JUMO diraTRON/diraVIEW 104/108/116/132

Régulateurs compacts/Indicateurs numériques

Notice Editeur ST

70211000T96Z002K000

V2.00/FR/00688821

Sommaire

Sommaire

1.1 Instructions relatives à la sécurité

Symboles indiquant une remarque

REMARQUE !

Ce pictogramme renvoie à une **information importante** sur le produit, sur son maniement ou ses applications annexes.

Renvoi !

Ce pictogramme renvoie à des **informations supplémentaires** dans d'autres sections, chapitres ou notices.

1.2 Utilisation conforme aux prescriptions

L'appareil est conçu pour une utilisation dans un environnement industriel, comme spécifié dans les caractéristiques techniques des différents modules du système. Toute autre utilisation ou hors de ce cadre est considérée comme non conforme.

L'appareil est fabriqué conformément aux normes et directives applicables ainsi qu'aux règles de sécurité en vigueur. Toutefois une utilisation inappropriée peut provoquer des dommages corporels ou des dégâts matériels.

Pour écarter tout danger, l'appareil ne peut être utilisé que :

- conformément à sa destination
- dans des conditions de sécurité irréprochables
- dans le respect de la documentation technique fournie

Même si l'appareil est utilisé de façon appropriée ou conformément à sa destination, il peut être une source de danger lié à l'application, par ex. à cause de réglages incorrects ou l'absence de dispositifs de sécurité.

1.3 Qualification du personnel

Ce document contient les informations nécessaires pour une utilisation conformément à leur destination des modules décrits.

Il s'adresse à un personnel qualifié du point de vue technique, formé spécialement et qui possède des connaissances en matière d'automatisation.

La connaissance et l'application techniquement parfaite des conseils de sécurité et des avertissement contenus dans la documentation technique livrée sont les conditions préalables à un montage, une installation et une mise en service sans danger ainsi qu'à la sécurité pendant le fonctionnement des modules décrits. Seul un personnel qualifié dispose des connaissances techniques nécessaires pour interpréter correctement, sur des cas concrets, les conseils de sécurité et les avertissements utilisés dans ce document ainsi que pour les mettre en oeuvre.

1.4 Contenu de ce document

REMARQUE !

Ce document est aussi bien valable pour les appareils de la série 70211x (régulateurs compacts) que les appareils de la série 70151x (indicateurs).

Ce document décrit l'application de l'éditeur ST, qui permet à l'utilisateur de créer sa propre application dans le langage de programmation d'automate "Texte structuré" (ST). Le document est destiné aux utilisateurs ayant des compétences en programmation.

1 Introduction

Le langage de programmation d'automate "Texte structuré" (ST) est décrit dans la norme CEI 61131-3. Vous trouverez des informations détaillées pour la programmation dans cette norme. Le module ST de l'appareil en question ne prend en charge qu'une partie du langage de programmation décrit dans la norme.

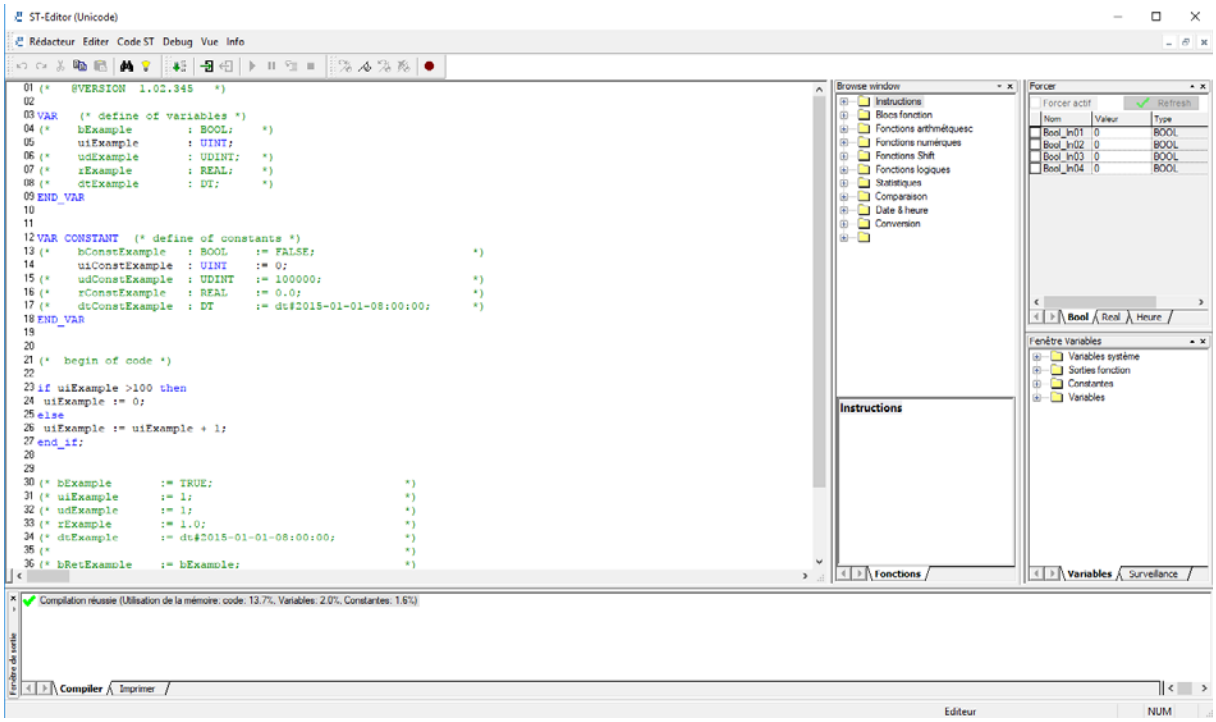
Outre ce document, il faut tenir compte de la notice de mise en service de la série d'appareils concernée :

- Types 701510, 701511, 701512, 701513, 701514 (indicateurs numériques) :
  document 70151000T90Z...K...
- Types 702110, 702111, 702112, 702113, 702114 (régulateurs compacts) :
  document 70211000T90Z...K...

L'éditeur ST fait partie du programme Setup et est démarré à partir de là en cliquant sur le bouton correspondant dans la fenêtre „Code ST“ (voir notice de mise en service).

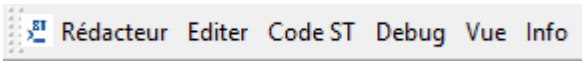
L'application achevée est transmise à l'appareil comme code ST et est traitée en permanence dans le module ST intégré.

Vue d'ensemble



L'interface du programme se compose de plusieurs barres et fenêtres, qui sont brièvement décrites dans les chapitres suivants.

Barre de menu



Les menus individuels contiennent des fonctions d'édition et de traduction d'un programme, de recherche d'erreurs, de masquage et d'affichage des barres d'outils et des fenêtres de l'interface du programme ainsi que des informations de version pour l'éditeur ST.

De plus amples informations sur les fonctions de la barre de menus s'affichent dans la barre d'état lorsque le pointeur de la souris pointe sur la fonction individuelle dans le menu respectif.

2 Interface du programme

Barre d'outils

Certaines fonctions de la barre de menu sont également disponibles dans la barre d'outils et peuvent être sélectionnées par un simple clic de souris. Une Infobulle donne la signification des icônes (avec le pointeur de la souris, pointer sur le symbole correspondant). De plus amples informations concernant les fonctions concernées s'affichent dans la barre d'état.

Symbole	Signification	Description
	Annuler (ctrl+Z)	Annuler la dernière action
	Restaurer	Répéter l'action précédemment annulée
	Couper (Ctrl+X)	Supprimer les données sélectionnées et les transférer dans le presse-papiers
	Copier (Ctrl+C)	Copier les données sélectionnées et les transférer dans le presse-papiers
	Coller (Ctrl+V)	Coller les données du presse-papier
	Rechercher (Ctrl+F)	Rechercher texte saisi
	Texte de fonction (F4)	Insérer le texte de fonction de la fenêtre de navigateur Voir également chapitre "Browse window ", page 11.
	Traduire (F7)	Traduire code ST
	Démarrer débogage	Démarrer débogage (démarrage à froid) Après le démarrage, un programme déjà présent dans l'appareil est terminé. Au lieu de cela, le programme en cours est chargé dans l'appareil. Voir également chapitre "Débogage ", page 13.
	Quitter débogage	Quitter débogage Après l'arrêt, le programme actuel est maintenu dans l'appareil et est exécuté. Voir également chapitre "Quitter l'éditeur ST ", page 13.
	Démarrer/Suivant (F5)	Démarrer le programme ou continuer après l'interruption
	Interrompre le programme	Interrompre le programme
	Etape unique (F11)	Exécuter le programme en une seule étape
	Quitter	Quitter le programme
	Signet suivant (F2)	Déplacer le curseur sur le signet suivant
	Commuter signet (Ctrl+F2)	Commuter le signet pour la ligne actuelle
	Signet précédent (maj.+F2)	Déplacer le curseur sur le signet précédent
	Effacer le signet (maj.+Ctrl+F2)	Effacer tous les signets

Symbole	Signification	Description
	Commuter le point d'arrêt (F9)	Commuter le point d'arrêt pour la ligne actuelle

Fenêtre d'édition

```
01 (*   @VERSION  1.02.345   *)
02
03 VAR   (* define of variables *)
04 (*    bExample        : BOOL;    *)
05       uiExample       : UINT;
06 (*    udExample       : UDINT;   *)
07 (*    rExample        : REAL;    *)
08 (*    dtExample       : DT;      *)
09 END_VAR
10
11
12 VAR CONSTANT  (* define of constants *)
13 (*    bConstExample   : BOOL    := FALSE;                      *)
14       uiConstExample  : UINT    := 0;
15 (*    udConstExample  : UDINT   := 100000;                     *)
16 (*    rConstExample   : REAL    := 0.0;                        *)
17 (*    dtConstExample  : DT      := dt#2015-01-01-08:00:00;     *)
18 END_VAR
19
20
21 (*  begin of code *)
22
23 if uiExample >100 then
24  uiExample := 0;
25 else
26  uiExample := uiExample + 1;
27 end_if;
28
```

Dans la fenêtre d'édition est créé le programme dans lequel sont déclarés les variables et les constantes, est créé le code de programme et dans lequel les textes de commentaire sont utilisés

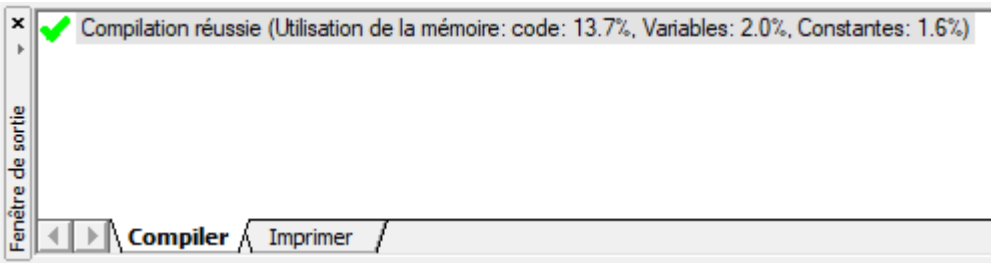
Afin d'attribuer un nom de version au programme, le mot-clé @VERSION suivi du nom de la version doit être utilisé dans une ligne de commentaire. Si le programme contient plusieurs lignes de commentaire avec ce mot-clé, seule la première ligne de commentaire est analysée. Après transfert permanent du code ST vers l'appareil, l'intitulé de la version peut être affiché dans l'appareil (Info appareil > Versions > Version-Code-ST).

