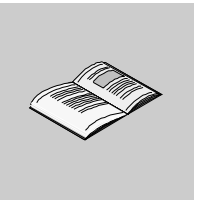
Concept 2.6
Bibliothèque de blocs IEC
Intercalaire : IEC

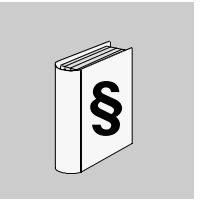
01/2007

33002230.03

Table des matières



Consignes de sécurité



Informations importantes

AVIS

Veuillez lire soigneusement ces consignes et examiner l'appareil afin de vous familiariser avec lui avant son installation, son fonctionnement ou son entretien. Les messages particuliers qui suivent peuvent apparaître dans la documentation ou sur l'appareil. Ils vous avertissent de dangers potentiels ou attirent votre attention sur des informations susceptibles de clarifier ou de simplifier une procédure.

L'apposition de ce symbole à un panneau de sécurité Danger ou Avertissement signale un risque électrique pouvant entraîner des lésions corporelles en cas de non-respect des consignes.

Ceci est le symbole d'une alerte de sécurité. Il vous avertit d'un risque de blessures corporelles. Respectez scrupuleusement les consignes de sécurité associées à ce symbole pour éviter de vous blesser ou de mettre votre vie en danger.

⚠ DANGER

DANGER indique une situation immédiatement dangereuse qui, si elle n'est pas évitée, **entraînera** la mort ou des blessures graves.

⚠ AVERTISSEMENT

AVERTISSEMENT indique une situation présentant des risques susceptibles de **provoquer** la mort, des blessures graves ou des dommages matériels.

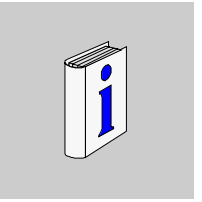
⚠ ATTENTION

ATTENTION indique une situation potentiellement dangereuse et susceptible d'**entraîner** des lésions corporelles ou des dommages matériels.

REMARQUE IMPORTANTE

Les équipements électriques doivent être installés, exploités et entretenus par un personnel d'entretien qualifié. Schneider Electric n'assume aucune responsabilité des conséquences éventuelles découlant de l'utilisation de cette documentation.

A propos de ce manuel



Présentation

Objectif du document

Cette documentation vous aidera à configurer les fonctions et les modules de fonctions.

Champ d'application

Cette documentation s'applique à la version 2.6 de Concept pour Microsoft Windows 98, Microsoft Windows Version 2000, Microsoft Windows XP ou Microsoft Windows NT 4.x.

Note : Vous trouverez d'autres notes actuelles dans le fichier README.WRI de Concept.

Document à consulter

Titre	Référence
Instructions d'installation de Concept	840 USE 502 01
Manuel utilisateur de Concept	840 USE 503 01
Concept EFB User Manual	840 USE 505 00
Bibliothèque de blocs LL984 de Concept	840 USE 506 01

Vous pouvez télécharger ces publications techniques ainsi que d'autres informations techniques à partir de notre site Web : www.telemecanique.com

Commentaires utilisateur

Envoyez vos commentaires à l'adresse e-mail techpub@schneider-electric.com

Présentation générale de la bibliothèque de modules IEC

I

Aperçu

Introduction

Cette section donne des informations générales sur la bibliothèque de modules IEC.

Contenu de cette partie

Cette partie contient les chapitres suivants :

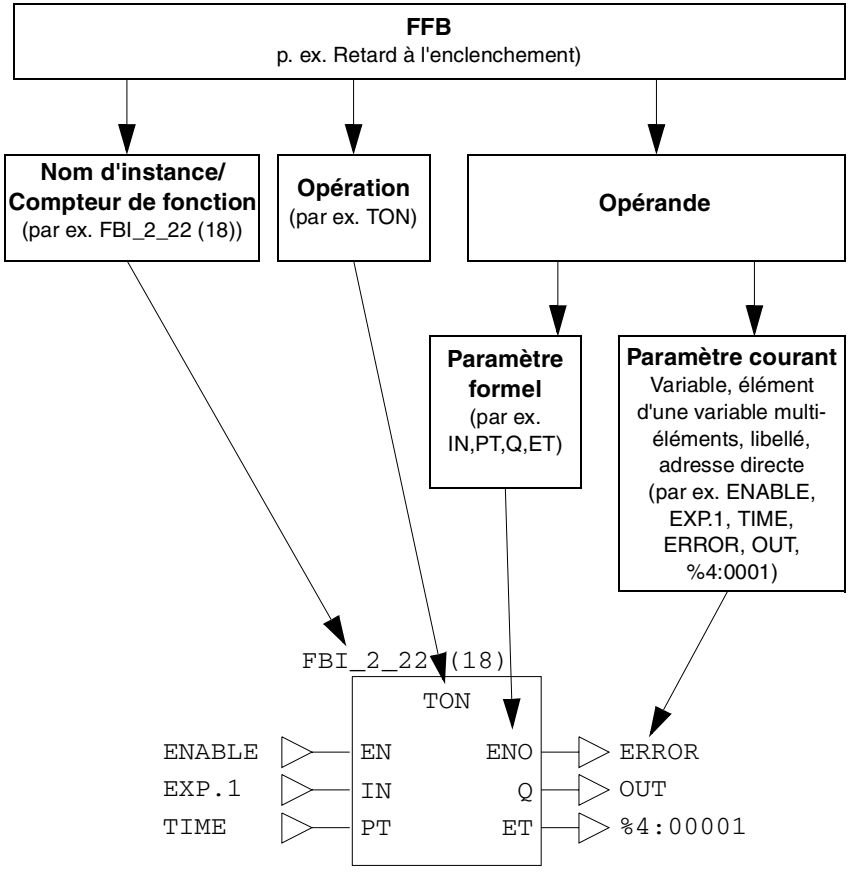
Paramétrage des fonctions et blocs fonction

1

Paramétrage des fonctions et blocs fonction

Généralités

Tout FFB se compose d'une opération, des opérandes nécessaires à l'opération et d'un nom d'instance/numéro de fonction.



Opération

L'opération détermine la fonctionnalité qui doit être exécutée par le FFB, p. ex. registre à décalage ou opérations de conversion.

Opérande

L'opérande détermine avec quoi l'opération doit être exécutée. Dans les FFB, il est constitué de paramètres formels et de paramètres réels.

Paramètre formel/paramètre réel

Le paramètre formel réserve la place pour un opérande. Lors du paramétrage, un paramètre actualisé (paramètre réel) est affecté au paramètre formel.

Le paramètre réel peut être une variable, une variable multi-éléments, un élément d'une variable multi-éléments, un libellé ou une adresse directe.

Lancement conditionnel/ inconditionnel

Chaque FFB peut disposer d'un lancement "conditionnel" ou "non conditionnel". La condition est réalisée par une connexion préalable de l'entrée EN.

- EN démasqué
  appel conditionnel (le FFB est traité uniquement lorsque EN = 1)
- EN masqué
  appel non conditionnel (le FFB est toujours traité)

Note : Si elle n'est pas paramétrée, l'entrée EN doit être masquée. Étant donné que les entrées non paramétrées sont automatiquement occupées par un "0", le FFB ne serait jamais exécuté.

Note : Dans le cas des blocs fonction bloqués (EN = 0) disposant d'une fonction temporelle interne (par exemple, DELAY), il semble que le temps continue de s'écouler, car il est calculé à l'aide de l'horloge système, le rendant indépendant du cycle programme et de la validation du bloc.

Appel de fonctions et DE blocs fonction en IL et ST

Pour l'appel des fonctions et des blocs fonction dans IL (liste d'instructions) et ST (littéral structuré), veuillez vous référer aux chapitres correspondants du manuel de l'utilisateur.

Descriptions des EFB

II

Aperçu

Introduction

Les descriptions des EFB sont classées par ordre alphabétique.

Note : Le nombre des entrées de certains EFBs peut être augmenté à 32 max. par modification verticale du symbole FFB. Veuillez consulter la description de chaque EFB pour savoir de quel EFB il s'agit.

Contenu de cette partie

Cette partie contient les chapitres suivants :

ABS_*** : Création de valeur absolue

2

Aperçu

Introduction

Ce chapitre décrit le module ABS_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction génère la valeur absolue de la valeur d'entrée et la délivre en sortie.

Le traitement concerne les types de données du groupe ANY_NUM

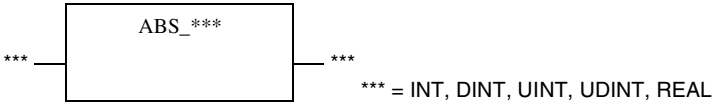
Les types de données de la valeur d'entrée et de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

$$OUT = |IN|$$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Type de données	Signification
IN	INT, DINT, UINT, UDINT, REAL	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

Si un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

ACOS_REAL : Arc cosinus en radian

3

Aperçu

Introduction

Ce chapitre décrit le module ACOS_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

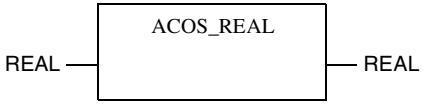
La fonction calcule l'arc cosinus de la valeur d'entrée et le délivre en sortie, exprimé en radians.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = arc cos IN

Conditions préalables requises :

$-1 \leq IN \leq 1$

soit :

$0 \leq OUT \leq \pi$

Description des paramètres

Description des paramètres de module

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	REAL	Valeur de sortie en radian

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de fonction, la plage de valeurs d'entrée n'est pas respectée ou en présence d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

ADD_*** : Addition

4

Aperçu

Introduction

Ce chapitre décrit le module ADD_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction additionne les valeurs d'entrée du groupe ANY_NUM ou du type de données TIME et délivre le résultat en sortie.

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

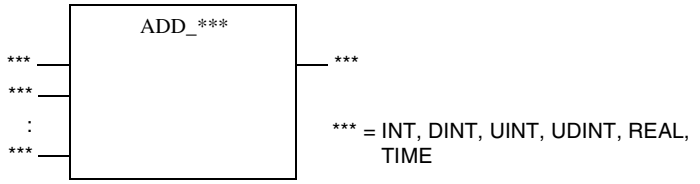
Il est possible d'augmenter le nombre d'entrées pour toutes les fonctions, sauf pour ADD_TIME.

Les paramètres EN et ENO peuvent être configurés en tant que paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

INT, DINT, UINT, UDINT, REAL :

OUT = IN1 + IN2 + .. INn

TIME :

OUT = IN1 + IN2

Description des paramètres

Description des paramètres de module

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME	Opérande
IN2	INT, DINT, UINT, UDINT, REAL, TIME	Opérande
INn	INT, DINT, UINT, UDINT, REAL	Opérande
OUT	INT, DINT, UINT, UDINT, REAL, TIME	Somme

Erreur d’exécution

Message d’erreur

Affichage d’un message d’erreur dans le cas où

- dépassement de la plage de valeurs de sortie,
- un paramètre d’entrée du type de données REAL est assorti d’un nombre en virgule flottante non admis.

AND_*** : Fonction ET

5

Aperçu

Introduction

Ce chapitre décrit le module AND_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction procède au chaînage (en logique ET) des séquences binaires aux entrées et délivre le résultat en sortie. Le chaînage s'effectue bit à bit.

Le traitement concerne les types de données du groupe ANY_BIT .

Note : Cette fonction n'est pas disponible, avec des variables de Boole, dans le langage de programmation LD (Ladder Diagram) , étant donné que cette même fonctionnalité peut être réalisée, dans ce cas, à l'aide de contacts .

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

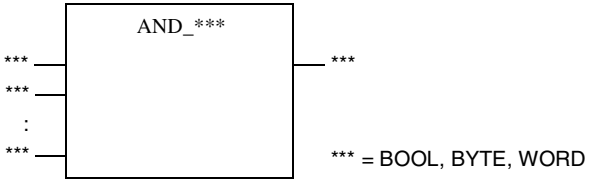
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 & IN2 & INn

Description des paramètres

Description des paramètres de module

Paramètres	Types de données	Signification
IN1	BOOL, BYTE, WORD	Séquence binaire d'entrée
IN2	BOOL, BYTE, WORD	Séquence binaire d'entrée
INn	BOOL, BYTE, WORD	Séquence binaire d'entrée
OUT	BOOL, BYTE, WORD	Séquence binaire de sortie

ASIN_REAL : Arc sinus en radian

6

Aperçu

Introduction

Ce chapitre décrit le module ASIN_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

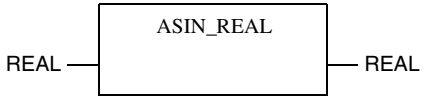
La fonction calcule l'arc sinus de la valeur d'entrée et délivre le résultat en sortie, exprimé en radians.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = arc sin (IN)

Condition :

IN -1 ≤ IN ≤ 1

Dans ce cadre est valable :

$$-\frac{\pi}{2} \leq \mathrm{OUT} \leq \frac{\pi}{2}$$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	REAL	Valeur de sortie en radian

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de fonction, la plage de valeurs d'entrée n'est pas respectée ou en présence d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

ATAN_REAL : Arc tangente en radian

7

Aperçu

Introduction

Ce chapitre décrit le module ATAN_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

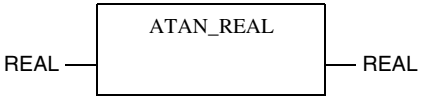
La fonction calcule l'arc tangente de la valeur d'entrée et délivre le résultat en sortie, exprimé en radians.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = arc tan (IN)

Dans ce cadre est valable :

$$-\frac{\pi}{2} \leq \mathrm{OUT} \leq \frac{\pi}{2}$$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	REAL	Valeur de sortie en radian

Erreur d'exécution

Message d'erreur

La présence d'un nombre en virgule flottante non admis à l'entrée entraîne l'affichage d'un message d'erreur.

BOOL_TO_*** : Conversion de type

8

Aperçu

Introduction

Ce chapitre décrit le module BOOL_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée de type de données BOOL en type de données du groupe ANY_NUM ou en types de données BYTE, WORD ou TIME.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

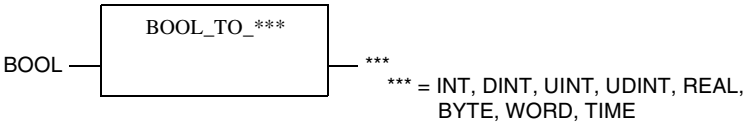
Le bit de poids faible de la valeur de sortie est pris pour paramétrer la valeur d'entrée. Tous les autres bits de la valeur de sortie sont positionnés sur zéro.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BOOL	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL, BYTE, WORD, TIME	Valeur de sortie

BYTE_TO_*** : Conversion de type

9

Aperçu

Introduction

Ce chapitre décrit le module BYTE_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée de type de données BYTE en type de données du groupe ANY_NUM ou en types de données BOOL, WORD ou TIME.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. BYTE_ TO_TIME. Ceci étant, il faut retenir que le profil de l'octet d'entrée est transféré dans le mot de poids fort du mot de sortie .

Lors de la conversion du type de données BYTE en type de données du groupe ANY_NUM ou WORD le profil binaire d'entrée est transféré dans les bits de poids faible de la sortie. Les bits de poids fort de la sortie sont positionnés sur zéro.

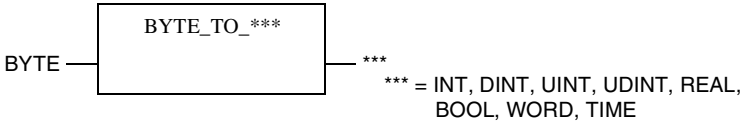
Lors de la conversion du type de données BYTE en type de données BOOL le bit de poids faible de la valeur d'entrée est transféré à la sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BYTE	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL, BOOL, WORD, TIME	Valeur de sortie

Erreur d'exécution

Message d'erreur

Lors de la conversion en type de données REAL la génération d'un nombre en virgule flottante non admis entraîne l'affichage d'un message d'erreur.

