schéma à contacts).

Lorsque vous écrivez un bloc fonctionnel ou une fonction que vous désirez appeler en CONT, vous devez gérer le bit RB et ce, indépendamment du langage de programmation (LIST ou CONT) utilisé pour écrire le FB et la FC. Le bit RB correspond à la sortie de validation (ENO) d'un pavé CONT. L'opération SAVE (en LIST) ou la bobine —(SAVE) (en CONT) vous permettent d'enregistrer le RLG dans le bit RB selon les critères suivants :

- Un résultat logique (RLG) de « 1 » est enregistré dans le bit RB lorsque le FB ou la FC a été exécutée sans erreur.
- Un résultat logique (RLG) de « 0 » est enregistré dans le bit RB lorsqu'une erreur s'est produite à l'exécution du FB ou de la FC.

Vous devez programmer ces opérations à la fin du FB ou de la FC de façon à ce qu'elles soient exécutées en dernier dans le bloc.



Attention

Le bit RB peut avoir été mis à « 0 » involontairement.

Lorsque vous écrivez des FB ou des FC en CONT et que vous ne gérez pas le bit RB comme décrit plus haut, un FB ou une FC peut écraser le bit RB d'un autre FB ou d'une autre FC.

Pour éviter cette erreur, enregistrez le RLG à la fin de chaque FB ou FC comme décrit plus haut.

Bloc de code

Les blocs de code sont des blocs dans STEP 7 contenant le programme pour la logique de commande. Il s'agit des blocs d'organisation (OB), des fonctions et des blocs fonctionnels (FC et FB), des fonctions système et des blocs fonctionnels système (SFC et SFB). Un bloc de données (DB) n'est pas considéré comme un bloc de code.

Bloc de données (DB)

Un bloc de données (DB) contient les données pour le programme utilisateur. Vous définissez la structure des informations enregistrées dans le bloc de données. Ces informations peuvent être soit utilisées par tous les blocs de code d'un programme, soit par une instance spécifique d'un FB (la structure du bloc de données dépendant alors de la table de déclaration des variables du FB).

Bloc de données d'instance	<p>Un bloc de données d'instance fournit de la mémoire pour un appel spécifique ou « instance » d'un bloc fonctionnel. Vous pouvez, en créant plusieurs instances (DB d'instance) d'un FB, utiliser le même FB pour commander plusieurs appareils.</p> <p>L'organisation d'un DB d'instance reflète la table de déclaration des variables d'un FB. Le DB d'instance mémorise les paramètres effectifs pour les variables IN, OUT, IN_OUT et VAR.</p>
Bloc fonctionnel (FB)	<p>Un bloc fonctionnel (FB) est un bloc de code contenant un segment de programme et disposant d'une zone de mémoire. Il faut indiquer un bloc de données d'instance à chaque appel de FB. Un FB peut être appelé plusieurs fois, à chaque fois avec un bloc de données d'instance différent. Les paramètres et les variables statiques du FB sont rangés dans le bloc de données d'instance.</p>
Bloc fonctionnel système (SFB)	<p>Un bloc fonctionnel système (SFB) est un bloc fonctionnel intégré au système d'exploitation S7. Vous pouvez appeler un SFB à partir de votre programme. Comme pour un FB, un SFB a une zone de mémoire propre dans laquelle des données peuvent être stockées jusqu'au prochain appel du SFB. Cette mémoire est réalisée sous forme de bloc de données d'instance (DB d'instance). Vous devez créer ce bloc de données (qui est ouvert comme partie de l'opération d'appel). Comme les SFB font partie du système d'exploitation, vous ne devez pas les charger.</p>
C	
CPU	<p>La CPU (Central Processing Unit : unité centrale) contient le programme utilisateur et traite les données pour l'automate programmable (AP).</p>
F	
Fonction (FC)	<p>Une FC est un bloc de code contenant un segment de programme mais qui ne dispose pas d'une zone de mémoire propre. Une fonction opère comme un sous-programme d'un programme d'ordinateur. Vous créez des FC et les appelez dans votre programme. Comme votre programme peut appeler une FC plusieurs fois (et donner différentes valeurs à chaque appel), une FC est définie comme bloc « réutilisable ». Après l'exécution de la FC, les données locales temporaires ayant été utilisées par la FC sont réallouées.</p>
Fonction système (SFC)	<p>Une fonction système (SFC) est une fonction testée préprogrammée et intégrée au système d'exploitation S7. Vous pouvez appeler une SFC à partir de votre programme. Comme les SFC sont une partie du système d'exploitation, elles ne nécessitent pas d'espace dans la mémoire principale. Comme les FC, les SFC n'utilisent pas de DB d'instance.</p>