Ctrl + molette de la souris peut être utilisé pour changer la taille de la police.

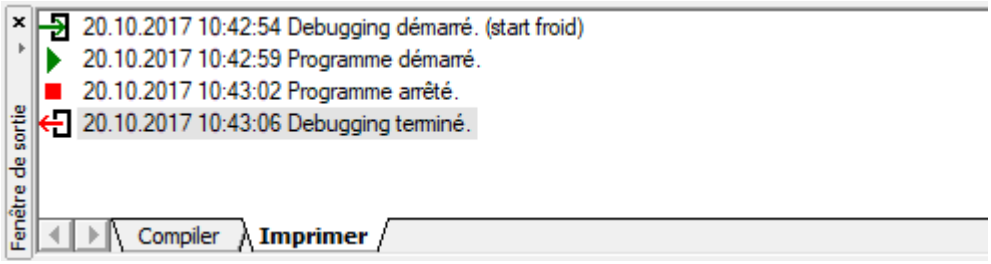
2 Interface du programme

Fenêtre de sortie

La fenêtre de sortie se compose de 2 parties (onglets)

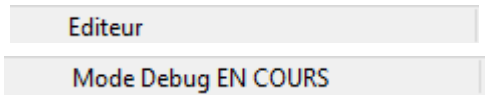


L'onglet „Compilateur“ affiche des messages concernant la compilation du code de programme.



L'onglet „Imprimer“ contient des messages liés au mode de débogage.

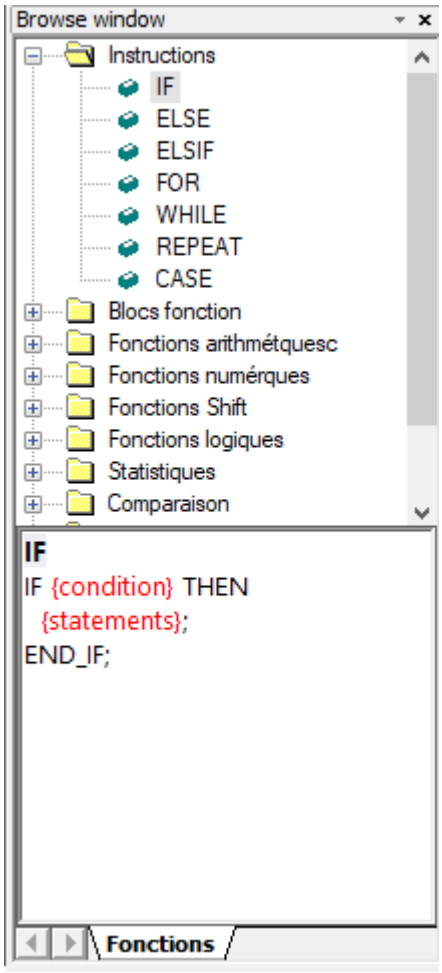
Barre d'état



La barre d'état se trouve en bas de l'interface du programme et indique si le mode éditeur ou débogage est actif. En mode Debug, les différents états sont affichés (MARCHE, ARRET, PAUSE).

De plus amples informations sur les fonctions de la barre de menus et de la barre d'outils s'affichent dans la barre d'état (avec le pointeur de la souris pointant sur la fonction individuelle dans le menu respectif ou sur l'icône dans la barre d'outils).

Browse window



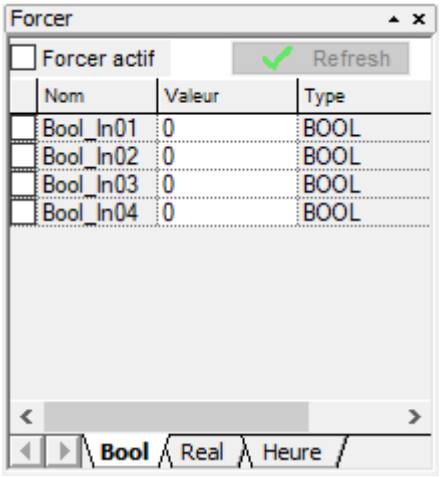
Browse window répertorie toutes les instructions et fonctions pouvant être utilisées dans l'éditeur. La partie inférieure de la fenêtre montre l'utilisation de l'instruction ou de la fonction.

Le texte de fonction affiché dans la partie inférieure de Browse window peut être inséré dans le code ST à l'emplacement souhaité dans le code ST par glisser-déposer (en sélectionnant le texte), en cliquant sur le symbole "Fonction" ou en utilisant la touche F4 au lieu d'un texte marqué). Après l'insertion, l'espace réservé suivant est mis en surbrillance dans le texte de la fonction en cliquant plusieurs fois sur le symbole ou en appuyant sur F4.

2 Interface du programme

Force window

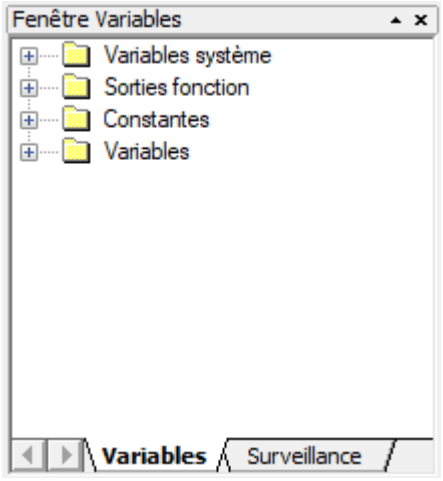
Force-window se compose de 3 parties (onglets). Seul l'onglet „Bool“ est représenté.



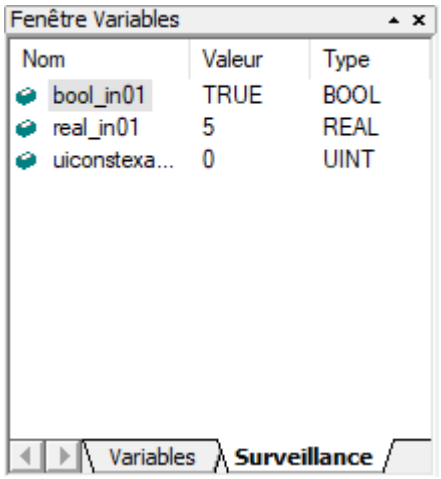
L'utilisateur peut activer les variables d'entrée de type „Bool“ et „Real“ ainsi que la date et l'heure dans Force-window afin de tester le code du programme.

Fenêtre Variables

La fenêtre Variables se compose de 2 parties (onglets).



Toutes les variables système et les sorties de fonction (sorties des blocs fonctionnels) ainsi que les constantes et variables déclarées par l'utilisateur (drapeau) sont répertoriées dans l'onglet „Variables“. Elles peuvent être copiées de la fenêtre Variables par glisser-déposer et insérées dans la fenêtre d'édition à l'endroit souhaité dans le code ST.



Dans l'onglet "Surveillance", les variables et les constantes peuvent être affichées et observées pendant le débogage. Pour ce faire, ils doivent être copiés depuis la fenêtre d'édition ou l'onglet "Variables" par glisser-déposer et insérés dans l'onglet "Surveillance".

L'insertion et la suppression ne sont possibles que si le programme n'est pas en cours d'exécution. Ensuite, diverses fonctions sont également disponibles via le menu contextuel (bouton droit de la souris).

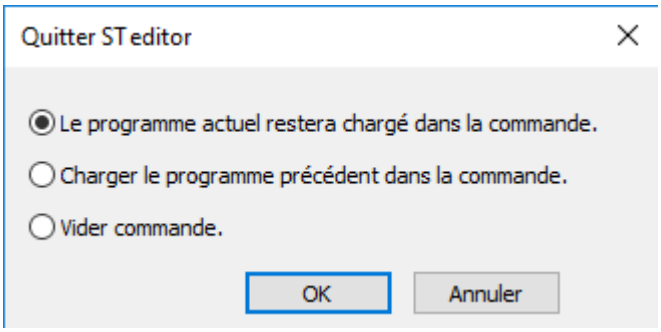
Débogage

Avec la fonction *Debug > Démarrer débogage* (barre de menu) les variantes suivantes sont disponibles :

- Démarrage à froid : un programme déjà en cours d'exécution dans l'appareil est terminé et le programme en cours de l'éditeur ST est chargé dans l'appareil. Toutes les variables (variables retain et non-retain) sont réinitialisées sur leur valeur par défaut.
- Démarrage à chaud : les variables retain ne sont pas réinitialisées (appareils de types 70211x et 70151x ne prennent pas en charge les variables retain).
- Démarrage de service : le programme en cours dans l'appareil est utilisé dans l'éditeur ST et peut être analysé. Aucune variable n'est réinitialisée. La condition du démarrage du service est que le code compilé dans l'éditeur ST concorde avec le code de l'appareil.

Quitter l'éditeur ST

En quittant l'éditeur ST avec *Editeur > Enregistrer et quitter* (barre de menu) les choix suivants sont possibles (si le mode de débogage était actif) :



- *Le programme reste chargé dans la commande*
  Le dernier programme chargé en mode débogage doit être utilisé.
- *Charger le programme dans la commande.*
  Le dernier programme chargé en mode débogage ne doit pas être utilisé. Au lieu de cela, le programme précédent (chargé en permanence dans l'appareil) est utilisé, si disponible.
- *Vider la commande.*
  Ni le dernier programme chargé en mode débogage ni le programme précédent ne doit être utilisé.

2 Interface du programme

REMARQUE !

En mode débogage, le programme est chargé temporairement (pas de sauvegarde en cas de panne de courant). Pour effectuer des modifications permanentes sur l'appareil, le fichier Setup doit être transféré sur l'appareil après la sortie de l'éditeur ST (fermer la fenêtre du code ST avec „OK“, transfert des données vers l'appareil).

