

# ***Boîtes Fonctionnelles IEC 61131-3***

Généralités

Editeurs terminaux

Développement d'applications

Outils de tests et d'animation

Documentation

**Eléments de langage**

Modules annexes

**A.2**



## Sommaire détaillé

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Types de données simples	6
1.2	Types de données Tableaux	6
<b>2</b>	<b>Opérations sur variables booléennes</b>	<b>7</b>
2.1	R_TRIG	8
2.2	F_TRIG	9
2.3	RS	10
2.4	SR	11
<b>3</b>	<b>Opérations sur les bits d'un mot</b>	<b>12</b>
3.1	AND	13
3.2	OR	14
3.3	XOR	15
3.4	NOT	16
<b>4</b>	<b>Décalages et rotations</b>	<b>17</b>
4.1	SHL	18
4.2	SHR	19
4.3	ROL	20
4.4	ROR	21
<b>5</b>	<b>Sélection sur variables</b>	<b>22</b>
5.1	SEL	23
5.2	MUX	24
5.3	MAX	25
5.4	MIN	26
5.5	LIMIT	27
<b>6</b>	<b>Opérations de comparaison</b>	<b>28</b>
6.1	EQ	29
6.2	GE	30
6.3	GT	31
6.4	LE	32
6.5	LT	33
6.6	NE	34
<b>7</b>	<b>Opérations arithmétiques</b>	<b>35</b>
7.1	ADD	36
7.2	MUL	37
7.3	SUB	38
7.4	DIV	39
7.5	MOD	40
7.6	EXPT	41
7.7	CONCAT	42

<b>8</b>	<b>Fonctions numériques</b>	<b>43</b>
8.1	ABS	44
8.2	SQRT	45
8.3	LN	46
8.4	LOG	47
8.5	EXP	48
8.6	SIN	49
8.7	COS	50
8.8	TAN	51
8.9	ASIN	52
8.10	ACOS	53
8.11	ATAN	54
<b>9</b>	<b>Comptage</b>	<b>55</b>
9.1	CTU	56
9.2	CTD	57
9.3	CTUD	58
<b>10</b>	<b>Temporisation</b>	<b>59</b>
10.1	TP	60
10.2	TON	61
10.3	TOF	62
<b>11</b>	<b>Opérations sur chaînes de caractères</b>	<b>63</b>
11.1	LEN	64
11.2	LEFT	65
11.3	RIGHT	66
11.4	MID	67
11.5	CONCAT	68
11.6	INSERT	69
11.7	DELETE	70
11.8	REPLACE	71
11.9	FIND	72
<b>12</b>	<b>Opérations de conversion</b>	<b>73</b>
12.1	BOOL_TO_SINT	75
12.2	BOOL_TO_INT	76
12.3	BOOL_TO_DINT	77
12.4	BOOL_TO_BYTE	78
12.5	BOOL_TO_WORD	79
12.6	BOOL_TO_DWORD	80
12.7	BOOL_TO_REAL	81
12.8	SINT_TO_BOOL	82
12.9	SINT_TO_INT	83
12.10	SINT_TO_DINT	84
12.11	SINT_TO_REAL	85
12.12	INT_TO_BOOL	86
12.13	INT_TO_SINT	87
12.14	INT_TO_DINT	88



12.15	INT_TO_REAL	89
12.16	DINT_TO_BOOL	90
12.17	DINT_TO_SINT	91
12.18	DINT_TO_INT	92
12.19	DINT_TO_REAL	93
12.20	REAL_TO_BOOL	94
12.21	REAL_TO_SINT	95
12.22	REAL_TO_INT	96
12.23	REAL_TO_DINT	97
12.24	REAL_TO_BYTE	98
12.25	REAL_TO_WORD	99
12.26	REAL_TO_DWORD	100
12.27	TRUNC_REAL_TO_SINT	101
12.28	TRUNC_REAL_TO_INT	102
12.29	TRUNC_REAL_TO_DINT	103
12.30	BYTE_BCD_TO_SINT	104
12.31	SINT_TO_BCD_BYTE	105
12.32	WORD_BCD_TO_INT	106
12.33	INT_TO_BCD_WORD	107
12.34	DWORD_BCD_TO_DINT	108
12.35	DINT_TO_BCD_DWORD	109
12.36	BYTE_TO_BOOL	110
12.37	BYTE_TO_WORD	111
12.38	BYTE_TO_DWORD	112
12.39	WORD_TO_BOOL	113
12.40	WORD_TO_BYTE	114
12.41	WORD_TO_DWORD	115
12.42	DWORD_TO_BOOL	116
12.43	DWORD_TO_BYTE	117
12.44	DWORD_TO_WORD	118
12.45	BYTE_TO_SINT	119
12.46	BYTE_TO_INT	120
12.47	BYTE_TO_DINT	121
12.48	BYTE_TO_REAL	122
12.49	BYTE_TO_STRING	123
12.50	WORD_TO_SINT	124
12.51	WORD_TO_INT	125
12.52	WORD_TO_DINT	126
12.53	WORD_TO_REAL	127
12.54	WORD_TO_STRING	128
12.55	DWORD_TO_SINT	129
12.56	DWORD_TO_INT	130
12.57	DWORD_TO_DINT	131
12.58	DWORD_TO_REAL	132
12.59	DWORD_TO_STRING	133
12.60	SINT_TO_BYTE	134
12.61	SINT_TO_WORD	135
12.62	SINT_TO_DWORD	136
12.63	SINT_TO_STRING	137



12.64	INT_TO_BYTE	138
12.65	INT_TO_WORD	139
12.66	INT_TO_DWORD	140
12.67	INT_TO_STRING	141
12.68	DINT_TO_BYTE	142
12.69	DINT_TO_WORD	143
12.70	DINT_TO_DWORD	144
12.71	DINT_TO_STRING	145
12.72	REAL_TO_STRING	146
12.73	STRING_TO_BYTE	147
12.74	STRING_TO_WORD	148
12.75	STRING_TO_DWORD	149
12.76	STRING_TO_SINT	150
12.77	STRING_TO_INT	151
12.78	STRING_TO_DINT	152
12.79	STRING_TO_REAL	153
12.80	DT_TO_DATE	154
12.81	DT_TO_TOD	155



## 1 Introduction

Ce document détaille les boîtes fonctionnelles utilisables dans les éditeurs de l'outil.

### 1.1 Types de données simples

Chaque variable d'entrée et de sorties d'une boîte fonctionnelle correspond à l'un des types de données élémentaires disponible dans les éditeurs.

ANY

ANY_NUM	
REAL	Nombre réel sur 32 bits
ANY_INT	
SINT	Entier signé sur 8 bits
INT	Entier signé sur 16 bits
DINT	Entier signé sur 32 bits
ANY_BIT	
BOOL	Booléen
BYTE	Mot de 8 bits (non signé)
WORD	Mot de 16 bits (non signé)
DWORD	Mot de 32 bits (non signé)
CHAR	Caractère ASCII sur 8 bits
ANY_TIME	
TIME	Durée ou temps
DATE_AND_TIME	Date et heure
DATE	Date
TIME_OF_DAY	Heure du jour

### 1.2 Types de données Tableaux

A partir de ces types simples, les éditeurs permettent de manipuler des types complexes appelés tableaux de :

REAL	Tableau de réel sur 32 bits
SINT	Tableau d'entier signé sur 8 bits
INT	Tableau d'entier signé sur 16 bits
DINT	Tableau d'entier signé sur 32 bits
BOOL	Tableau de booléen
STRING	Chaîne de caractères
BYTE	Tableau de mots de 8 bits
WORD	Tableau de mots de 16 bits
DWORD	Tableau de mots de 32 bits



## 2 Opérations sur variables booléennes

FB contenues dans cette famille :

opérateur	Libellé
<b>R_TRIG</b>	Front montant
<b>F_TRIG</b>	Front descendant
<b>RS</b>	Bascule à mise à 0 prioritaire
<b>SR</b>	Bascule à mise à 1 prioritaire



## 2.1 R\_TRIG

### Front montant

#### Paramètres

entrées			
CLK	BOOL	Signal à détecter	
sorties			
Q	BOOL	Valeur du front montant	
locales			
i_rt	INT	Mémorisation de la valeur de CLK	

#### Description

Cette opération détecte le passage de 0 à 1 de l'entrée **CLK**.  
A chaque cycle programme, l'état de **CLK** est comparé avec l'état de **i\_rt** qui contient l'état précédent de **CLK**.  
Le résultat de la sortie **Q** est vrai le cycle pendant lequel **CLK** vaut 1 et **i\_rt** vaut 0.

	t <sub>0</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>
CLK	0	0	1	1	1	0	0	0
i_rt	0	0	0	1	1	1	0	0
Q	0	0	1	0	0	0	0	0

#### Equivalence

```
Q := CLK AND NOT i_rt;  
i_rt := CLK;
```

#### Exemple ST

```
A := R_TRIG(B, temporaire);
```



2.2 F\_TRIG

Front descendant d'un booléen

Paramètres

entrées			
CLK	BOOL	Signal à détecter	
sorties			
Q	BOOL	Valeur du front descendant	
locales			
i_ft	INT	Mémorisation de la valeur de CLK	

Description

Cette opération détecte le passage de 1 à 0 de l'entrée **CLK**.  
A chaque cycle programme, l'état de **CLK** est comparé avec l'état de **i\_ft** qui contient l'état précédent de **CLK**.  
Le résultat de la sortie **Q** est vrai le cycle pendant lequel **CLK** vaut 0 et **i\_ft** vaut 1.

	t <sub>0</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>
CLK	0	0	1	1	1	0	0	0
i_ft	0	0	0	1	1	1	0	0
Q	0	0	0	0	0	1	0	0

Equivalence

Q := NOT CLK AND i\_ft;  
i\_ft := CLK;

Exemple ST

A := F\_TRIG(B, temporaire);



## 2.3 RS

### Bascule à mise à 0 prioritaire

#### Paramètres

entrées			
S1	BOOL	Entrée SET de la bascule	
R	BOOL	Entrée RESET de la bascule	
sorties			
Q1	BOOL	Résultat de la bascule	
locales			
i rs	INT	Mémorisation de l'état précédent	

#### Description

Cette opération représente le fonctionnement d'une bascule SR (mise à 0 prioritaire).

- Si le signal d'entrée sur **S1** est 1 et que le signal d'entrée sur **R** est 0 la sortie **Q1** est positionnée à 1.
- Si le signal d'entrée sur **S1** est 0 et que le signal d'entrée sur **R** est 1, la sortie **Q1** est positionnée à 0.
- Si le signal d'entrée sur **S1** est 0 et que le signal d'entrée sur **R** est 0, la sortie **Q1** reste dans le dernier état calculé.
- Si le signal d'entrée sur **S1** est 1 et que le signal d'entrée sur **R** est 1, la sortie **Q1** est positionnée à 0 (le reset est prioritaire).

	t <sub>0</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>
<b>S1</b>	0	1	0	0	0	1	1	1
<b>R</b>	0	0	0	1	0	0	1	0
<b>i_sr</b>	0	1	1	0	0	1	0	1
<b>Q1</b>	0	1	1	0	0	1	0	1

#### Exemple ST

```
A := RS(B, C, temporaire);
```

## 2.4 SR

### Bascule à mise à 1 prioritaire

#### Paramètres

entrées		
S	BOOL	Entrée SET de la bascule
R1	BOOL	Entrée RESET de la bascule
sorties		
Q1	BOOL	Résultat de la bascule
locales		
i_sr	INT	Mémorisation de l'état précédent

#### Description

Cette opération représente le fonctionnement d'une bascule RS (mise à 1 prioritaire).

- Si le signal d'entrée sur S est 1 et que le signal d'entrée sur R1 est 0 la sortie Q1 est positionnée à 1.
- Si le signal d'entrée sur S est 0 et que le signal d'entrée sur R1 est 1, la sortie Q1 est positionnée à 0.
- Si le signal d'entrée sur S est 0 et que le signal d'entrée sur R1 est 0, la sortie Q1 reste dans le dernier état calculé.
- Si le signal d'entrée sur S est 1 et que le signal d'entrée sur R1 est 1, la sortie Q1 est positionnée à 1 (le set est prioritaire).

	t <sub>0</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>
<b>S</b>	0	1	0	0	0	1	<b>1</b>	1
<b>R1</b>	0	0	0	1	0	0	<b>1</b>	0
<b>i_rs</b>	0	1	1	0	0	1	<b>0</b>	1
<b>Q1</b>	0	1	1	0	0	1	<b>1</b>	1

#### Exemple ST

```
A := SR(B, C, temporaire);
```



### 3 Opérations sur les bits d'un mot

Les opérations booléennes peuvent être réalisées sur les bits des mots suivants :

- mot de 8 bits (BYTE),
- mot de 16 bits (WORD),
- mot de 32 bits (DWORD).

FB contenues dans cette famille :

opérateur	Libellé
<b>AND</b>	ET logique sur bits
<b>OR</b>	OU logique sur bits
<b>XOR</b>	OU exclusif sur bits
<b>NOT</b>	Inversion de bits



## 3.1

**AND***ET logique sur bit***Paramètres**

entrées

IN1	ANY_BIT	Première entrée de la fonction
IN2	ANY_BIT	Deuxième entrée de la fonction

sorties

OUT	ANY_BIT	Résultat du ET logique sur mot
-----	---------	--------------------------------

**Description**

Cette fonction réalise le ET logique sur les bits de même rang des deux variables d'entrées **IN1** et **IN2** et range le résultat dans la sortie **OUT**.

**Exemple**

value of IN1:	2#0000_0001_0011_0111
value of IN2:	2#0011_0011_0011_0011
OUT result:	2#0000_0001_0011_0011

**BF typées**

<b>AND_BYTE</b>	ET logique sur les bits de 2 mots de 8 bits
<b>AND_WORD</b>	ET logique sur les bits de 2 mots de 16 bits
<b>AND_DWORD</b>	ET logique sur les bits de 2 mots de 32 bits

**Exemple ST**

```
A := AND_WORD(B, C) ;
```



### 3.2 OR

#### OU logique sur bits

---

##### Paramètres

entrées			
IN1	ANY_BIT	Première entrée de la fonction	
IN2	ANY_BIT	Deuxième entrée de la fonction	
sorties			
OUT	ANY_BIT	Résultat du OU logique sur mot	

---

##### Description

Cette fonction réalise le OU logique sur les bits de même rang des deux variables d'entrées **IN1** et **IN2** et range le résultat dans la sortie **OUT**.

---

##### Exemple

value of IN1:	2#0000_0001_0011_0111
value of IN2:	2#0011_0011_0011_0011
OUT result:	2#0011_0011_0011_0111

---

##### BF typées

<b>OR_BYTE</b>	OU logique sur les bits de 2 mots de 8 bits
<b>OR_WORD</b>	OU logique sur les bits de 2 mots de 16 bits
<b>OR_DWORD</b>	OU logique sur les bits de 2 mots de 32 bits

---

##### Exemple ST

A := OR\_WORD(B, C) ;

---

## 3.3

**XOR***OU exclusif sur bits***Paramètres**

entrées			
IN1	ANY_BIT	Première entrée de la fonction	
IN2	ANY_BIT	Deuxième entrée de la fonction	
sorties			
OUT	ANY_BIT	Résultat du OU Exclusif sur mot	

**Description**

Cette fonction réalise le OU Exclusif sur les bits de même rang des deux variables d'entrées **IN1** et **IN2** et range le résultat dans la sortie **OUT**.