COS_REAL : Cosinus

10

Aperçu

Introduction

Ce chapitre décrit le module COS_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction calcule le cosinus de la valeur d'entrée et le délivre en sortie. La valeur d'entrée doit être saisie en radian.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :

Formule

OUT = cos (IN)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée en radian
OUT	REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

La présence d'un nombre en virgule flottante non admis à l'entrée entraîne l'affichage d'un message d'erreur.

CTD : Compteur dégressif

11

Aperçu

Introduction

Ce chapitre décrit le module CTD.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le bloc fonction sert au décompte des valeurs INT.

Les modules de fonction servant au comptage des valeurs DINT, UDINT et UINT se trouvent dans la bibliothèque Extended.

En présence d'un signal "1" à l'entrée LD, la valeur de l'entrée PV est attribuée à la sortie CV. Chaque fois que la valeur, à l'entrée CD, passe de "0" à "1", la valeur de CV est décrémentée de 1.

Si CV ≤ est à 0, la sortie Q est à "1".

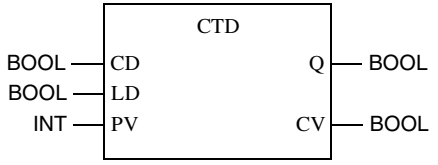
Note : Le compteur ne fonctionne que jusqu'à la valeur minimum de la sortie CV. Aucun débordement n'a lieu.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
CD	BOOL	Entrée Trigger
LD	BOOL	Chargement des données
PV	INT	Valeur de préréglage
Q	BOOL	Sortie
CV	INT	Valeur de comptage (valeur réelle)

CTU : Compteur progressif

12

Aperçu

Introduction

Ce chapitre décrit le module CTU.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le bloc fonction sert au compte progressif de valeursINT.

Les modules de fonction servant au comptage des valeurs DINT, UDINT et UINT se trouvent dans la bibliothèque Extended.

En présence d'un signal "1" à l'entrée R, la valeur "0" est attribuée à la sortie CV. Chaque fois que la valeur, à l'entrée CU, passe de "0" à "1", la valeur de CV est incrémentée de 1.

SI CV ≥ est égal à PV, la sortie Q passe à "1".

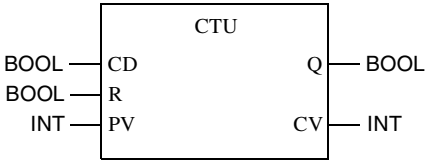
Note : Le compteur ne fonctionne que jusqu'à la valeur maximum de la sortie CV. Aucun débordement n'a lieu.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
CU	BOOL	Entrée Trigger
R	BOOL	Reset
PV	INT	Valeur de préréglage
Q	BOOL	Sortie
CV	INT	Valeur de comptage (valeur réelle)

CTUD : Compteur bidirectionnel

13

Aperçu

Introduction

Ce chapitre décrit le module CTUD.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction sert au comptage bidirectionnel des valeurs INT.

Les modules de fonction servant au comptage des valeurs DINT, UDINT et UINT se trouvent dans la bibliothèque Extended.

En présence d'un signal "1" à l'entrée R, la valeur "0" est attribuée à la sortie CV. En présence d'un signal "1" à l'entrée LD, la valeur de l'entrée PV est attribuée à la sortie CV. Chaque fois que la valeur, à l'entrée CU, passe de "0" à "1", la valeur de CV est incrémentée de 1. Chaque fois que la valeur, à l'entrée CD, passe de "0" à "1", la valeur de CV est décrémentée de 1.

En cas de présence simultanée du signal "1" aux entrées CU et CD, l'entrée CU (compteur progressif) est supérieure.

En cas de présence simultanée du signal "1" aux entrées R et LD, l'entrée R est supérieure.

Si CV ≥ est égal à PV, la sortie QU passe à "1".

Si CV ≤ est à 0, la sortie QD est à "1".

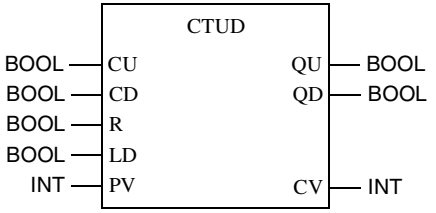
Note : Le compteur ne fonctionne que jusqu'à la valeur maximum (décomptage) ou maximum (comptage) de la sortie CV. Aucun débordement n'a lieu.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
CU	BOOL	Entrée de comptage
CD	BOOL	Entrée de décomptage
R	BOOL	Reset
LD	BOOL	Chargement des données
PV	INT	Valeur de préréglage
QU	BOOL	Affichage comptage
QD	BOOL	Affichage décomptage
CV	INT	Valeur de comptage (valeur réelle)

DINT_EXPT_REAL : Exponentiation

14

Aperçu

Introduction

Ce chapitre décrit le module DINT_EXPT_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

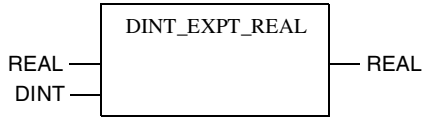
La fonction sert au calcul exponentiel. La valeur de l'entrée IN1 (base) fait l'objet d'un calcul exponentiel avec la valeur de l'entrée IN2 comme exposant, la puissance étant ensuite délivrée en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 exp IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	REAL	Base
IN2	DINT	Exposant
OUT	REAL	Puissance

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, dans le cas où

- (IN1 = 0) & (IN2 < 0),
- un nombre en virgule flottante non admis est affecté à IN1 ou
- la plage des valeurs est dépassée en sortie.

DINT_TO_*** : Conversion de type

15

Aperçu

Introduction

Ce chapitre décrit le module DINT_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée de type de données DINT en une valeur de sortie de type de données INT, UDINT, UINT, REAL, TIME, BOOL, BYTE ou WORD.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. DINT_ TO_BOOL.

Lors de la conversion de type de données DINT en type BOOL, BYTE ou WORD, les bits d'entrée de poids faible sont transmis à la sortie.

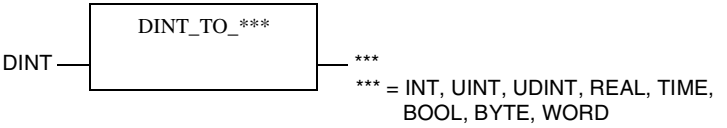
Les valeurs d'entrée négatives ne peuvent être converties en types de données UDINT ou UINT.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	DINT	Valeur d'entrée
OUT	INT, UDINT, UINT, REAL, TIME, BOOL, BYTE, WORD	Valeur de sortie

Erreur d’exécution

Message d’erreur

Affichage d’un message d’erreur, dans le cas où

- la plage de valeurs de sortie est dépassée ou
- une valeur d’entrée négative est susceptible d’être convertie en une valeur de sortie UDINT ou UINT.

DIV_*** : Division

16

Aperçu

Introduction

Ce chapitre décrit le module DIV_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction divise la valeur de l'entrée IN1 par celle de l'entrée IN2 et délivre le résultat en sortie.

Les types de données des valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Lors de la division de types de données du groupe ANY_INT la troncation d'une position après la virgule éventuellement présente est opérée vers zéro dans le résultat p.ex.

$7 \div 3 = 2$

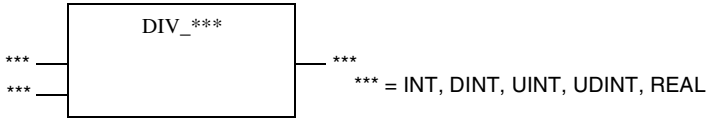
$(-7) \div 3 = -2$

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

$OUT = ((IN1) \div (IN2))$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL	Dividende
IN2	INT, DINT, UINT, UDINT, REAL	Diviseur
OUT	INT, DINT, UINT, UDINT, REAL	Quotient

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur dans le cas où

- $IN2 = 0$ ou
- un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis.

EQ_*** : Egal

17

Aperçu

Introduction

Ce chapitre décrit le module EQ_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction contrôle l'égalité des entrées, c.-à-d. que la sortie passe à "1", s'il y a égalité des valeurs sur toutes les entrées; sinon la sortie reste sur "0".

Les types de données de toutes les valeurs d'entrée doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

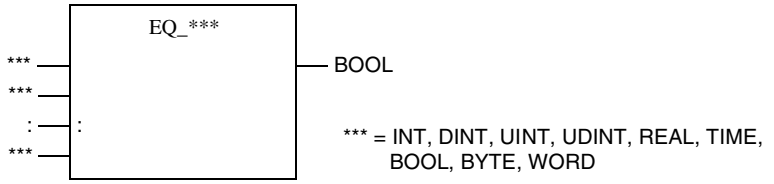
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = 1, si (IN1 = IN2) & (IN2 = IN3) & .. & ($IN_{(n-1)}$ = IN_n)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. entrée
OUT	BOOL	Sortie

Erreur d’exécution

Message d’erreur	Si un paramètre d’entrée du type de données REAL est assorti d’un nombre en virgule flottante non admis, il s’ensuit l’affichage d’un message d’erreur.

EXP_REAL : Fonction exponentielle

18

Aperçu

Introduction

Ce chapitre décrit le module EXP_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

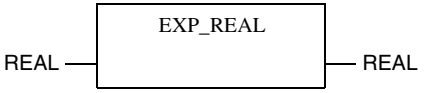
La fonction sert au calcul de la fonction exponentielle. La valeur d'entrée est utilisée comme exposant de la valeur de base e, la puissance étant délivrée en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = e exp IN

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Exposant de la base e
OUT	REAL	Puissance

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur dans le cas où

- un nombre en virgule flottante non admis est présent en entrée ou
- la plage des valeurs est dépassée en sortie.

F_TRIG : Détection de flancs descendants

19

Aperçu

Introduction

Ce chapitre décrit le module F_TRIG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction sert à la détection des flancs descendants 1 -> 0.

Note : Ce module de fonction n'est pas disponible en langage de programmation LD (Ladder Diagram), étant donné que celui-ci contient le "Contact - Flanc négatif", remplissant la même fonctionnalité.

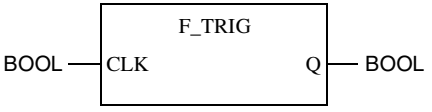
Lorsque l'entrée CLK passe de "1" à "0", la sortie Q passe à l'état "1". La sortie reste sur "1" d'un cycle d'exécution du module de fonction jusqu'au cycle suivant; ensuite, elle revient à "0".

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
CLK	BOOL	Entrée Horloge
Q	BOOL	Sortie

GE_*** : Supérieur/égal

20

Aperçu

Introduction

Ce chapitre décrit le module GE_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction contrôle les valeurs des entrées successives, d'après les critères d'ordre décroissant ou d'égalité.

Les types de données de toutes les valeurs d'entrée doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

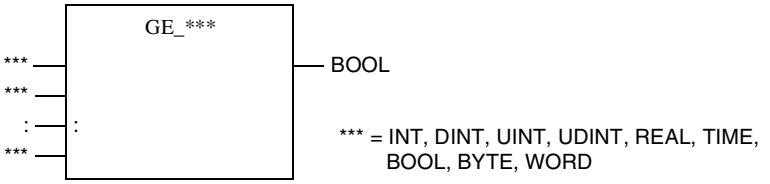
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = 1, si (IN1 ≥ IN2) & (IN2 ≥ IN3) & .. & ($IN_{(n-1)} \geq IN_n$)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. entrée
OUT	BOOL	Sortie

Erreur d'exécution

Message d'erreur

Si un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

GT_*** : Supérieur

21

Aperçu

Introduction

Ce chapitre décrit le module GT_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction contrôle les valeurs des entrées successives, d'après le critère d'ordre décroissant.

Les types de données de toutes les valeurs d'entrée doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

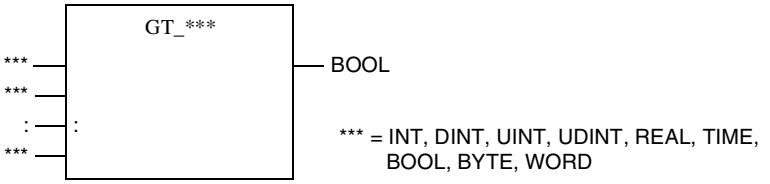
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = 1, si (IN1 > IN2) & (IN2 > IN3) & .. ($IN_{(n-1)}$ > IN_n)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. entrée
OUT	BOOL	Sortie

Erreur d’exécution

Message d’erreur	Si un paramètre d’entrée du type de données REAL est assorti d’un nombre en virgule flottante non admis, il s’ensuit l’affichage d’un message d’erreur.

INT_EXPT_REAL : Exponentiation

22

Aperçu

Introduction

Ce chapitre décrit le module INT_EXPT_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

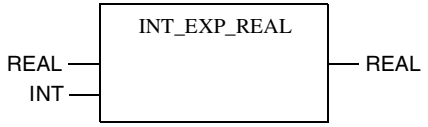
La fonction sert au calcul exponentiel. La valeur de l'entrée IN1 (base) fait l'objet d'un calcul exponentiel avec la valeur de l'entrée IN2 comme exposant, la puissance étant ensuite délivrée en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 exp (IN2)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	REAL	Base
IN2	INT	Exposant
OUT	REAL	Puissance

Erreur d'exécution

Message d'erreur

Un message d'erreur s'affiche, dans le cas où

- (IN1 = 0) & (IN2 < 0),
- un nombre en virgule flottante non admis est présent sur IN1 ou
- la plage des valeurs est dépassée en sortie.

INT_TO_*** : Conversion de type

23

Aperçu

Introduction

Ce chapitre décrit le module INT_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée du type de données INT en une valeur de sortie du type BOOL, BYTE, WORD, DINT, UDINT, UINT, REAL ou TIME.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. INT_TO_BOOL.

Les valeurs d'entrée négatives ne peuvent être converties en types de donnéesUDINT, UINT ou TIME.

Lors de la conversion d'une valeur d'entrée de type INT en type WORD, le profil binaire d'entrée est transmis à la sortie sans modification.

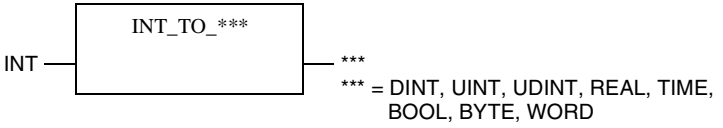
Lors de la conversion d'une valeur d'entrée de type INT en type de données BOOL ou BYTE, les bits d'entrée de poids faible sont transmis à la sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	INT	Valeur d'entrée
OUT	BOOL, BYTE, WORD, DINT, UDINT, UINT, REAL, TIME	Valeur de sortie

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, dans le cas où

- la plage de valeurs de sortie est dépassée ou
- une valeur d'entrée négative est susceptible d'être convertie en UDINT, UINT ou TIME.

LE_*** : Inférieur/égal

24

Aperçu

Introduction

Ce chapitre décrit le module LE_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction contrôle les valeurs des entrées successives, d'après les critères d'ordre croissant ou d'égalité .

Les types de données de toutes les valeurs d'entrée doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

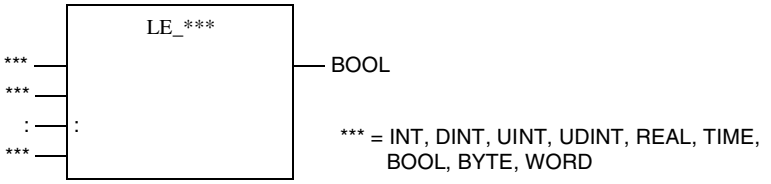
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = 1, si (IN1 ≤ In2) & (IN2 ≤ In3) & .. & ($IN_{(n-1)} \leq IN_n$)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. entrée
OUT	BOOL	Sortie

Erreur d'exécution

Message d'erreur

Si un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

LIMIT_*** : Limites

25

Aperçu

Introduction

Ce chapitre décrit le module LIMIT_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction transmet ,à la sortie, la valeur d'entrée inchangée (IN), si celle-ci n'est pas inférieure à la valeur minimum (MN) ni supérieure à la valeur maximum (MX). Si la valeur d'entrée (IN) est inférieure à la valeur minimum (MN), cette dernière est transmise à la sortie. Si la valeur d'entrée (IN) est supérieure à la valeur maximum (MX), cette dernière est transmise à la sortie.