REMARQUE !

Le code source créé dans l'éditeur ST est transféré compilé dans l'appareil. Il est donc nécessaire de sauvegarder le code source séparément dans un fichier Setup.

Les variables système incluent les variables d'entrée et de sortie ainsi que les variables internes.

Les valeurs des différents types de données peuvent être échangées entre l'appareil et le module ST intégré dans l'appareil via les variables d'entrée et de sortie. Les variables internes ne sont pertinentes que dans le module ST et peuvent être utilisées pour réaliser certaines fonctions.

3.1 Variables d'entrée

Les variables d'entrée fournissent des valeurs de l'appareil à utiliser dans le module ST.

L'affectation des valeurs aux variables d'entrée s'effectue dans le programme Setup en sélectionnant les signaux respectifs des sélecteurs (sélecteur analogique ou sélecteur numérique). En outre, certaines variables sont affectées de manière permanente dans l'appareil et n'apparaissent donc pas dans les sélecteurs.

bool_in

Désignation	Désignation dans le programme Setup	Description
bool_in01 à bool_in04	bool_in01 à bool_in04	Variables d'entrée booléennes ; affectation flexible dans le programme Setup (sélecteur numérique)

real_in

Désignation	Désignation dans le programme Setup	Description
real_in01 à real_in06	real_in01 à real_in06	Variables d'entrée real ; affectation flexible dans le programme Setup (sélecteur analogique)

dword_in

Désignation	Désignation dans le programme Setup	Description
dword_in01 dword_in02	(pas de désignation, pas d'affectation fixe)	(Ces variables d'entrée à double mot ne sont pas utilisées.)

rtc.cdtrtc.cdt

Désignation	Désignation dans le programme Setup	Description
rtc.cdtrtc.cdt	(pas de désignation, affectation fixe)	Ces variables d'entrée de type DT donnent le temps écoulé depuis la mise sous tension de l'appareil (date et heure commencent à chaque mise sous tension à partir de 01.01.1970 00:00:00).

3 Variables système

3.2 Variables de sortie

Les variables de sortie sont utilisées pour transférer les valeurs générées dans le module ST vers l'appareil.

Les variables de sortie sont disponibles pour la configuration dans le programme Setup et aux niveau des sélecteurs dans l'appareil (sélecteur analogique et/ou numérique) et peuvent être utilisées individuellement. En outre, il existe des variables qui ont des fonctions définies dans l'appareil (par ex. affichage de texte) et qui par conséquent ne peuvent être sélectionnées via les sélecteurs.

Pour certaines variables de sortie, il existe dans la fenêtre des variables des variables avec les désignations „...conf“ et „...sec“. Ces variables peuvent être utilisées pour implémenter, en cas d'erreur, un comportement défini dans le code ST.

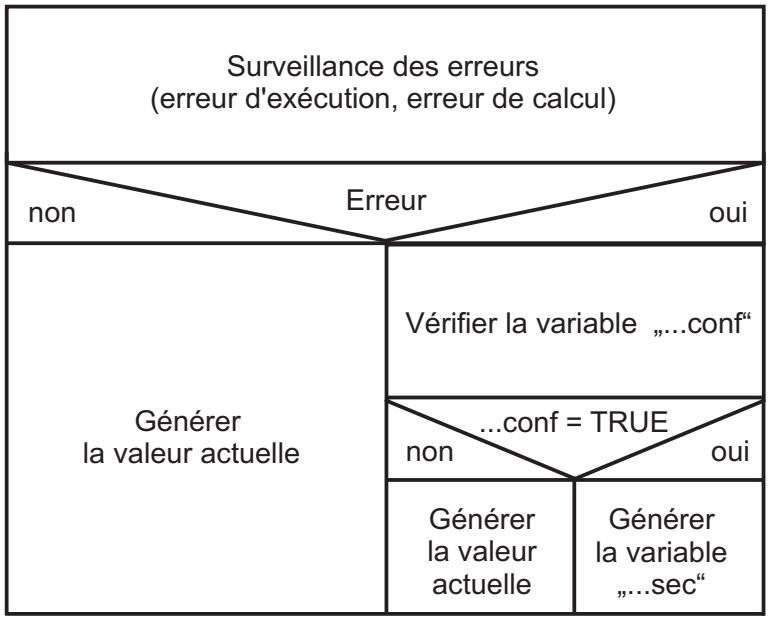
- La v**ariable „...conf“** peut être réglée dans le code ST, pour définir quelle valeur prend, en cas d'erreur, en charge la variable de sortie correspondante : FALSE = valeur actuelle ; TRUE = valeur de remplacement.

  Réglage par défaut : FALSE
- La v**ariable „...sec“** comprend la valeur de remplacement ; dans le code ST, elle peut être réglée sur une valeur définie ou sur la valeur par défaut.

  Dans le cas des variables REAL, la conformité de la valeur de remplacement avec les limites d'échelle est vérifiée. Si elle se situe en dehors des limites, la valeur par défaut sera utilisée.

  La fonction SET_DEFAULT est disponible pour régler la valeur par défaut.

  ⇨ 8.9 "Autres fonctions", page 42



bool_out

Désignation	Désignation dans le programme Setup et dans l'appareil	Description
bool_out01 à bool_out04	1ère sortie numérique ST jusqu'à 4ème sortie numérique ST	Variables de sortie booléennes ; utilisation flexible dans le programme Setup et dans l'appareil (sélecteur numérique) Dans le programme Setup, un texte descriptif peut être attribué à chaque variable.

real_out

Désignation	Désignation dans le programme Setup et dans l'appareil	Description
real_out01 à real_out06	1ère sortie analogique ST jusqu'à 6ème sortie analogique ST	Variables de sortie real ; utilisation flexible dans le programme Setup et dans l'appareil (sélecteur analogique) Dans le programme Setup, un texte descriptif peut être attribué à chaque variable. En outre, des réglages concernant les variables sont nécessaire (par ex. mise à l'échelle).

alarm

Désignation	Désignation dans le programme Setup et dans l'appareil	Description
alarm	Alarme ST	Variables de sortie booléennes ; utilisation flexible dans le programme Setup et dans l'appareil (sélecteur numérique) Cette variable a contrairement aux variables bool_out une désignation spéciale. Elle peut, cependant, être utilisée librement dans le code ST.

error_out

Désignation	Désignation dans le programme Setup et dans l'appareil	Description
error_out	Erreur ST	Variables de sortie booléennes ; utilisation flexible dans le programme Setup et dans l'appareil (sélecteur numérique) Cette variable est définie sur TRUE si le code ST ne peut plus être exécuté, comme dans les cas suivants : • Division par zéro • Accès array erroné • Dépassement du temps d'exécution La variable est réinitialisée sur FALSE lorsque le code ST a été exécuté complètement, une fois l'erreur survenue

dword_out

Désignation	Désignation dans le programme Setup et dans l'appareil	Description
dword_out01, dword_out02	(pas de désignation, affectation fixe)	Les variables de sortie à double mot de l'application ; affectation fixe : affichage du texte à l'écran Les textes doivent être saisis dans le programme Setup, fenêtre de dialogue „Affichage/commande“ sous „Textes d'affichage“.

3 Variables système

Variable	Valeur	Fonction
dword_out01	0	Pas d'affichage de texte
	1 à 10	Appareils en formats 108H, 108Q et 104 : le texte respectif (1 à 10) s'affiche sur la 3ème ligne. Appareils en formats 132, 116 : le texte respectif (1 à 10) s'affiche sur la 1ère ligne.
dword_out02	0	Pas d'affichage de texte
	1 à 10	Appareils en formats 108H, 108Q et 104 : le texte respectif (1 à 10) s'affiche sur la 4ème ligne. Appareils en formats 132, 116 : le texte (1 à 10) s'affiche sur la 2e ligne.

3.3 Variables internes

Les variables internes peuvent être utilisées pour réaliser certaines fonctions dans le code ST.

ext_error

Désignation	Description
ext_error	Variable de type entier La variable est 0, lorsque la fonction Set_CP (définir les paramètres du régulateur) a été exécutée sans erreur. En cas d'erreur, la variable livre une valeur > 0.

reset_flag

Désignation	Description
reset_flag	Variable booléenne La valeur est réinitialisée sur TRUE après mise sous tension. Lorsqu'après mise sous tension, le code ST a été complètement exécuté sans qu'une erreur ne se produise, la variable est réglée sur FALSE. Autrement elle reste sur TRUE. Mode Debug : Lorsque vous démarrez en mode débogage ou démarrez après un arrêt de programme, la variable est TRUE. Après une deuxième exécution, elle est définie sur FALSE.

sys_error

Désignation	Description
sys_error	Variable real La variable livre une valeur d'erreur si une erreur se produit pendant l'exécution du programme: 1.00E+37 = valeur d'erreur générale 1.0E+37 = dépassement inférieur de la plage 2.0E+37 = dépassement supérieur de la plage 3.0E+37 = valeur d'entrée invalide 4.0E+37 = division par zéro 5.0E+37 = valeur mathématique erronée 7.0E+37 = valeur invalide (générale)

week_day

Désignation	Description
week_day	Variable de type entier La variable comprend le jour de semaine actuel : 0 =dimanche, 1 = lundi, 2 = mardi, 3 = mercredi, 4 = jeudi, 5 = vendredi, 6 = samedi Le jour de la semaine est dérivé de la variable d'entrée rtc.cdt.

3 Variables système

Le module ST prend en charge les types de données suivants :

- Valeur booléenne (BOOL)
- Nombre entier non signé, 2 octets (UINT)
- Nombre entier double non signé, 4 octets (UDINT)
- Nombre à virgule flottante (REAL)
- Date et heure (DT ou DATE_AND_TIME)

REMARQUE !

Pour les opérations, il n'y a pas de surveillance des types de données.
Exceptions : fonction racine carrée (SRQT), valeur inverse (1/x)

REMARQUE !