**Exemple**

```
value of IN1:  2#0000_0001_0011_0111
value of IN2:  2#0011_0011_0011_0011
OUT result:    2#0011_0010_0000_0100
```

**BF typées**

<b>XOR_BYTE</b>	OU Exclusif sur les bits de 2 mots de 8 bits
<b>XOR_WORD</b>	OU Exclusif sur les bits de 2 mots de 16 bits
<b>XOR_DWORD</b>	OU Exclusif sur les bits de 2 mots de 32 bits

**Exemple ST**

```
A := XOR_WORD(B, C) ;
```



### 3.4 NOT

#### Négation des bits d'un mot

---

##### Paramètres

entrées			
IN	ANY_BIT	Mot à inverser	
sorties			
OUT	ANY_BIT	Résultat de l'inversion des bits du mot	

---

##### Description

Cette fonction réalise la négation de tous les bits du mot **IN**.

---

##### Exemple

value of IN:	2#0000_0001_0011_0111
OUT result:	2#1111_1110_1100_1000

---

##### BF typées

<b>NOT_BYTE</b>	Négation des bits d'un mot de 8 bits
<b>NOT_WORD</b>	Négation des bits d'un mot de 16 bits
<b>NOT_DWORD</b>	Négation des bits d'un mot de 32 bits

---

##### Exemple ST

A := NOT\_WORD(B) ;

---





## 4 Décalages et rotations

Les opérations de décalages et de rotations permettent de manipuler les variables de types suivants :

- mots de 8 bits (BYTE),
- mots de 16 bits (WORD),
- mots de 32 bits (DWORD).

FB contenues dans cette famille :

opérateur	Libellé
<b>SHL</b>	Décalage à gauche de N bits
<b>SHR</b>	Décalage à droite de N bits
<b>ROL</b>	Décalage circulaire à gauche de N bits
<b>ROR</b>	Décalage circulaire à droite de N bits



## 4.1 SHL

*Décalage à gauche de N bits*

### Paramètres

entrées			
IN	ANY_BIT	Mot à décaler	
N	INT	Nombre de bits à décaler	
sorties			
OUT	ANY_BIT	Résultat après décalage de N bits	

### Description

Cette fonction réalise le décalage à gauche de **N** positions de tous les bits du mot **IN** et range le résultat dans la sortie **OUT**.  
Les **N** bits qui sont sortis à gauche sont perdus .  
Des zéros ont été entrés à droite pour remplacer les cases laissées libres par le décalage.

### Limitation

Si le nombre **N** de bits à décaler est inférieur à 1 et supérieur à 15 pour un WORD (ou 7 pour un BYTE et 31 pour un DWORD), la sortie **OUT** reçoit le contenu de l'entrée **IN**.

### Exemple

value of IN:	2#0001_0011_0111_1111
value of N:	4
OUT result:	2#0011_0111_1111_0000

### BF typées

<b>SHL_BYTE</b>	Décalage à gauche de N bits d'un mot de 8 bits
<b>SHL_WORD</b>	Décalage à gauche de N bits d'un mot de 16 bits
<b>SHL_DWORD</b>	Décalage à gauche de N bits d'un mot de 32 bits

### Exemple ST

```
A := SHL_WORD(B,C) ;
```

## 4.2

## SHR

*Décalage à droite de N bits***Paramètres**

entrées			
IN	ANY_BIT	Mot à décaler	
N	INT	Nombre de bits à décaler	
sorties			
OUT	ANY_BIT	Résultat après décalage de N bits	

**Description**


Cette fonction réalise le décalage à droite de **N** positions de tous les bits du mot **IN** et range le résultat dans la sortie **OUT**.  
 Les **N** bits qui sont sortis à droite sont perdus .  
 Des zéros ont été entrés à gauche pour remplacer les cases laissées libres par le décalage.

**Limitation**

Si le nombre **N** de bits à décaler est inférieur à 1 et supérieur à 15 pour un WORD (ou 7 pour un BYTE et 31 pour un DWORD), la sortie **OUT** reçoit le contenu de l'entrée **IN**.

**Exemple**

value of IN: 2# 0001 0011 0111 1111  
 value of N: 4  
 OUT result: 2# 0000 0001 0011 0111


**BF typées**

<b>SHR_BYTE</b>	Décalage à droite de N bits d'un mot de 8 bits
<b>SHR_WORD</b>	Décalage à droite de N bits d'un mot de 16 bits
<b>SHR_DWORD</b>	Décalage à droite de N bits d'un mot de 32 bits

**Exemple ST**

A := SHR\_WORD(B,C) ;



### 4.3 ROL

*Décalage circulaire à gauche de N bits*

#### Paramètres

entrées			
IN	ANY_BIT	Mot à décaler	
N	INT	Nombre de bits à décaler	
sorties			
OUT	ANY_BIT	Résultat après décalage de N bits	

#### Description

Cette fonction réalise le décalage circulaire à gauche de **N** positions de tous les bits du mot **IN** et range le résultat dans la sortie **OUT**. Les bits qui sortent à gauche sont entrés à droite pour remplacer les cases laissées libres par le décalage.

#### Limitation

Si le nombre **N** de bits à décaler est inférieur à 1 et supérieur à 15 pour un WORD (ou 7 pour un BYTE et 31 pour un DWORD), la sortie **OUT** reçoit le contenu de l'entrée **IN**.

#### Exemple

value of IN: 2#0001\_0011\_0111\_1111  
value of N: 4  
OUT result: 2#0011\_0111\_1111\_0001

#### BF typées

<b>ROL_BYTE</b>	Décalage circulaire à gauche de N bits d'un mot de 8 bits
<b>ROL_WORD</b>	Décalage circulaire à gauche de N bits d'un mot de 16 bits
<b>ROL_DWORD</b>	Décalage circulaire à gauche de N bits d'un mot de 32 bits

#### Exemple ST

```
A := ROL_WORD(B,C) ;
```

## 4.4

**ROR***Décalage circulaire à droite de N bits***Paramètres**

entrées			
IN	ANY_BIT	Mot à décaler	
N	INT	Nombre de bits à décaler	
sorties			
OUT	ANY_BIT	Résultat après décalage de N bits	

**Description**

Cette fonction réalise le décalage circulaire à droite de **N** positions de tous les bits du mot **IN** et range le résultat dans la sortie **OUT**. Les bits qui sortent à droite sont entrés à gauche pour remplacer les cases laissées libres par le décalage.

**Limitation**

Si le nombre **N** de bits à décaler est inférieur à 1 et supérieur à 15 pour un WORD (ou 7 pour un BYTE et 31 pour un DWORD), la sortie **OUT** reçoit le contenu de l'entrée **IN**.

**Exemple**

value of IN: 2#0001\_0011\_0111\_1111  
 value of N: 4  
 OUT result: 2#1111\_0001\_0011\_0111


**BF typées**

<b>ROR_BYTE</b>	Décalage circulaire à droite de N bits d'un mot de 8 bits
<b>ROR_WORD</b>	Décalage circulaire à droite de N bits d'un mot de 16 bits
<b>ROR_DWORD</b>	Décalage circulaire à droite de N bits d'un mot de 32 bits

**Exemple ST**

```
A := ROR_WORD(B,C) ;
```



## 5 Sélection sur variables

Les opérations de sélection permettent de manipuler les variables de types suivants :

- Booléens (BOOL),
- mots de 8 bits (BYTE),
- mots de 16 bits (WORD),
- mots de 32 bits (DWORD)
- entiers de 8 bits (SINT),
- entiers de 16 bits (INT),
- entiers de 32 bits (DINT),
- réels (à virgule flottante IEEE de 32 bits),
- chaînes de caractères (STRING),
- durée (TIME),
- date (DATE),
- heure (TIME\_OF\_DAY),
- date et heure (DATE\_AND\_TIME),

FB contenues dans cette rubrique :

opérateur	Libellé
<b>SEL</b>	Choix binaire sur 2 variables
<b>MUX</b>	Multiplexeur sur N variables
<b>MAX</b>	Recherche de la plus grande variable
<b>MIN</b>	Recherche de la plus petite variable
<b>LIMIT</b>	Limitation (min. et max.) d'une variable

### Note pour les chaînes de caractères :

*Pour les opérateurs de sélection (MIN, MAX, LIMIT) :*

*L'opération est effectuée sur les caractères de chaque chaîne de la gauche vers la droite en prenant en compte la valeur ASCII de chaque caractère (selon la table de codage ISO 646).*

*Si les deux chaînes n'ont pas la même longueur utile, la plus petite chaîne est considérée comme ayant la même longueur que la plus grande. Les caractères temporaires supplémentaires prennent la valeur ASCII 0.*

## 5.1

**SEL***Choix binaire sur deux variables***Paramètres**

entrées		
G	BOOL	Sélecteur binaire ( 0 ou 1)
IN0	ANY	Entrée pour la valeur <b>0</b> du sélecteur
IN1	ANY	Entrée pour la valeur <b>1</b> du sélecteur
sorties		
OUT	ANY	Résultat selon la valeur du sélecteur

**Description**

Cette fonction permet de sélectionner l'un des deux nombres en fonction de la valeur d'un sélecteur binaire.

- si **G** est égal à 0 alors **OUT** prend la valeur de **IN0**.
- si **G** est égal à 1 alors **OUT** prend la valeur de **IN1**.

**BF typées**

<b>SEL_BOOL</b>	Sélection de l'un des deux booléens
<b>SEL_BYTE</b>	Sélection de l'un des deux mots de 8 bits
<b>SEL_WORD</b>	Sélection de l'un des deux mots de 16 bits
<b>SEL_DWORD</b>	Sélection de l'un des deux mots de 32 bits
<b>SEL_SINT</b>	Sélection de l'un des deux entiers 8 bits
<b>SEL_INT</b>	Sélection de l'un des deux entiers 16 bits
<b>SEL_DINT</b>	Sélection de l'un des deux entiers 32 bits
<b>SEL_REAL</b>	Sélection de l'un des deux réels
<b>SEL_STR</b>	Sélection de l'une des deux chaînes
<b>SEL_TIME</b>	Sélection de l'une des deux durées
<b>SEL_DATE</b>	Sélection de l'une des deux dates
<b>SEL_TOD</b>	Sélection de l'une des deux heures
<b>SEL_DT</b>	Sélection de l'une des deux dates

**Exemple ST**

```
A := SEL_INT(B, C0, C1) ;
```



## 5.2 MUX

### Multiplexeur sur N variables

#### Paramètres

entrées			
K	INT		Valeur du sélecteur ( $0 \leq K \leq 9$ )
IN0	ANY		Entrée pour la valeur <b>0</b> du sélecteur
IN1	ANY		Entrée pour la valeur <b>1</b> du sélecteur
...			
IN9	ANY		Entrée pour la valeur <b>9</b> du sélecteur
sorties			
OUT	ANY		Résultat selon la valeur du sélecteur

#### Description

Cette fonction permet de sélectionner l'un des dix nombres en fonction de la valeur d'un sélecteur entier.

- si **K** est égal à 0 alors **OUT** prend la valeur de **IN0**.
- si **K** est égal à 1 alors **OUT** prend la valeur de **IN1**.
- ...
- si **K** est égal à 9 alors **OUT** prend la valeur de **IN9**.

#### Limitation

Le sélecteur **K** doit être compris entre les valeurs 0 et 9 incluses. En dehors de ces valeurs, la sortie **OUT** prend la valeur de **IN0**.

#### BF typées

<b>MUX_BOOL</b>	Sélection de l'un des <b>K</b> booléens
<b>MUX_BYTE</b>	Sélection de l'un des <b>K</b> mots de 8 bits
<b>MUX_WORD</b>	Sélection de l'un des <b>K</b> mots de 16 bits
<b>MUX_DWORD</b>	Sélection de l'un des <b>K</b> mots de 32 bits
<b>MUX_SINT</b>	Sélection de l'un des <b>K</b> entiers 8 bits
<b>MUX_INT</b>	Sélection de l'un des <b>K</b> entiers 16 bits
<b>MUX_DINT</b>	Sélection de l'un des <b>K</b> entiers 32 bits
<b>MUX_REAL</b>	Sélection de l'un des <b>K</b> réels
<b>MUX_STR</b>	Sélection de l'une des <b>K</b> chaînes
<b>MUX_TIME</b>	pour sélectionner l'une des <b>K</b> durées
<b>MUX_DATE</b>	pour sélectionner l'une des <b>K</b> dates
<b>MUX_TOD</b>	pour sélectionner l'une des <b>K</b> heures du jour
<b>MUX_DT</b>	pour sélectionner l'une des <b>K</b> dates et heures

#### Exemple ST

```
A := MUX_INT(B, C0, C1, ..., C8, C9) ;
```



## 5.3

**MAX***Recherche de la plus grande variable***Paramètres**

entrées			
IN1	ANY	Premier terme de la fonction	
IN2	ANY	Deuxième terme de la fonction	
sorties			
OUT	ANY	Contient la valeur du plus grand	

**Description**

Cette fonction permet de comparer deux variables et de retourner la plus grande valeur des deux.

**BF typées**

<b>MAX_BYTE</b>	Plus grand des deux mots de 8 bits
<b>MAX_WORD</b>	Plus grand des deux mots de 16 bits
<b>MAX_DWORD</b>	Plus grand des deux mots de 32 bits
<b>MAX_SINT</b>	Plus grand des deux entiers 8 bits
<b>MAX_INT</b>	Plus grand des deux entiers 16 bits
<b>MAX_DINT</b>	Plus grand des deux entiers 32 bits
<b>MAX_REAL</b>	Plus grand des deux réels
<b>MAX_STR</b>	Plus grande des deux chaînes
<b>MAX_TIME</b>	Plus grande des deux durées
<b>MAX_DATE</b>	Plus grande des deux dates
<b>MAX_TOD</b>	Plus grande des deux heures du jour
<b>MAX_DT</b>	Plus grande des deux dates et heures

**Exemple ST**

```
A := MAX_INT(B, C) ;
```



### 5.4 MIN

#### Recherche de la plus petite variable

---

##### Paramètres

entrées			
IN1	ANY		Premier terme de la fonction
IN2	ANY		Deuxième terme de la fonction
sorties			
OUT	ANY		Contient la valeur du plus petit

---

##### Description

Cette fonction permet de comparer deux variables et de retourner la plus petite valeur des deux.

---

##### BF typées

<b>MIN_BYTE</b>	Plus petit des deux mots de 8 bits
<b>MIN_WORD</b>	Plus petit des deux mots de 16 bits
<b>MIN_DWORD</b>	Plus petit des deux mots de 32 bits
<b>MIN_SINT</b>	Plus petit des deux entiers 8 bits
<b>MIN_INT</b>	Plus petit des deux entiers 16 bits
<b>MIN_DINT</b>	Plus petit des deux entiers 32 bits
<b>MIN_REAL</b>	Plus petit des deux réels
<b>MIN_STR</b>	Plus petite des deux chaînes
<b>MIN_TIME</b>	Plus petite des deux durées
<b>MIN_DATE</b>	Plus petite des deux dates
<b>MIN_TOD</b>	Plus petite des deux heures du jour
<b>MIN_DT</b>	Plus petite des deux dates et heures

---

##### Exemple ST

```
A := MIN_INT(B, C);
```

---

## 5.5

**LIMIT***Limitation min et max d'une variable***Paramètres**

entrées			
MN	ANY	Borne minimale	
IN	ANY	Nombre à borner	
MX	ANY	Borne maximale	
sorties			
OUT	ANY	Valeur bornée de l'entrée IN	

**Description**

Cette fonction permet de limiter la valeur d'un nombre entre deux bornes (min et max).