Le traitement concerne les types de données du groupe ANY_ELEM.

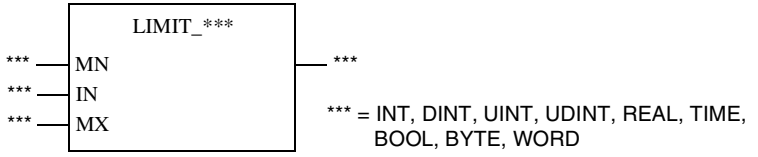
Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN, si (IN ≥ MN) & (IN ≤ MX)

OUT = MN, si (IN < MN)

OUT = MX, si (IN > MX)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
MN	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Valeur limite inférieure
IN	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Entrée
MX	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Valeur limite supérieure
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Sortie

Erreur d’exécution

Message d’erreur	La présence d’un nombre en virgule flottante non admis à l’entrée entraîne l’affichage d’un message d’erreur.

LN_REAL : Logarithme naturel

26

Aperçu

Introduction

Ce chapitre décrit le module LN_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

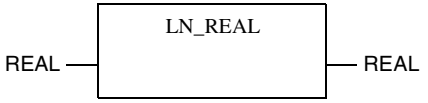
La fonction calcule le logarithme naturel de la valeur d'entrée et délivre le résultat en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = ln (IN)

Condition :

IN > 0

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de fonction, la plage de valeurs d'entrée n'est pas respectée ou en présence d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

LOG_REAL : Logarithme de base 10

27

Aperçu

Introduction

Ce chapitre décrit le module LOG_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction calcule le logarithme afférent à la base 10 de la valeur d'entrée et délivre le résultat en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :

Formule

OUT = log (IN)

Condition :

IN > 0

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de fonction, la plage de valeurs d'entrée n'est pas respectée ou en présence d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

LT_*** : Inférieur

28

Aperçu

Introduction

Ce chapitre décrit le module LT_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction contrôle les valeurs des entrées successives, d'après le critère d'ordre croissant.

Les types de données de toutes les valeurs d'entrée doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

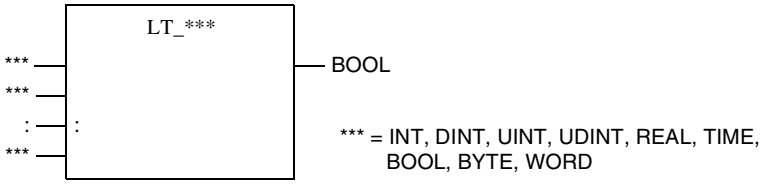
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = 1, si (IN1 < IN2) & (IN2 < IN3) & .. & ($IN_{(N-1)}$ < IN_n)

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Valeur d'entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Valeur d'entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. valeur d'entrée
OUT	BOOL	Valeur de sortie

Erreur d’exécution

Message d’erreur

Si un paramètre d’entrée du type de données REAL est assorti d’un nombre en virgule flottante non admis, il s’ensuit l’affichage d’un message d’erreur.

MAX_*** : Choix de valeur maximum

29

Aperçu

Introduction

Ce chapitre décrit le module MAX_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction délivre en sortie la plus grande valeur d'entrée.

Le traitement concerne les types de données du groupe ANY_ELEM.

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

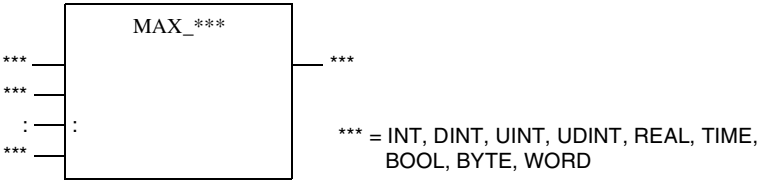
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = MAX {IN1, In2, .., In}

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Valeur d'entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Valeur d'entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Valeur maximale

Erreur d’exécution

Message d’erreur

Si un paramètre d’entrée du type de données REAL est assorti d’un nombre en virgule flottante non admis, il s’ensuit l’affichage d’un message d’erreur.

MIN_*** : Choix de valeur minimum

30

Aperçu

Introduction

Ce chapitre décrit le module MIN_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction délivre en sortie la plus petite valeur d'entrée.

Le traitement concerne les types de données du groupe ANY_ELEM.

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

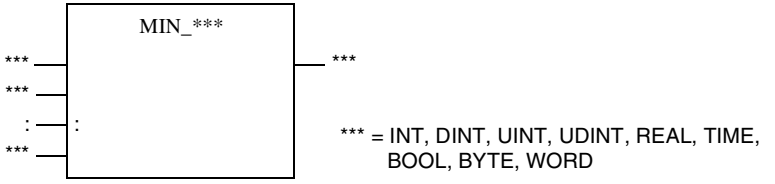
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = MIN {IN1, IN2, .., INn}

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Valeur d'entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Valeur d'entrée
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Valeur minimum

Erreur d’exécution

Message d’erreur	Si un paramètre d’entrée du type de données REAL est assorti d’un nombre en virgule flottante non admis, il s’ensuit l’affichage d’un message d’erreur.

MOD_*** : Modulo

31

Aperçu

Introduction

Ce chapitre décrit le module MOD_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction divise la valeur de l'entrée IN1 par celle de l'entrée IN2 et délivre le reste de division (Modulo) en sortie.

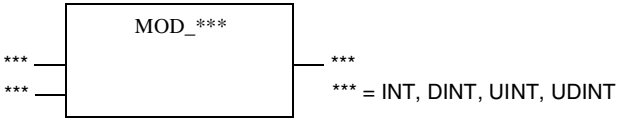
Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 mod IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT	Dividende
IN2	INT, DINT, UINT, UDINT	Diviseur
OUT	INT, DINT, UINT, UDINT	Modulo

MOVE : Assignation

32

Aperçu

Introduction

Ce chapitre décrit le module MOVE.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction assigne la valeur d'entrée à la sortie.

La fonction est générique, cela signifie que le type de données à traiter est déterminé par la variable affectée en premier à la fonction.

S'il s'agit d'assigner une adresse directe à une variable ou inversement, il est impératif de toujours assigner en premier la variable à la fonction. Il est interdit d'assigner une adresse directe à l'entrée et à la sortie de la fonction, le type de données ne pouvant, dans ce cas, être défini avec précision.

Les types de données de la valeur d'entrée et de la valeur de sortie doivent être identiques.

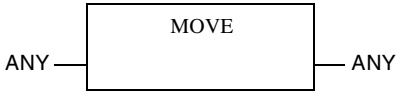
Note : En langage de programmation LD (Ladder Diagram) cette fonction ne peut être utilisée avec le type de données BOOL, étant donné que cette même fonctionnalité peut être réalisée, dans ce cas, à l'aide de constantes et de bobines.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	ANY	Valeur d'entrée
OUT	ANY	Valeur de sortie

MUL_*** : Multiplication

33

Aperçu

Introduction

Ce chapitre décrit le module MUL_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction multiplie les valeurs d'entrée et délivre le résultat en sortie.

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

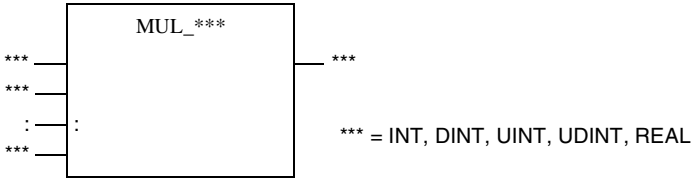
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

$OUT = IN1 \times IN2 \times .. \times IN_n$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL	Multiplicande (facteur)
IN2	INT, DINT, UINT, UDINT, REAL	Multiplicateur (facteur)
INn	INT, DINT, UINT, UDINT, REAL	Multiplicateur (facteur)
OUT	INT, DINT, UINT, UDINT, REAL	Produit

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur dans le cas où

- un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis ou
- la plage des valeurs est dépassée en sortie.

MUX_*** : Multiplexeur

34

Aperçu

Introduction

Ce chapitre décrit le module MUX_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction transmet à la sortie l'entrée adéquate, en fonction de la valeur de l'entrée K.

Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Exemple

K = 0 : L'entrée IN0 est transmise à la sortie

K = 1 : L'entrée IN1 est transmise à la sortie

K = 5 : L'entrée IN5 est transmise à la sortie

K = n : L'entrée INn est transmise à la sortie

Type de données

Le traitement concerne les types de données du groupe ANY.

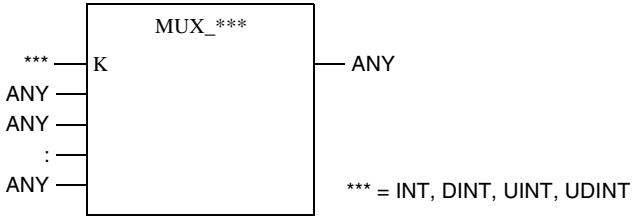
Les types de données des entrées IN0 à INn et en sortie doivent être identiques .

Pour le traitement des différents types de données(ANY_INT) de l'entrée K, l'on dispose respectivement d'une fonction particulière.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
K	INT, DINT, UINT, UDINT	Entrée de sélection
IN0	ANY	0. Entrée
IN1	ANY	1. Entrée
IN2	ANY	2. Entrée
INn	ANY	n. entrée
OUT	ANY	Sortie

NE_*** : Différent de

35

Aperçu

Introduction

Ce chapitre décrit le module NE_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction contrôle les inégalités des valeurs d'entrée.

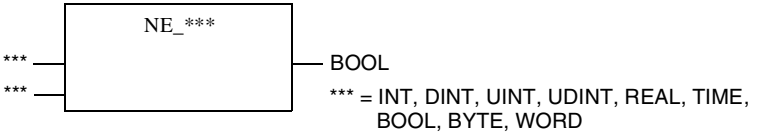
Les types de données des valeurs d'entrée doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = 1, si IN1 < > IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Entrée
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Entrée
OUT	BOOL	Sortie

Erreur d'exécution

Message d'erreur

Si un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

NOT_*** : Négation

36

Aperçu

Introduction

Ce chapitre décrit le module NOT_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction procède à l'inversion logique bit à bit de la séquence binaire d'entrée et délivre le résultat en sortie.

Le traitement concerne les types de données du groupe ANY_BIT .

Note : Cette fonction n'est pas disponible avec des variables de Boole, en langage de programmation LD (Ladder Diagram) , étant donné que cette même fonctionnalité peut, dans ce cas, être réalisée à l'aide de contacts (de fermeture) .

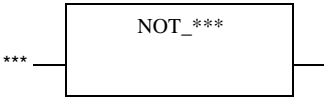
Les types de données de la valeur d'entrée et de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



*** = BOOL, BYTE, WORD

Formule

OUT = NOT IN

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BOOL, BYTE, WORD	Séquence binaire d'entrée
OUT	BOOL, BYTE, WORD	Séquence binaire inversée

OR_*** : Fonction OU

37

Aperçu

Introduction

Ce chapitre décrit le module OR_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction procède au chaînage (en logique OU) des séquences binaires aux entrées et délivre le résultat en sortie. Le chaînage s'effectue bit à bit.

Le traitement concerne les types de données du groupe ANY_BIT .

Note : Cette fonction n'est pas disponible, avec des variables de Boole, dans le langage de programmation LD (Ladder Diagram) , étant donné que cette même fonctionnalité peut être réalisée, dans ce cas, à l'aide de contacts .

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

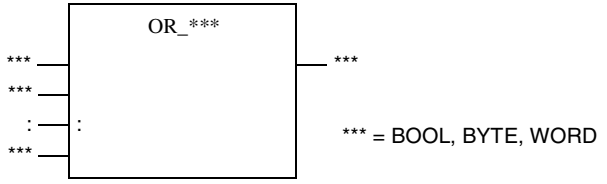
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 OR IN2 OR .. OR INn

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	BOOL, BYTE, WORD	Séquence binaire d'entrée
IN2	BOOL, BYTE, WORD	Séquence binaire d'entrée
INn	BOOL, BYTE, WORD	Séquence binaire d'entrée
OUT	BOOL, BYTE, WORD	Séquence binaire de sortie

R_TRIG : Détection de flancs montants

38

Aperçu

Introduction

Ce chapitre décrit le module R_TRIG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction sert à détecter les flancs montants 0 -> 1.

Note : En langage de programmation LD (Ladder Diagram) ce module de fonction n'est pas disponible, étant donné que celui-ci contient le "Contact - Flanc positif", remplissant la même fonctionnalité.

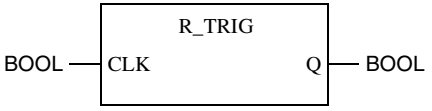
Lorsque l'entrée CLK passe de "1" à "0", la sortie Q passe à l'état "1". La sortie reste à "1" d'une séquence d'exécution de la fonction à la suivante (un cycle); ensuite, la sortie revient à "0".

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
CLK	BOOL	Entrée Horloge
Q	BOOL	Sortie

REAL_EXPT_REAL : Exponentiation

39

Aperçu

Introduction

Ce chapitre décrit le module REAL_EXPT_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

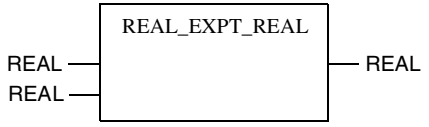
La fonction sert au calcul exponentiel. La valeur de l'entrée IN1 (base) fait l'objet d'un calcul exponentiel avec la valeur de l'entrée IN2 comme exposant, la puissance étant ensuite délivrée en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 exp IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	REAL	Base
IN2	REAL	Exposant
OUT	REAL	Puissance

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, dans le cas où

- un nombre en virgule flottante est présent sur l'une des entrées ou
- la plage des valeurs est dépassée en sortie.

REAL_TO_*** : Conversion de type

40

Aperçu

Introduction

Ce chapitre décrit le module REAL_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée du type de données REAL en type de données du groupe ANY_BIT ou ANY_INT ou en type de données TIME.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. REAL_TO_BOOL.

Lors de la conversion en ANY_BIT les bits de poids faible de la valeur d'entrée sont transmis à la sortie.

Note : Pour la conversion de REAL en WORD, l'on dispose également des modules R_INT_WORD et R_UINT_WORD de la bibliothèque de modules ANA_IO et du module REAL_AS_WORD de la bibliothèque de modules EXTENDED.

Lors de la conversion en ANY_INT et TIME les valeurs sont arrondies conformément aux règles spécifiées par CEI 559.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Exemple

L'exemple suivant montre la manière d'arrondir selon CEI 559.

1,4 -> 1

1,5 -> 2

2,5 -> 2

3,5 -> 4

4,5 -> 4

4,6 -> 5

Types de données

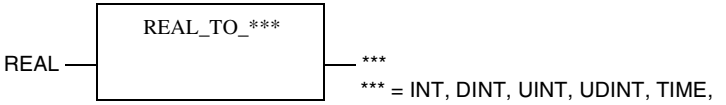
Les valeurs d'entrée négatives ne peuvent être converties en types de donnéesUDINT, UINT ou TIME.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT, TIME, BOOL, BYTE, WORD	Valeur de sortie

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, dans le cas où

- un nombre en virgule flottante non admis est présent à l'entrée,
- la plage de valeurs de sortie est dépassée ou
- une valeur d'entrée négative est susceptible d'être convertie en UDINT, UINT ou TIME.