Le module ST ne prend en charge aucune variable qui a été enregistrée hors tension (variables retain)

Valeur booléenne

Mot-clé : BOOL

Plage de valeurs : TRUE (1) ou FALSE (0)

Déclaration d'une variable (exemple):

```
VAR
    bExample : BOOL;
END_VAR
```

Déclaration d'une constante (exemple) :

```
VAR CONSTANT
    bConstExample : BOOL := FALSE;
END_VAR
```

Affectation (exemple) :

```
bExample := TRUE;
```

Nombre entier (2 octets)

Mot-clé : UINT

Plage de valeurs : 0 à 65535 (0 à 2^{16}-1)

Déclaration d'une variable (exemple):

```
VAR
    uiExample : UINT;
END_VAR
```

Déclaration d'une constante (exemple) :

```
VAR CONSTANT
    uiConstExample : UINT := 0;
END_VAR
```

Affectation (exemple) :

```
uiExample := 1;
```

4 Types de données

Nombre entier double (4 octets)

Mot-clé : UDINT

Plage de valeurs : 0 à 4294967295 (0 à 232-1)

Déclaration d'une variable (exemple):

```
VAR
    udExample : UDINT;
END_VAR
```

Déclaration d'une constante (exemple) :

```
VAR CONSTANT
    udConstExample : UDINT := 100000;
END_VAR
```

Affectation (exemple) :

```
udExample := 200000;
```

Nombre à virgule flottante

Mot-clé : REAL

Plage de valeurs : -1.5E-45 à 1.0E37
(les valeurs ≥ 1.0E37 sont interprétées comme des valeurs d'erreur.)

Déclaration d'une variable (exemple):

```
VAR
    rExample : REAL;
END_VAR
```

Déclaration d'une constante (exemple) :

```
VAR CONSTANT
    rConstExample : REAL := 0.0;
END_VAR
```

Affectation (exemple) :

```
rExample := 1.0;
```

Date et heure

Mots-clés : DT ou DATE_AND_TIME

Plage de valeurs **:** yyyy-mm-dd-hh:mm:ss

Déclaration d'une variable (exemple):

```
VAR
    dtExample : DT;
END_VAR
```

Déclaration d'une constante (exemple) :

```
VAR CONSTANT
    dtConstExample : DT := dt#2017-12-31-23:59:59;
END_VAR
```

Affectation (exemple) :

```
dtExample := dt#2017-01-01-08:00:00;
```

4.1 Champs

Tous les types de données peuvent également être déclarés comme champ unidimensionnel ou bidimensionnel (Array) (voir norme CEI 61131-3). Les chapitres suivants utilisent le type de données REAL comme exemple.

Champ unidimensionnel

Déclaration de variables (exemple) :

```
VAR
    rArrayExample : ARRAY [0..2] OF REAL; (* champ de 3 variables *)
END_VAR
```

Déclaration de constantes (exemple) :

```
VAR CONSTANT
    rArrayConstExample : ARRAY [0..2] OF REAL := [0.0, 0.1, 0.2]; (* champ de 3 constantes *)
END_VAR
```

Affectation (exemple) :

```
rArrayExample[0] := 0.0;
rArrayExample[1] := 0.1;
rArrayExample[2] := 0.2;
```

Champ bidimensionnel

Déclaration de variables (exemple) :

```
VAR
    rArrayExample : ARRAY [0..2, 0..1] OF REAL; (* champ de 3 x 2 variables *)
END_VAR
```

Déclaration de constantes (exemple) :

```
VAR CONSTANT
    rArrayConstExample : ARRAY [0..2, 0..1] OF REAL :=
    [0.0, 0.1, 0.2,
    1.0, 1.1, 1.2]; (* champ de 3 x 2 constantes *)
END_VAR
```

Affectation (exemple) :

```
rArrayExample[0,0] := 0.0;
rArrayExample[1,0] := 0.1;
rArrayExample[2,0] := 0.2;
rArrayExample[0,1] := 1.0;
rArrayExample[1,1] := 1.1;
rArrayExample[2,1] := 1.2;
```

4 Types de données

Le tableau suivant répertorie tous les opérateurs pris en charge par le module ST. L'ordre dans le tableau dépend du classement des opérateurs, en commençant par le rang le plus élevé.

Opération	Symbole	Types de données admissibles	Exemple
Parenthèse	(expression)		a := 3.0 * (b - 1.0);
Fonction	Identifiant (liste d'arguments)		i := MIN (3, j);
Négation	-	REAL	a := -a;
Complément	NOT	BOOL, UINT, UDINT[a]	a := NOT b;
Multiplication	*	UINT, UDINT, REAL	i := 5 * j;
Division	/		a := 5.0 / b;
Modulo	MOD	UINT, UDINT	j := i MOD 10;
Addition	+	UINT, UDINT, REAL	i := 5 + j;
Soustraction	-		a := b - 5.0E20;
Comparaison	<, >, <=, >=	UINT, UDINT, REAL, DATE_AND_TIME	bExample := 5 <= j;
Egalité	=	BOOL, UINT, UDINT, REAL, DATE_AND_TIME	bExample := 5 = i;
Inégalité	<>		bExample := 5 <> j;
UND	& ou AND	BOOL, UINT, UDINT[a]	bExample := x AND y;
OU exclusif	XOR	BOOL, UINT, UDINT[a]	bExample := x XOR y;
OU	OR	BOOL, UINT, UDINT[a]	bExample := x OR y;

[a] Opérateur logique pour les variables booléennes, bit à bit pour les variables entières.

5 Opérateurs

6 Commandes de sélection

Une commande de sélection exécute une instruction ou un ensemble d'instructions en fonction d'une condition spécifiée.

Instruction IF

L'instruction IF spécifie qu'un ensemble d'instructions n'est exécuté que si l'expression booléenne associée livre la valeur TRUE (vraie). Si la condition est fausse, aucune instruction ne peut être exécutée ou le groupe d'instructions suivant le mot-clé ELSE (ou le mot-clé ELSIF si sa condition booléenne associée est vraie) doit être exécuté.

Mots-clés :

IF, THEN, ELSIF, ELSE, END_IF

Exemple :

```
VAR
    i : UINT;
END_VAR

IF reset_flag THEN
    bool_out01 := false; (* après remise à zéro, les sorties bool_out01 undr *)
    bool_out02 := false; (* bool_out02 sont au préalable réinitialisés et *)
    i := 1; (* i forcé à 1 *)
END_IF;

if i>0 and i<=100 then
    bool_out01 := true; (* de 1 à 100, la sortie bool_out01 est forcé ... *)
    bool_out02 := false; (* et bool_out02 est réinitialisé *)
elsif i>100 and i<=200 then
    bool_out01 := false; (* de 101 à 200, la sortie bool_out01 est réinitialisée ... *)
    bool_out02 := false; (* et bool_out02 est forcé *)
else
    i := 1; (* durant le cycle 201, i est forcé à 1 et ... *)
    bool_out01 := false; (* les sorties bool_out01 et ... *)
    bool_out02 := false; (* bool_out02 sont réinitialisées ; tout recommence *)
end_if;

i := i + 1; (* i est incrémenté de 1 *)
```

6 Commandes de sélection

Instruction CASE

L'instruction CASE combine plusieurs instructions conditionnelles. Elle se compose d'une expression contenant une variable de type entier et une liste de groupes d'instruction. Chaque groupe se caractérise par une marque constituée d'un ou de plusieurs nombres entier (séparés par une virgule). La valeur des variables déterminent la marque et ainsi le groupe d'instruction qui doit être exécuté. Le mot-clé ELSE peut également être utilisé en option. Ensuite, si aucune valeur n'est valable, le groupe d'instructions qui suit le mot clé ELSE est exécuté.

Mots-clés :

CASE, OF, ELSE, END_CASE

Exemple :

```
CASE i OF (* la variable i doit être UINT ou UDINT *)
    1: b1 := TRUE;
    2, 3, 4: b2 := TRUE;
ELSE
    b1 := FALSE;
    b2 := FALSE;
END_CASE;
```

7 Instructions de répétition

Les instructions répétitives (itérations) exécutent plusieurs fois des instructions et des groupes d'instructions.

Il existe trois types d'instructions de répétition

- FOR
- WHILE
- REPEAT UNTIL

EXIT

EXIT permet d'arrêter une instruction de répétition avant que la condition finale de l'itération ne soit atteinte. Si l'instruction EXIT est utilisée dans une imbrication de fonctions de répétition, la boucle interne, contenant l'EXIT, est quittée

REMARQUE !

Lors d'instructions répétées, il faut veiller à ce qu'il n'y ait pas de bouclages infinis ou de boucles très longues. La durée de fonctionnement est surveillée.

Instruction FOR

L'instruction FOR est utilisée lorsque le nombre de répétitions est défini.

Mots-clés :

FOR, TO, BY, DO, END_FOR

Exemples :

```
FOR i := 10 TO 100 BY 10 DO (* la variable de comptage doit être de type UINT *)
    j := j + i; (* i fonctionne de 10 à 100 en pas de 10 (10, 20, 30, …, 100)  *)
END_FOR;


FOR i := 1 TO 5 DO (* sans indication de „BY x“ ... *)
    a := a + b; (* la variable de comptage est incrémentée de 1 à chaque passage (1, 2, 3, 4, 5) *)
    a := a * 3.0;
END_FOR;
```

Instruction WHILE

L'instruction WHILE déclenche l'exécution répétée d'un ensemble d'instructions jusqu'à ce que l'expression booléenne associée soit fausse (FALSE, fausse). Si l'expression booléenne est erronée depuis le début, l'ensemble des instructions ne sera pas exécuté.

Mots-clés :

WHILE, DO, END_WHILE

Exemple :

```
WHILE j < 100 DO
    j := j + 2;
END_WHILE;
```

7 Instructions de répétition

Instruction REPEAT

L'instruction REPEAT fait en sorte qu'un ensemble d'instructions (et au moins une fois) soit répété jusqu'au mot-clé UNTIL jusqu'à ce que la condition booléenne associée devienne vraie (TRUE).

Mots-clés :

REPEAT, UNTIL, END_REPEAT

Exemple :

```
REPEAT
    i := i - 2;
    a := a + 2.0;
UNTIL i = 0 END_REPEAT;
```

Le module ST prend en charge les fonctions suivantes :

- Conversion de type (conversion)
- Fonctions arithmétiques
- Fonctions numériques
- Fonctions séquence de bits (fonctions majuscule)
- Fonctions logiques
- Sélection (statistique)
- Comparaison
- Date et heure
- Fonctions spécifiques (ne font pas partie de la norme)

8.1 Conversion de type

Types de données admissibles

Argument: UINT, UDINT

Résultat: BOOL, UINT, UDINT, REAL

INT_TO_REAL

Convertit un nombre ENTIER (INTEGER) en un nombre REAL.

Exemple :

```
a := INT_TO_REAL(10); (* a := 10.0 *)
```

INT_TO_DINT

Convertit un nombre ENTIER (INTEGER) en un nombre ENTIER (INTEGER) DOUBLE.

Exemple :

```
a := INT_TO_DINT(10); (* a := 10 *)
```

INT_TO_BOOL

Convertit un nombre ENTIER (INTEGER) en un nombre BOOL.