**BF typées**

<b>LIMIT_SINT</b>	Bornage d'un mot de 8 bits
<b>LIMIT_INT</b>	Bornage d'un mot de 16 bits
<b>LIMIT_DINT</b>	Bornage d'un mot de 32 bits
<b>LIMIT_SINT</b>	Bornage d'un entier 8 bits
<b>LIMIT_INT</b>	Bornage d'un entier 16 bits
<b>LIMIT_DINT</b>	Bornage d'un entier 32 bits
<b>LIMIT_REAL</b>	Bornage d'un réel
<b>LIMIT_STR</b>	Bornage d'une chaîne
<b>LIMIT_TIME</b>	Bornage d'une durée
<b>LIMIT_DATE</b>	Bornage d'une date
<b>LIMIT_TOD</b>	Bornage d'une heure
<b>LIMIT_DT</b>	Bornage d'une date et heure

**Exemple ST**

```
A := LIMIT_INT(MIN, B , MAX) ;
```



## 6 Opérations de comparaison

Les opérations de comparaison permettent de comparer les paires suivantes de variables IN1 et IN2 :

- deux mots de 8 bits (BYTE),
- deux mots de 16 bits (WORD),
- deux mots de 32 bits (DWORD),
- deux entiers de 8 bits (SINT),
- deux entiers de 16 bits (INT),
- deux entiers de 32 bits (DINT),
- deux réels (à virgule flottante IEEE de 32 bits),
- deux chaînes de caractères (STRING),
- deux durées (TIME),
- deux dates (DATE),
- deux heures (TIME\_OF\_DAY),
- deux dates et heures (DATE\_AND\_TIME),

Les opérations de comparaison sont les suivantes :

opérateur	Libellé
<b>EQ</b>	Test si IN1 est <b>égal</b> à IN2
<b>GE</b>	Test si IN1 est <b>supérieur ou égal</b> à IN2
<b>GT</b>	Test si IN1 est <b>strictement supérieur</b> à IN2
<b>LE</b>	Test si IN1 est <b>inférieur ou égal</b> à IN2
<b>LT</b>	Test si IN1 est <b>strictement inférieur</b> à IN2
<b>NE</b>	Test si IN1 est <b>différent</b> de IN2

### Note pour les chaînes de caractères :

*Pour les opérateurs de comparaison, l'opération est effectuée sur les caractères de chaque chaîne de la gauche vers la droite en prenant en compte la valeur ASCII de chaque caractère (selon la table de codage ISO 646).*

*Si les deux chaînes n'ont pas la même longueur utile, la plus petite chaîne est considérée comme ayant la même longueur que la plus grande. Les caractères temporaires supplémentaires prennent la valeur ASCII 0.*

## 6.1

**EQ***Test si IN1 est égal à IN2***Paramètres**

entrées			
IN1	ANY	Premier terme de la comparaison	
IN2	ANY	Deuxième terme de la comparaison	
sorties			
OUT	BOOL	Résultat de la comparaison	

**Description**

L'opération permet de tester si les deux variables **IN1** et **IN2** sont égales.

**BF typées**

<b>EQ_BYTE</b>	égalité de deux mots de 8 bits
<b>EQ_WORD</b>	égalité de deux mots de 16 bits
<b>EQ_DWORD</b>	égalité de deux mots de 32 bits
<b>EQ_SINT</b>	égalité de deux entiers 8 bits
<b>EQ_INT</b>	égalité de deux entiers 16 bits
<b>EQ_DINT</b>	égalité de deux entiers 32 bits
<b>EQ_REAL</b>	égalité de deux réels (flottant)
<b>EQ_STR</b>	égalité de deux chaînes
<b>EQ_TIME</b>	égalité de deux durées
<b>EQ_DATE</b>	égalité de deux dates
<b>EQ_TOD</b>	égalité de deux heures
<b>EQ_DT</b>	égalité de deux "date et heure"

**Exemple ST**

```
A := EQ_INT(B, C);
```



### 6.2

### GE

*Test si IN1 est supérieur ou égal à IN2*

---

#### Paramètres

entrées			
IN1	ANY	Premier terme de la comparaison	
IN2	ANY	Deuxième terme de la comparaison	
sorties			
OUT	BOOL	Résultat de la comparaison	

---

#### Description

L'opération permet de tester si la variable **IN1** est supérieure ou égale à la variable **IN2**.

---

#### BF typées

<b>GE_BYTE</b>	supériorité entre deux mots de 8 bits
<b>GE_WORD</b>	supériorité entre deux mots de 16 bits
<b>GE_DWORD</b>	supériorité entre deux mots de 32 bits
<b>GE_SINT</b>	supériorité entre deux entiers 8 bits
<b>GE_INT</b>	supériorité entre deux entiers 16 bits
<b>GE_DINT</b>	supériorité entre deux entiers 32 bits
<b>GE_REAL</b>	supériorité entre deux réels 32 bits
<b>GE_STR</b>	supériorité entre deux chaînes
<b>GE_TIME</b>	supériorité entre deux durées
<b>GE_DATE</b>	supériorité entre deux dates
<b>GE_TOD</b>	supériorité entre deux heures
<b>GE_DT</b>	supériorité entre deux "date et heure"

---

#### Exemple ST

```
A := GE_INT(B, C);
```

---

## 6.3

**GT***Test si IN1 est strictement supérieur à IN2***Paramètres**

entrées			
IN1	ANY	Premier terme de la comparaison	
IN2	ANY	Deuxième terme de la comparaison	
sorties			
OUT	BOOL	Résultat de la comparaison	

**Description**

L'opération permet de tester si la variable **IN1** est strictement supérieure à la variable **IN2**.

**BF typées**

<b>GT_BYTE</b>	supériorité stricte entre deux mots de 8 bits
<b>GT_WORD</b>	supériorité stricte entre deux mots de 16 bits
<b>GT_DWORD</b>	supériorité stricte entre deux mots de 32 bits
<b>GT_SINT</b>	supériorité stricte entre deux entiers 8 bits
<b>GT_INT</b>	supériorité stricte entre deux entiers 16 bits
<b>GT_DINT</b>	supériorité stricte entre deux entiers 32 bits
<b>GT_REAL</b>	supériorité stricte entre deux réels 32 bits
<b>GT_STR</b>	supériorité stricte entre deux chaînes
<b>GT_TIME</b>	supériorité stricte entre deux durées
<b>GT_DATE</b>	supériorité stricte entre deux dates
<b>GT_TOD</b>	supériorité stricte entre deux heures
<b>GT_DT</b>	supériorité stricte entre deux "date et heure"

**Exemple ST**

```
A := GT_INT(B, C) ;
```



### 6.4 LE

*Test si IN1 est inférieur ou égal à IN2*

---

#### Paramètres

entrées			
IN1	ANY		Premier terme de la comparaison
IN2	ANY		Deuxième terme de la comparaison
sorties			
OUT	BOOL		Résultat de la comparaison

---

#### Description

L'opération permet de tester si la variable **IN1** est inférieure ou égale à la variable **IN2**.

---

#### BF typées

<b>LE_BYTE</b>	infériorité entre deux mots de 8 bits
<b>LE_WORD</b>	infériorité entre deux mots de 16 bits
<b>LE_DWORD</b>	infériorité entre deux mots de 32 bits
<b>LE_SINT</b>	infériorité entre deux entiers 8 bits
<b>LE_INT</b>	infériorité entre deux entiers 16 bits
<b>LE_DINT</b>	infériorité entre deux entiers 32 bits
<b>LE_REAL</b>	infériorité entre deux réels 32 bits
<b>LE_STR</b>	infériorité entre deux chaînes
<b>LE_TIME</b>	infériorité entre deux durées
<b>LE_DATE</b>	infériorité entre deux dates
<b>LE_TOD</b>	infériorité entre deux heures
<b>LE_DT</b>	infériorité entre deux "date et heure"

---

#### Exemple ST

A := LE\_INT(B, C) ;

---



## 6.5

## LT

*Test si IN1 est strictement inférieur à IN2***Paramètres**

entrées			
IN1	ANY	Premier terme de la comparaison	
IN2	ANY	Deuxième terme de la comparaison	
sorties			
OUT	BOOL	Résultat de la comparaison	

**Description**

L'opération permet de tester si la variable **IN1** est strictement inférieure à la variable **IN2**.

**BF typées**

<b>LT_BYTE</b>	infériorité stricte entre deux mots de 8 bits
<b>LT_WORD</b>	infériorité stricte entre deux mots de 16 bits
<b>LT_DWORD</b>	infériorité stricte entre deux mots de 32 bits
<b>LT_SINT</b>	infériorité stricte entre deux entiers 8 bits
<b>LT_INT</b>	infériorité stricte entre deux entiers 16 bits
<b>LT_DINT</b>	infériorité stricte entre deux entiers 32 bits
<b>LT_REAL</b>	infériorité stricte entre deux réels 32 bits
<b>LT_STR</b>	infériorité stricte entre deux chaînes
<b>LT_TIME</b>	infériorité stricte entre deux durées
<b>LT_DATE</b>	infériorité stricte entre deux dates
<b>LT_TOD</b>	infériorité stricte entre deux heures
<b>LT_DT</b>	infériorité stricte entre deux "date et heure"

**Exemple ST**

```
A := LT_INT(B, C) ;
```



### 6.6 NE

*Test si IN1 est différent de IN2*

---

#### Paramètres

entrées			
IN1	ANY	Premier terme de la comparaison	
IN2	ANY	Deuxième terme de la comparaison	
sorties			
OUT	BOOL	Résultat de la comparaison	

---

#### Description

L'opération permet de tester si les deux variables d'entrée **IN1** et **IN2** sont différentes.

---

#### BF typées

<b>NE_BYTE</b>	test d'inégalité de deux mots de 8 bits
<b>NE_WORD</b>	test d'inégalité de deux mots de 16 bits
<b>NE_DWORD</b>	test d'inégalité de deux mots de 32 bits
<b>NE_SINT</b>	test d'inégalité de deux entiers 8 bits
<b>NE_INT</b>	test d'inégalité de deux entiers 16 bits
<b>NE_DINT</b>	test d'inégalité de deux entiers 32 bits
<b>NE_REAL</b>	test d'inégalité de deux réels 32 bits
<b>NE_STR</b>	test d'inégalité de deux chaînes
<b>NE_TIME</b>	test d'inégalité de deux durées
<b>NE_DATE</b>	test d'inégalité de deux dates
<b>NE_TOD</b>	test d'inégalité de deux heures
<b>NE_DT</b>	test d'inégalité de deux "date et heure"

---

#### Exemple ST

A := NE\_INT(B, C) ;

---

## 7 Opérations arithmétiques

Les boîtes fonctionnelles suivantes permettent de réaliser des opérations arithmétiques sur les paires suivantes de variables IN1 et IN2 :

- deux entiers de 8 bits,
- deux entiers de 16 bits,
- deux entiers de 32 bits (entiers doubles),
- deux réels (à virgule flottante IEEE de 32 bits),
- deux durées (TIME),
- deux dates (DATE),
- deux heures (TIME\_OF\_DAY),
- deux dates et heures (DATE\_AND\_TIME),

FB contenues dans cette rubrique :

opérateur	Libellé
<b>ADD</b>	Addition de 2 variables
<b>MUL</b>	Multiplication 2 variables
<b>SUB</b>	Soustraction de 2 variables
<b>DIV</b>	Division de 2 variables
<b>MOD</b>	Reste d'une division de 2 variables
<b>EXPT</b>	Exponentiation d'un variables
<b>CONCAT</b>	Concaténation d'une date et de l'heure du jour

**Note :** La variable résultat est toujours de même type que les deux entrées de la fonction.



### 7.1 ADD

#### Addition de 2 variables

---

##### Paramètres

entrées			
IN1	ANY	Premier terme de l'opération	
IN2	ANY	Deuxième terme de l'opération	
sorties			
OUT	ANY	Résultat de l'opération	

---

##### Description

Cette fonction réalise l'addition du contenu des deux variables **IN2** et **IN1** et range le résultat dans la sortie **OUT**.

---

##### BF typées

<b>ADD_SINT</b>	Addition de deux entiers 8 bits
<b>ADD_INT</b>	Addition de deux entiers 16 bits
<b>ADD_DINT</b>	Addition de deux entiers 32 bits
<b>ADD_REAL</b>	Addition de deux réels
<b>ADD_TIME</b>	Addition d'une durée à une autre
<b>ADD_DT_T</b>	Addition d'une durée à une date & time
<b>ADD_TOD_T</b>	Addition d'une durée à l'heure du jour

---

##### Exemple ST

A := ADD\_INT(B, C) ;

---

## 7.2

**MUL***Multiplication de 2 variables***Paramètres**

entrées			
IN1	ANY	Premier terme de l'opération	
IN2	ANY	Deuxième terme de l'opération	
sorties			
OUT	ANY	Résultat de l'opération	

**Description**

Cette fonction réalise la multiplication du contenu des deux variables **IN2** et **IN1** et range le résultat dans la sortie **OUT**.

**BF typées**

<b>MUL_SINT</b>	Multiplication de deux entiers 8 bits
<b>MUL_INT</b>	Multiplication de deux entiers 16 bits
<b>MUL_DINT</b>	Multiplication de deux entiers 32 bits
<b>MUL_REAL</b>	Multiplication de deux réels
<b>MUL_T_SINT</b>	Multiplication d'une durée par un entier 8 bits
<b>MUL_T_INT</b>	Multiplication d'une durée par un entier 16 bits
<b>MUL_T_DINT</b>	Multiplication d'une durée par un entier 32 bits
<b>MUL_T_REAL</b>	Multiplication d'une durée par un réel

**Exemple ST**

```
A := MUL_INT(B1, B2) ;
```



### 7.3

## SUB

### Soustraction de deux variables

#### Paramètres

entrées			
IN1	ANY		Premier terme de l'opération
IN2	ANY		Deuxième terme de l'opération
sorties			
OUT	ANY		Résultat de l'opération

#### Description

Cette fonction réalise la soustraction du contenu de la variable **IN2** à la variable **IN1** et range le résultat dans la sortie **Q**.

#### BF typées

<b>SUB_SINT</b>	Soustraction de deux entiers 8 bits
<b>SUB_INT</b>	Soustraction de deux entiers 16 bits
<b>SUB_DINT</b>	Soustraction de deux entiers 32 bits
<b>SUB_REAL</b>	Soustraction de deux réels
<b>SUB_DATE_DATE</b>	Soustraction d'une date à une autre
<b>SUB_TOD_TOD</b>	Soustraction d'une heure à une autre
<b>SUB_DT_DT</b>	Soustraction de d'une date&time à une autre
<b>SUB_DT_T</b>	Soustraction d'une durée à une heure
<b>SUB_TOD_T</b>	Soustraction d'une durée à l'heure du jour

#### Exemple ST

```
A := SUB_INT(B, C) ;
```

## 7.4

**DIV***Division de deux variables***Paramètres**

entrées			
IN1	ANY	Premier terme de la fonction	
IN2	ANY	Deuxième terme de la fonction	
sorties			
OUT	ANY	Résultat de la division	

**Description**

Cette fonction réalise la division du contenu de la variable **IN1** par le contenu de la variable **IN2** et range le résultat dans la variable de sortie **Q**.

**Remarque**

Le résultat d'une division de deux entiers est un entier de même type avec une troncatisation (les chiffres après la virgule sont perdus).