REAL_TRUNC_*** : Conversion de type

41

Aperçu

Introduction

Ce chapitre décrit le module REAL_TRUNC_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit (avec troncation vers zéro) une valeur d'entrée du type de données REAL en type de données du groupe ANY_INT.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Exemple

L'exemple suivant montre la manière de convertir.

1,6 -> 1

-1,6 -> -1

1,4 -> 1

-1,4 -> -1

Types de données

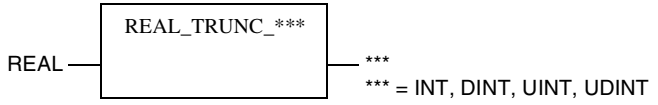
Les valeurs d'entrée négatives ne peuvent être converties en types de données UDINT ou UINT.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT	Valeur de sortie

Erreur d’exécution

Message d’erreur

Affichage d’un message d’erreur, dans le cas où

- une valeur d’entrée négative est susceptible d’être convertie en UDINT ou UINT,
- un nombre en virgule flottante non admis est présent à l’entrée ou
- la plage de valeurs de sortie est dépassée.

ROL_*** : Rotation à gauche

42

Aperçu

Introduction

Ce chapitre décrit le module ROL_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction fait tourner la séquence binaire, à l'entrée IN, de n Bits (valeur de l'entrée N), selon un mouvement giratoire vers la gauche.

Le traitement concerne les types de données du groupe ANY_BIT .

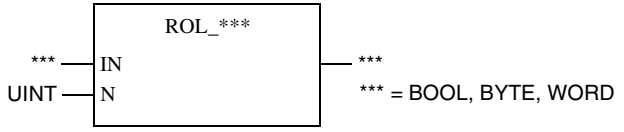
Les types de données de la valeur d'entrée IN et celui de la valeur de sortie OUT doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BOOL, BYTE, WORD	Séquence binaire objet de la rotation
N	UINT	Nombre de positions prises en compte pour la rotation
OUT	BOOL, BYTE, WORD	Séquence binaire ayant subi la rotation

ROR_*** : Rotation à droite

43

Aperçu

Introduction

Ce chapitre décrit le module ROR_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction fait tourner la séquence binaire, à l'entrée IN, de n Bits (valeur de l'entrée N) selon un mouvement giratoire vers la droite.

Le traitement concerne les types de données du groupe ANY_BIT .

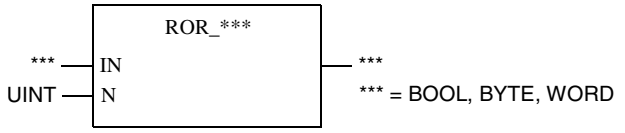
Les types de données de la valeur d'entrée IN et celui de la valeur de sortie OUT doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Type de données	Signification
IN	BOOL, BYTE, WORD	Séquence binaire objet de la rotation
N	UINT	Nombre de positions prises en compte pour la rotation
OUT	BOOL, BYTE, WORD	Séquence binaire ayant subi la rotation

RS : Module de fonction bistable, Reset dominant

44

Aperçu

Introduction

Ce chapitre décrit le module RS.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction est utilisé comme mémoire RS ayant la propriété de "réinitialisation prioritaire".

La sortie Q1 passe à "1", si l'entrée S passe à "1". Cet état est maintenu même si l'entrée S est remise à "0". La sortie Q1 n'est remise à "0" que si l'entrée R1 passe à "1". Si les entrées S et R1 ont simultanément l'état "1", l'entrée dominante R1 remet la sortie Q1 à "0".

L'état initial de Q1 lors du premier appel du module de fonction est "0".

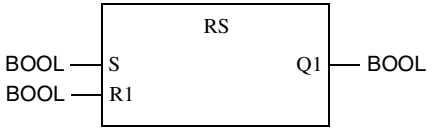
EN et ENO peuvent être gérés comme paramètres supplémentaires.

Note : Ce module de fonction utilise une 'Unlocated Variable' et se comporte donc comme une mémoire. Cela signifie que si la sortie "Q1" est liée à une sortie matérielle, la sortie est maintenue sur la valeur "1" lorsque l'API est activée/désactivée.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
S	BOOL	Initialisation
R1	BOOL	Reset (dominant)
Q1	BOOL	Sortie

SEL : Choix binaire

45

Aperçu

Introduction

Ce chapitre décrit le module SEL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction sert au choix binaire entre deux valeurs d'entrée.

En fonction de l'état de l'entrée G, l'entrée transmise à la sortie OUT est soit IN0 soit IN1.

G = 0 -> OUT = IN0

G = 1 -> OUT = IN1

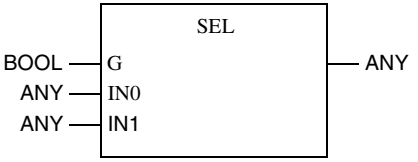
Les types de données des valeurs d'entrée IN0 et IN1 et de la valeur de sortie OUT doivent être identiques.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
G	BOOL	Entrée de sélection
IN0	ANY	Entrée 0
IN1	ANY	Entrée 1
OUT	ANY	Sortie

SHL_*** : Décalage à gauche

46

Aperçu

Introduction

Ce chapitre décrit le module SHL_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction décale la séquence binaire, à l'entrée IN, de n Bits (valeur de l'entrée N) vers la gauche.

Insertion de zéros de remplissage à partir de la droite.

Le traitement concerne les types de données du groupe ANY_BIT .

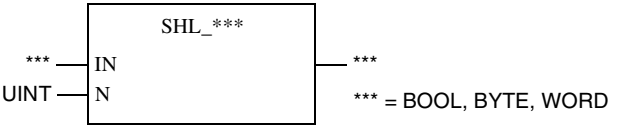
Les types de données de la valeur d'entrée IN et celui de la valeur de sortie OUT doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BOOL, BYTE, WORD	Séquence binaire à décaler
N	UINT	Nombre de positions prises en compte pour le décalage
OUT	BOOL, BYTE, WORD	Séquence binaire décalée

SHR_*** : Décalage à droite

47

Aperçu

Introduction

Ce chapitre décrit le module SHR_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction décale la séquence binaire, à l'entrée IN, de n Bits (valeur de l'entrée N) vers la droite.

Insertion de zéros de remplissage à partir de la gauche.

Le traitement concerne les types de données du groupe ANY_BIT .

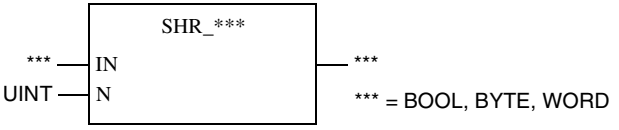
Les types de données de la valeur d'entrée IN et celui de la valeur de sortie OUT doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BOOL, BYTE, WORD	Séquence binaire à décaler
N	BOOL	Nombre de positions prises en compte pour le décalage
OUT	BOOL, BYTE, WORD	Séquence binaire décalée

SIN_REAL : Sinus

48

Aperçu

Introduction

Ce chapitre décrit le module SIN_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

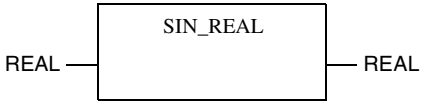
La fonction calcule le sinus de la valeur d'entrée et délivre le résultat en sortie. La valeur d'entrée doit être saisie en radian.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = sin IN

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée en radian
OUT	REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

La présence d'un nombre en virgule flottante non admis à l'entrée entraîne l'affichage d'un message d'erreur.

SQRT_REAL : Racine carrée

49

Aperçu

Introduction

Ce chapitre décrit le module SQRT_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

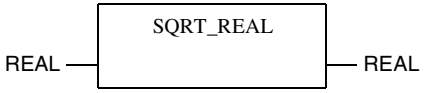
La fonction calcule la racine carrée de la valeur d'entrée et délivre le résultat en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

$$OUT = \sqrt{IN}$$

Condition :

$IN \geq 0$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée
OUT	REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de fonction, la plage de valeurs d'entrée n'est pas respectée ou en présence d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

SR : Module de fonction bistable, Initialisation dominante

50

Aperçu

Introduction

Ce chapitre décrit le module SR.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction est utilisé comme mémoire SR ayant la propriété de "positionnement prioritaire".

La sortie Q1 passe à "1", si l'entrée S1 passe à "1". Cet état est maintenu même si l'entrée S1 est remise à "0". La sortie Q1 n'est remise à "0" que si l'entrée R passe à "1". Si les entrées S1 et R ont simultanément l'état "1", l'entrée dominante S1 remet la sortie Q1 à "1".

L'état initial de Q1 lors du premier appel du module de fonction est "0".

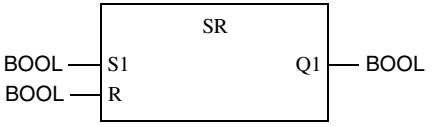
EN et ENO peuvent être gérés comme paramètres supplémentaires.

Note : Ce module de fonction utilise une 'Unlocated Variable' et se comporte donc comme une mémoire. Cela signifie que si la sortie "Q1" est liée à une sortie matérielle, la sortie est maintenue sur la valeur "1" lorsque l'API est activée/ désactivée.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
S1	BOOL	Initialisation (dominante)
R	BOOL	Reset
Q	BOOL	Sortie

SUB_*** : Soustraction

51

Aperçu

Introduction

Ce chapitre décrit le module SUB_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction soustrait la valeur de l'entrée IN2 de celle de l'entrée IN1 et délivre le résultat en sortie.

Le traitement concerne les types de données du groupe ANY_NUM et le type de données TIME.

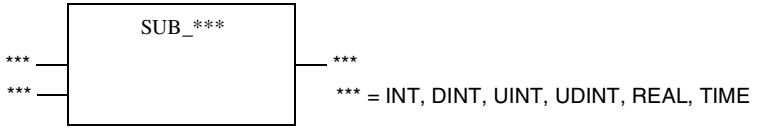
Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	INT, DINT, UINT, UDINT, REAL, TIME	Diminuende
IN2	INT, DINT, UINT, UDINT, REAL, TIME	Diminuteur
OUT	INT, DINT, UINT, UDINT, REAL, TIME	Différence

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur dans le cas où

- un paramètre d'entrée du type de données REAL est assorti d'un nombre en virgule flottante non admis ou
- la plage des valeurs est dépassée en sortie.

TAN_REAL : Tangente

52

Aperçu

Introduction

Ce chapitre décrit le module TAN_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

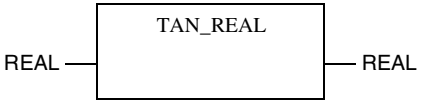
La fonction calcule la tangente de la valeur d'entrée et délivre le résultat en sortie. La valeur d'entrée doit être saisie en radian.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = tan IN

Condition :

$$IN \neq \frac{\langle 2n + 1 \rangle \times \pi}{2}$$

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	REAL	Valeur d'entrée en radian
OUT	REAL	Valeur de sortie

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de fonction, la plage de valeurs d'entrée n'est pas respectée ou en présence d'un nombre en virgule flottante non admis, il s'ensuit l'affichage d'un message d'erreur.

TIME_DIV_*** : Division de valeurs de temps

53

Aperçu

Introduction

Ce chapitre décrit le module TIME_DIV_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction divise la valeur de l'entrée IN1 par celle de l'entrée IN2 et délivre le résultat en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Exemple

Troncation vers zéro d'une position après la virgule éventuellement présente dans le résultat :

67 / 3 = 22

-67 / 3 = -22

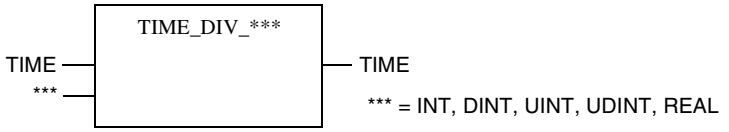
Types de données

Pour le traitement des différents types de données l'on dispose respectivement d'une fonction particulière.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 / IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	TIME	Dividende
IN2	INT, DINT, UINT, UDINT, REAL	Diviseur
OUT	TIME	Quotient

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de la fonction , la plage de valeurs est dépassée en sortie, il s'ensuit l'affichage d'un message d'erreur.

TIME_MUL_*** : Multiplication de valeurs de temps

54

Aperçu

Introduction

Ce chapitre décrit le module TIME_MUL_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction multiplie la valeur de l'entrée IN1 par celle de l'entrée IN2 et délivre le résultat en sortie.

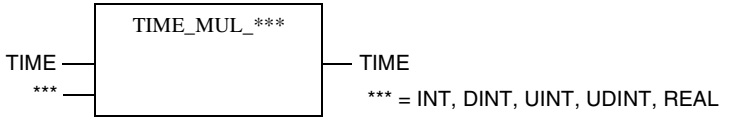
Pour le traitement des différents types de données l'on dispose respectivement d'une fonction particulière.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 x IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	TIME	Multiplicande (facteur)
IN2	INT, DINT, UINT, UDINT, REAL	Multiplicateur (facteur)
OUT	TIME	Produit

Erreur d'exécution

Message d'erreur

Si, au cours de l'exécution de la fonction , la plage de valeurs est dépassée en sortie, il s'ensuit l'affichage d'un message d'erreur.

TIME_TO_*** : Conversion de type

55

Aperçu

Introduction

Ce chapitre décrit le module TIME_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée du type de données TIME en type de données du groupe ANY_BIT ou ANY_NUM.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques , par ex. TIME_TO_BOOL.

Lors de la conversion d'une valeur d'entrée de type TIME en une valeur de sortie de type BOOL, BYTE, WORD, INT ou UINT, les bits d'entrée de poids faible sont respectivement transmis à la sortie.

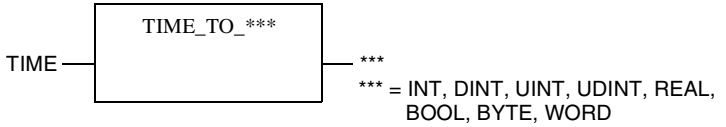
Note : Pour une conversion de TIME en WORD, l'on dispose également du module TIME_AS_WORD de la bibliothèque de modules EXTENDED.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	TIME	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL, BOOL, BYTE, WORD	Valeur de sortie

Erreur d’exécution

Message d’erreur

Si, au cours de l’exécution de la fonction , la plage de valeurs est dépassée en sortie, il s’ensuit l’affichage d’un message d’erreur.

TOF : Temporisation de désactivation

56

Aperçu

Introduction

Ce chapitre décrit le module TOF.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction sert de temporisateur de désactivation.

Un front 0 -> 1 à l'entrée IN entraîne une réinitialisation.

Un front 1 -> 0 à l'entrée IN lance la fonction de temporisation.

Si le temps écoulé (sortie ET) atteint la valeur spécifiée à l'entrée PT, la sortie Q passe à "0".

L'état initial de ET lors du premier appel du module de fonction est "0".

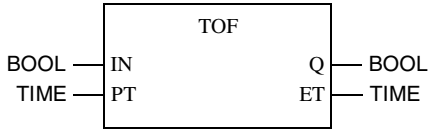
EN et ENO peuvent être gérés comme paramètres supplémentaires.

Note : Il n'est pas possible d'utiliser l'entrée EN comme fonction de pause pour le bloc fonction.
Même si l'entrée EN passe à "0", le temps écoulé continue à être mesuré. Si l'entrée EN passe de nouveau à "1", la sortie ET est mise à jour et exécute alors un saut.
Si vous avez besoin d'une fonction de pause, vous disposez du bloc fonction TOF_P de la bibliothèque de blocs EXTENDED.