Le résultat est FALSE si l'argument est 0. Dans tous les autres cas, le résultat est TRUE.

Exemples :

```
a := INT_TO_BOOL(0); (* a = FALSE *)
b := INT_TO_BOOL(1); (* b = TRUE *)
c := INT_TO_BOOL(8); (* c = TRUE *)
```

DINT_TO_REAL

Convertit un nombre ENTIER (INTEGER) DOUBLE en un nombre REAL.

Exemples :

```
a := DINT_TO_REAL(100000); (* a = 100000.0 *)
```

DINT_TO_INT

Convertit un nombre ENTIER (INTEGER) DOUBLE en un nombre ENTIER (INTEGER).

Seuls les deux octets inférieurs sont évalués (Bit 0 à Bit 15).

Exemples :

a := DINT_TO_INT(1); (* a = 1; 1 = 0000 0000 0000 0001 = 0x0001 *)

b := DINT_TO_INT(65535); (* b = 65535; 65535 = 1111 1111 1111 1111 = 0xFFFF *)

c := DINT_TO_INT(65536); (* c = 0; 65536 = 0001 0000 0000 0000 0000 = 0x10000 *)

d := DINT_TO_INT(65537); (* d = 1, 65537 = 0001 0000 0000 0000 0001 =0x10001 *)

DINT_TO_BOOL

Convertit un nombre ENTIER (INTEGER) DOUBLE en une valeur de type BOOL.

Le résultat est FALSE si l'argument est 0. Dans tous les autres cas, le résultat est TRUE.

Exemples :

a := DINT_TO_BOOL(0); (* a = FALSE: *)

b := DINT_TO_BOOL(1); (* b = TRUE *)

c := DINT_TO_BOOL(100000); (* c = TRUE *)

REAL_TO_INT

Convertit un nombre REAL (REEL) en un nombre INTEGER (ENTIER) en supprimant les décimales.

Exemples :

a := REAL_TO_INT(1.378); (* a = 1 *)

b := REAL_TO_INT(1.897); (* b = 2 *)

REAL_TO_DINT

Convertit un nombre REAL (REEL) en un nombre DOUBLE INTEGER (ENTIER DOUBLE) en supprimant les décimales.

Exemples :

a := REAL_TO_DINT(100000.378); (* a = 100000 *)

b := REAL_TO_DINT(100000.897); (* b = 100001 *)

8.2 Fonctions arithmétiques

Types de données admissibles

Argument: UINT, UDINT, REAL (pour Négation seulement REAL, pour Modulo seulement UINT ou UDINT)

Résultat: UINT, UDINT, REAL

Addition

L'addition est une fonction extensible. Par conséquent, la somme des arguments est restituée

Exemple :

OUT := IN1 + IN2 + ... INn;

Soustraction

Par conséquent, la seconde est déduite du premier argument.

Exemple :

OUT := IN1 - IN2;

Multiplication

La multiplication est une fonction extensible. Le résultat résulte de la multiplication des arguments.

Exemple :

OUT := IN1 * IN2 * ...INn;

Division

Le résultat retourné contient le quotient provenant des 2 arguments.

Le résultat de la division de deux nombres entiers (UINT ou UDINT) est un nombre dont on a coupé la partie décimale (exemple : 7/3 = 2).

Exemple :

OUT := IN1 / IN2;

Négation

Les nombres REAL peuvent être négatifs.

Exemple :

OUT := -IN;

Modulo

Les arguments de la fonction Modulo (%) doivent être des nombres entiers (UINT ou UDINT). Le résultat de la fonction modulo est égal au traitement de l'instruction suivante :

```
IF (IN2 = 0) THEN OUT := 0;
ELSE OUT := IN1 - (IN1 / IN2) * IN2;
END_IF
```

Le résultat de la division des nombres entiers (UINT ou UDINT) dans la branche ELSE de la commande de sélection est un nombre dont on a coupé la partie décimale.

Exemple :

```
IN1 := 17;
IN2 := 3;
OUT := IN1 % IN2; (* OUT := 2 *)
```

8 Fonctions

8.3 Fonctions numériques

Types de données admissibles

Argument: REAL

Résultat: REAL

ABS (IN)

Livre la valeur absolue de IN.

Exemple :

```
IN := -2.3;
OUT := ABS(IN); (* OUT = 2.3 *)
```

SQRT (IN)

Livre la racine carrée de IN.

Si IN est négatif, la valeur d'erreur 5.0E+37 est livrée.

Exemple :

```
IN := 4.0;
OUT := SQRT(IN); (* OUT = 2.0 *)
```

LN (IN)

Livre le logarithme naturel de IN.

Exemple :

```
IN := 2.718282; (* nombre e (arrondi) *)
OUT := LN(IN); (* OUT = 1.0 (arrondi) *)
```

LOG (IN)

Livre le logarithme naturel sur la base 10 de IN.

Exemple :

```
IN := 100.0;
OUT := LOG(IN); (* OUT = 2.0 *)
```

EXP (IN)

Livre la fonction exponentielle naturelle de IN.

Exemple :

```
IN := 2.0;
OUT := EXP(IN); (* OUT = 7.389056 *)
```

SIN (IN)

Livre le sinus de IN (radians).

Exemple :

```
IN := 1.5708; (* correspond à 90° *)
OUT := SIN(IN); (* OUT = 1.0 *)
```

COS (IN)

Livre le cosinus de IN (radians).

Exemple :

```
IN := 0.0; (* correspond à 0° *)
OUT := COS(IN); (* OUT = 1.0 *)
```

TAN (IN)

Livre la tangente de IN (radians).

Exemple :

```
IN := 0.7854; (* correspond à 45° *)
OUT := TAN(IN); (* OUT = 1.0 *)
```

ASIN (IN)

Livre l'arc sinus de IN (radians).

Exemple :

```
IN := 1.0;
OUT := ASIN(IN); (* OUT = 1.5708; entspricht 90° *)
```

ACOS (IN)

Livre l'arc sinus (radians) de IN.

Exemple :

```
IN := 1.0;
OUT := ASIN(IN); (* OUT = 0.0 ; correspond à 0° *)
```

ATAN (IN)

Livre l'arc tangente (radians) de IN.

Exemple :

```
IN := 1.0;
OUT := ATAN(IN); (* OUT = 0.7854; correspond à 45° *)
```

8.4 Fonctions séquence de bits

Types de données admissibles

Argument: UINT, UDINT

Résultat: UINT, UDINT

SHL (IN, n)

Déplace la séquence de bits de l'argument IN vers la gauche de n bits. Les positions vacantes à droite sont remplacées par des 0.

Exemple :

```
IN := 255; (* séquence de bits : 0000 0000 1111 1111 *)
OUT := SHL(IN, 4); (* OUT = 4080 ; séquence de bits: 0000 1111 1111 0000 *)
```

SHR (IN, n)

Déplace la séquence de bits de l'argument IN vers la droite de n bits. Les positions vacantes à gauche sont remplacées par des 0.

Exemple :

```
IN := 255; (* séquence de bits : 0000 0000 1111 1111 *)
OUT := SHR(IN, 4); (* OUT = 15 ; séquence de bits : 0000 0000 0000 1111 *)
```

ROL (IN, n)

Fait pivoter la séquence de bits de l'argument IN vers la gauche de n bits dans le cercle.

Exemple :

```
IN := 43690; (* séquence de bits : 1010 1010 1010 1010 *)
OUT := ROL(IN, 1); (* OUT = 21845 ; séquence de bits : 0101 0101 0101 0101 *)
```

ROR (IN, n)

Fait pivoter la séquence de bits de l'argument IN vers la droite de n bits dans le cercle.

Exemple :

```
IN := 21845; (* séquence de bits : 0101 0101 0101 0101 *)
OUT := ROR(IN, 1); (* OUT = 43690 ; séquence de bits : 1010 1010 1010 1010 *)
```

8.5 Fonctions logiques

Les variables booléennes sont combinées logiquement, tandis que les variables UINT et UDINT le sont au niveau des bits

Lorsqu'il y a plus de deux paramètres, les combinaisons se font toujours par paire de gauche vers la droite.

Types de données admissibles

Opérande : BOOL, UINT, UDINT

Résultat : BOOL, UINT, UDINT

AND

Opérateur ET

Exemples :

Opérateur logique :

```
x := TRUE;
y := FALSE;
z := x AND y; (* z = FALSE *)
```

Opérateur bit à bit :

```
a := 5;. . . . . 0101 = 5
b := 6;. . . . . .0110 = 6
     . . . . . ._____
c := a AND b; (* c = 4 *) . . . . . 0100 = 4
```

OR

Opérateur OU

Exemples :

Opérateur logique :

```
x := TRUE;
y := FALSE;
z := x OR y; (* z = TRUE *)
```

Opérateur bit à bit :

```
a := 5;. . . . . 0101 = 5
b := 6;. . . . . .0110 = 6
     . . . . . ._____
c := a OR b; (* c = 7 *) . . . . . .0111 = 7
```

XOR

Opérateur OU exclusif

Exemples :

Opérateur logique :

```
x := TRUE;
y := FALSE;
z := x XOR y; (* z = TRUE *)
```

Opérateur bit à bit :

```
a := 5;. . . . . 0101 = 5
b := 6;. . . . . .0110 = 6
     . . . . . ._____
c := a XOR b; (* c = 3 *). . . . . .0011 = 3
```

NOT

Inversion (former un complément)

Exemples :

```
IN := 1;
OUT := NOT IN; (* OUT = 0 *)
IN := 43690; (* IN = 1010 1010 1010 1010 *)
OUT := NOT IN; (* OUT = 21845; OUT = 0101 0101 0101 0101 *)
```

8.6 Sélection

Types de données admissibles

Argument: UINT, UDINT, REAL (pour SEL, BOOL supplémentaire pour le sélecteur)

Résultat: UINT, UDINT, REAL

MAX

Le résultat retourné est l'argument le plus élevé.

Exemple :

OUT := MAX(8, 9, 10, 11, 12); (* OUT := 12 *)

MIN

Le résultat retourné est l'argument le plus faible.

Exemple :

OUT := MIN(8, 9, 10, 11, 12); (* OUT := 8 *)

LIMIT

Vérifie si une valeur (IN) se trouve dans une plage définie (MIN, MAX). Lorsque la valeur se situe dans la plage, la valeur elle-même est retransmise. Lorsque la valeur se situe en-dessous de la plage, la limite inférieure est retransmise. Lorsque la valeur se situe au-dessus de la plage, la limite supérieure est retransmise.