**Exemples**

DIV\_INT(7, 3) = 2  
DIV\_INT(-7, 3) = -2

**Limitation**

Si le deuxième terme **IN2** de la fonction est nul, le résultat **OUT** prend la valeur 0.

**BF typées**

<b>DIV_SINT</b>	Résultat entier d'une division de deux entiers 8 bits
<b>DIV_INT</b>	Résultat entier d'une division de deux entiers 16 bits
<b>DIV_DINT</b>	Résultat entier d'une division de deux entiers 32 bits
<b>DIV_REAL</b>	Résultat réel d'une division de deux réels
<b>DIV_T_SINT</b>	Résultat de la division d'une durée par un entier de 8 bits
<b>DIV_T_INT</b>	Résultat de la division d'une durée par un entier de 16 bits
<b>DIV_T_DINT</b>	Résultat de la division d'une durée par un entier de 32 bits
<b>DIV_T_REAL</b>	Résultat de la division d'une durée par un réel

**Exemple ST**

A := DIV\_INT(B, C) ;



### 7.5

### MOD

*Reste d'une division de deux variables*

---

#### Paramètres

entrées	IN1	ANY_INT	Premier terme de la fonction
	IN2	ANY_INT	Deuxième terme de la fonction
sorties	OUT	ANY_INT	Reste de la division

---

#### Description

Cette fonction réalise la division du contenu de la variable **IN1** par le contenu de la variable **IN2** et range le reste dans la variable de sortie **Q**.

---

#### Limitation

Si le deuxième terme **IN2** de la fonction est nul, le résultat **OUT** prend la valeur 0.

---

#### BF typées

<b>MOD_SINT</b>	Reste d'une division de deux entiers 8 bits
<b>MOD_INT</b>	Reste d'une division de deux entiers 16 bits
<b>MOD_DINT</b>	Reste d'une division de deux entiers 32 bits

---

#### Exemple ST

A := MOD\_INT(B, C) ;

---





## 7.6

**EXPT***Exponentiation d'une variable***Paramètres**

entrées		
IN1	ANY_NUM	Premier terme de la fonction
IN2	ANY_NUM	Deuxième terme de la fonction
sorties		
OUT	ANY_NUM	Résultat de l'exponentiation

**Description**

Cette opération réalise la mise à la puissance **IN2** de la variable **IN1** et range le résultat dans la variable **OUT**.

**BF typées**

<b>EXPT_SINT</b>	exponentiation d'un entier 8 bits par un autre
<b>EXPT_INT</b>	exponentiation d'un entier 16 bits par un autre
<b>EXPT_DINT</b>	exponentiation d'un entier 32 bits par un autre
<b>EXPT_REAL</b>	exponentiation d'un réel par un autre

**Exemple ST**

```
A := EXPT_INT(B, C);
```



### 7.7

## CONCAT

*Concaténation d'une date et d'une heure*

### Paramètres

entrées			
IN1	DATE	Premier terme de la fonction	
IN2	TOD	Deuxième terme de la fonction	
sorties			
OUT	DT	Résultat de la concaténation	

### Description

Cette opération réalise la concaténation de la variable **IN1** avec la variable **IN2** et range le résultat dans la variables de sortie **OUT**.

### BF typées

**CONCAT\_DATE\_TOD** concaténation d'une date avec une heure

### Exemple ST

```
A:= CONCAT_DATE_TOD(B, C);
```

## 8 Fonctions numériques

FB contenues dans cette rubrique :

opérateur	Libellé
<b>ABS</b>	Valeur absolue d'un nombre
<b>SQRT</b>	Racine carrée d'un nombre réel
<b>LN</b>	Logarithme népérien d'un nombre réel
<b>LOG</b>	Logarithme base 10 d'un nombre réel
<b>EXP</b>	Exponentiel d'un nombre réel
<b>SIN</b>	Sinus d'un nombre réel en radian
<b>COS</b>	Cosinus d'un nombre réel en radian
<b>TAN</b>	Tangente d'un nombre réel en radian
<b>ASIN</b>	Arc sinus d'un nombre réel
<b>ACOS</b>	Arc cosinus d'un nombre réel
<b>ATAN</b>	Arc tangente d'un nombre réel



### 8.1 ABS

*Valeur absolue d'un nombre*

---

#### Paramètres

entrées			
IN	ANY_NUM	Entrée à traiter par la fonction	
sorties			
OUT	ANY_NUM	Résultat de la fonction	

---

#### Description

Cette fonction retourne la valeur absolue de l'entrée.

---

#### BF typées

<b>ABS_SINT</b>	valeur absolue d'un entier 8 bits
<b>ABS_INT</b>	valeur absolue d'un entier 16 bits
<b>ABS_DINT</b>	valeur absolue d'un entier 32 bits
<b>ABS_REAL</b>	valeur absolue d'un réel

---

#### Exemple ST

A := ABS\_INT(B)

---



## 8.2

**SQRT***Racine carrée d'un réel***Paramètres**

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

**Description**

Cette fonction retourne la racine carrée d'un nombre.

**Limitation**

L'entrée **IN** de la fonction doit être supérieure ou égale à 0.  
Dans le cas contraire, le résultat **OUT** vaut 0.0.

**Exemple ST**

```
A := SQRT(B) ;
```



### 8.3

### LN

*Logarithme népérien d'un réel*

---

#### Paramètres

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

---

#### Description

Cette fonction retourne le logarithme népérien d'un nombre

---

#### Limitation

L'entrée **IN** de la fonction doit être supérieure ou égale à 0.0.  
Dans le cas contraire, le résultat **OUT** vaut 0.0.

---

#### Exemple ST

A := LN(B) ;

---



## 8.4

**LOG***Logarithme base 10 d'un réel***Paramètres**

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

**Description**

Cette fonction retourne le logarithme base 10 d'un nombre.

**Limitation**

L'entrée **IN** de la fonction doit être supérieure ou égale à 0.  
Dans le cas contraire, le résultat **OUT** vaut 0.0.

**Exemple ST**

```
A := LOG(B) ;
```



### 8.5 EXP

*Exponentiel d'un réel*

---

#### Paramètres

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

---

#### Description

Cette fonction retourne la valeur exponentielle d'un nombre.

---

#### Exemple ST

A := EXP(B) ;

---





## 8.6

**SIN***Sinus d'un réel***Paramètres**

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

**Description**

Cette fonction retourne la valeur du sinus d'un nombre.

**Exemple ST**

```
A := SIN(B) ;
```



## 8.7

### COS

*Cosinus d'un réel*

---

#### Paramètres

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

---

#### Description

Cette fonction retourne la valeur du cosinus d'un nombre.

---

#### Exemple ST

A := COS(B) ;

---



## 8.8

**TAN***Tangente d'un réel***Paramètres**

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

**Description**

Cette fonction retourne valeur de la tangente d'un nombre.

**Exemple ST**

```
A := TAN(B) ;
```



### 8.9

### ASIN

*Arc sinus d'un réel*

---

#### Paramètres

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

---

#### Description

Cette fonction retourne la valeur de l'arc sinus d'un nombre.

---

#### Limitation

La valeur **EN** en entrée de la fonction doit être comprise entre les valeurs -1 et 1 incluses.  
Dans le cas contraire, le résultat **OUT** vaut 0.0.

---

#### Exemple ST

A := ASIN(B)

---



## 8.10 ACOS

*Arc cosinus d'un réel*

---

### Paramètres

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

---

### Description

Cette fonction retourne la valeur de l'arc cosinus d'un nombre.

---

### Limitation

La valeur **EN** en entrée de la fonction doit être comprise entre les valeurs -1 et 1 incluses.  
Dans le cas contraire, le résultat **OUT** vaut 0.0.

---

### Exemple ST

A := ACOS(B) ;

---



## 8.11 ATAN

*Arc tangente d'un réel*

---

### Paramètres

entrées			
IN	REAL	Entrée à traiter par la fonction	
sorties			
OUT	REAL	Résultat de la fonction	

---

### Description

Cette fonction retourne la valeur de l'arc tangente d'un nombre.

---

### Exemple ST

A := ATAN(B) ;

---



## 9 Comptage

FB contenues dans cette rubrique :

opérateur	Libellé
<b>CTU</b>	Compteur
<b>CTD</b>	Décompteur
<b>CTUD</b>	Compteur - décompteur



## 9.1 CTU

### Compteur

#### Paramètres

entrées			
CU	BOOL	Entrée de comptage	
R	BOOL	Remise à zéro du compteur	
PV	INT	Présélection du seuil	
sorties			
Q	BOOL	Dépassement de seuil supérieur	
CV	INT	Valeur du compteur	
locales			
cv_ctu	INT	Mémorisation interne de la valeur du compteur	
t_ctu	INT	Mémorisation horloge interne du compteur	

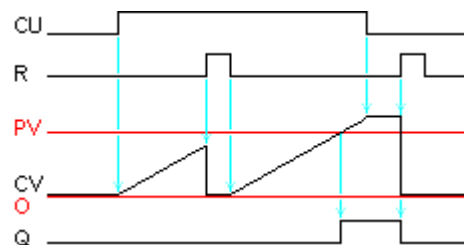
#### Description

SI l'entrée R est à 1 ALORS la sortie CV est forcée à 0,  
SINON  
    SI l'entrée CU est à 1 ET TANT QUE CV est inférieur à la  
    valeur maximum autorisée pour un entier simple ALORS la  
    sortie CV est incrémentée.  
TANT QUE la sortie CV est supérieure à la valeur de présélection PV  
ALORS la sortie Q prend la valeur 1.

avec

PVmax = 257 pour un SINT.

PVmax = 32767 pour un INT.



#### Exemple ST

```
ETAT := CTU(Compte, Reset, Preselection, temp1, temp2).Q;  
VALEUR := CTU(Compte, Reset, Preselection, temp1, temp2).CV;
```



## 9.2

**CTD***Décompteur***Paramètres**

## entrées

CD	BOOL	Entrée de décomptage
LD	BOOL	Remise à la valeur de présélection
PV	INT	Présélection du seuil

## sorties

Q	BOOL	Dépassement inférieur de seuil
CV	INT	Valeur du compteur

## locales

cv_ctd	INT	Mémorisation interne de la valeur du compteur
t_ctd	INT	Mémorisation horloge interne du compteur

**Description**

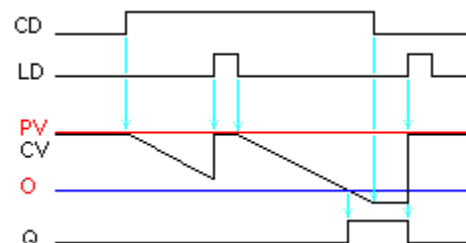
SI l'entrée R est à 1 ALORS la sortie CV est forcée à 0,  
SINON

SI l'entrée CU est à 1 ET TANT QUE CV est inférieur à la  
valeur maximum autorisée pour un entier simple ALORS la  
sortie CV est incrémentée.

TANT QUE la sortie CV est supérieure à la valeur de présélection PV  
ALORS la sortie Q prend la valeur 1.

avec

PVmax = 257 pour un SINT.  
PVmax = 32767 pour un INT.

**Exemple ST**

```
ETAT := CTD(Decompte, Charge, Preselection, temp1, temp2).Q;
VALEUR := CTD(Decompte, Charge, Preselection, temp1,
temp2).CV;
```

## 9.3 CTUD

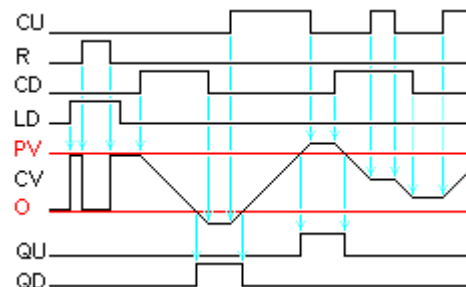
### Compteur - décompteur

#### Paramètres

entrées			
CU	BOOL	Entrée de comptage	
CD	BOOL	Entrée de décomptage	
R	BOOL	Rémise à zéro du compteur	
LD	BOOL	Rémise à la valeur de présélection	
PV	INT	Valeur de présélection du seuil	
sorties			
QU	BOOL	Dépassement supérieur de seuil	
QD	BOOL	Dépassement inférieur de seuil	
CV	INT	Valeur du compteur	
locales			
cv_ctud	INT	Mémorisation interne de la valeur du compteur	
t_ctud	INT	Mémorisation horloge interne du compteur	

#### Description

SI l'entrée R est à 1 ALORS la sortie CV est forcée à 0,  
SINON  
SI l'entrée CU est à 1 ET TANT QUE CV est inférieur à la  
valeur maximum autorisée pour un entier simple ALORS la  
sortie CV est incrémentée.  
TANT QUE CV est supérieure à la valeur de présélection PV ALORS  
la sortie Q prend la valeur 1.



#### Exemple ST

```

ETAT1 := CTUD(Cpt, Dcpt, Reset, Charge, Presel, temp1,
temp2).QU;
ETAT2 := CTUD(Cpt, Dcpt, Reset, Charge, Presel, temp1,
temp2).QD;
VALEUR := CTUD(Cpt, Dcpt, Reset, Charge, Presel, temp1,
temp2).CV;

```



## 10 Temporisation

FB contenues dans cette rubrique :

opérateur	Libellé
<b>TP</b>	Mise a niveau de signal
<b>TON</b>	Temporisation à l'activation
<b>TOF</b>	Temporisation à la desactivation

**Remarque :** Pour chaque temporisateur, trois horloges sont disponibles :

- **Txx\_SEC** (secondes),
- **Txx\_CS** (centièmes),
- **Txx\_MS** (millièmes)



## 10.1 TP

### Mise à niveau de signal

#### Paramètres

entrées						
IN	BOOL	Lancement de la temporisation				
PT	ANY_INT	Valeur de présélection				
sorties						
Q	BOOL	Sortie en fin de temporisation				
CV	ANY_INT	Valeur courante de la temporisation				
locales						
i_tp	INT	Horloge interne de la temporisation				
t1_tp	REAL	Mémorisation	valeur	réelle	de	la
temporisation						
t2_tp	INT	Mémorisation	résultat	entier	de	la
temporisation						

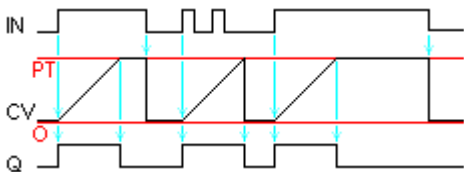
#### Description

Sur front montant de **IN**, la sortie **CV** s'incrémente et la sortie **OUT** est positionnée à 1.

Quand la valeur de **CV** est supérieure ou égale à la valeur de présélection **PV** :

- la valeur de **OUT** est positionnée à 0 quelque soit l'état de l'entrée **IN**.
- la valeur de **CV** est figée tant que l'entrée **IN** est à 1.
- la valeur de **CV** remise à 0 si l'entrée **IN** passe à 0.

Pendant l'incrémentation de **CV**, l'entrée **IN** peut changer plusieurs fois d'état sans influencer CV.