Représentation

Symbole

Représentation du bloc :



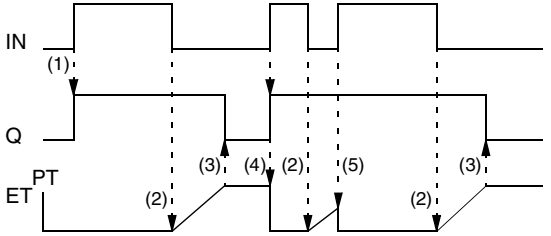
Description des paramètres

Description des paramètres du bloc :

Paramètres	Type de données	Signification
IN	BOOL	0 -> 1: Reset 1 -> 0: Démarrage de la temporisation
PT	TIME	Préréglage du retard
Q	BOOL	Sortie
ET	TIME	Temps écoulé

Description détaillée

Chronogramme Représentation de la temporisation de désactivation TOF :



(1) Si IN passe à "1", Q passe à "1".
(2) Si IN passe à "0", l'horloge interne (ET) se déclenche.
(3) Si l'horloge interne atteint la valeur de PT, Q passe à "0".
(4) Si IN passe à "1", Q passe à "1" et l'horloge interne s'arrête/est remise à zéro.
(5) Si IN passe à "1" avant que l'horloge interne n'ait atteint la valeur de PT, l'horloge interne s'arrête/est remise à zéro sans que Q ne soit remis à "0".

TON : Temporisation d'activation

57

Aperçu

Introduction

Ce chapitre décrit le module TON.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction sert de temporisateur d’activation.

Un front 1 -> 0 à l'entrée IN entraîne une réinitialisation.

Un front 0 -> 1 à l'entrée IN lance la fonction de temporisation.

Si le temps écoulé (sortie ET) atteint la valeur spécifiée à l'entrée PT, la sortie Q passe à "1".

L’état initial de ET lors du premier appel du module de fonction est "0".

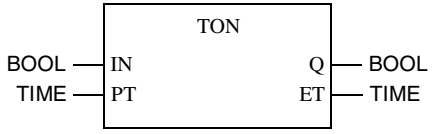
EN et ENO peuvent être gérés comme paramètres supplémentaires.

Note : Il n'est pas possible d'utiliser l'entrée EN comme fonction de pause pour le bloc fonction.
Même si l'entrée EN passe à "0", le temps écoulé continue à être mesuré. Si l'entrée EN passe de nouveau à "1", la sortie ET est mise à jour et exécute alors un saut.
Si vous avez besoin d'une fonction de pause, vous disposez du bloc fonction TON_P de la bibliothèque de blocs EXTENDED.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

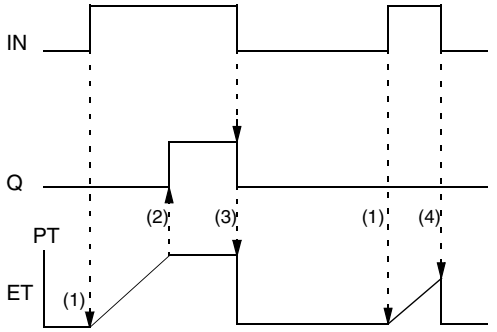
Description des paramètres du bloc :

Paramètres	Type de données	Signification
IN	BOOL	0 -> 1: Démarrage de la temporisation 1 -> 0: Reset
PT	TIME	Préréglage du retard
Q	BOOL	Sortie
ET	TIME	Temps écoulé

Description détaillée

Chronogramme

Représentation de la temporisation d'activation TON :



(1) Si IN passe à "1", l'horloge interne (ET) se déclenche.
(2) Si l'horloge interne atteint la valeur de PT, Q passe à "1".
(3) Si IN passe à "0", Q passe à "0" et l'horloge interne s'arrête/est remise à zéro.
(4) Si IN passe à "0" avant que l'horloge interne n'ait atteint la valeur de PT, l'horloge interne s'arrête/est remise à zéro sans que Q ne passe à "1".

TP : Impulsion

58

Aperçu

Introduction

Ce chapitre décrit le module TP.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

Le module de fonction sert à générer une impulsion de durée définie.

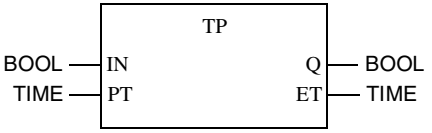
L'état initial de ET lors du premier appel du module de fonction est "0".

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



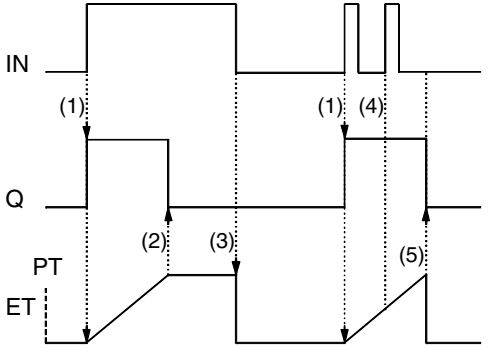
Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	BOOL	Déclenchement d'impulsion
PT	TIME	Préréglage de la durée d'impulsion
Q	BOOL	Sortie
ET	TIME	Horloge interne

Description détaillée

Chronogramme Représentation de l'impulsion TP :



(1) Si IN passe à "1", Q passe à "1" et l'horloge interne (ET) se déclenche.
(2) Si l'horloge interne atteint la valeur de PT, Q est remis à "0" (indépendamment de IN).
(3) L'horloge interne s'arrête/est remise à zéro, si IN est remis à "0".
(4) Si l'horloge interne n'a pas encore atteint la valeur de PT, la présence d'un cycle d'impulsion au niveau de IN n'a aucune influence sur l'horloge interne.
(5) Si l'horloge interne a atteint la valeur de PT et que IN est à l'état "0", l'horloge interne s'arrête/est remise à zéro et Q est remis à "0".

UDINT_EXPT_REAL : Exponentiation

59

Aperçu

Introduction

Ce chapitre décrit le module UDINT_EXPT_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

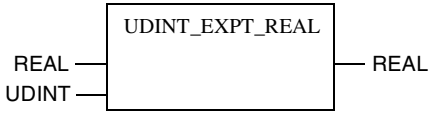
La fonction sert au calcul exponentiel. La valeur de l'entrée IN1 (base) fait l'objet d'un calcul exponentiel avec la valeur de l'entrée IN2 comme exposant, la puissance étant ensuite délivrée en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 exp IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	REAL	Base
IN2	UDINT	Exposant
OUT	REAL	Puissance

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, dans le cas où

- (IN1 = 0) & (IN2 < 0),
- un nombre en virgule flottante non admis est affecté à IN1 ou
- la plage des valeurs est dépassée en sortie.

UDINT_TO_*** : Conversion de type

60

Aperçu

Introduction

Ce chapitre décrit le module UDINT_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée du type de données UDINT en valeur de sortie du type DINT, INT, UINT, REAL, TIME, BOOL, BYTE ou WORD.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. UDINT_TO_BOOL.

Lors de la conversion du type de données UDINT en type de données BOOL, BYTE ou WORD, les bits de poids faible de la valeur d'entrée sont transmis à la sortie.

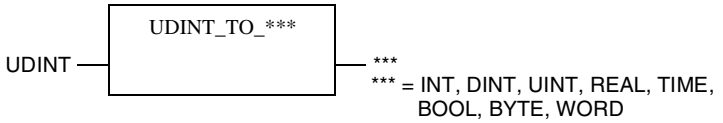
Note : Pour une conversion de UDINT en WORD, l'on dispose également du module UDINT_AS_WORD de la bibliothèque de modules EXTENDED.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description du module :

Paramètres	Types de données	Signification
IN	UDINT	Valeur d'entrée
OUT	DINT, INT, UINT, REAL, TIME, BOOL, BYTE, WORD	Valeur de sortie

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, si la plage de valeurs de la sortie est dépassée .

UINT_EXPT_REAL : Exponentiation

61

Aperçu

Introduction

Ce chapitre décrit le module UINT_EXPT_REAL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

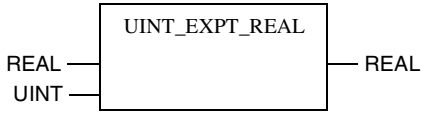
La fonction sert au calcul exponentiel. La valeur de l'entrée IN1 (base) fait l'objet d'un calcul exponentiel avec la valeur de l'entrée IN2 comme exposant, la puissance étant ensuite délivrée en sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

OUT = IN1 exp IN2

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	REAL	Base
IN2	UINT	Exposant
OUT	REAL	Puissance

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, dans le cas où

- (IN1 = 0) & (IN2 < 0),
- un nombre en virgule flottante non admis est affecté à IN1 ou
- la plage des valeurs est dépassée en sortie.

UINT_TO_*** : Conversion de type

62

Aperçu

Introduction

Ce chapitre décrit le module UINT_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée du type de données UINT en valeur de sortie du type BOOL, BYTE, WORD, DINT, INT, UDINT, REAL ou TIME.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. UINT_ TO_BOOL.

Lors de la conversion d'une valeur d'entrée de type UINT en type WORD, le profil binaire d'entrée est transmis à la sortie sans modification.

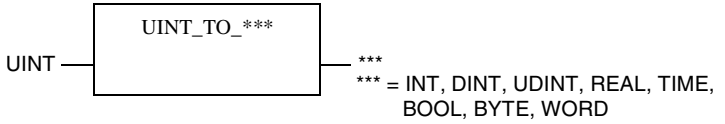
Lors de la conversion d'une valeur d'entrée du type de données UINT en type BOOL ou BYTE, les bits d'entrée de poids faible sont transmis à la sortie.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	UINT	Valeur d'entrée
OUT	BOOL, BYTE, WORD, DINT, INT, UDINT, REAL, TIME	Valeur de sortie

Erreur d'exécution

Message d'erreur

Affichage d'un message d'erreur, si la plage de valeurs de la sortie est dépassée .

WORD_TO_*** : Conversion de type

63

Aperçu

Introduction

Ce chapitre décrit le module WORD_TO_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction convertit une valeur d'entrée du type de données WORD en type de données du groupe ANY_NUM ou en types de données BOOL, BYTE ou TIME.

Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

Note : EFB procède à la conversion en respectant strictement les règles CEI. Cet EFB étant réalisé en tant que fonction générique, il en résulte aussi quelques conversions illogiques, par ex. WORD_ TO_TIME. Ceci étant, il faut retenir que le profil de l'octet d'entrée est transféré dans le mot de poids fort du mot de sortie .

Lors de la conversion du type de données WORD en type de données DINT, UDINT ou REAL, le profil binaire d'entrée est transféré dans les bits de poids faible de la sortie. Les bits de poids fort de la sortie sont positionnés sur zéro.

Lors de la conversion du type de données WORD en type de données BOOL ou BYTE, les bits de poids faible de la valeur d'entrée sont transmis à la sortie.

Note : Pour une conversion de WORD en REAL, l'on dispose également des modules W_INT_REAL et W_UINT_REAL de la bibliothèque de modules ANA_IO et du module WORD_AS_REAL de la bibliothèque de modules EXTENDED.

Note : Pour une conversion de WORD en TIME, l'on dispose également du module WORD_AS_TIME de la bibliothèque de modules EXTENDED.

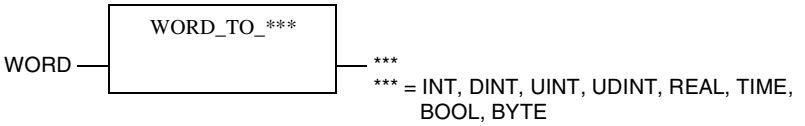
Note : Pour une conversion de WORD en UDINT, l'on dispose également du module WORD_AS_UDINT de la bibliothèque de modules EXTENDED.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN	WORD	Valeur d'entrée
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE	Valeur de sortie

Erreur d'exécution

Message d'erreur

Lors de la conversion en type de données REAL la génération d'un nombre en virgule flottante non admis entraîne l'affichage d'un message d'erreur.

XOR_*** : Fonction OU exclusif

64

Aperçu

Introduction

Ce chapitre décrit le module XOR_***.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Présentation

Description de la fonction

La fonction procède au chaînage (en logique OU exclusif) des séquences binaires arrivant aux entrées; ensuite, elle délivre le résultat en sortie. Le chaînage s'effectue bit à bit.

Le traitement concerne les types de données du groupe ANY_BIT .

Les types de données de toutes les valeurs d'entrée et celui de la valeur de sortie doivent être identiques. Pour le traitement des différents types de données, l'on dispose respectivement d'une fonction particulière.

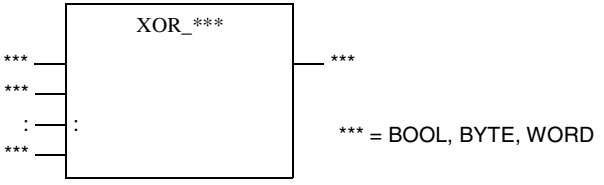
Il est possible d'augmenter le nombre d'entrées.

EN et ENO peuvent être gérés comme paramètres supplémentaires.

Représentation

Symbole

Représentation du bloc :



Formule

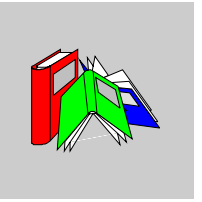
OUT = IN1 XOR IN2 XOR .. XOR INn

Description des paramètres

Description des paramètres du bloc :

Paramètres	Types de données	Signification
IN1	BOOL, BYTE, WORD	Séquence binaire d'entrée
IN2	BOOL, BYTE, WORD	Séquence binaire d'entrée
INn	BOOL, BYTE, WORD	Séquence binaire d'entrée
OUT	BOOL, BYTE, WORD	Séquence binaire de sortie

Glossaire



A

Abonné de réseau

Un abonné est un appareil avec une adresse (1 à 64) sur le réseau Modbus Plus.

Abonné local du réseau

L'abonné local est celui qui est projeté à l'instant.

Adresse abonné

L'adresse abonné sert à la désignation univoque d'un abonné du réseau dans l'itinéraire de routage. L'adresse est réglée directement sur l'abonné, p. ex. via le commutateur rotatif situé sur la face arrière du module.

Adresses

Les adresses (directes) sont des zones de mémoire dans l'API. Celles-ci se trouvent dans la mémoire d'état et peuvent être affectées à des modules d'entrée/sortie. L'affichage/la saisie d'adresses directes est possible dans les formats suivants :

- Format standard (400001)
- Format séparateur (4:00001)
- Format compact (4:1)
- Format CEI (QW1)

Affectation des E/S

L'affectation des E/S est une liste d'affectation générée à partir de la liste d'affectation de l'utilisateur. L'affectation des E/S est gérée dans l'API et contient p. ex. des informations sur l'état des stations et modules E/S, en supplément de la liste d'affectation de l'utilisateur.

ANL_IN ANL_IN est le type de données "entrée analogique" et est utilisé pour le traitement des valeurs analogiques. Les références 3x du module d'entrée analogique configuré déterminées dans la liste d'affectation des E/S sont affectées automatiquement au type de données et doivent de ce fait être occupées uniquement par des variables non localisées.

ANL_OUT ANL_OUT est le type de données "sortie analogique" et est utilisé pour le traitement des valeurs analogiques. Les références 4x du module de sortie analogique configuré déterminées dans la liste d'affectation des E/S sont affectées automatiquement au type de données et doivent de ce fait être occupées uniquement par des variables non localisées.

ANY Dans la présente version, "ANY" comprend les types de données élémentaires BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME et WORD ainsi que les types de données qui en sont dérivés.

ANY_BIT Dans la présente version, "ANY_BIT" comprend les types de données BOOL, BYTE et WORD.

ANY_ELEM Dans la présente version, "ANY_ELEM" comprend les types de données BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME et WORD.

ANY_INT Dans la présente version, "ANY_INT" comprend les types de données DINT, INT, UDINT et UINT.

ANY_NUM Dans la présente version, "ANY_NUM" comprend les types de données DINT, INT, REAL, UDINT et UINT.