I

Identificateur d'opérande

Un identificateur d'opérande est la partie de l'opérande d'une opération qui fournit les informations relatives à la zone de mémoire où l'opération trouve la valeur (objet de données) sur laquelle elle doit porter ainsi qu'à la taille de cette valeur. Pour l'opérande « EB10 », « EB » est l'identificateur d'opérande : « E » indique la zone de mémoire des entrées et « B » un octet dans cette zone.

L

Liste d'instructions

La liste d'instructions (LIST) est l'un des langages de programmation du progiciel STEP 7 vous permettant de communiquer avec votre automate programmable S7-300. Chaque instruction de votre programme comprend une opération dont l'abréviation mnémotechnique représente une fonction de l'automate programmable.

O

Opérande

L'opérande d'une opération du schéma à contacts indique une constante ou une adresse à laquelle l'opération trouve la valeur sur laquelle opérer. L'opérande peut être un mnémonique ou une adresse absolue ou une combinaison des deux. L'opérande peut désigner :

- une constante, la valeur d'une temporisation ou d'un compteur ou une chaîne de caractères ASCII,
- une adresse dans le mot d'état de l'automate programmable,
- un bloc de données et une adresse à l'intérieur de ce bloc de données,
- une fonction (FC), un bloc fonctionnel (FB), une fonction système intégrée (SFC) ou un bloc fonctionnel système intégré (SFB) et le numéro de la fonction ou du bloc,
- un repère pour une opération de saut,
- un identificateur d'opérande et une adresse dans la zone de mémoire indiquée par l'identificateur d'opérande (par exemple, E 1.0),
- le numéro d'une temporisation ou d'un compteur.

Opération

Une opération CONT indique à la CPU de votre automate programmable quelle fonction celui-ci doit exécuter. Les opérations CONT peuvent être représentées sous forme d'éléments ou de pavés.

P

- Paramètre effectif** Un paramètre effectif est une adresse ou une valeur fournie comme entrée ou sortie lors de l'appel d'un bloc fonctionnel (FB) ou d'une fonction (FC). Les paramètres effectifs correspondent aux paramètres formels déclarés dans la table de déclaration des variables du FB ou de la FC.
- Paramètre formel** Les paramètres formels sont déclarés dans la table de déclaration des variables d'un FB ou d'une FC. Lorsque vous appelez un FB ou une FC, vous devez fournir un paramètre effectif (adresse ou valeur) pour chaque paramètre formel.
- Pointeur** Un pointeur est un élément identifiant l'adresse d'une variable. Un pointeur contient une adresse à la place d'une valeur. Lorsque vous affectez un paramètre effectif au type de paramètre POINTER, vous indiquez l'adresse de mémoire. Dans STEP 7, vous pouvez indiquer le pointeur soit en format pointeur, soit simplement comme adresse (par exemple, M 50.0). L'exemple suivant montre un pointeur en format pointeur pour accéder aux données commençant à M 50.0 :
- P#M50.0
- Programme utilisateur** Le programme utilisateur contient la logique de commande pour un projet d'automatisation. Cette logique de commande est enregistrée sous forme d'opérations destinées à l'automate programmable pour la commande de l'installation ou du processus.
-
- R**
- Relais de masquage** Le relais de masquage (Master Control Relay, MCR) est un commutateur principal de logique à relais permettant l'activation ou la désactivation du flux d'énergie dans le circuit (trajet du courant). Un trajet du courant désexcité correspond à une séquence d'opérations écrivant la valeur zéro au lieu de la valeur calculée ou à une séquence d'opérations laissant inchangée la valeur existant en mémoire.
- Réseau** Dans un schéma à contacts STEP 7, un réseau est un circuit avec des opérations CONT. Il contient généralement des contacts d'entrée, des opérations sous forme de pavés et une opération de sortie à la fin de la ligne. Dans STEP 7, un trajet de courant dans un schéma à contacts constitue un réseau.