#### Exemple ST

```
ETAT := TP_SEC(Lance, Presel, temp1, temp2, temp3).Q;  
VALEUR := TP_SEC(Lance, Presel, temp1, temp2, temp3).CV;
```

10.2 TON

Temporisation à l'activation

Paramètres

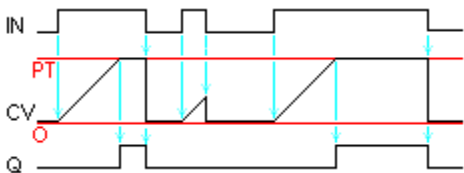
entrées						
IN	BOOL	Lancement de la temporisation				
PT	ANY_INT	Valeur de présélection				
sorties						
Q	BOOL	Sortie en fin de temporisation				
CV	ANY_INT	Valeur courante de la temporisation				
locales						
i_ton	INT	Horloge interne de la temporisation				
t1_ton	REAL	Mémorisation	valeur	réelle	de	la
temporisation						
t2_ton	INT	Mémorisation	résultat	entier	de	la
temporisation						

Description

Sur front montant de l'entrée **IN**, la sortie **CV** s'incrémente.  
Quand la valeur de **CV** est supérieure ou égale à la valeur de présélection **PV** :

- la sortie **OUT** est positionnée à 1 et la valeur de la sortie **CV** est figée tant que l'entrée **IN** est à 1.
- la sortie **CV** et la sortie **OUT** sont remises à 0 dès que l'entrée **IN** passe à 0.

Pendant l'incrémementation de **CV**, si l'entrée **IN** repasse à 0, la valeur de CV est remise à 0. Dans ce cas, la sortie Q reste positionnée à 0.



Exemple ST

```
ETAT := TON_SEC(Lance, Preselection, temp1, temp2, temp3).Q;  
VALEUR := TON_SEC(Lance, Preselection, temp1, temp2,  
temp3).CV;
```



## 10.3 TOF

### Temporisation à la désactivation

#### Paramètres

entrées			
IN	BOOL	Lancement de la temporisation	
PT	ANY_INT	Valeur de présélection	
sorties			
Q	BOOL	Sortie en fin de temporisation	
CV	ANY_INT	Valeur courante de la temporisation	
locales			
i_tof	INT	Horloge interne de la temporisation	
t1_tof	REAL	Mémorisation	valeur réelle de la
temporisation			
t2_tof	INT	Mémorisation	résultat entier de la
temporisation			

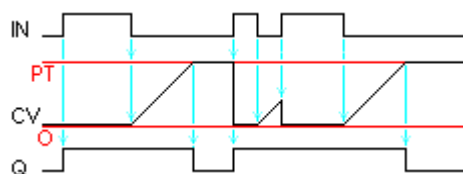
#### Description

Sur front montant de l'entrée **IN**, la sortie **OUT** est positionnée à 1 et la sortie **CV** est remise à 0.

Quand l'entrée **IN** est à 0, la sortie **CV** s'incrémente.

Quand la valeur de **CV** est supérieure ou égale à la valeur de présélection **PV** :

- la sortie **OUT** est positionnée à 0.
- la sortie **CV** est figée tant que l'entrée **IN** est à 0.



#### Exemple ST

```
ETAT := TOF_SEC(Lance, Preselection, temp1, temp2, temp3).Q;  
VALEUR := TOF_SEC(Lance, Preselection, temp1, temp2,  
temp3).CV;
```

# 11 Opérations sur chaînes de caractères

FB contenues dans cette famille :

opérateur	Libellé
<b>LEN</b>	Calcule la longueur utile d'une chaîne
<b>LEFT</b>	Extrait les N caractères de gauche
<b>RIGHT</b>	Extrait les N caractères de droite
<b>MID</b>	Extrait N caractères à partir d'une position
<b>CONCAT</b>	Concatène deux chaînes
<b>INSERT</b>	Insère une chaîne dans une deuxième à partir d'une position dans celle ci
<b>DELETE</b>	Supprime N caractères à partir d'une position
<b>REPLACE</b>	Remplace N caractères d'une chaîne à partir d'une position par une autre chaîne
<b>DELETE</b>	Recherche une chaîne dans une autre

**Remarques:** *Lors des opérations, si la variable réceptrice de la chaîne de sortie **OUT** n'a pas été déclarée avec une taille suffisante pour recevoir le résultat, les caractères de position supérieure à la longueur de la chaîne réceptrice sont perdus.*



### 11.1 LEN

*Calcule la longueur utile d'une chaîne*

---

#### Paramètres

entrées			
IN	STRING	Chaîne à mesurer	
sorties			
OUT	INT	Longueur de la chaîne	

---

#### Description

Cet boîte fonctionnelle calcule et retourne dans **OUT** la longueur utile de la chaîne d'entrée **IN**.

---

#### Exemples

IN	OUT
'ABCDE'	5
'XXX'	3

---

#### Exemple ST

```
OUT := LEN(IN);  
A := LEN('ABCDE');
```

---



## 11.2 LEFT

### Extrait N caractères de gauche

#### Paramètres

entrées			
IN	STRING	Chaîne en entrée	
L	INT	Nombre de caractères à extraire	
sorties			
OUT	STRING	Résultat de l'extraction	

#### Description

Cette boîte fonctionnelle extrait les **L** caractères de gauche de la chaîne **IN** passée en argument et les stocke dans la chaîne résultat **OUT**.

#### Limitation

Si le nombre **L** de caractères à extraire est supérieur à la longueur utile de la chaîne **IN**, la chaîne **OUT** reçoit tous les caractères de la chaîne **IN**.

#### Exemples

IN	L	OUT
'ABCDE'	2	'AB'
'12345'	10	'12345'

#### Exemple ST

```
OUT := LEFT(IN, L);
A := LEFT('ABCDE', 4);
```



## 11.3 RIGHT

*Extrait N caractères de droite*

### Paramètres

entrées			
IN	STRING	Chaîne en entrée	
L	INT	Nombre de caractères à extraire	
sorties			
OUT	STRING	Résultat de l'extraction	

### Description

Cette boîte fonctionnelle extrait les **L** caractères de droite de la chaîne **IN** passée en argument et les stocke dans la chaîne résultat **OUT**.

### Limitation

Si le nombre **L** de caractères à extraire est supérieur à la longueur utile de la chaîne **IN**, la chaîne **OUT** reçoit tous les caractères de la chaîne **IN**.

### Exemples

IN	L	OUT
'ABCDE'	2	'DE'
'12345'	10	'12345'

### Exemple ST

```
OUT := RIGHT(IN, L);  
A := RIGHT('ABCDE', 4);
```

## 11.4 MID

*Extrait N caractères à partir d'une position*

### Paramètres

entrées			
IN	STRING	Chaîne en entrée	
L	INT	Nombre de caractères à extraire	
P	INT	Position de début de l'extraction	
sorties			
OUT	STRING	Résultat de l'extraction	

### Description

Cette boîte fonctionnelle extrait les **L** caractères à partir de la position **P** dans la chaîne **IN** et les stocke dans la chaîne **OUT**.

### Limitation

Si le nombre **L** de caractères à extraire est supérieur au nombre de caractères restant entre la position **P** et la longueur utile de la chaîne **IN**, la chaîne **OUT** reçoit tous les caractères de la chaîne **IN** à droite de la position **P**.

Si la position **P** est supérieure à la longueur utile de la chaîne **IN**, la chaîne **OUT** est retournée vide car l'opération est annulée.

### Exemples

IN	L	P	OUT
'ABCDE'	2	3	'CD'
'ABCDE'	10	3	'CDE'
'ABCDE'	2	10	"

### Exemple ST

```
OUT := MID(IN, L, P);
A := MID('ABCDE', 10, 3);
```



## 11.5 CONCAT

### Concatène deux chaînes

#### Paramètres

entrées			
IN1	STRING	Chaîne 1 en entrée	
IN2	STRING	Chaîne 2 en entrée	
sorties			
OUT	STRING	Résultat de l'extraction	

#### Description

Cette boîte fonctionnelle stocke, dans la chaîne de sortie **OUT**, tous les caractères de la chaîne d'entrée **IN1** et, à leur suite, tous les caractères de la chaîne d'entrée **IN2**.

#### Exemples

IN1	IN2	OUT
'XXX'	'ABCDE'	'XXXABCDE'

#### Exemple ST

```
OUT := CONCAT(IN1, IN2);  
A := CONCAT('XXX', 'ABCDE');
```

## 11.6 INSERT

*Insère une chaîne dans une autre*

### Paramètres

entrées			
IN1	STRING	Chaîne 1 en entrée	
IN2	STRING	Chaîne 2 en entrée	
P	INT	Position de début de l'insertion	
sorties			
OUT	STRING	Résultat de l'insertion	

### Description

Cette boîte fonctionnelle insère la totalité des caractères de la chaîne **IN2** dans la chaîne **IN1** après le **P**ième caractère et stocke le résultat dans la chaîne **OUT**.

### Limitation

Si la position **P** de l'insertion est supérieure à la longueur utile de la chaîne **IN1**, la chaîne **OUT** reçoit le résultat de la concaténation de **IN1** avec **IN2**.

### Exemples

IN1	IN2	P	OUT
'ABCDE'	'XX'	3	'ABCXXDE'
'ABCDE'	'XX'	10	'ABCDEXX'

### Exemple ST

```
OUT := INSERT(IN1, IN2, P);
A := INSERT('ABCDE', 'XXX', 3);
```



## 11.7 DELETE

### Suppression de *N* caractères

#### Paramètres

entrées			
IN	STRING	Chaîne en entrée	
L	INT	Nombre de caractères à supprimer	
P	INT	Position de début de la suppression	
sorties			
OUT	STRING	Résultat de la suppression	

#### Description

Cette boîte fonctionnelle supprime les **L** caractères à partir de la position **P** dans la chaîne **IN** et les stocke dans la chaîne **OUT**.

#### Limitation

Si le nombre **L** de caractères à supprimer est supérieur au nombre de caractères restant entre la position **P** et la longueur utile de la chaîne **IN**, la chaîne **OUT** reçoit tous les caractères de la chaîne **IN** à gauche de la position **P**.

Si la position **P** est supérieure à la longueur utile de la chaîne **IN**, la chaîne **OUT** est retournée avec la valeur de **IN**.

#### Exemples

IN	L	P	OUT
'ABCDE'	2	2	'ADE'
'ABCDE'	10	2	'A'
'ABCDE'	3	10	'ABCDE'

#### Exemple ST

```
OUT := DELETE(IN, L, P);  
A := DELETE('ABCDE', 2, 2);
```

## 11.8 REPLACE

*Remplace N caractères*

### Paramètres

entrées			
IN1	STRING	Chaîne 1 en entrée	
IN2	STRING	Chaîne 2 en entrée	
P	INT	Position de début du remplacement	
sorties			
OUT	STRING	Résultat de du remplacement	

### Description

Cette boîte fonctionnelle remplace les **L** caractères à partir de la position **P** de la chaîne **IN1** par le contenu de la chaîne **IN2** et stocke le résultat dans la chaîne **OUT**.

### Limitation

Si le nombre **L** de caractères à remplacer est supérieur au nombre de caractères restant entre la position **P** et la longueur utile de la chaîne **IN1**, la chaîne **OUT** reçoit tous les caractères de la chaîne **IN1** à droite de la position **P** auxquels s'ajoutent tous les caractères de **IN2**.

Si la position **P** du remplacement est supérieure à la longueur utile de la chaîne **IN1**, la chaîne **OUT** reçoit le résultat de la concaténation de **IN1** avec **IN2**.

### Exemples

IN1	IN2	P	L	OUT
'ABCDE'	'XX'	2	2	'ABXXE'
'ABCDE'	'XX'	10	2	'ABCDEXX'
'ABCDE'	'XX'	2	10	'ABXX'

### Exemple ST

```
OUT := REPLACE(IN1, IN2, L, P);
A := REPLACE('ABCDE', 'XX', 10, 3);
```



## 11.9 FIND

*Recherche l'occurrence de N caractères*

### Paramètres

entrées			
IN1	STRING	Chaîne en entrée	
IN2	STRING	Chaîne à rechercher dans chaîne	
sorties			
OUT	INT	Résultat de la recherche	

### Description

Cette boîte fonctionnelle recherche la position du début de la première occurrence de **IN2** dans **IN1** et retourne cette position dans l'entier **OUT**.

Si l'occurrence n'est pas trouvée, la boîte fonctionnelle retourne la valeur 0.

### Exemples

IN1	IN2	OUT
'XX'	'AXBCXXDE'	5
'YY'	'AXBCXXDE'	0

### Exemple ST

```
OUT := FIND(IN1, IN2);  
A := FIND('AXBCXXDE', 'XX');
```



## 12 Opérations de conversion

FB contenues dans cette famille :

opérateur	Libellé
<b>BOOL_TO_SINT</b>	Convertit un booléen en entier court (8)
<b>BOOL_TO_INT</b>	Convertit un booléen en entier normal (16)
<b>BOOL_TO_DINT</b>	Convertit un booléen en entier double (32)
<b>BOOL_TO_BYTE</b>	Convertit un booléen dans un mot de 8 bits (format BCD)
<b>BOOL_TO_WORD</b>	Convertit booléen dans un mot de 16 bits (format BCD)
<b>BOOL_TO_DWORD</b>	Convertit booléen dans un mot de 32 bits (format BCD)
<b>BOOL_TO_REAL</b>	Convertit booléen dans un réel
<b>SINT_TO_BOOL</b>	Convertit un entier court (8) en booléen
<b>SINT_TO_INT</b>	Convertit un entier court (8) en entier normal (16)
<b>SINT_TO_DINT</b>	Convertit un entier court (8) en entier double (32)
<b>SINT_TO_REAL</b>	Convertit un entier court (8) en réel
<b>INT_TO_BOOL</b>	Convertit un entier normal (16) en booléen
<b>INT_TO_SINT</b>	Convertit un entier normal (16) en entier court (8)
<b>INT_TO_DINT</b>	Convertit un entier normal (16) en entier double (32)
<b>INT_TO_REAL</b>	Convertit un entier normal (16) en réel
<b>DINT_TO_BOOL</b>	Convertit un entier double (32) en booléen
<b>DINT_TO_SINT</b>	Convertit un entier double (32) en entier court (8)
<b>DINT_TO_INT</b>	Convertit un entier double (32) en entier normal (16)
<b>DINT_TO_REAL</b>	Convertit un entier double (32) en réel
<b>REAL_TO_BOOL</b>	Convertit un réel en booléen
<b>REAL_TO_SINT</b>	Convertit un réel en entier court (8)
<b>REAL_TO_INT</b>	Convertit un réel en entier normal (16)
<b>REAL_TO_DINT</b>	Convertit un réel en entier double (32)
<b>TRUNC_REAL_TO_SINT</b>	Convertit un real dans un entier simple
<b>TRUNC_REAL_TO_INT</b>	Convertit un real dans un entier normal
<b>TRUNC_REAL_TO_DINT</b>	Convertit un real dans un entier double
<b>BYTE_BCD_TO_SINT</b>	Convertit un mot de 8 bits (format BCD) dans un entier simple
<b>SINT_TO_BCD_BYTE</b>	Convertit un entier simple dans un mot de 8 bits (format BCD)
<b>WORD_BCD_TO_INT</b>	Convertit un mot de 16 bits (format BCD) dans un entier
<b>INT_TO_BCD_WORD</b>	Convertit un entier dans un mot de 16 bits (format BCD)
<b>DWORD_BCD_TO_DINT</b>	Convertit un mot de 32 bits (format BCD) dans un entier double
<b>DINT_TO_BCD_DWORD</b>	Convertit un entier double dans un mot de 32 bits (format BCD)
<b>BYTE_TO_BOOL</b>	Convertit un mot de 8 bits en booléen
<b>BYTE_TO_WORD</b>	Convertit un mot de 8 bits en un mot de 16 bits
<b>BYTE_TO_DWORD</b>	Convertit un mot de 8 bits en un mot de 32 bits
<b>WORD_TO_BOOL</b>	Convertit un mot de 16 bits en booléen
<b>WORD_TO_BYTE</b>	Convertit un mot de 16 bits en un mot de 8 bits
<b>WORD_TO_DWORD</b>	Convertit un mot de 16 bits en un mot de 32 bits
<b>DWORD_TO_BOOL</b>	Convertit un mot de 32 bits en booléen
<b>DWORD_TO_BYTE</b>	Convertit un mot de 32 bits en un mot de 8 bits
<b>DWORD_TO_WORD</b>	Convertit un mot de 32 bits en un mot de 16 bits
<b>BYTE_TO_SINT</b>	Convertit un mot de 8 bits en un entier simple
<b>BYTE_TO_INT</b>	Convertit un mot de 8 bits en un entier normal
<b>BYTE_TO_DINT</b>	Convertit un mot de 8 bits en un entier double
<b>BYTE_TO_REAL</b>	Convertit un mot de 8 bits en un réel
<b>BYTE_TO_STRING</b>	Convertit un mot de 8 bits en une chaîne de caractères
<b>WORD_TO_SINT</b>	Convertit un mot de 16 bits en un entier simple
<b>WORD_TO_INT</b>	Convertit un mot de 16 bits en un entier normal