ANY_REAL Dans la présente version, "ANY_REAL" correspond au type de données REAL.

API Automate programmable industriel

Appel La procédure par laquelle l'exécution d'une opération est lancée.

Argument Synonyme de paramètre réel.

Atrium L'automate basé sur PC est monté sur platine standard AT et s'utilise au sein d'un ordinateur hôte dans un emplacement de bus ISA. Ce module possède une carte mère (nécessite un pilote SA85) avec deux emplacements pour cartes filles PC104. L'une des cartes filles PC104 sert d'UC et l'autre à la commande INTERBUS.

Avertissement Si un état critique est identifié lors du traitement d'un FFB ou d'une étape (p. ex. des valeurs d'entrée critiques ou des limites temporelles dépassées), un avertissement est généré. Celui-ci peut être visualisé à l'aide de la commande **En ligne** → **Affichage événements...**. Sur les FFB, la sortie ENO reste sur "1".

B

Base de données de projet La base de données du PC, contenant les informations de configuration d'un projet.

Bibliothèque Ensemble d'objets logiciels prévus pour la réutilisation lors de la programmation de nouveaux projets, ou bien même pour l'élaboration de nouvelles bibliothèques. Les exemples sont les bibliothèques des types de blocs fonction élémentaires.
Les bibliothèques EFBpeuvent être subdivisées en groupes.

Bits d'entrée (Références 1x) L'état 1/0 des bits d'entrée est commandé par les données du procédé arrivant depuis un périphérique d'entrée dans l'UC.

Note : Le x suivant le premier chiffre du type de référence représente un emplacement à cinq chiffres dans la mémoire de données utilisateur, p. ex. la référence 100201 signifie un bit d'entrée à l'adresse 201 de la mémoire d'état.

Bits d'état Il existe un bit d'état pour chaque abonné à entrée globale, entrée ou sortie spécifique de données de diffusion. Si un groupe de données défini a pu être transmis avec succès avant écoulement du timeout réglé, le bit d'état correspondant est mis à 1. Dans le cas contraire, ce bit est mis à 0 et toutes les données appartenant à ce groupe (à 0) sont effacées.

Bits de sortie/ bits internes (Références 0x) Un bit de sortie/bit interne peut être utilisé pour commander des données de sortie réelles via une unité de sortie du système de contrôle, ou pour définir une ou plusieurs sorties TOR dans la mémoire d'état. Remarque : le x suivant immédiatement le premier chiffre du type de référence, représente un emplacement mémoire sur 5 chiffres dans la mémoire de données utilisateur, p. ex. la référence 000201 signifie un bit interne ou de sortie à l'adresse 201 de la mémoire d'état.

Bloc fonction (instance) (BF)

Un bloc fonction est une unité d'organisation de programme, qui, en fonction de sa fonctionnalité définie dans la description de type de bloc fonction, calcule des valeurs pour ses sorties et variable(s) interne(s), lorsqu'elle est appelée comme instance particulière. Toutes les valeurs des sorties et variables internes d'une instance particulière de bloc fonction sont conservées d'un appel du bloc fonction au suivant. Des appels répétés de la même instance de bloc fonction avec les mêmes arguments (valeurs des paramètres d'entrée) ne délivrent de ce fait pas forcément la (les) même(s) valeur(s) de sortie.
Chaque instance de bloc fonction est représentée graphiquement par un symbole rectangulaire. Le nom du type de bloc fonction est situé en haut au milieu, à l'intérieur du rectangle. Le nom de l'instance de bloc fonction est également en haut, bien qu'à l'extérieur du rectangle. Il est généré automatiquement à la création d'une instance mais peut, le cas échéant, être modifié par l'utilisateur. Les entrées sont représentées à gauche, les sorties à droite du bloc. Les noms des paramètres formels d'entrée/sortie sont indiqués à l'intérieur du rectangle aux places correspondantes.
La description ci-dessus de la représentation graphique est valable de principe également pour lesappels de fonction et pour les appels DFB. Les différences sont décrites dans les définitions correspondantes.

Bobine

Une bobine est un élément LD transmettant sans le modifier l'état de la liaison horizontale sur sa gauche à la liaison horizontale sur sa droite. L'état est alors mémorisé dans la variable/adresse directe associée.

BOOL

BOOL signifie type de données "booléen". La longueur des éléments de données est 1 bit (stocké en mémoire sur 1 octet). La plage de valeurs des variables de ce type de données est 0 (FALSE) et 1 (TRUE).

Bridge

Un bridge est un dispositif permettant de relier des réseaux. Il permet la communication entre abonnés de deux réseaux. Chaque réseau possède sa propre séquence de rotation de jeton - le jeton n'est pas transmis par les bridges.

BYTE

BYTE est le type de données "cordon de bits 8". L'entrée peut se faire en libellé en base 2, libellé en base 8 ou libellé en base 16. La longueur des éléments de données est de 8 bits. Il n'est pas possible d'affecter une plage de valeurs numériques à ce type de données.

C

CEI 611313

Norme internationale : Automates programmables Partie 3 : Langages de programmation.

Code de section	Le code de section est le code exécutable d'une section. La taille du code de section dépend principalement du nombre de blocs dans la section.
Code DFB	Le code DFB est le code DFB exécutable d'une section. La taille du code DFB dépend principalement du nombre de modules dans la section.
Code EFB	Le code EFB est le code exécutable de tous les EFB utilisés. Les EFB utilisés dans les DFB sont également pris en compte.
Configuration de transmission de données	Paramètres déterminant comment les informations sont transmises depuis votre PC vers l'API.
Connexion série	En connexion série (COM), les informations sont transmises bit par bit.
Constantes	Les constantes sont des variables non localisées, auxquelles est affectée une valeur qui ne peut être modifiée par la logique de programme (lecture seule).
Contact	Un contact est un élément LD transmettant un état sur la liaison horizontale située à sa droite. Cet état est le résultat d'une liaison ET booléenne entre l'état de la liaison horizontale sur sa gauche et l'état de la variable/adresse directe qui lui est affectée. Un contact ne modifie pas la valeur de la variable/adresse directe associée.
Convention CEI sur les noms (Identificateur)	Un identificateur est une suite de lettres, chiffres et caractères de soulignement devant commencer par une lettre ou un caractère de soulignement (p. ex. nom d'un type de bloc fonction, d'une instance, d'une variable ou d'une section). Les lettres des polices de caractères nationales (p. ex. : ö, ü, é, õ) peuvent être utilisées sauf dans les noms de projets et de DFB. Les caractères de soulignement sont significatifs dans les identificateurs ; p. ex. "A_BCD" et "AB_CD" seront interprétés comme des identificateurs différents. Plusieurs caractères de soulignement de tête ou de suite ne sont pas autorisés. Les identificateurs ne doivent pas comporter d'espaces. Les majuscules/minuscules ne sont pas significatives ; p. ex. "ABCD" et "abcd" seront interprétés comme le même identificateur. Les identificateurs ne doivent pas être des mots-clés.
Cordon de bits	C'est un élément de données constitué d'un ou de plusieurs bits.
Cycle programme	Un cycle programme consiste en la lecture des entrées, le traitement de la logique de programme et l'édition des sorties.

D

DDE (Echange dynamique de données)

L'interface DDE permet à deux programmes sous Windows d'échanger des données en dynamique. L'utilisateur peut se servir de l'interface DDE en moniteur étendu afin d'appeler ses propres applications d'affichage. Avec cette interface, l'utilisateur (c.-à-d. le client DDE) peut non seulement lire des données du moniteur étendu (le serveur DDE), mais peut également écrire des données sur l'API via le serveur. L'utilisateur peut ainsi modifier directement des données dans l'API tout en surveillant et en analysant les résultats. Lors de l'utilisation de cette interface, l'utilisateur peut créer son propre "Outil graphique", "Face Plate" ou "Outil de réglage", et intégrer celui-ci dans le système. Ces outils peuvent être écrits dans n'importe quel langage que le DDE prend en charge, p. ex. Visual Basic, VisualC++. Ils sont appelés lorsque l'utilisateur actionne l'un des boutons de commande de la boîte de dialogue Moniteur étendu. Outil graphique Concept : grâce au lien DDE entre Concept et l'outil Graphique Concept, il est possible de représenter les signaux d'une configuration sous forme de chronogramme.

Déclaration

Le mécanisme qui permet d'établir la définition d'un élément de langage. Normalement, une déclaration nécessite le rattachement d'un identificateur à l'élément de langage et l'affectation d'attributs, tels que lestypes de données et les algorithmes.

Défaut

Si, lors du traitement d'un FFB ou d'une étape, une erreur est détectée (p. ex. valeurs d'entrée non autorisées ou erreur de durée), un message d'erreur est généré, lequel peut être visualisé à l'aide de la commande **En ligne** → **Affichage événements...**. Sur les FFB la sortie ENOest mise à "0".

Défragmentation

La défragmentation permet de supprimer les trous indésirables dans la zone mémoire (générés, p. ex., en effaçant des variables inutilisées).

Derived Function Block (DFB) (Bloc fonction dérivé)

Un bloc fonction dérivé représente l'appel d'un type de bloc fonction dérivé. Vous trouverez des détails de la forme graphique de l'appel dans la définition "Bloc fonction (instance)". Contrairement aux appels de types d'EFB, les appels de types DFB sont caractérisés par des lignes verticales doubles sur les côtés gauche et droit du symbole rectangulaire du bloc.
Le corps d'un type de bloc fonction dérivé est projeté en langage FBD, langage LD, langage ST et langage IL quoique seulement dans la version actuelle du système de programmation. Les fonctions dérivées ne peuvent pas encore être définies dans la version actuelle.
On fait la distinction entre les DFB locaux et globaux.

DFB globaux Les DFB globaux sont disponibles dans tout projet Concept. Le stockage des DFB globaux dépend de la configuration dans le fichier CONCEPT.INI.

DFB locaux Les DFB locaux ne sont disponibles que dans un seul projet Concept et sont enregistrés dans le répertoire DFB sous le répertoire de projet.

Diagramme fonctionnel en séquence (SFC) Les éléments de langage SFC permettent de subdiviser une unité d'organisation de programme en un certain nombre d'étapes et de transitions, reliées entre elles par des liaisons dirigées. A chaque étape correspond un nombre d'actions et à chaque transition est associée une condition de transition.

DINT DINT signifie type de données "entier double (double integer)". L'entrée s'effectue en libellé entier, libellé en base 2, libellé en base 8 ou libellé en base 16. La longueur des éléments de données est de 32 bits. La plage de valeurs pour les variables de ce type de données va de -2 exp (31) à 2 exp (31) -1.

Données d'instance DFB Les données d'instance DFB sont des données internes des instructions chargeables dérivées utilisées dans le programme.

Données de section Les données de section sont les données locales d'une section, comme par ex. les libellés, les liaisons entre blocs, les entrées et sorties de bloc non liées, la mémoire d'état interne des EFB.

Note : Les données qui sont configurées dans les DFB de cette section ne sont pas des données de section.

Données globales Les données globales sont des variables non localisées.

DP (PROFIBUS) DP = Dezentrale Peripherie (périphérie décentralisée)

DX Zoom Cette caractéristique vous permet de vous raccorder sur un objet de programmation afin d'en surveiller des valeurs et de les modifier, si nécessaire.

E

Elément de langage Chaque élément de base dans l'un des langages de programmation CEI, p. ex. une étape en SFC, une instance de bloc fonction en FBD ou la valeur de départ d'une variable.

EN / ENO (autorisation / affichage d'erreur)

Si la valeur de EN vaut "0", lorsque le FFB est lancé, les algorithmes définis par le FFB ne sont pas exécutés et toutes les sorties conservent leur valeur précédente. La valeur de ENO est dans ce cas mise automatiquement à "0". Si la valeur de EN est "1" lors de l'appel du FFB, les algorithmes définis par le FFB seront exécutés. Après l'exécution sans erreur de ces algorithmes, la valeur de ENO est mise automatiquement à "1". Si une erreur survient lors de l'exécution de ces algorithmes, ENO est mis automatiquement à "0". Le comportement de sortie des FFB est indépendant du fait que ceux-ci sont appelés sans EN/ENO ou avec EN=1. Si l'affichage de EN/ENO est activé, l'entrée EN doit absolument être câblée. Le FFB n'est sinon jamais exécuté. L'activation/la désactivation de EN et ENO se fait dans la boîte de dialogue des caractéristiques du bloc fonction. Cette boîte de dialogue est appelée via **Objets** → **Propriétés...** ou en double-cliquant sur le FFB.

Erreur d'exécution

Erreur survenant lors du traitement du programme sur l'API sur des objets SFC (p. ex. des étapes) ou des FFB. Il s'agit p. ex. de dépassement de plage de valeurs sur les compteurs ou bien d'erreurs temporelles sur les étapes.

Etape

Elément de langage SFC : situation dans laquelle le comportement d'un programme suit, en fonction de ses entrées et sorties, les opérations définies par les actions correspondantes de l'étape.

Etape initiale (Etape de départ)

L'étape de démarrage d'une séquence. Une étape initiale doit être définie dans chaque séquence. La séquence est démarrée à son premier appel par l'étape initiale.

Evaluation

C'est le processus par lequel est déterminé une valeur d'une fonction ou des sorties d'un bloc fonction lors de l'exécution du programme.

Expression

Les expressions sont constituées d'opérateurs et d'opérandes.

F

Fenêtre active

Il s'agit de la fenêtre momentanément sélectionnée. Pour un instant donné, seule une fenêtre peut être active. Lorsqu'une fenêtre devient active, la couleur de sa barre de titre change afin de la distinguer des autres fenêtres. Les fenêtres non sélectionnées ne sont pas actives.

Fenêtre d'application

Il s'agit de la fenêtre contenant l'espace de travail, la barre de menus et la barre d'outils du programme applicatif. Le nom du programme applicatif apparaît dans la barre de titre. Une fenêtre d'application peut contenir plusieurs fenêtres de document. Dans Concept, la fenêtre d'application correspond à un projet.

Fenêtre de document — Une fenêtre contenue dans une fenêtre d'application. Plusieurs fenêtres de document peuvent être ouvertes simultanément dans une fenêtre d'application. Mais seule une fenêtre de document peut être active. Les fenêtres de document dans Concept sont p. ex. les sections, la fenêtre des messages, l'éditeur de données de référence et la configuration de l'automate.

FFB (fonctions/ blocs fonction) — Terme générique désignant les EFB (fonctions/blocs fonction élémentaires) et les DFB (blocs fonction dérivés)

Fichier de code source (Concept-EFB) — Le fichier de code source est un fichier source ordinaire en C++. Après exécution de la commande **Bibliothèque** → **Créer des fichiers**, ce fichier contient un cadre de code EFB dans lequel vous devez porter un code spécifique de l'EFB sélectionné. Pour ce faire, lancez la commande **Objets** → **Source**.

Fichier de définition (Concept-EFB) — Le fichier de définition contient des informations générales de description de l'EFB sélectionné et ses paramètres formels.

Fichier de sauvegarde (Concept-EFB) — Le fichier de sauvegarde est une copie du dernier fichier de code source. Le nom de ce fichier de sauvegarde est "backup??.c" (on suppose ce faisant que vous n'avez jamais plus de 100 copies de votre fichier de sauvegarde). Le premier fichier de sauvegarde porte le nom "backup00.c". Si vous avez procédé à des modifications dans le fichier de définition n'entraînant pas de modification d'interface pour l'EFB, vous pouvez vous dispenser de créer un fichier de sauvegarde en éditant son fichier de code source (**Objets** → **Source**). Si un fichier de sauvegarde est créé, vous pouvez lui donner le nom Fichiersource.