Résultat logique (RLG)

Le bit 1 du mot d'état est appelé bit RLG (résultat logique). Ce bit mémorise le résultat d'une combinaison sur bits ou d'une comparaison arithmétique. L'état de signal du bit RLG donne des informations relatives au flux d'énergie. L'état de signal « 1 » indique qu'il y a flux d'énergie (activé) ; l'état de signal « 0 » indique qu'il n'y a pas de flux d'énergie (désactivé).

Par exemple, la première opération dans un réseau CONT interroge l'état de signal d'un contact et donne le résultat « 1 » ou « 0 ». L'opération range ce résultat dans le bit RLG. La deuxième opération dans un réseau de combinaisons sur bits interroge également l'état de signal d'un contact et donne un résultat. Puis, elle combine ce résultat au bit RLG dans le mot d'état selon les règles de la logique booléenne. Le résultat de cette opération logique est enregistré dans le bit RLG du mot d'état et remplace la valeur précédemment mémorisée dans le bit RLG. Chacune des opérations suivantes dans le réseau effectue une combinaison de deux valeurs : le résultat produit lorsque l'opération interroge le contact et le RLG en cours.

Vous pouvez utiliser une combinaison sur bits lors d'une première interrogation pour affecter au RLG l'état du contenu d'une adresse de mémoire (bit). Vous pouvez également vous servir du RLG pour déclencher des opérations de saut.

S

Schéma à contacts (CONT)

Le schéma à contacts (CONT) est l'un des langages de programmation du progiciel STEP 7 vous permettant de programmer votre automate programmable (AP) S7-300. Le langage de programmation CONT utilise des symboles graphiques semblables aux éléments de relais de commande câblés.

T

Table de déclaration des variables

Tous les blocs de code ont une table de déclaration des variables. Lorsque vous entrez des informations dans cette table, vous déclarez (définissez) les paramètres et les variables utilisés par le bloc.

Types de données

Il est possible d'affecter un type aux données devant être utilisées dans un programme. Vous devez préciser un type de données lorsque vous définissez des mnémoniques avec l'éditeur de mnémoniques ou des variables locales dans la table de déclaration des variables. Le type de données définit la longueur et l'organisation des bits dans la mémoire réservée par la CPU.

- Types de données simples : BOOL (booléen), BYTE (octet), WORD (mot), DWORD (double mot), CHAR (caractère), INT (nombre entier de 16 bits), DINT (nombre entier de 32 bits), REAL (nombre à virgule flottante), TIME (durée), DATE (date), TOD (heure) et S5TIME (durée S5). Le système d'exploitation attribue à chaque type de données élémentaire une longueur fixe en mémoire. Un type de données booléen (BOOL), par exemple, dispose d'un bit, un octet (BYTE) de 8 bits, un mot (WORD) de 2 octets (ou 16 bits) et un double mot (DWORD) de 4 octets (ou 32 bits).
- Types de données complexes : DT (DATE_AND_TIME, date et heure), STRING (chaîne de 255 caractères au maximum), STRUCT (structure), UDT (type de données utilisateur) et ARRAY (tableau). Les types de données complexes dépassent généralement 32 bits (ou 4 octets). Vous pouvez combiner des types de données en définissant soit un groupe de types de données dans une structure (STRUCT), soit un même type de données plusieurs fois dans un tableau (ARRAY).
- Types de paramètres : TIMER (numéro de temporisation), COUNTER (numéro de compteur), BLOCK_[DB, FB, FC, SDB, SFC, SFB] (numéro du type de bloc identifié), POINTER (référence de pointeur à une adresse) ou ANY (type de données indéfini)

Z

Zone de mémoire

Une zone de mémoire est la zone de la CPU dans laquelle une opération trouve une valeur (objet de données) sur laquelle opérer. Votre automate programmable dispose des zones de mémoire suivantes que vous pouvez indiquer comme partie de l'opérande d'une opération :