<b>WORD_TO_DINT</b>	Convertit un mot de 16 bits en un entier double
<b>WORD_TO_REAL</b>	Convertit un mot de 16 bits en un réel
<b>WORD_TO_STRING</b>	Convertit un mot de 16 bits en une chaîne de caractères
<b>DWORD_TO_SINT</b>	Convertit un mot de 32 bits en un entier simple
<b>DWORD_TO_INT</b>	Convertit un mot de 32 bits en un entier normal
<b>DWORD_TO_DINT</b>	Convertit un mot de 32 bits en un entier double
<b>DWORD_TO_REAL</b>	Convertit un mot de 32 bits en un réel
<b>DWORD_TO_STRING</b>	Convertit un mot de 32 bits en une chaîne de caractères
<b>SINT_TO_BYTE</b>	Convertit un entier simple en un mot de 8 bits
<b>SINT_TO_WORD</b>	Convertit un entier simple en un mot de 16 bits
<b>SINT_TO_DWORD</b>	Convertit un entier simple en un mot de 32 bits
<b>SINT_TO_STRING</b>	Convertit un entier simple en une chaîne de caractères
<b>INT_TO_BYTE</b>	Convertit un entier normal en un mot de 8 bits
<b>INT_TO_WORD</b>	Convertit un entier normal en un mot de 16 bits
<b>INT_TO_DWORD</b>	Convertit un entier normal en un mot de 32 bits
<b>INT_TO_STRING</b>	Convertit un entier normal en une chaîne de caractères
<b>DINT_TO_BYTE</b>	Convertit un entier double en un mot de 8 bits
<b>DINT_TO_WORD</b>	Convertit un entier double en un mot de 16 bits
<b>DINT_TO_DWORD</b>	Convertit un entier double en un mot de 32 bits
<b>DINT_TO_STRING</b>	Convertit un entier double en une chaîne de caractères
<b>REAL_TO_BYTE</b>	Convertit un réel en un mot de 8 bits
<b>REAL_TO_WORD</b>	Convertit un réel en un mot de 16 bits
<b>REAL_TO_DWORD</b>	Convertit un réel en un mot de 32 bits
<b>REAL_TO_STRING</b>	Convertit un réel en une chaîne de caractères
<b>STRING_TO_BYTE</b>	Convertit une chaîne de caractères en un mot de 8 bits
<b>STRING_TO_WORD</b>	Convertit une chaîne de caractères en un mot de 16 bits
<b>STRING_TO_DWORD</b>	Convertit une chaîne de caractères en un mot de 32 bits
<b>STRING_TO_SINT</b>	Convertit une chaîne de caractères en un entier simple
<b>STRING_TO_INT</b>	Convertit une chaîne de caractères en un entier normal
<b>STRING_TO_DINT</b>	Convertit une chaîne de caractères en un entier double
<b>STRING_TO_REAL</b>	Convertit une chaîne de caractères en un réel
<b>DT_TO_DATE</b>	Convertit une variable DATE_AND_TIME en DATE
<b>DT_TO_TOD</b>	Convertit une variable DATE_AND_TIME en TIME_OF_DAY



## 12.1 BOOL\_TO\_SINT

*Convertit un booléen en entier court*

---

### Paramètres

entrées			
IN	BOOL	Booléen à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un entier court (8)	

---

### Description

Cette opération convertit le booléen **IN** en un entier court et charge le résultat dans **OUT**. Si **IN** est vrai, **OUT** sera égal à 1, sinon il sera égal à 0.

---

### Exemple ST

```
A := BOOL_TO_SINT(B);
```

---



### 12.2 BOOL\_TO\_INT

*Convertit un booléen en entier normal*

---

#### Paramètres

entrées			
IN	BOOL	Booléen à convertir	
sorties			
OUT	INT	Résultat en sortie dans un entier normal (16)	

---

#### Description

Cette opération convertit le booléen **IN** en un entier normal et charge le résultat dans **OUT**. Si **IN** est vrai, **OUT** sera égal à 1, sinon il sera égal à 0.

---

#### Exemple ST

```
A := BOOL_TO_INT(B);
```

---



## 12.3    **BOOL\_TO\_DINT**

*Convertit un booléen en entier double*

<b>Paramètres</b>			
entrées	IN	BOOL	Booléen à convertir
	sorties		
	OUT	DINT	Résultat en sortie dans un entier double (32)
<b>Description</b>			
Cette opération convertit le booléen <b>IN</b> en un entier double et charge le résultat dans <b>OUT</b> . Si <b>IN</b> est vrai, <b>OUT</b> sera égal à 1, sinon il sera égal à 0.			
<b>Exemple ST</b>			
A := BOOL_TO_DINT(B);			



### 12.4 BOOL\_TO\_BYTE

*Convertit un booléen en un mot de 8 bits*

---

#### Paramètres

entrées			
IN	BOOL	Booléen à convertir	
sorties			
OUT	BYTE	Résultat en sortie dans un mot de 8 bits	

---

#### Description

Cette opération convertit le booléen **IN** en un mot de 8 bits et charge le résultat dans **OUT**. Si **IN** est vrai, **OUT** sera égal à 1, sinon il sera égal à 0.

---

#### Exemple ST

```
A := BOOL_TO_BYTE(B);
```

---



## 12.5 BOOL\_TO\_WORD

*Convertit un booléen en un mot de 16 bits*

---

### Paramètres

entrées			
IN	BOOL	Booléen à convertir	
sorties			
OUT	WORD	Résultat en sortie dans un mot de 16 bits	

---

### Description

Cette opération convertit le booléen **IN** en un mot de 16 bits et charge le résultat dans **OUT**. Si **IN** est vrai, **OUT** sera égal à 1, sinon il sera égal à 0.

---

### Exemple ST

```
A := BOOL_TO_WORD(B);
```

---



### 12.6 BOOL\_TO\_DWORD

*Convertit un booléen en un mot de 32 bits*

---

#### Paramètres

entrées			
IN	BOOL	Booléen à convertir	
sorties			
OUT	DWORD	Résultat en sortie dans un mot de 32 bits	

---

#### Description

Cette opération convertit le booléen **IN** en un mot de 32 bits et charge le résultat dans **OUT**. Si **IN** est vrai, **OUT** sera égal à 1, sinon il sera égal à 0.

---

#### Exemple ST

```
A := BOOL_TO_DWORD(B);
```

---





## 12.7 BOOL\_TO\_REAL

*Convertit un booléen en un réel*

---

### Paramètres

entrées			
IN	BOOL	Booléen à convertir	
sorties			
OUT	REAL	Résultat en sortie dans réel	

---

### Description

Cette opération convertit le booléen **IN** en un réel et charge le résultat dans **OUT**. Si **IN** est vrai, **OUT** sera égal à 1.0, sinon il sera égal à 0.0

---

### Exemple ST

A := BOOL\_TO\_REAL(B);

---



### 12.8 SINT\_TO\_BOOL

*Convertit entier court en booléen*

---

#### Paramètres

entrées			
IN	SINT	Entier court (8) à convertir	
sorties			
OUT	BOOL	Résultat en sortie dans un booléen	

---

#### Description

Cette opération convertit l'entier court **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0, **OUT** sera vrai sinon il sera faux.

---

#### Exemple ST

A := SINT\_TO\_BOOL(B);

---



## 12.9 SINT\_TO\_INT

*Convertit entier court en entier normal*

---

### Paramètres

entrées			
IN	SINT	Entier court (8) à convertir	
sorties			
OUT	INT	Résultat en sortie dans un entier normal (16)	

---

### Description

Cette opération convertit l'entier court **IN** en un entier normal et charge le résultat dans **OUT**.

---

### Exemple ST

A := SINT\_TO\_INT(B);

---



### 12.10 SINT\_TO\_DINT

*Convertit entier court en entier double*

---

#### Paramètres

entrées			
IN	SINT	Entier court (8) à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un entier double (32)	

---

#### Description

Cette opération convertit l'entier court **IN** en un entier double et charge le résultat dans **OUT**.

---

#### Exemple ST

```
A := SINT_TO_DINT(B);
```

---



## 12.11 SINT\_TO\_REAL

*Convertit un entier court en réel*

---

### Paramètres

entrées			
IN	SINT	Entier court (8) à convertir	
sorties			
OUT	REAL	Résultat en sortie dans un réel	

---

### Description

Cette opération convertit l'entier court **IN** en un réel et charge le résultat dans **OUT**.

---

### Exemple ST

```
A := SINT_TO_REAL(B);
```

---



### 12.12 INT\_TO\_BOOL

*Convertit entier normal en booléen*

---

#### Paramètres

entrées			
IN	INT	Entier normal (16) à convertir	
sorties			
OUT	BOOL	Résultat en sortie dans un booléen	

---

#### Description

Cette opération convertit l'entier normal **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0, **OUT** sera vrai sinon il sera faux.

---

#### Exemple ST

```
A := INT_TO_BOOL(B);
```

---



## 12.13 INT\_TO\_SINT

*Convertit entier normal en entier court*

### Paramètres

entrées			
IN	INT	Entier normal (16) à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un entier court (8)	

### Description

Cette opération convertit l'entier normal **IN** en un entier court **OUT**.  
 Les 8 bits de poids faible de **IN** sont chargés dans **OUT**.  
 Les 8 bits de poids fort de **IN** sont perdus.

### Exemples

IN	OUT
0	0
1	1
127	127
128	-128
129	-127
255	-1
256	0
257	1

### Exemple ST

```
A := INT_TO_SINT(B);
```



## 12.14 INT\_TO\_DINT

*Convertit entier normal en entier double*

---

### Paramètres

entrées			
IN	INT	Entier normal (16) à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un entier double (32)	

---

### Description

Cette opération convertit l'entier normal **IN** en un entier double et charge le résultat dans **OUT**.

---

### Exemple ST

```
A := INT_TO_DINT(B);
```

---





## 12.15 INT\_TO\_REAL

*Convertit un entier normal en réel*

---

### Paramètres

entrées			
IN	INT	Entier normal (16) à convertir	
sorties			
OUT	REAL	Résultat en sortie dans un réel	

---

### Description

Cette opération convertit l'entier normal **IN** en un réel et charge le résultat dans **OUT**.

---

### Exemple ST

```
A := INT_TO_REAL(B);
```

---



### 12.16 DINT\_TO\_BOOL

*Convertit entier double en booléen*

---

#### Paramètres

entrées			
IN	DINT	Entier normal (32) à convertir	
sorties			
OUT	BOOL	Résultat en sortie dans un booléen	

---

#### Description

Cette opération convertit l'entier double **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0, **OUT** sera vrai sinon il sera faux.

---

#### Exemple ST

```
A := DINT_TO_BOOL(B);
```

---

## 12.17 DINT\_TO\_SINT

*Convertit entier double en entier court*

### Paramètres

entrées			
IN	DINT	Entier double (32) à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un entier court (8)	

### Description

Cette opération convertit l'entier double **IN** en un entier court et charge le résultat dans **OUT**.

Les 8 bits de poids faible de **IN** sont chargés dans **OUT**.

Les 24 bits de poids fort de **IN** sont perdus.

### Exemples

IN	OUT
0	0
1	1
127	127
128	-128
129	-127
255	-1
256	0
257	1

### Exemple ST

```
A := DINT_TO_SINT(B);
```



### 12.18 DINT\_TO\_INT

*Convertit entier double en entier normal*

#### Paramètres

entrées			
IN	DINT	Entier double (32) à convertir	
sorties			
OUT	INT	Résultat en sortie dans un entier normal (16)	

#### Description

Cette opération convertit l'entier double **IN** en un entier normal et charge le résultat dans **OUT**.

Les 16 bits de poids faible de **IN** sont chargés dans **OUT**.

Les 16 bits de poids fort de **IN** sont perdus.

#### Exemples

IN	OUT
0	0
1	1
32767	32767
32768	-32768
32769	-32767
65535	-1
65536	0
65537	1

#### Exemple ST

```
A := DINT_TO_INT(B);
```



## 12.19 DINT\_TO\_REAL

*Convertit un entier double en réel*

---

### Paramètres

entrées			
IN	DINT	Entier double (32) à convertir	
sorties			
OUT	REAL	Résultat en sortie dans un réel	

---

### Description

Cette opération convertit l'entier double **IN** en un réel et charge le résultat dans **OUT**.

---

### Exemple ST

```
A := DINT_TO_REAL(B);
```

---



### 12.20 REAL\_TO\_BOOL

*Convertit réel en booléen*

---

#### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	BOOL	Résultat en sortie dans un booléen	

---

#### Description

Cette opération convertit le réel **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0.0, **OUT** sera vrai sinon il sera faux.

---

#### Exemple ST

```
A := REAL_TO_BOOL(B);
```

---



## 12.21 REAL\_TO\_SINT

*Convertit un réel en entier 16*

---

### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un entier 16	

---

### Description

Cette opération convertit le réel **IN** en un entier simple et charge le résultat dans **OUT**.

La conversion arrondit à la valeur de l'entier le plus proche :

REAL\_TO\_SINT(1.6) est équivalent à 2

REAL\_TO\_SINT(-1.6) est équivalent à -2

REAL\_TO\_SINT(1.5) est équivalent à 2

REAL\_TO\_SINT(1.4) est équivalent à 1

---

### Limite

Cette opération est correcte dans les limites des bornes de valeurs du SINT.

---

### Exemple ST

```
A := REAL_TO_SINT(B);
```

---



### 12.22 REAL\_TO\_INT

*Convertit un réel en entier 16*

---

#### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	INT	Résultat en sortie dans un entier 16	

---

#### Description

Cette opération convertit le réel **IN** en un entier simple et charge le résultat dans **OUT**.

La conversion arrondit à la valeur de l'entier le plus proche :

REAL\_TO\_INT(1.6) est équivalent à 2

REAL\_TO\_INT(-1.6) est équivalent à -2

REAL\_TO\_INT(1.5) est équivalent à 2

REAL\_TO\_INT(1.4) est équivalent à 1

---

#### Limite

Cette opération est correcte dans les limites des bornes de valeurs du INT.

---

#### Exemple ST

A := REAL\_TO\_INT(B);

---





## 12.23 REAL\_TO\_DINT

*Convertit un réel en entier 32*

---

### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un entier 32	

---

### Description

Cette opération convertit le réel **IN** en un entier double et charge le résultat dans **OUT**.