Exemples :

OUT := LIMIT(IN, MIN, MAX);

OUT := LIMIT(15, 10, 20); (* OUT = 15 *)

OUT := LIMIT(5, 10, 20); (* OUT = 10 *)

OUT := LIMIT(25, 10, 20); (* OUT = 20 *)

SEL

Sélectionne l'une des deux valeurs (IN0, IN1) en fonction d'une variable booléenne (G). Le type de données des valeurs IN0 et IN1 doit être identique.

Exemples :

OUT := SEL(G, IN0, IN1);

OUT := SEL(G, 10, 20); (* G = FALSE: OUT = 10 *)

OUT := SEL(G, 10, 20); (* G = TRUE: OUT = 20 *)

8.7 Comparaison

Types de données admissibles

Argument pour LT, LE, GT, GE: UINT, UDINT, REAL, DT

Argument pour EQ et NE: BOOL, UINT, UDINT, REAL, DT

Résultat : BOOL

< (LT)

Compare les arguments pour voir si l'un est plus petit qu'un autre.

Exemples :

bOUT := IN1 < IN2; (* bOUT := TRUE, lorsque IN1 est inférieur à IN2 *)

bOUT := (IN1 < IN2) & (IN2 > IN3) & ... & (INn-1 > INn);

<= (LE)

Compare les arguments pour voir si l'un est inférieur ou égal à un autre.

Exemples :

bOUT := IN1 <= IN2; (* bOUT := TRUE, quand IN1 est inférieur ou égal à IN2 *)

bOUT := (IN1 <= IN2) & (IN2 <= IN3) & ... & (INn-1 <= INn);

> (GT)

Compare les arguments pour voir si l'un est supérieur à un autre.

Exemples :

bOUT := IN1 > IN2; (* bOUT := TRUE, lorsque IN1 est supérieur à IN2 *)

bOUT := (IN1 > IN2) & (IN2 > IN3) & ... & (INn-1 > INn);

>= (GE)

Compare les arguments pour voir si l'un est supérieur ou égal à un autre.

Exemple :

bOUT := IN1 >= IN2; (* bOUT := TRUE, quand IN1 est supérieur ou égal à IN2 *)

= (EQ)

Compare les arguments pour voir s'ils sont égaux.

Exemple :

bOUT := IN1 = IN2; (* bOUT := TRUE, quand IN1 est égal à IN2 *)

<> (NE)

Compare les arguments pour voir s'ils sont différents.

Exemple :

bOUT := IN1 <> IN2; (* bOUT := TRUE, quand IN1 est différent de IN2 *)

8.8 Date et heure

Il s'agit de fonctions spécifiques ne faisant pas partie de la norme CEI 61131-3.

Les fonctions peuvent être utilisées, par exemple, pour interroger la variable d'entrée rtc.cdt, qui indiquent le temps écoulé depuis la mise sous tension de l'appareil. (date et heure commencent à chaque mise sous tension à partir de 01.01.1970 00:00:00).

Types de données admissibles

Argument: DT (ou DATE_AND_TIME)

Résultat : UINT

GET_YEAR

Restitue l'année (avec le siècle).

Exemples :

iOUT := GET_YEAR(dt#2017-03-20-17:45:12); (* iOUT = 2017 *)

GET_MONTH

Restitue le mois.

Exemple :

iOUT := GET_MONTH(dt#2017-03-20-17:45:12); (* iOUT = 03 *)

GET_DAY

Restitue le jour.

Exemple :

iOUT := GET_DAY(dt#2017-03-20-17:45:12); (* iOUT = 20 *)

GET_HOUR

Restitue les heures.

Exemple :

iOUT := GET_HOUR(dt#2017-03-20-17:45:12); (* iOUT = 17 *)

GET_MINUTE

Restitue les minutes.

Exemple :

iOUT := GET_MINUTE(dt#2017-03-20-17:45:12); (* iOUT = 45 *)

GET_SECOND

Restitue les secondes.

Exemple :

iOUT := GET_SECOND(dt#2017-03-20-17:45:12); (* iOUT = 12 *)

iOUT := GET_SECOND(rtc_cdt); (* iOUT = 59, si rtc_cdt = 1970-01-01 00:00:59 *)

8.9 Autres fonctions

Il s'agit de fonctions spécifiques ne faisant pas partie de la norme CEI 61131-3.

IS_VALID

Vérifie la validité du nombre REAL. Le résultat est une valeur booléenne (valide = TRUE, invalide = FALSE).

Les valeurs ≥ 1.00E+37 (valeur d'erreur générale) sont considérées comme invalides, tout comme les valeurs „infini“ (INF) et „Aucun chiffre“ (NaN).

Exemples :

```
bOUT := IS_VALID(1.2); (* bOUT =  TRUE *)
```

SET_DEFAULT

Définit un nombre REAL ou BOOL à la valeur par défaut.

Valeur par défaut REAL: 5.0E+37 (valeur d'erreur mathématique)
Valeur par défaut BOOL: FALSE

Exemple :

```
SET_DEFAULT(bOUT); (* bOUT =  FALSE *)
SET_DEFAULT(rOUT); (* rOUT = 5.0E+37 *)
```

Instance

Un module fonctionnel - appelé bloc fonctionnel dans l'éditeur ST - est une unité d'organisation de programme qui fournit une ou plusieurs valeurs lorsqu'il est exécuté (voir également norme CEI 61131-3). Un module fonctionnel peut exister dans plusieurs instances (copies). Chaque instance possède un identificateur propre . Les instances d'un module fonctionnel sont indépendantes les unes des autres.

S'il y a un module fonctionnel s'appelant TON par ex. dans 4 instances, les différentes instances seront alors adressées avec TON01, TON02, TON03, TON04.

Le module ST comprend les blocs de fonction suivants :

Identificateur	Instances	Fonction
CTUD	4	Logiciel-Comptage/décomptage
TP	4	Compteurs d'impulsion
TON	4	Retard à l’enclenchement
TOF	4	Retard au déclenchement
R_TRIG	4	Détection de front montant (0 -> 1)
F_TRIG	4	Détection de front descendant (1 -> 0)
SR	4	Modules fonctionnels bistables (prioritaire)
RS	4	Modules fonctionnels bistables (à réinitialiser en priorité)
SET_CP	1	Prédéfinir les paramètres du régulateur

Paramètre

Les modules fonctionnels reçoivent un ou plusieurs paramètres (entrées). Les paramètres suivent la spécification de l'instance entre parenthèses. L'ordre des paramètres est fixe. L'appel d'un module fonctionnel, comme toute instruction, s'achève par un „;“.

Exemple :

TON01(TRUE,5,1);

Entrées/sorties

Les états des entrées et sorties de toutes les instances des modules fonctionnels restent toujours constants jusqu'au prochain appel de l'instance du module fonctionnel. Les sorties peuvent être consultées et utilisées dans n'importe quelle opération

Accès aux sorties

On accède aux sorties (éléments de structure) des modules fonctionnels comme habituellement dans le langage de programmation de type Pascal.

Exemple :

Interrogation de la sortie ET du module fonctionnel TON via l'instance 01:

OUT := TON01.ET;

9 Modules fonctionnels

9.1 Logiciel-Comptage/décomptage

Le compteur met à disposition les fonctions suivantes.

- Comptage
- Décomptage
- Réinitialisation du compteur à 0
- Réglage d'une valeur de comptage supérieure
- Requête pour dépasser la valeur de comptage supérieure
- Requête valeur du compteur 0
- Interroger valeur de comptage

Le compteur compte les fronts positifs aux entrées CU (comptage) et CD (décomptage).

Appel

CTUD<Instance> (CU, CD, R, LD, PV);

Entrées

Les paramètres R, LD, CU et CD sont triés par ordre décroissant dans le tableau en fonction de leur priorité.

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
R	BOOL	TRUE = régler le compteur à 0
LD	BOOL	TRUE = régler la valeur de comptage supérieure
CU	BOOL	Front positif = comptage
CD	BOOL	Front positif = décomptage
PV	UINT	Valeur de comptage supérieure

Sorties

Paramètre	Type de données	Requête ; (* Description *)
CV	UINT	OUT := CTUD<Instance>.CV; (* OUT = relevé du compteur *)
QU	BOOL	OUT := CTUD<Instance>.QU; (* OUT = TRUE, lorsque le compteur est > à la valeur de comptage supérieure *)
QD	BOOL	OUT := CTUD<Instance>.QD; (* OUT = TRUE, lorsque le compteur <= 0 *)

Remarque

Pour forcer la valeur de comptage supérieure avec LD, R ne doit pas être TRUE.

Les valeurs d'entrée/de sortie ne sont pas conservées via la mise hors tension

Exemples

CTUD01 (IN, FALSE, FALSE, FALSE, 100); (* comptage des fronts positifs de IN *)

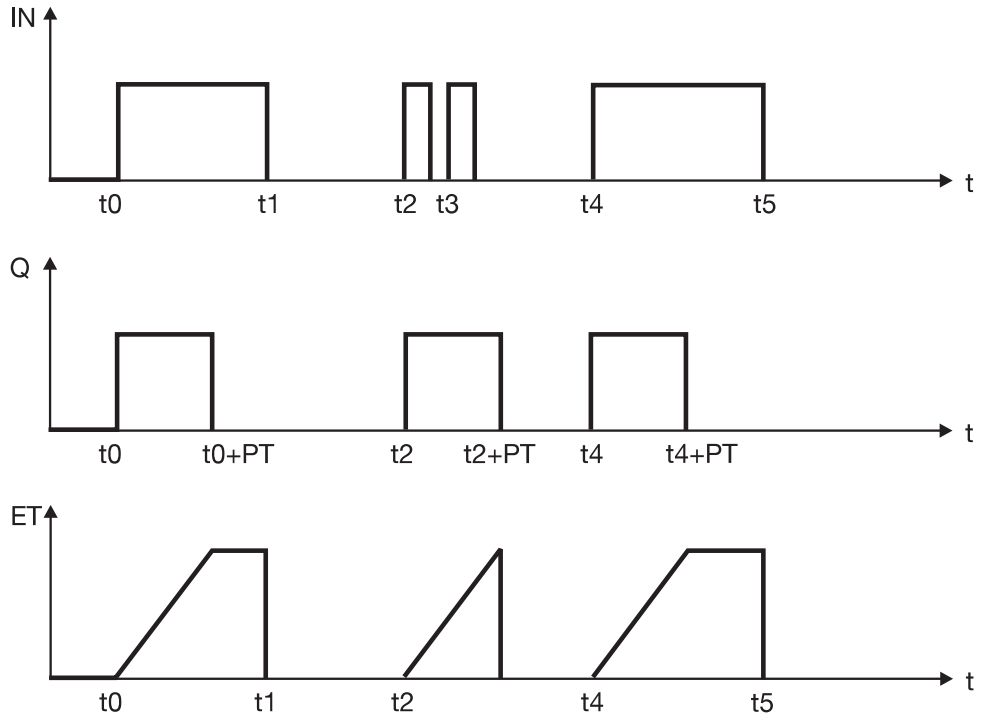
CTUD01 (IN, FALSE, FALSE, FALSE, 100); (* décomptage des fronts positifs de IN *)