- Mémoire image des entrées
- Mémoire image des sorties
- Mémentos
- Périphérie d'entrée et de sortie
- Temporisations
- Compteurs
- Blocs de données
- Données temporaires (données locales dynamiques)

Index

A

Ajouter valeur dans table (ATT), 2-2
Algorithme d'avance et de retard de phase
(LEAD_LAG), 6-16
Annuler la mise à l'échelle (UNSCALE), 6-14
Arithmétique sur nombres à virgule flottante, Ecart
type (DEV), 7-2
Assistance technique, v

B

Barillet d'événement avec masquage, 5-10
Blocs fonctionnels (FB)
copier, iii
liste, vi
Blocs fonctionnels de comparaison
Comparaison de colonne de matrice (IMC), 8-2
Comparaison séquentielle de colonne de matrice
(SMC), 8-6

C

Combinaison de bits
Mettre à un plage de sorties directes (SETI), 1-8
Mettre à un zone de mémentos ou de périphérie
dans la mémoire image (SET), 1-6
Remettre à zéro plage de sorties directes
(RSETI), 1-4
Remettre à zéro zone de mémentos ou de
périphérie dans la mémoire image (RSET),
1-2
Combiner valeur logiquement avec élément de
table et mémoriser (WRD_TBL), 2-15
Comparaison
Comparaison de colonne de matrice (IMC), 8-2
Comparaison séquentielle de colonne de matrice
(SMC), 8-6
Comparaison de colonne de matrice (IMC), 8-2
Comparaison séquentielle de colonne de matrice
(SMC), 8-6
Complément à 10 (BCDCPL), 6-10
Compter bits à 1 (BITSUM), 6-11

Conversion

Algorithme d'avance et de retard de phase
(LEAD_LAG), 6-16
Annuler la mise à l'échelle (UNSCALE), 6-14
Complément à 10 (BCDCPL), 6-10
Compter bits à 1 (BITSUM), 6-11
Conversion ASCII-hexa (ATH), 6-4
Conversion hexa-ASCII (HTA), 6-6
Décoder position binaire (DECO), 6-9
Décodeur 7 segments (SEG), 6-2
Encoder position binaire (ENCO), 6-8
Mise à l'échelle (SCALE), 6-12
Conversion ASCII-hexa (ATH), 6-4
Conversion hexa-ASCII (HTA), 6-6
Copie des fonctions (FC) et de blocs fonctionnels
(FB), iii
Copier valeur de la table (TBL_WRD), 2-13

D

Décalage
Déplacer bit vers registre à décalage (SHRB),
3-4
Déplacer mot vers registre à décalage (WSR),
3-2
Décoder position binaire (DECO), 6-9
Décodeur 7 segments (SEG), 6-2
Déplacer bit vers registre à décalage (SHRB), 3-4
Déplacer mot vers registre à décalage (WSR), 3-2
Dernière valeur entrée, première sortie (LIFO), 2-9

E

Ecart type (DEV), 7-2
Encoder position binaire (ENCO), 6-8
Exécuter opération sur table (TBL), 2-11
Exécuter opération sur tables et mémoriser dans
table cible (TBL_TBL), 2-19

F

- Fonction arithmétique sur nombres à virgule flottante, Ecart type (DEV), 7-2
- Fonction et bloc fonctionnel de transfert
 - Rassembler/répartir données de table (PACK), 4-4
 - Transfert indirect de blocs (IBLKMOV), 4-2
- Fonction et blocs fonctionnels de temporisation
 - Barillet d'événement avec masquage (DRUM), 5-10
 - Temporisation d'alarme avec commande moteur (MCAT), 5-7
 - Temporisation d'alarme avec commande tout ou rien (DCAT), 5-4
 - Temporisation sous forme de retard à la montée mémorisé (TONR), 5-2
- Fonctions (FC)
 - copier, iii
 - liste, vi
- Fonctions de combinaison de bits
 - Mettre à un plage de sorties directes (SETI), 1-8
 - Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET), 1-6
 - Remettre à zéro plage de sorties directes (RSETI), 1-4
 - Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET), 1-2
- Fonctions de décalage
 - Déplacer bit vers registre à décalage (SHRB), 3-4
 - Déplacer mot vers registre à décalage (WSR), 3-2
- Fonctions de table
 - Ajouter valeur dans table (ATT), 2-2
 - Combiner valeur logiquement avec élément de table et mémoriser (WRD_TBL), 2-15
 - Copier valeur de la table (TBL_WRD), 2-13
 - Dernière valeur entrée, première sortie (LIFO), 2-9
 - Exécuter opération sur table (TBL), 2-11
 - Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL), 2-19
 - Première valeur entrée, première sortie (FIFO), 2-4
 - Recherche de valeur dans table (TBL_FIND), 2-6
 - Tables de données corrélées (CDT), 2-17
- Fonctions et bloc fonctionnel de conversion
 - Algorithme d'avance et de retard de phase (LEAD_LAG), 6-16
 - Annuler la mise à l'échelle (UNSCALE), 6-14
 - Complément à 10 (BCDCPL), 6-10
 - Compter bits à 1 (BITSUM), 6-11
 - Conversion ASCII-hexa (ATH), 6-4