La conversion arrondit à la valeur de l'entier le plus proche :

REAL\_TO\_DINT(1.6) est équivalent à 2

REAL\_TO\_DINT(-1.6) est équivalent à -2

REAL\_TO\_DINT(1.5) est équivalent à 2

REAL\_TO\_DINT(1.4) est équivalent à 1

---

### Limite

Cette opération est correcte dans les limites des bornes de valeurs du DINT.

---

### Exemple ST

```
A := REAL_TO_DINT(B);
```

---



### 12.24 REAL\_TO\_BYTE

*Convertit un réel en byte*

---

#### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	BYTE	Résultat en sortie dans un byte	

---

#### Description

Cette opération convertit le réel **IN** en un byte et charge le résultat dans **OUT**.

La conversion arrondit à la valeur de l'entier le plus proche :

REAL\_TO\_BYTE(1.6) est équivalent à 2

REAL\_TO\_BYTE(1.5) est équivalent à 2

REAL\_TO\_BYTE(1.4) est équivalent à 1

---

#### Limite

Cette opération est correcte dans les limites des bornes de valeurs du BYTE.

---

#### Exemple ST

A := REAL\_TO\_BYTE(B);

---



## 12.25 REAL\_TO\_WORD

*Convertit un réel en word*

---

### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	WORD	Résultat en sortie dans un word	

---

### Description

Cette opération convertit le réel **IN** en un word et charge le résultat dans **OUT**.

La conversion arrondit à la valeur de l'entier le plus proche :

REAL\_TO\_WORD(1.6) est équivalent à 2

REAL\_TO\_WORD(1.5) est équivalent à 2

REAL\_TO\_WORD(1.4) est équivalent à 1

---

### Limite

Cette opération est correcte dans les limites des bornes de valeurs du WORD.

---

### Exemple ST

A := REAL\_TO\_WORD(B);

---



### 12.26 REAL\_TO\_DWORD

*Convertit un réel en dword*

---

#### Paramètres

entrées			
IN	REAL	Réel à convertir	
sorties			
OUT	DWORD	Résultat en sortie dans un dword	

---

#### Description

Cette opération convertit le réel **IN** en un byte et charge le résultat dans **OUT**.

La conversion arrondit à la valeur de l'entier le plus proche :

REAL\_TO\_DWORD(1.6) est équivalent à 2

REAL\_TO\_DWORD(1.5) est équivalent à 2

REAL\_TO\_DWORD(1.4) est équivalent à 1

---

#### Limite

Cette opération est correcte dans les limites des bornes de valeurs du DWORD.

---

#### Exemple ST

```
A := REAL_TO_DWORD(B);
```

---



## 12.27 TRUNC\_REAL\_TO\_SINT

*Convertit un nombre réel dans un entier 8 bits*

---

### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un entier 16	

---

### Description

Cette opération convertit le réel **IN** en un entier simple et charge le résultat dans **OUT**.  
Les chiffres après la virgule sont perdus après transformation en DINT.  
Les 8 bits de poids faible de **IN** (en équivalence DINT) sont chargés dans **OUT**.  
Les 24 bits de poids fort de **IN** (en équivalence DINT) sont perdus.

---

### Limite

Cette opération est correcte dans les limites des bornes de valeurs du SINT.

---

### Exemple ST

A:= TRUNC\_REAL\_TO\_SINT(B);

---



## 12.28 TRUNC\_REAL\_TO\_INT

*Convertit un nombre réel dans un entier 16 bits*

---

### Paramètres

entrées			
IN	REAL	Réel à convertir	
sorties			
OUT	INT	Résultat en sortie dans un entier 16	

---

### Description

Cette opération convertit le réel **IN** en un entier simple et charge le résultat dans **OUT**.

Les chiffres après la virgule sont perdus après transformation en DINT.

Les 16 bits de poids faible de **IN** (en équivalence DINT) sont chargés dans **OUT**.

Les 16 bits de poids fort de **IN** (en équivalence DINT) sont perdus.

---

### Limite

Cette opération est correcte dans les limites des bornes de valeurs du INT.

---

### Exemple ST

A:= TRUNC\_REAL\_TO\_INT(B);

---



## 12.29 TRUNC\_REAL\_TO\_DINT

*Convertit un nombre réel dans un entier 32 bits*

---

### Paramètres

entrées			
IN	REAL	Réal à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un entier 32	

---

### Description

Cette opération convertit le réel **IN** en un entier double et charge le résultat dans **OUT**.  
Les chiffres après la virgule sont perdus après transformation en DINT.

---

### Limite

Cette opération est correcte dans les limites des bornes de valeurs du DINT.

---

### Exemple ST

```
A:= TRUNC_REAL_TO_DINT(B);
```

---



### 12.30 BYTE\_BCD\_TO\_SINT

*Convertit un mot de 8 bits (codage BCD) dans un entier 8 bits*

---

#### Paramètres

entrées			
IN	BYTE	Mot de 8 bits à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un entier codé sur 8 bits	

---

#### Description

Cette opération convertit le mot **IN** en un entier codé sur 8 bits et place le résultat dans la variable **OUT**.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du SINT.

---

#### Exemple ST

A:= BYTE\_BCD\_TO\_SINT(B);

---





12.31

**SINT\_TO\_BCD\_BYTE**  
*Convertit un entier de 8 bits dans un mot de 8 bits*

<b>Paramètres</b>			
entrées	IN	SINT	Entier codé sur 8 bits à convertir
	sorties	OUT	BYTE
<b>Description</b>			
Cette opération convertit l'entier <b>IN</b> dans un mot de 8 bits et range le résultat dans la variable <b>OUT</b> .			
<b>Limite</b>			
Cette opération est correcte dans la limite des bornes de valeurs du BYTE.			
<b>Exemple ST</b>			
A:= SINT_TO_BCD_BYTE(B);			



## 12.32 WORD\_BCD\_TO\_INT

*Convertit un mot de 16 bits (codage BCD) dans un entier 16 bits*

---

### Paramètres

entrées			
IN	WORD		Mot de 16 bits à convertir
sorties			
OUT	INT		Résultat en sortie dans un entier codé sur 16 bits

---

### Description

Cette opération convertit le mot **IN** en un entier codé sur 16 bits et place le résultat dans la variable **OUT**.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du INT.

---

### Exemple ST

```
A:= WORD_BCD_TO_INT(B);
```

---



## 12.33 INT\_TO\_BCD\_WORD

*Convertit un entier de 16 bits dans un mot de 16 bits*

---

### Paramètres

entrées			
IN	INT	Entier codé sur 16 bits à convertir	
sorties			
OUT	WORD	Résultat en sortie dans un mot de 8 bits	

---

### Description

Cette opération convertit l'entier **IN** dans un mot de 16 bits et range le résultat dans la variable **OUT**.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

### Exemple ST

A:= INT\_TO\_BCD\_WORD(B);

---



### 12.34 DWORD\_BCD\_TO\_DINT

*Convertit un mot de 32 bits (codage BCD) dans un entier 32 bits*

---

#### Paramètres

entrées			
IN	DWORD	Mot de 32 bits à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un entier codé sur 32 bits	

---

#### Description

Cette opération convertit le mot **IN** en un entier codé sur 8 bits et place le résultat dans la variable **OUT**.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DINT.

---

#### Exemple ST

```
A:= DWORD_BCD_TO_DINT(B);
```

---



12.35

**DINT\_TO\_BCD\_DWORD**  
*Convertit un entier de 32 bits dans un mot de 32 bits*

<b>Paramètres</b>	
entrées	
IN	DINT Entier 32 bits à convertir
sorties	
OUT	DWORD Résultat en sortie dans un mot de 32 bits
<b>Description</b>	
Cette opération convertit l'entier <b>IN</b> dans un mot de 32 bits et range le résultat dans la variable <b>OUT</b> .	
<b>Limite</b>	
Cette opération est correcte dans la limite des bornes de valeurs du DWORD.	
<b>Exemple ST</b>	
A:= DINT_TO_BCD_DWORD(B);	



### 12.36 BYTE\_TO\_BOOL

*Convertit un mot de 8 bits en booléen*

---

#### Paramètres

entrées			
IN	BYTE	Mots de 8 bits à convertir	
sorties			
OUT	BOOL	Résultat en sortie dans un booléen	

---

#### Description

Cette opération convertit le mot de 8 bits **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0, **OUT** sera vrai sinon il sera faux.

---

#### Exemple ST

```
A := BYTE_TO_BOOL(B);
```

---



## 12.37 BYTE\_TO\_WORD

*Charge un mot de 8 bits dans un mot de 16 bits*

---

### Paramètres

entrées			
IN	BYTE	Mot de 8 bits à convertir	
sorties			
OUT	WORD	Résultats en sortie dans un mot de 16 bits	

---

### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Aucune.

---

### Exemple ST

A:= BYTE\_TO\_WORD(B);

---



### 12.38 BYTE\_TO\_DWORD

*Charge un mot non signé de 8 bits dans un mot non signé de 32 bits*

---

#### Paramètres

entrées			
IN	BYTE	Mot non signé de 8 bits à convertir	
sorties			
OUT	DWORD	Résultats en sortie dans un mot non signé de 32 bits	

---

#### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

#### Limite

Aucune.

---

#### Exemple ST

A:= BYTE\_TO\_DWORD(B);

---





## 12.39 WORD\_TO\_BOOL

*Convertit un mot de 16 bits en booléen*

---

### Paramètres

entrées			
IN	WORD	Mots de 16 bits à convertir	
sorties			
OUT	BOOL	Résultat en sortie dans un booléen	

---

### Description

Cette opération convertit le mot de 16 bits **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0, **OUT** sera vrai sinon il sera faux.

---

### Exemple ST

```
A := WORD_TO_BOOL(B);
```

---



### 12.40 WORD\_TO\_BYTE

*Charge les 8 bits de poids faible d'un mot de 16 bits dans un mot de 8 bits*

---

#### Paramètres

entrées			
IN	WORD	Mot de 16 bits à convertir	
sorties			
OUT	WORD	Résultat en sortie dans un mot de 8 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot de 8 bits et stocke le résultat dans la variable **OUT**.  
Les 8 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 8 bits de poids fort de **IN** sont perdus.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

#### Exemple ST

```
A:= WORD_TO_BYTE(B);
```

---



## 12.41 WORD\_TO\_DWORD

*Charge un mot de 16 bits dans un mot de 32 bits*

---

### Paramètres

entrées	
IN	WORD Mot de 16 bits à convertir
sorties	
OUT	DWORD Résultat en sortie dans un mot de 32 bits

---

### Description

Cette opération charge les 16 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Aucune.

---

### Exemple ST

A:= WORD\_TO\_DWORD(B);

---



### 12.42 DWORD\_TO\_BOOL

*Convertit un mot de 32 bits en booléen*

---

#### Paramètres

entrées		
IN	DWORD	Mots de 32 bits à convertir
sorties		
OUT	BOOL	Résultat en sortie dans un booléen

---

#### Description

Cette opération convertit le mot de 32 bits **IN** en un booléen et charge le résultat dans **OUT**. Si **IN** est différent de 0, **OUT** sera vrai sinon il sera faux.

---

#### Exemple ST

```
A := DWORD_TO_BOOL(B);
```

---



## 12.43 DWORD\_TO\_BYTE

*Charge les 8 bits de poids faible d'un mot non signé de 32 bits dans un mot non signé de 8 bits*

---

### Paramètres

entrées			
IN	DWORD	Mot non signé de 32 bits à convertir	
sorties			
OUT	BYTE	Résultat en sortie dans un mot non signé de 8 bits	

---

### Description

Cette opération convertit la variable **IN** dans un mot de 8 bits et stocke le résultat dans la variable **OUT**.  
Les 8 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 24 bits de poids fort de **IN** sont perdus.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du BYTE.

---

### Exemple ST

```
A:= DWORD_TO_BYTE(B);
```

---



### 12.44 DWORD\_TO\_WORD

*Charge les 16 bits de poids faible d'un mot de 32 bits dans un mot de 16 bits*

---

#### Paramètres

entrées	
IN	DWORD Mot de 32 bits à convertir
sorties	
OUT	WORD Résultat en sortie dans un mot de 16 bits

---

#### Description

Cette opération convertit la variable **IN** dans un mot de 16 bits et stocke le résultat dans la variable **OUT**.  
Les 16 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 16 bits de poids fort de **IN** sont perdus.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

#### Exemple ST

```
A:= DWORD_TO_WORD(B);
```

---



12.45 **BYTE\_TO\_SINT**

*Charge les 8 bits d'un mot non signé de 8 bits dans un mot signé de 8 bits*

<b>Paramètres</b>			
entrées	IN	BYTE	Mot non signé de 8 bits à convertir
	sorties		
	OUT	SINT	Résultat en sortie dans un mot signé de 8 bits
<b>Description</b>			
Cette opération convertit la variable <b>IN</b> dans un mot signé de 8 bits et stocke le résultat dans la variable <b>OUT</b> .			
<b>Limite</b>			
Cette opération est correcte dans la limite des bornes de valeurs du SINT.			
<b>Exemple ST</b>			
A:= BYTE_TO_SINT(B);			



### 12.46 BYTE\_TO\_INT

*Charge les 8 bits d'un mot non signé de 8 bits dans un mot signé de 16 bits*

---

#### Paramètres

entrées			
IN	BYTE	Mot non signé de 8 bits à convertir	
sorties			
OUT	INT	Résultat en sortie dans un mot signé de 16 bits	

---

#### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

#### Limite

Aucune

---

#### Exemple ST

A:= BYTE\_TO\_INT(B);

---





## 12.47 BYTE\_TO\_DINT

*Charge les 8 bits d'un mot non signé de 8 bits dans un mot signé de 32 bits*

---

### Paramètres

entrées			
IN	BYTE	Mot non signé de 8 bits à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un mot signé de 32 bits	

---

### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Aucune

---

### Exemple ST

A:= BYTE\_TO\_DINT(B);

---



## 12.48 BYTE\_TO\_REAL

*Charge les 8 bits d'un mot non signé de 8 bits dans un réel*

---

### Paramètres

entrées			
IN	BYTE	Mot non signé de 8 bits à convertir	
sorties			
OUT	REAL	Résultat en sortie dans un réel	

---

### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Aucune

---

### Exemple ST

```
A:= BYTE_TO_REAL(B);
```

---



## 12.49 BYTE\_TO\_STRING

*Convertit un mot non signé de 8 bits en une chaîne de caractères*

---

### Paramètres

entrées			
IN	BYTE	Mot non signé de 8 bits à convertir	
sorties			
OUT	STRING	Résultat en sortie dans une chaîne de 6 caractères	

---

### Description

Cette opération convertit la variable **IN** dans une chaîne de 6 caractères et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères renvoyée est la représentation hexadécimale du résultat « 16#xx », où xx est la représentation hexadécimale du mot de 8 bits à convertir.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du BYTE.

---

### Exemple ST

A:= BYTE\_TO\_STRING(B);

---



### 12.50 WORD\_TO\_SINT

*Charge les 8 bits de poids faible d'un mot non signé de 16 bits dans un mot signé de 8 bits*

---

#### Paramètres

entrées			
IN	WORD	Mot non signé de 16 bits à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un mot signé de 8 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot signé de 8 bits et stocke le résultat dans la variable **OUT**.  
Les 8 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 8 bits de poids fort de **IN** sont perdus.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du SINT.