CTUD01 (IN, FALSE, TRUE, FALSE, 100); (* forcer le compteur à 0 *)

CTUD01 (IN, FALSE, FALSE, TRUE, 100); (* forcer la valeur de comptage supérieure : 100 *)

9.2 Compteurs d'impulsion

Le mode de fonctionnement du compteur d'impulsion illustre le diagramme suivant :



Appel

TP<Instance> (IN, PT, TimeBase);

Entrées

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
IN	BOOL	Le front positif sur IN place Q pour la durée PT sur TRUE
PT	UINT	Période (unité: TimeBase), pour laquelle Q = TRUE devient pour IN = TRUE
TimeBase	UINT	Unité de PT : 1 = ms 2 = s 3 = min

Sorties

Paramètre	Type de données	Requête ; (* Description *)
Q	BOOL	OUT := TP<Instance>.Q ; (* OUT = TRUE pour la période PT, si IN := 0 -> 1 *)
ET	UINT	OUT := TP<Instance>.ET; (* OUT = temps écoulé depuis IN = TRUE jusqu'à Q = FALSE (temps écoulé depuis le début de la phase d'impulsion active) *)

Remarque

Le paramètre TimeBase est une extension de la norme CEI 61131-3. Cette extension n'est pas une restriction de la norme. Le temps imparti est selon la norme „fonction de l'implémentation“.

Les valeurs d'entrée/de sortie ne sont pas conservées via la mise hors tension

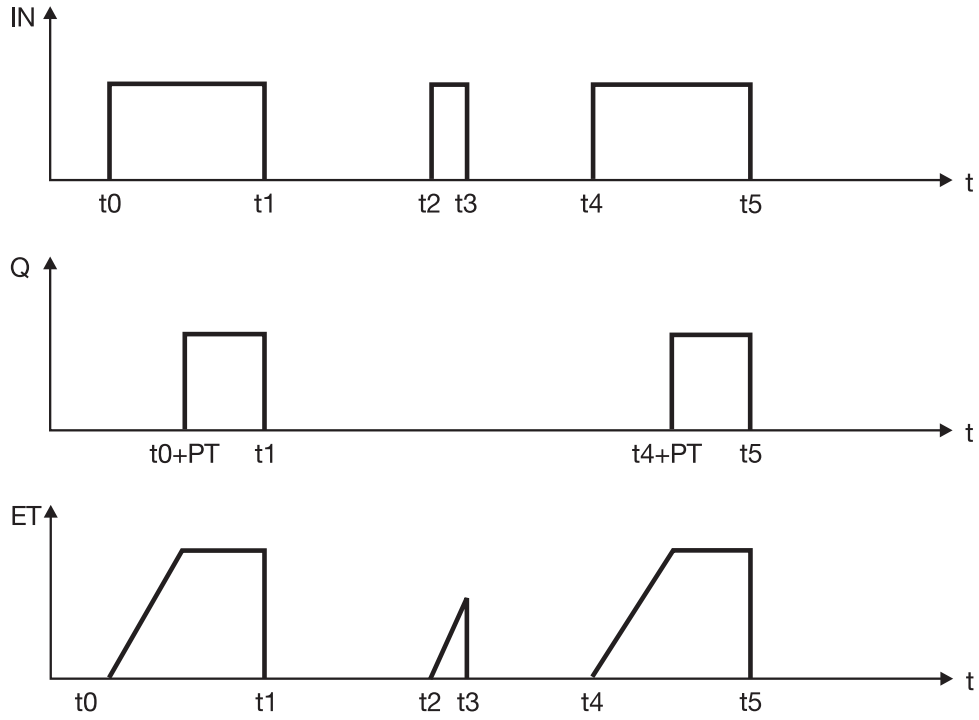
La résolution temporelle de la longueur d'impulsion est de 150 ms (temps de cycle de l'appareil).

Exemple

```
TP01(IN, 5, 3);
(* le front positif à IN force TP01.Q pour env. 5 minutes sur TRUE (respecter la résolution temporelle).
TP01.ET renvoie le temps écoulé en minutes depuis le début de la phase d'impulsion active. *)
```

9.3 Retard à l'enclenchement

Le mode de fonctionnement de l'enclenchement retardé illustre le diagramme suivant :



Appel

TON<Instance> (IN, PT, TimeBase) ;

Entrées

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
IN	BOOL	IN = TRUE force Q = TRUE, **après** expiration de la période PT
PT	UINT	Temporisation (unité: TimeBase)
TimeBase	UINT	Unité de PT : 1 = ms 2 = s 3 = min

Sorties

Paramètre	Type de données	Requête ; (* Description *)
Q	BOOL	OUT := TON<Instance>.Q; (* OUT = TRUE, lorsque IN = TRUE et PT expiré *)
ET	UINT	OUT := TON<Instance>.ET; (* OUT = temps écoulé depuis IN = TRUE jusqu'à Q = TRUE ou jusqu'à IN = FALSE *)

Remarque

Le paramètre TimeBase est une extension de la norme CEI 61131-3. Cette extension n'est pas une restriction de la norme. Le temps imparti est selon la norme „fonction de l'implémentation“.

Les valeurs d'entrée/de sortie ne sont pas conservées via la mise hors tension

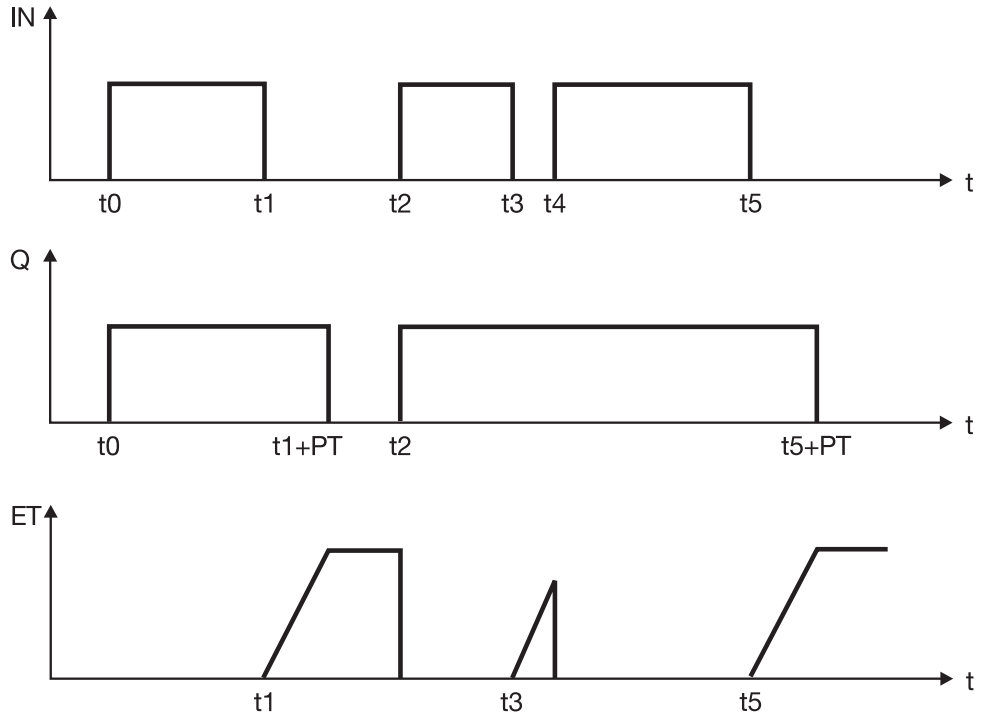
La résolution temporelle de la temporisation est de 150 ms (temps de cycle de l'appareil).

Exemple

```
TON01(IN, 6, 2);
(* IN = TRUE force TON01.Q surTRUE après 6 secondes.
TON01.Q reste VRAI tant que IN = TRUE.
TON01.ET renvoie le temps écoulé en secondes de IN = TRUE à TON01.Q = TRUE (6 secondes max.)
ou à IN = FALSE. *)
```

9.4 Retard au déclenchement

Le mode de fonctionnement du déclenchement retardé illustre le diagramme suivant :



Appel

TOF<Instance> (IN, PT, TimeBase);

Entrées

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
IN	BOOL	IN = TRUE force Q = TRUE IN = FALSE force Q au temps PT retardé sur FALSE
PT	UINT	Temporisation (unité: TimeBase)
TimeBase	UINT	Unité de PT : 1 = ms 2 = s 3 = min

Sorties

Paramètre	Type de données	Requête ; (* Description *)
Q	BOOL	OUT := TOF<Instance>.Q; (* OUT = TRUE, lorsque IN = TRUE ou lorsque IN = FALSE et PT n'est pas encore écoulé *)
ET	UINT	OUT := TOF<Instance>.ET; (* OUT = temps écoulé depuis IN = FALSE jusqu'à Q = FALSE ou jusqu'à IN = TRUE *)

Remarque

Le paramètre TimeBase est une extension de la norme CEI 61131-3. Cette extension n'est pas une restriction de la norme. Le temps imparti est selon la norme „fonction de l'implémentation“.

Les valeurs d'entrée/de sortie ne sont pas conservées via la mise hors tension

La résolution temporelle de la temporisation est de 150 ms (temps de cycle de l'appareil).

Exemple

```
TOF01(IN, 450, 1);
(* IN = TRUE force TOF01.Q à TRUE.
Après IN = FALSE reste TOF01.Q TRUE, jusqu'à ce que le temps PT = 450  ms soit écoulé.
TOF01.ET donne le temps passé en millisecondes de IN = FALSE à TOF01.Q = FALSE (max. 450 ms)
ou à IN = TRUE. *)
```

9.5 Détection de front montant

Ce module fonctionnel détecte un front montant (passage 0 -> 1) .

Appel

R_TRIG<Instance> (CLK);

Entrée

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
CLK	BOOL	Par un front montant (0 -> 1) à CLK ; Q = TRUE

Sortie

Paramètre	Type de données	Requête
Q	BOOL	OUT := R_TRIG<Instance>.Q;

Remarque

Le mode de fonctionnement du module fonctionnel correspond au code de programme suivant :

VAR CLK : BOOL; END_VAR

VAR Q : BOOL; END_VAR

VAR M : BOOL := FALSE; END_VAR

Q := CLK AND NOT M;

M := CLK;

La sortie Q conserve sa valeur booléenne d'un appel à l'autre. Elle suit la transition du signal d'entrée (CLK) de „0“ à "1" et revient à l'appel suivant à „0“.