- Conversion hexa-ASCII (HTA), 6-6
- Décoder position binaire (DECO), 6-9
- Décodeur 7 segments (SEG), 6-2
- Encoder position binaire (ENCO), 6-8
- Mise à l'échelle (SCALE), 6-12

M

- Mettre à un plage de sorties directes (SETI), 1-8
- Mettre à un zone de mémentos ou de périphérie dans la mémoire image (SET), 1-6
- Mise à l'échelle (SCALE), 6-12

P

- Première valeur entrée, première sortie (FIFO), 2-4

R

- Rassembler/répartir données de table (Pack), 4-4
- Recherche de valeur dans table (TBL_FIND), 2-6
- Remettre à zéro plage de sorties directes (RSETI), 1-4
- Remettre à zéro zone de mémentos ou de périphérie dans la mémoire image (RSET), 1-2

T

Table

- Ajouter valeur dans table (ATT), 2-2
- Combiner valeur logiquement avec élément de table et mémoriser (WRD_TBL), 2-15
- Copier valeur de la table (TBL_WRD), 2-13
- Dernière valeur entrée, première sortie (LIFO), 2-9
- Exécuter opération sur table (TBL), 2-11
- Exécuter opération sur tables et mémoriser dans table cible (TBL_TBL), 2-19
- Première valeur entrée, première sortie (FIFO), 2-4
- Recherche de valeur dans table (TBL_FIND), 2-6
- Tables de données corrélées (CDT), 2-17
- Tables de données corrélées (CDT), 2-17
- Temporisation
 - Barillet d'événement avec masquage (DRUM), 5-10
- Temporisation d'alarme avec commande moteur (MCAT), 5-7
- Temporisation d'alarme avec commande tout ou rien (DCAT), 5-4
- Temporisation sous forme de retard à la montée mémorisé (TONR), 5-2
- Transfert
 - Rassembler/répartir données de table (PACK), 4-4
 - Transfert indirect de blocs (IBLKMOV), 4-2

Siemens AG
A&D AS E 81

Oestliche Rheinbrueckenstr. 50
D-76181 Karlsruhe
République Fédérale d'Allemagne

Expéditeur :

Vos . Nom : _ _ _ _ _
Fonction : _ _ _ _ _
Entreprise : _ _ _ _ _
Rue : _ _ _ _ _
Code postal : _ _ _ _ _
Ville : _ _ _ _ _
Pays : _ _ _ _ _
Téléphone : _ _ _ _ _

Indiquez votre secteur industriel :

- | | |
|---|---|
| <input type="checkbox"/> Industrie automobile | <input type="checkbox"/> Industrie pharmaceutique |
| <input type="checkbox"/> Industrie chimique | <input type="checkbox"/> Traitement des matières plastiques |
| <input type="checkbox"/> Industrie électrique | <input type="checkbox"/> Industrie du papier |
| <input type="checkbox"/> Industrie alimentaire | <input type="checkbox"/> Industrie textile |
| <input type="checkbox"/> Contrôle/commande | <input type="checkbox"/> Transports |
| <input type="checkbox"/> Construction mécanique | <input type="checkbox"/> Autres _ _ _ _ _ |
| <input type="checkbox"/> Pétrochimie | |