Fichier factice — Il s'agit d'un fichier vide constitué d'un en-tête contenant diverses informations générales sur le fichier, comme l'auteur, la date de création, la désignation de l'EFB, etc. L'utilisateur doit procéder à la préparation de ce fichier factice à l'aide d'entrées supplémentaires.

Fichier prototype (Concept-EFB) — Le fichier prototype contient tous les prototypes des fonctions affectées. On indique en outre, si elle existe, une définition type de la structure de la situation interne.

Fichier Template (Concept-EFB) — Le fichier Template est un fichier ASCII contenant des informations de mise en page pour l'éditeur FBD de Concept, ainsi que des paramètres pour la génération de code.

Filtre RIF — (Filtre Finite Impulse Response) Filtre à réponse impulsionnelle finie

Filtre RII — (Filtre Infinite Impulse Response) Filtre à réponse impulsionnelle infinie

Fonction (FUNK)

Une unité d'organisation de programme délivrant à l'exécution exactement un élément de donnée. Une fonction ne dispose pas d'information de situation interne. Les appels répétés de la même fonction avec les mêmes paramètres d'entrée délivrent toujours les mêmes valeurs de sortie.
Vous trouverez des détails de la forme graphique des appels de fonction dans la définition "Bloc fonction (instance)". Contrairement aux appels de blocs fonction, les appels de fonction ne disposent que d'une unique sortie sans nom, son nom étant le nom de la fonction elle-même. En FBD, chaque appel est caractérisé par un numéro unique par le bloc graphique ; ce numéro est créé automatiquement et ne peut pas être modifié.

Fonctions/blocs fonction élémentaires (EFB)

Caractérisation des fonctions ou des blocs fonction, dont les définitions de type n'ont pas été formulées dans l'un des langages CEI, c.-à-d. dont les corps p. ex. ne peuvent être modifiés à l'aide de l'éditeur DFB (Concept-DFB). Les types EFB sont programmés en "C" et sont mis à disposition en forme précompilée par les bibliothèques.

Format CEI (QW1)

Au début de l'adresse se trouve un identificateur conforme à CEI, suivi de l'adresse à cinq chiffres :

- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

Format compact (4:1)

Le premier chiffre (la référence) est séparé par deux points (:) de l'adresse suivante, les zéros de tête n'étant pas indiqués dans l'adresse.

Format séparateur (délimiteur) (4:00001)

Le premier chiffre (la référence) est séparé par deux-points (:) de l'adresse à cinq caractères.

Format standard (400001)

L'adresse à cinq positions se situe juste après le premier chiffre (la référence).

G

Groupes (EFB)

Quelques bibliothèques EFB (p. ex. la bibliothèque CEI) sont subdivisées en groupes. Cela simplifie, particulièrement dans les importantes bibliothèques, la recherche des EFB.

I

Instanciation La création d'une instance.

Instruction (IL) Les instructions sont des "commandes" du langage de programmation IL. Chaque instruction commence à une nouvelle ligne et est suivie d'un opérateur, le cas échéant avec modificateur, et, si nécessaire pour l'opération concernée, d'un ou de plusieurs opérandes. Si l'instruction utilise plusieurs opérandes, ceux-ci sont séparés par des virgules. Devant l'instruction peut se trouver une étiquette suivie de deux points. Le commentaire doit, s'il existe, être le dernier élément de la ligne.

Instruction (LL984) La mission d'un utilisateur lors de la programmation d'automatismes électriques est de mettre en oeuvre des instructions codées de façon opérationnelle sous forme d'objets imagés classés selon les formes identifiables de contact. Les objets du programme ainsi conçus sont convertis au niveau utilisateur en codes opérandes utilisables par l'ordinateur, et ce lors de la procédure de chargement. Les codes opérandes sont décodés dans l'UC et traités par les fonctions micrologicielles du contrôleur, de sorte que la commande désirée soit ainsi mise en oeuvre.

Instruction (ST) Les instructions sont des "commandes" du langage de programmation ST. Les instructions doivent se terminer par des points-virgules. Plusieurs instructions (séparées par des points-virgules) peuvent se trouver sur une même ligne.

INT INT correspond au type de données "nombre entier (integer)". L'entrée s'effectue en libellé entier, libellé en base 2, libellé en base 8 ou libellé en base 16. La longueur des éléments de données est de 16 bits. La plage de valeurs pour les variables de ce type de données va de -2 exp (15) à 2 exp (15) -1.

Interbus S (PCP) Afin d'utiliser le canal PCP de l'Interbus S et le prétraitement de données de procédé Interbus S (PDV), le configurateur Concept propose maintenant le nouveau type de station d'E/S Interbus S (PCP). A ce type de station d'E/S est affecté de manière fixe le module de connexion Interbus 180-CRP-660-01.
Le module 180-CRP-660-01 se distingue du 180-CRP-660-00 seulement par une plage d'E/S sensiblement plus importante dans la mémoire d'état de l'automate.

J

Jeton

Le jeton du réseau régit la possession momentanée du droit de transmission d'un abonné individuel. Le jeton circule entre les abonnés dans un sens circulaire (croissant) des adresses. Tous les abonnés suivent la rotation du jeton et peuvent obtenir toute sorte de données qui y sont véhiculées.

L

Langage en blocs fonctionnels (FBD)

Une ou plusieurs sections contenant des réseaux représentés graphiquement composés de fonctions, blocs fonction et liaisons.

Liaison

Une liaison de contrôle ou de données entre objets graphiques (p. ex. étapes dans l'éditeur SFC, blocs fonction dans l'éditeur FBD) au sein d'une section, graphiquement représenté par une ligne.

Liaison locale (Local Link)

La liaison locale de réseau est le réseau reliant l'abonné local à d'autres abonnés, soit directement soit par l'amplificateur de bus.

Liaisons binaires

Il s'agit de liaisons entre des sorties et des entrées de FFB de type de données BOOL.

Libellé

Les libellés servent à fournir des valeurs directement aux entrées des FFB, conditions de transition etc... Ces valeurs ne peuvent pas être écrasées par la logique du programme (lecture seule). Le système distingue les libellés génériques des libellés classés par type.
De plus, les libellés servent à affecter une valeur à une constante ou une valeur initiale à une variable.
L'entrée se fait en libellé en base 2, libellé en base 8, libellé en base 16, libellé entier, libellé réel ou libellé réel avec exposant.

Libellé de durée

Les unités permises pour les durées (TIME) sont les jours (J), les heures (H), les minutes (M), les secondes (S) et les millisecondes (MS) ou une combinaison de ceux-ci. La durée doit être caractérisée par le préfixe t#, T#, time# ou TIME#. Le "dépassement" de l'unité de plus grande valeur est admise; p. ex. l'entrée T#25H15M est permise.

Exemple
t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Libellé en base 16

Les libellés en base 16 servent à codifier les entiers dans le système hexadécimal. La base doit être repérée par le préfixe 16#. Les valeurs doivent être non signées (+/). Les caractères de soulignement individuels (_) entre les chiffres ne sont pas significatifs.

Exemple
16#F_F ou 16#FF (décimal 255)
16#E_0 ou 16#E0 (décimal 224)

Libellé en base 2

Les libellés en base 2 servent à la codification de valeurs entières dans le système de base 2. La base doit être repérée par le préfixe 2#. Les valeurs doivent être non signées (+/). Les caractères de soulignement individuels (_) entre les chiffres ne sont pas significatifs.

Exemple
2#1111_1111 ou 2#11111111 (255 décimal)
2#1110_0000 ou 2#11100000 (224 décimal)

Libellé en base 8

Les libellés en base 8 servent à codifier les entiers dans le système de base 8. La base doit être repérée par le préfixe 8#. Les valeurs doivent être non signées (+/). Les caractères de soulignement individuels (_) entre les chiffres ne sont pas significatifs.

Exemple
8#3_77 ou 8#377 (255 décimal)
8#34_0 ou 8#340 (décimal 224)

Libellé entier

Les libellés entiers servent à indiquer des valeurs entières dans le système décimal. Les valeurs peuvent être signées (+/). Les caractères de soulignement individuels (_) entre les chiffres ne sont pas significatifs.

Exemple
-12, 0, 123_456, +986

Libellés classés par type

Si vous voulez déterminer le type de données d'un libellé, vous pouvez le faire avec la construction suivante : 'nomtypedonnée'#'Valeur du libellé'

Exemple
INT#15 (type de données : entier, valeur : 15),
BYTE#00001111 (type de données : octet, valeur : 00001111)
REAL#23.0 (type de données : réel, valeur : 23,0)

Pour l'affectation du type de données REAL, vous pouvez indiquer la valeur de la manière suivante : 23.0.
En indiquant ce point décimal, le type de données REAL est affecté automatiquement.

Libellés génériques

Si le type de données d'un libellé n'a pas d'importance pour vous, indiquez la valeur du libellé. Dans ce cas, Concept affecte automatiquement un type de données adéquat au libellé.

Libellés réels

Les libellés réels servent à indiquer les valeurs à virgule flottante dans le système décimal. Les libellés réels s'identifient au point décimal. Les valeurs peuvent être signées (+/). Les caractères de soulignement individuels (_) entre les chiffres ne sont pas significatifs.

Exemple
-12.0, 0.0, +0.456, 3.14159_26

Libellés réels avec exposant

Les libellés réels avec exposant servent à indiquer les valeurs à virgule flottante dans le système décimal. Les libellés réels avec exposant se caractérisent par le point décimal. L'exposant donne la puissance de dix avec lequel le chiffre de devant doit être multiplié pour obtenir la valeur à représenter. La base peut être précédée d'un signe moins (). L'exposant peut être signé (+/-). Les caractères de soulignement individuels (_) entre les chiffres ne sont pas significatifs. (Uniquement entre les chiffres, et non avant ou après la virgule ou avant ou après "E", "E+" ou "E-")

Exemple
-1.34E-12 ou -1.34e-12
1.0E+6 ou 1.0e+6
1.234E6 ou 1.234e6

Liste d'affectation des E/S

Dans la liste d'affectation des E/S, on configure les modules d'E/S et modules experts des différentes unités centrales.

Liste d'instructions (IL)

IL est un langage littéral conforme à la norme CEI 1131, dans lequel les opérations, telles que les appels sur ou sans condition de blocs fonction et de fonctions, les sauts conditionnels ou sans condition, etc., sont représentées par des instructions.

Littéral structuré (ST)

ST est un langage littéral conforme à la CEI 1131, dans lequel les opérations, comme le lancement de blocs fonction et de fonctions, les exécutions conditionnelles d'instructions, la réitération d'instructions, etc. sont représentés par des instructions.

M

Macro

Les macros sont créées à l'aide du logiciel Concept-DFB.
Les macros servent à dupliquer des sections et des réseaux fréquemment utilisés (y compris leur logique, leurs variables et leur déclaration de variable).
On fait la distinction entre les macros locales et globales.

Les macros possèdent les caractéristiques suivantes :

- Les macros ne peuvent être créées qu'avec les langages FBD et LD
- Les macros ne contiennent qu'une seule section
- Elles peuvent contenir une section d'une complexité quelconque
- D'un point de vue programme, une macro instanciée, c.-à-d. une macro insérée dans une section, ne se distingue pas d'une section créée de manière conventionnelle.
- Appel de DFB dans une macro
- Déclaration de variables
- Utilisation de structures de données propres aux macros
- Validation automatique des variables déclarées dans la macro
- Valeurs initiales des variables
- Instanciation multiple d'une macro dans tout le programme avec différentes variables
- Le nom de la section, les noms des variables et le nom de la structure de données peuvent comporter jusqu'à 10 marques d'échange (@0 à @9) différentes.

Macros globales

Les macros globales sont disponibles dans tout projet Concept et sont enregistrées dans le répertoire DFB directement situé sous le répertoire Concept.

Macros locales

Les macros locales ne sont disponibles que dans un seul projet Concept et sont enregistrées dans le répertoire DFB sous le répertoire de projet.

Mémoire d'état La mémoire d'état est l'emplacement mémoire pour toutes les grandeurs sollicitées dans le programme utilisateur par des références (représentation directe). Par exemple les bits d'entrée, les bits de sortie/bits internes, les mots d'entrée et mots de sortie/mots internes se trouvent en mémoire d'état.

Mémoire du programme CEI La mémoire du programme CEI comprend le code programme, le code EFB, les données de section et les données d'instance DFB.

MMI Interface Homme-Machine

Mode ASCII American Standard Code for Information Interchange. Le mode ASCII est utilisé pour la communication avec différents équipements hôte. ASCII fonctionne sur 7 bits de données.

Mode RTU Remote Terminal Unit
Le mode RTU est utilisé pour la communication entre l'API et un ordinateur personnel compatible IBM. RTU fonctionne sur 8 bits de données.

Module SA85 Le module SA85 est une carte Modbus Plus pour ordinateur IBM-AT ou compatible.

Mots d'entrée (Références 3x) Un mot d'entrée contient des informations émanant d'une source externe et par lesquelles un nombre sur 16 bits est représenté. Un registre 3x peut également contenir 16 bits successifs lus dans le registre au format binaire ou BCD (binaire codé décimal). Remarque : le x suivant immédiatement le premier chiffre du type de référence, représente un emplacement mémoire à cinq chiffres dans la mémoire de données utilisateur, p.ex. la référence 300201 signifie un mot d'entrée de 16 bits à l'adresse 201 de la mémoire d'état.

Mots de sortie/ mots internes (Références 4x) Un mot de sortie/mot interne peut être utilisé pour la mémorisation de données numériques (binaires ou décimales) en mémoire d'état, ou bien pour envoyer des données depuis l'UC vers une unité de sortie du système de contrôle. Remarque : le x suivant immédiatement le premier chiffre du type de référence, représente un emplacement mémoire à cinq chiffres dans la mémoire de données utilisateur, p.ex. la référence 400201 signifie un mot de sortie/mot interne de 16 bits à l'adresse 201 de la mémoire d'état.

Mots-clés Les mots-clés sont des combinaisons uniques de caractères utilisés comme éléments spéciaux de syntaxe comme il est défini à l'annexe B de la CEI 1131-3. Tous les mots-clés utilisés dans la CEI 1131-3 et donc dans Concept, sont listés en annexe C de la CEI 1131-3. Ces mots-clés répertoriés ne doivent être utilisés à aucune autre fin, p. ex. pas comme nom de variable, nom de section, nom d'instance, etc.

N

Node

Un node est une cellule de programmation dans un réseau LL984. Une cellule/un node comprend une matrice 7x11, c.-à-d. 7 lignes de 11 éléments.

Nom d'étape

Le nom d'étape sert à la désignation unique d'une étape dans une unité d'organisation de programme. Le nom d'étape est créé automatiquement, mais peut être édité. Il doit être unique dans toute l'unité d'organisation de programme, sinon un message d'erreur apparaît.
Le nom d'étape créé automatiquement a toujours la structure suivante : S_n_m

S = Etape
n = Numéro de la section (numéro courant)
m = Numéro de l'étape dans la section (numéro courant)

Nom d'instance

Un identificateur, associé à une instance spécifique de bloc fonction.. Le nom d'instance sert au repérage sans univoque d'un bloc fonction au sein d'une unité d'organisation de programme. Le nom d'instance est créé automatiquement, mais peut être édité. Le nom d'instance doit être unique dans toute l'unité d'organisation de programme, la distinction Majuscule/Minuscule n'est pas faite. Si le nom saisi existe déjà, vous en êtes averti et vous devez choisir un autre nom. Le nom d'instance doit satisfaire aux conventions de noms CEI, sinon un message d'erreur apparaît. Le nom d'instance créé automatiquement a toujours la structure suivante : FBI_n_m

FBI = Instance de bloc fonction
n = Numéro de la section (numéro courant)
m = Numéro de l'objet FFB dans la section (numéro courant)

Numéro d'identification

Le numéro d'identification sert à caractériser de manière unique une fonction dans un programme ou DFB. Le numéro d'identification ne peut être édité et est attribué automatiquement. Il a toujours la structure : .n.m

n = Numéro de la section (numéro courant)
m = Numéro de l'objet FFB dans la section (numéro courant)

O

Opérande

Un opérande est un libellé, une variable, un appel de fonction ou une expression.