Exemple

```
IF reset_flag THEN
bool_out01 := FALSE;
END_IF;

R_TRIG01 (bool_in01); (* détection d'un front montant à l'entrée bool_in01 *)
bool_out01 := R_TRIG01.Q; (* impulsion brève à la sortie bool_out01 pour front montant à l'entrée *)
```

9 Modules fonctionnels

9.6 Détection de front descendant

Ce module fonctionnel détecte un front descendant (passage 1 -> 0) .

Appel

F_TRIG<Instance> (CLK);

Entrée

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
CLK	BOOL	Par un front descendant (1 -> 0) à CLK ; Q = TRUE

Sortie

Paramètre	Type de données	Requête
Q	BOOL	OUT := F_TRIG<Instance>.Q;

Remarque

Le mode de fonctionnement du module fonctionnel correspond au code de programme suivant :

VAR CLK : BOOL; END_VAR

VAR Q : BOOL; END_VAR

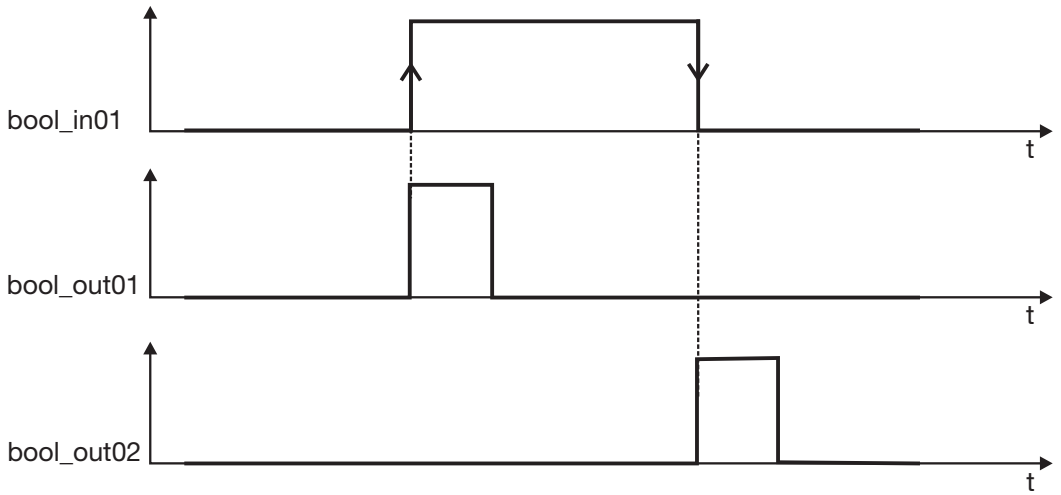
VAR M : BOOL := TRUE; END_VAR

Q := NOT CLK AND NOT M;

M := NOT CLK;

La sortie Q conserve sa valeur booléenne d'un appel à l'autre. Elle suit la transition du signal d'entrée (CLK) de „1“ à „0“ et revient à l'appel suivant à „0“.

Exemple avec R_TRIG et F_TRIG



IF reset_flag THEN
bool_out01 := FALSE;
bool_out02 := FALSE;
END IF;

R_TRIG01 (bool_in01); (* détection d'un front montant à l'entrée bool_in01 *)
bool_out01 := R_TRIG01.Q; (* impulsion brève à la sortie bool_out01 pour front montant à l'entrée *)

```
F_TRIG01 (bool_in01); (* détection d'un front descendant à l'entrée bool_in01 *)
bool_out02 := F_TRIG01.Q; (* impulsion brève à la sortie bool_out02 pour front descendant à l'entrée *)
```

9 Modules fonctionnels

9.7 Fonction bistable SR

Fonction bistable (prioritaire) avec auto-maintien

Appel

SR<Instance> (S1, R) ;

Entrées

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
S1	BOOL	S1 = TRUE force Q1 = TRUE
R	BOOL	R = TRUE force Q1 = FALSE, quand S1 = FALSE

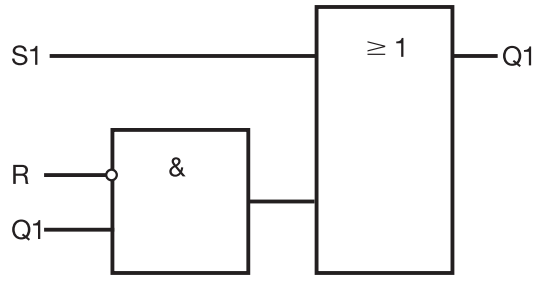
Sorties

Paramètre	Type de données	Requête
Q1	BOOL	OUT := SR<Instance>.Q1;

Remarque

Sortie Q1 maintenue après activation par S1 tant que TRUE, jusqu'à ce qu'elle soit réinitialisée avec R. Lorsque S1 et R sont définis simultanément (TRUE), la sortie Q1 est également activée (TRUE).

Fonction logique



Exemple

SR1 (bool_in01, bool_in02); (* activer avec entrée bool_in01, réinitialiser avec entrée bool_in02 *)
bool_out01 := SR1.Q1; (* l'état est généré via la sortie bool_out01. *)

9.8 Fonction bistable RS

Fonction bistable (à réinitialiser en priorité) avec auto-maintien

Appel

RS<Instance> (S, R1);

Entrées

Paramètre	Type de données	Description
<Instance>		01 à 04 (indiquer une instance à deux chiffres)
R1	BOOL	R1 = TRUE active Q1 = FALSE
S	BOOL	S = TRUE force Q1 = TRUE, quand R1 = FALSE

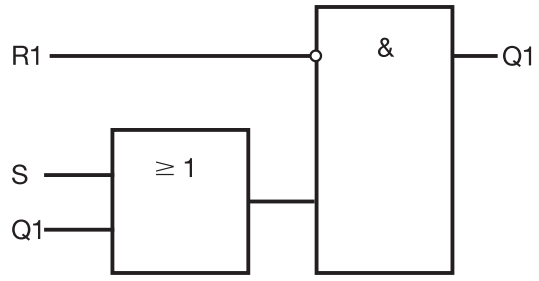
Sorties

Paramètre	Type de données	Requête
Q1	BOOL	OUT := RS<Instance>.Q1;

Remarque

Sortie Q1 maintenue après réinitialisation par R1 tant que FALSE, jusqu'à ce qu'elle soit réglée avec S. Lorsque R1 et S sont définis simultanément (TRUE), la sortie Q1 est réinitialisée (FALSE).

Fonction logique



Exemple

RS1 (bool_in01, bool_in02); (* réinitialiser avec l'entrée bool_in02, activer avec entrée bool_in01 *)
bool_out02 := RS1.Q1; (* l'état est généré via la sortie bool_out02. *)

9 Modules fonctionnels

9.9 Définir les paramètres du régulateur

Ce module fonctionnel permet de modifier les paramètres du régulateur du **second** jeu de paramètres (fonction spécifique, ne fait pas partie de la norme DIN IEC 61131-3).

Appel

Set_CP (Set, Key, Value);

Entrées

Paramètre	Type de données	Description
Controller	UINT	Régulateur n° 0 = régulateur 1 ; 1 = régulateur 2 ; ... Set = 0, lorsque l'appareil ne dispose que d'un régulateur (canal).
Paramètre	UINT	Paramètre : 0 = Xp1 (Bande proportionnelle 1) 1 = Xp2 (Bande proportionnelle 2) 2 = Tv1 (Temps de dérivée 1) 3 = Tv2 (Temps de dérivée 2) 4 = Tn1 (Temps d'intégrale 1) 5 = Tn2 (Temps d'intégrale 2) 6 = Cy1 (Durée du cycle de commutation 1) 7 = Cy2 (Durée du cycle de commutation 2) 8 = Y1 (Limitation max. du taux de modulation) 9 = Y2 (Limitation min. du taux de modulation)
Value	REAL	Valeur du paramètre concerné

Sorties

Ce module fonctionnel n'a pas de sortie.

Remarque

Les valeurs des paramètres sont vérifiées avant d'être transférées vers le régulateur. Si la valeur est en dehors de la plage de valeurs autorisée, le transfert est rejeté et une valeur d'erreur est renvoyée dans la variable ext_error (peut être évaluée dans le code ST). Si la valeur est valide, la nouvelle valeur sera utilisée à partir de ce moment.

Les valeurs modifiées ne sont pas sauvegardées dans la configuration du régulateur. Après un reset (mise sous tension) les valeurs sauvegardées seront à nouveau utilisées comme paramètres du régulateur.

Pour modifier plusieurs paramètres, le module fonction doit être appelé plusieurs fois (voir exemple: deux appels).

Exemple

```
VAR
    controller : UINT; (* régulateur n° *)
    para1 : UINT; (* Paramètre 1 *)
    value1 : REAL; (* valeur du paramètre 1 *)
    para2 : UINT; (* Paramètre 2 *)
    value2 : REAL; (* valeur du paramètre 2 *)
END_VAR
SET_CP (controller, para1, value1);
SET_CP (controller, para2, value2);
```

9 Modules fonctionnels

JUMO GmbH & Co. KG
Adresse :
Moritz-Juchheim-Straße 1
36039 Fulda, Allemagne
Adresse de livraison :
Mackenrodtstraße 14
36039 Fulda, Allemagne
Adresse postale :
36035 Fulda, Allemagne
Téléphone : +49 661 6003-0
Télécopieur : +49 661 6003-607
E-Mail: mail@jumo.net
Internet: www.jumo.net

JUMO-REGULATION SAS
7 rue des Drapiers
B.P. 45200
57075 Metz Cedex 3, France
Téléphone : +33 3 87 37 53 00
Télécopieur : +33 3 87 37 89 00
E-Mail: info.fr@jumo.net
Internet: www.jumo.fr
Service de soutien à la vente :
0892 700 733 (0,337 Euro/min)

JUMO Automation
S.P.R.L. / P.G.M.B.H. / B.V.B.A.
Industriestraße 18
4700 Eupen, Belgique
Téléphone : +32 87 59 53 00
Télécopieur : +32 87 74 02 03
E-Mail: info@jumo.be
Internet: www.jumo.be

JUMO Mess- und Regeltechnik AG
Laubisrütistrasse 70
8712 Stäfa, Suisse
Téléphone : +41 44 928 24 44
Télécopieur : +41 44 928 24 48
E-Mail: info@jumo.ch
Internet: www.jumo.ch