Opérateur Un opérateur est un symbole d'une opération arithmétique ou booléenne à exécuter.

P

Paramètre d'entrée (Entrée) Transmet lors de l'appel d'un FFB l'argument s'y rapportant.

Paramètre de sortie (Sortie) Un paramètre avec lequel est (sont) retourné(s) le(s) résultat(s) de l'évaluation d'un FFB.

Paramètre réel Paramètre d'entrée/sortie actuellement attribué.

Paramètres formels Paramètres d'entrée/sortie, utilisés au sein de la logique d'un FFB et sortant du FFB en entrées ou en sorties.

Paysage Le format paysage signifie que la page, au regard du texte imprimé, est plus large que haute.

PC Le matériel et le logiciel gérant (supportant) la programmation, l'élaboration, le test, la mise en service et la recherche de défauts dans les applications API ainsi que dans les applications système décentralisées, afin de rendre possible la documentation et l'archivage des sources. Le cas échéant, le PC peut également être utilisé pour la visualisation du procédé.

Portrait Portrait signifie que la page, au regard du texte imprimé, est plus haute que large.

Presse-papiers Le presse-papiers est une mémoire temporaire pour les objets coupés ou copiés. Ces objets peuvent être collés dans des sections. A chaque nouveau "couper" ou "copier", l'ancien contenu du presse-papiers est écrasé.

Processeur de communication Le processeur de communication traite les passages de jeton et le flux de données entre le réseau Modbus Plus et la logique utilisateur de l'API.

Programmation de la redondance d'UC (Hot Standby) Un système redondant est constitué de deux API configurés de manière identique qui communiquent entre eux à l'aide de processeurs redondants. En cas de panne de l'API primaire, l'API secondaire prend le contrôle de l'automatisme. Dans les conditions normales, l'API secondaire n'effectue aucune fonction de commande mais il vérifie les informations d'état afin de déceler les erreurs.

Programme La plus haute unité d'organisation de programme. Un programme est chargé en entier sur un seul API.

Projet

Appellation générale du niveau le plus élevé d'une arborescence logicielle, qui définit le nom de projet supérieur d'une application d'API. Après avoir défini le nom du projet, vous pouvez sauvegarder votre configuration système et votre programme de commande sous ce nom. Toutes les données apparaissant lors de la création de la configuration et du programme font partie de ce projet supérieur pour cette tâche spéciale d'automatisation.
Désignation générale du jeu complet d'informations de programmation et de configuration dans la base de données de projet, laquelle représente le code source décrivant l'automatisation d'une installation.

R

REAL

REAL correspond au type de données "nombre à virgule flottante". L'entrée se fait en libellé réel ou en libellé réel avec exposant. La longueur des éléments de données est de 32 bits. Plage des valeurs des variables de ce type de données : +/-3.402823E+38.

Note : En fonction du type de processeur mathématique de l'UC, différentes zones de cette plage de valeurs permise ne peuvent pas être affichées. Cela s'applique aux valeurs tendant vers ZERO et aux valeurs tendant vers l'INFINI. Dans ces cas, une valeur NAN (**N**ot **A N**umber) ou INF (**INF**inite (infini)) est affichée en mode Animation.

Référence

Toute adresse directe est une référence commençant par un code indiquant s'il s'agit d'une entrée ou d'une sortie et s'il s'agit d'un bit ou d'un mot. Les références commençant par le chiffre 6 représentent des registres de la mémoire étendue de la mémoire d'état.
Plage 0x = bits internes/de sortie
Plage 1x = bits d'entrée
Plage 3x = mots d'entrée
Plage 4x = mots internes/de sortie
Plage 6x = registres dans la mémoire étendue

Note : Le x suivant immédiatement le premier chiffre de chaque type de référence représente un emplacement mémoire à cinq chiffres dans la mémoire de données utilisateur, p.ex. la référence 400201 signifie un mot de sortie/mot interne de 16 bits à l'adresse 201 de la mémoire d'état.

Registres dans la mémoire étendue (référence 6x)

Les références 6x sont des mots indicateurs dans la mémoire étendue de l'API. Ils ne peuvent être utilisés que pour les programmes utilisateur LL984 et seulement sur les UC CPU 213 04 ou CPU 424 02.

Représentation directe

Une méthode pour représenter une variable dans un programme d'API, à partir de laquelle peut être déterminée directement une correspondance avec un emplacement logique, et indirectement avec l'emplacement physique.

Réseau

Un réseau est une connexion commune d'appareils sur une voie de données commune qui communiquent entre eux à l'aide d'un protocole commun.

Réseau décentralisé (DIO)

Une programmation décentralisée dans le réseau Modbus Plus permet une performance maximale de l'échange de données et n'a aucune exigence particulière sur les liaisons. La programmation d'un réseau décentralisé est simple. La configuration du réseau ne nécessite pas de logique de schéma à contacts supplémentaire. Toutes les conditions du transfert de données sont remplies en renseignant les paramètres correspondants du processeur de communication.

RIO (E/S décentralisée)

L'E/S décentralisée indique un emplacement physique des appareils E/S à commande par point par rapport au processeur qui les gère. Les entrées/sorties décentralisées sont reliées avec l'appareil de commande via un câble de communication.

S

Saut

Elément du langage SFC. Les sauts sont utilisés pour éviter des zones de la séquence.

Schéma à contacts (LD)

Le schéma à contacts est un langage de programmation graphique conforme à la CEI1131, dont l'aspect visuel suit les "échelons" d'un schéma à relayage.

Schéma à contacts 984 (LL)

Comme leur nom l'indique, les schémas à contacts comportent des contacts. Contrairement à un schéma électrique, les électrotechniciens se servent d'un schéma à contacts pour dessiner un circuit (à l'aide de symboles électriques). Celui-ci doit montrer l'évolution d'événements, et non les fils en présence qui relient les différentes parties entre elles. Une interface de schéma à contacts permet de réaliser une interface utilisateur traditionnelle pour commander les actions des constituants d'automatisme, afin que les électrotechniciens ne soient pas obligés d'apprendre un langage de programmation avec lequel ils ne seraient pas à l'aise. La construction d'un schéma à contacts effectif permet de relier des éléments électriques de manière à créer une sortie de commande. Celle-ci dépend d'un flux d'énergie logique passant par les objets électriques utilisés, lesquels représentent la condition préalable nécessaire d'un appareil électrique physique.
Sous une forme simple, l'interface utilisateur est un écran vidéo élaboré par l'application de programmation d'API, organisant un quadrillage vertical et horizontal dans lequel sont rangés des objets de programmation. Le schéma reçoit du courant par le côté gauche du quadrillage, et par connexion à des objets activés, le courant circule de gauche à droite.

Section

Une section peut par exemple être utilisée pour décrire le principe de fonctionnement d'une unité technologique telle qu'un moteur.
Un programme ou un DFB est constitué d'une ou de plusieurs sections. Les sections peuvent être programmées à l'aide des langages de programmation CEI FBD et SFC. Au sein d'une même section, seul un des langages de programmation mentionnés peut être utilisé.
Dans Concept, chaque section a sa propre fenêtre de document. Cependant, pour des raisons de clarté, il est conseillé de subdiviser une grande section en plusieurs petites. La barre de défilement sert à se déplacer au sein d'une section.

Station d'E/S DCP

A l'aide d'un processeur de contrôle distribué (D908), vous pouvez configurer un réseau décentralisé piloté par un API. Lorsque l'on utilise un D908 avec API décentralisé, l'API pilote considère l'API décentralisé comme une station d'E/S décentralisée. Le D908 et l'API décentralisé communiquent par le bus système, ce qui permet une grande performance pour un effet minimal sur le temps de cycle. L'échange de données entre le D908 et l'API pilote s'effectue par le bus d'E/S décentralisé à 1,5 Mégabit par seconde. Un API pilote peut gérer jusqu'à 31 processeurs D908 (adresse 2-32).

SY/MAX

Dans les automates Quantum, Concept gère la mise à disposition des modules d'E/S SY/MAX sur l'affectation des E/S pour la commande RIO par l'API Quantum. Le châssis distant SY/MAX dispose d'une carte d'E/S distante à l'emplacement 1, laquelle communique par un système d'E/S Modicon S908 R. Les modules d'E/S SY/MAX vous sont listés pour la sélection et la prise en compte dans l'affectation des E/S de la configuration Concept.

Symbole (icône) Représentation graphique de différents objets sous Windows, p. ex. lecteurs, programmes utilisateur et fenêtre de document.

T

Tas CEI Le tas CEI comprend la mémoire du programme CEI et les données globales.

TIME TIME est le type de données "durée". L'entrée se fait sous forme de libellé de durée. La longueur des éléments de données est de 32 bits. La plage de valeurs des variables de ce type de données va de 0 à 2exp(32)-1. L'unité du type de données TIME est 1 ms.

Transition La condition par laquelle la commande d'une ou de plusieurs étapes précédentes passe à une ou plusieurs étapes suivantes le long d'une liaison.

Type de bloc fonction Un élément de langage constitué de : 1. la définition d'une structure de données, subdivisée en variables d'entrée, de sortie et internes ; 2. un jeu d'opérations exécutées avec les éléments de la structure de données, lorsqu'une instance du type de bloc fonction est appelée. Ce jeu d'opérations peut être formulé soit dans l'un des langages CEI (type DFB) ou en "C" (type EFB). Un type de bloc fonction peut être instancié (appelé) plusieurs fois.

Type de données dérivé Les types de données dérivés sont des types de données qui ont été dérivés des types de données élémentaires et/ou d'autres types de données dérivés. La définition des types de données dérivés s'effectue dans l'éditeur de type de données de Concept.
On fait la distinction entre les types de données globaux et les types de données locaux.

Type de données générique Un type de données représentant plusieurs autres types de données.

Types de données

La vue d'ensemble montre la hiérarchie des types de données et comment ils sont utilisés aux entrées et sorties des fonctions et blocs fonction. Les types de données génériques sont caractérisés par le préfixe "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- Types de données système (Extension CEI)
- Dérivé (des types de données 'ANY')

Types de données dérivés globaux

Les types de données dérivés globaux sont disponibles dans tout projet Concept et sont enregistrés dans le répertoire DFB directement situé sous le répertoire Concept.

Types de données dérivés locaux

Les types de données dérivés locaux ne sont disponibles que dans un seul projet Concept et ses DFB locaux et sont enregistrés dans le répertoire DFB sous le répertoire de projet.

U

UDEFB

Fonctions/Blocs fonction élémentaires défini(e)s par l'utilisateur
Fonctions ou blocs fonction créés en langage de programmation C et que Concept met à votre disposition dans des bibliothèques.

UDINT

UDINT représente le type de données "entier double non signé (unsigned double integer)". L'entrée s'effectue en libellé entier, libellé en base 2, libellé en base 8 ou libellé en base 16. La longueur des éléments de données est de 32 bits. La plage de valeurs des variables de ce type de données va de 0 à 2exp(32)-1.

UINT

UINT représente le type de données "entier non signé (unsigned integer)". L'entrée s'effectue en libellé entier, libellé en base 2, libellé en base 8 ou libellé en base 16. La longueur des éléments de données est de 16 bits. La plage des valeurs des variables de ce type de données va de 0 à 2 exp(16) -1.

Unité d'organisation de programme

Une fonction, un bloc fonction ou un programme. Ce terme peut se rapporter à un type ou à une instance.

V

Valeur initiale

La valeur affectée à une variable lors du lancement du programme. L'affectation de la valeur s'effectue sous forme d'un libellé.

Variable localisée

Une adresse de mémoire d'état (adresses de références 0x, 1x, 3x, 4x) est affectée aux variables localisées. La valeur de ces variables est enregistrée dans la mémoire d'état et peut être modifiée en ligne au moyen de l'éditeur de données de référence. Ces variables peuvent être adressées avec leur nom symbolique ou avec leur adresse de référence.

Toutes les entrées et les sorties de l'API sont reliées à la mémoire d'état. L'accès du programme aux signaux des périphériques connectés à l'API ne se fait que via des variables localisées. Les accès de l'extérieur via les interfaces Modbus ou Modbus Plus de l'API, p. ex. des systèmes de visualisation, sont également possibles via des variables localisées.

Variable non localisée

Aucune adresse de mémoire d'état n'est affectée aux variables non localisées. Elles n'occupent donc pas non plus d'adresse de mémoire d'état. La valeur de ces variables est enregistrée dans le système et peut être modifiée en ligne au moyen de l'éditeur de données de référence. Ces variables ne sont adressées que par leur nom symbolique.

Les signaux ne disposant pas d'accès à la périphérie, p. ex, résultats intermédiaires, repères systèmes, etc., doivent être de préférence déclarés comme variable non localisée.

Variables

Les variables servent à l'échange de données au sein de sections, entre plusieurs sections et entre le programme et l'API.
Les variables consistent au moins en un nom de variable et un type de données. Si une adresse directe (référence) est affectée à une variable, on parle alors de variable localisée. Si aucune adresse directe n'est affectée à une variable, on parle alors de variable non localisée. Si un type de données dérivé est affecté à une variable, on parle alors d'une variable multi-éléments.
Il existe en outre des constantes et des libellés.

Variables de tableau

Variables auxquelles sont affectées untype de données dérivé défini à l'aide du mot clé ARRAY (tableau). Un tableau est un ensemble d'éléments de données appartenant au même type.

Variables multi-éléments

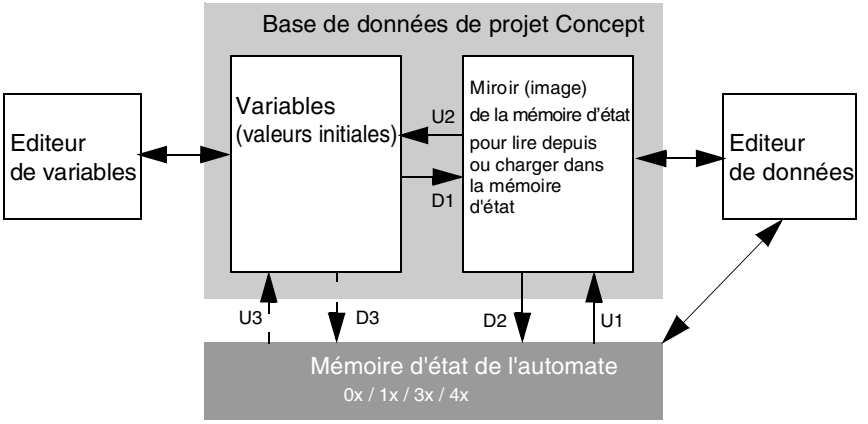
Variables, auxquelles est affecté un type de données dérivé défini avec STRUCT ou ARRAY.
On fait ici la distinction entre variables de tableau et variables structurées.

Variables structurées

Variables auxquelles est affecté un type de données dérivé défini avec STRUCT (structure).
Une structure est un ensemble d'éléments de données avec en général différents types de données (types de données élémentaires et/ou types de données dérivés).

Vue d'ensemble de la mémoire d'état lors de la lecture et du chargement

Vue d'ensemble :

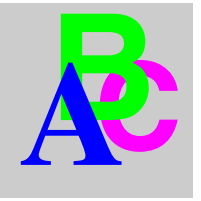


W

WORD

WORD correspond au type de données "Cordon de bits 16". L'entrée peut se faire en libellé en base 2, libellé en base 8 ou libellé en base 16. La longueur des éléments de données est de 16 bits. Il n'est pas possible d'affecter une plage de valeurs numériques à ce type de données.

Index



D

E

F

G

I

L

M

N

O

P

R

S

T

U

W

X