---

#### Exemple ST

```
A:= WORD_TO_SINT(B);
```

---



12.51    **WORD\_TO\_INT**

*Charge les 16 bits d'un mot non signé de 16 bits dans un mot signé de 16 bits*

<b>Paramètres</b>			
entrées	IN	WORD	Mot non signé de 16 bits à convertir
sorties	OUT	INT	Résultat en sortie dans un mot signé de 16 bits
<b>Description</b>			
Cette opération convertit la variable <b>IN</b> dans un mot signé de 16 bits et stocke le résultat dans la variable <b>OUT</b> .			
<b>Limite</b>			
Cette opération est correcte dans la limite des bornes de valeurs du INT.			
<b>Exemple ST</b>			
A:= WORD_TO_INT(B);			



### 12.52 WORD\_TO\_DINT

*Charge les 16 bits d'un mot non signé de 16 bits dans un mot signé de 32 bits*

---

#### Paramètres

entrées			
IN	WORD	Mot non signé de 16 bits à convertir	
sorties			
OUT	DINT	Résultat en sortie dans un mot signé de 32 bits	

---

#### Description

Cette opération charge les 16 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

#### Limite

Aucune

---

#### Exemple ST

A:= WORD\_TO\_DINT(B);

---



## 12.53 WORD\_TO\_REAL

*Charge les 16 bits d'un mot non signé de 16 bits dans un réel*

---

### Paramètres

entrées			
IN	WORD	Mot non signé de 16 bits à convertir	
sorties			
OUT	REAL	Résultat en sortie dans un réel	

---

### Description

Cette opération charge les 16 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Aucune

---

### Exemple ST

A:= WORD\_TO\_REAL(B);

---



### 12.54 WORD\_TO\_STRING

*Convertit un mot non signé de 16 bits en une chaîne de caractères*

---

#### Paramètres

entrées	
IN	WORD Mot non signé de 16 bits à convertir
sorties	
OUT	STRING Résultat en sortie dans une chaîne de 8 caractères

---

#### Description

Cette opération convertit la variable **IN** dans une chaîne de 8 caractères et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères renvoyée est la représentation hexadécimale du résultat « 16#xxxx », où xxxx est la représentation hexadécimale du mot de 16 bits à convertir.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

#### Exemple ST

```
A:= WORD_TO_STRING(B);
```

---





## 12.55 DWORD\_TO\_SINT

*Charge les 8 bits de poids faible d'un mot non signé de 32 bits dans un mot signé de 8 bits*

---

### Paramètres

entrées			
IN	DWORD	Mot non signé de 32 bits à convertir	
sorties			
OUT	SINT	Résultat en sortie dans un mot signé de 8 bits	

---

### Description

Cette opération convertit la variable **IN** dans un mot signé de 8 bits et stocke le résultat dans la variable **OUT**.  
Les 8 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 24 bits de poids fort de **IN** sont perdus.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du SINT.

---

### Exemple ST

A:= DWORD\_TO\_SINT(B);

---



### 12.56 DWORD\_TO\_INT

*Charge les 16 bits d'un mot non signé de 32 bits dans un mot signé de 16 bits*

---

#### Paramètres

entrées			
IN	DWORD	Mot non signé de 32 bits à convertir	
sorties			
OUT	INT	Résultat en sortie dans un mot signé de 16 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot signé de 16 bits et stocke le résultat dans la variable **OUT**.  
Les 16 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 16 bits de poids fort de **IN** sont perdus.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du INT.

---

#### Exemple ST

A:= DWORD\_TO\_INT(B);

---



12.57    **DWORD\_TO\_DINT**

*Charge les 32 bits d'un mot non signé de 32 bits dans un mot signé de 32 bits*

<b>Paramètres</b>	
entrées	
IN	DWORD Mot non signé de 32 bits à convertir
sorties	
OUT	DINT    Résultat en sortie dans un mot signé de 32 bits
<b>Description</b>	
Cette opération convertit la variable <b>IN</b> dans un mot signé de 32 bits et stocke le résultat dans la variable <b>OUT</b> .	
<b>Limite</b>	
Cette opération est correcte dans la limite des bornes de valeurs du DINT.	
<b>Exemple ST</b>	
A:= DWORD_TO_DINT(B);	



### 12.58 DWORD\_TO\_REAL

*Charge les 32 bits d'un mot non signé de 32 bits dans un réel*

---

#### Paramètres

entrées		
IN	DWORD	Mot non signé de 32 bits à convertir
sorties		
OUT	REAL	Résultat en sortie dans un réel

---

#### Description

Cette opération convertit la variable **IN** dans un mot signé de 32 bits et stocke le résultat dans la variable **OUT**.

---

#### Limite

Aucune.

---

#### Exemple ST

A:= DWORD\_TO\_REAL(B);

---



## 12.59 DWORD\_TO\_STRING

*Convertit un mot non signé de 32 bits en une chaîne de caractères*

---

### Paramètres

entrées	
IN	DWORD Mot non signé de 32 bits à convertir
sorties	
OUT	STRING Résultat en sortie dans une chaîne de 12 caractères

---

### Description

Cette opération convertit la variable **IN** dans une chaîne de 12 caractères et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères renvoyée est la représentation hexadécimale du résultat « 16#xxxxxxx », où xxxxxxxx est la représentation hexadécimale du mot de 32 bits à convertir.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DWORD.

---

### Exemple ST

```
A:= DWORD_TO_STRING(B);
```

---



### 12.60 SINT\_TO\_BYTE

*Charge les 8 bits d'un mot signé de 8 bits dans un mot non signé de 8 bits*

---

#### Paramètres

entrées			
IN	SINT	Mot signé de 8 bits à convertir	
sorties			
OUT	BYTE	Résultat en sortie dans un mot non signé de 8 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot non signé de 8 bits et stocke le résultat dans la variable **OUT**.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du BYTE.

---

#### Exemple ST

```
A:= SINT_TO_BYTE(B);
```

---



## 12.61 SINT\_TO\_WORD

*Charge les 8 bits d'un mot signé de 8 bits dans un mot non signé de 16 bits*

---

### Paramètres

entrées			
IN	SINT	Mot signé de 8 bits à convertir	
sorties			
OUT	INT	Résultat en sortie dans un mot non signé de 16 bits	

---

### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

### Exemple ST

```
A:= SINT_TO_WORD(B);
```

---



## 12.62 SINT\_TO\_DWORD

*Charge les 8 bits d'un mot signé de 8 bits dans un mot non signé de 32 bits*

---

### Paramètres

entrées			
IN	SINT	Mot signé de 8 bits à convertir	
sorties			
OUT	DWORD	Résultat en sortie dans un mot non signé de 32 bits	

---

### Description

Cette opération charge les 8 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DWORD.

---

### Exemple ST

```
A:= SINT_TO_DWORD(B);
```

---





## 12.63 SINT\_TO\_STRING

*Convertit un mot signé de 8 bits en une chaîne de caractères*

---

### Paramètres

entrées			
IN	SINT	Mot signé de 8 bits à convertir	
sorties			
OUT	STRING	Résultat en sortie dans une chaîne de 5 caractères	

---

### Description

Cette opération convertit la variable **IN** dans une chaîne de 5 caractères et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères renvoyée est justifiée à gauche dans la gamme '–128' à '127'. Le signe plus est omis. Les zéros de têtes sont supprimés.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du SINT.

---

### Exemple ST

```
A:= SINT_TO_STRING(B);
```

---



### 12.64 INT\_TO\_BYTE

*Charge les 8 bits de poids faible d'un mot signé de 16 bits dans un mot non signé de 8 bits*

---

#### Paramètres

entrées			
IN	INT	Mot signé de 16 bits à convertir	
sorties			
OUT	BYTE	Résultat en sortie dans un mot non signé de 8 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot non signé de 8 bits et stocke le résultat dans la variable **OUT**.  
Les 8 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 8 bits de poids fort de **IN** sont perdus.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du BYTE.

---

#### Exemple ST

```
A:= INT_TO_BYTE(B);
```

---



## 12.65 INT\_TO\_WORD

*Charge les 16 bits d'un mot signé de 16 bits dans un mot non signé de 16 bits*

---

### Paramètres

entrées			
IN	INT		Mot signé de 16 bits à convertir
sorties			
OUT	INT		Résultat en sortie dans un mot non signé de 16 bits

---

### Description

Cette opération convertit la variable **IN** dans un mot non signé de 16 bits et stocke le résultat dans la variable **OUT**.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

### Exemple ST

A:= INT\_TO\_WORD(B);

---



### 12.66 INT\_TO\_DWORD

*Charge les 16 bits de poids faible d'un mot signé de 16 bits dans un mot non signé de 32 bits*

---

#### Paramètres

entrées			
IN	INT		Mot signé de 16 bits à convertir
sorties			
OUT	DWORD		Résultat en sortie dans un mot non signé de 32 bits

---

#### Description

Cette opération charge les 16 bits de la variable **IN** et les range dans la zone de poids faible de la variable **OUT**.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DWORD.

---

#### Exemple ST

```
A:= INT_TO_DWORD(B);
```

---



12.67 INT\_TO\_STRING

Convertit un mot signé de 16 bits en une chaîne de caractères

Paramètres	
entrées	
IN	INT Mot signé de 16 bits à convertir
sorties	
OUT	STRING Résultat en sortie dans une chaîne de 7 caractères
Description	
Cette opération convertit la variable <b>IN</b> dans une chaîne de 7 caractères et stocke le résultat dans la variable <b>OUT</b> . La chaîne de caractères renvoyée est justifiée à gauche dans la gamme '–32 768' à '32 767'. Le signe plus est omis. Les zéros de têtes sont supprimés.	
Limite	
Cette opération est correcte dans la limite des bornes de valeurs du INT.	
Exemple ST	
A:= INT_TO_STRING(B);	



### 12.68 DINT\_TO\_BYTE

*Charge les 8 bits de poids faible d'un mot signé de 32 bits dans un mot non signé de 8 bits*

---

#### Paramètres

entrées			
IN	INT	Mot signé de 32 bits à convertir	
sorties			
OUT	BYTE	Résultat en sortie dans un mot non signé de 8 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot non signé de 8 bits et stocke le résultat dans la variable **OUT**.  
Les 8 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 24 bits de poids fort de **IN** sont perdus.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du BYTE.

---

#### Exemple ST

```
A:= DINT_TO_BYTE(B);
```

---



## 12.69 DINT\_TO\_WORD

*Charge les 16 bits de poids faible d'un mot signé de 32 bits dans un mot non signé de 16 bits*

---

### Paramètres

entrées			
IN	DINT	Mot signé de 32 bits à convertir	
sorties			
OUT	INT	Résultat en sortie dans un mot non signé de 16 bits	

---

### Description

Cette opération convertit la variable **IN** dans un mot non signé de 16 bits et stocke le résultat dans la variable **OUT**.  
Les 16 bits de poids faible de **IN** sont placés dans **OUT**.  
Les 16 bits de poids fort de **IN** sont perdus.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

### Exemple ST

```
A:= DINT_TO_WORD(B);
```

---



### 12.70 DINT\_TO\_DWORD

*Charge les 32 bits d'un mot signé de 32 bits dans un mot non signé de 32 bits*

---

#### Paramètres

entrées			
IN	DINT	Mot signé de 32 bits à convertir	
sorties			
OUT	DWORD	Résultat en sortie dans un mot non signé de 32 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot non signé de 32 bits et stocke le résultat dans la variable **OUT**.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DWORD.

---

#### Exemple ST

```
A:= DINT_TO_DWORD(B);
```

---





## 12.71 DINT\_TO\_STRING

*Convertit un mot signé de 32 bits en une chaîne de caractères*

---

### Paramètres

entrées			
IN	DINT	Mot signé de 32 bits à convertir	
sorties			
OUT	STRING	Résultat en sortie dans une chaîne de 12 caractères	

---

### Description

Cette opération convertit la variable **IN** dans une chaîne de 12 caractères et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères renvoyée est justifiée à gauche dans la gamme '-2 147 483 648' à '2 147 483 647'. Le signe plus est omis.  
Les zéros de têtes sont supprimés.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DINT.

---

### Exemple ST

```
A:= DINT_TO_STRING(B);
```

---



### 12.72 REAL\_TO\_STRING

*Convertit un réel en une chaîne de caractères*

---

#### Paramètres

entrées			
IN	DINT	Réel à convertir	
sorties			
OUT	STRING	Résultat en sortie dans une chaîne de 11 caractères	

---

#### Description

Cette opération convertit la variable **IN** dans une chaîne de 11 caractères et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères renvoyée est justifiée à gauche. Le signe plus est omis. Les zéros de têtes sont supprimés. La notation scientifique est utilisée pour les grandes valeurs.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du REAL.

---

#### Exemple ST

```
A:= REAL_TO_STRING(B);
```

---



## 12.73 STRING\_TO\_BYTE

*Convertit une chaîne de caractères en un mot non signé de 8 bits*

---

### Paramètres

entrées			
IN	STRING	Chaîne de 6 caractères	
sorties			
OUT	BYTE	Résultat en sortie dans un mot non signé de 8 bits	

---

### Description

Cette opération convertit la variable **IN** dans un mot non signé de 8 bits et stocke le résultat dans la variable **OUT**.

La chaîne de caractères en format hexadécimal attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis. La chaîne de caractères en format hexadécimal doit être de la forme « 16#xx », où xx est la représentation hexadécimale du mot de 8 bits à obtenir.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du BYTE.

---

### Exemple ST

```
A:= STRING_TO_BYTE(B);
```

---



### 12.74 STRING\_TO\_WORD

*Convertit une chaîne de caractères en un mot non signé de 16 bits*

---

#### Paramètres

entrées			
IN	STRING	Chaîne de 8 caractères	
sorties			
OUT	WORD	Résultat en sortie dans un mot non signé de 16 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot non signé de 16 bits et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères en format hexadécimal attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis. La chaîne de caractères en format hexadécimal doit être de la forme « 16#xxxx », où xxxx est la représentation hexadécimale du mot de 16 bits à obtenir.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du WORD.

---

#### Exemple ST

```
A:= STRING_TO_WORD(B);
```

---



## 12.75 STRING\_TO\_DWORD

*Convertit une chaîne de caractères en un mot non signé de 32 bits*

---

### Paramètres

entrées			
IN	STRING	Chaîne de 12 caractères	
sorties			
OUT	WORD	Résultat en sortie dans un mot non signé de 32 bits	

---

### Description

Cette opération convertit la variable **IN** dans un mot non signé de 32 bits et stocke le résultat dans la variable **OUT**.

La chaîne de caractères en format hexadécimal attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis. La chaîne de caractères en format hexadécimal doit être de la forme « 16#xxxxxxx », où xxxxxxxx est la représentation hexadécimale du mot de 32 bits à obtenir.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DWORD.

---

### Exemple ST

```
A:= STRING_TO_DWORD(B);
```

---



### 12.76 STRING\_TO\_SINT

*Convertit une chaîne de caractères en un mot signé de 8 bits*

---

#### Paramètres

entrées			
IN	STRING	Chaîne de 4 caractères	
sorties			
OUT	BYTE	Résultat en sortie dans un mot signé de 8 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot signé de 8 bits et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères en base décimale attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du SINT.

---

#### Exemple ST

```
A:= STRING_TO_SINT(B);
```

---



## 12.77 STRING\_TO\_INT

*Convertit une chaîne de caractères en un mot signé de 16 bits*

---

### Paramètres

entrées			
IN	STRING	Chaîne de 6 caractères	
sorties			
OUT	WORD	Résultat en sortie dans un mot signé de 16 bits	

---

### Description

Cette opération convertit la variable **IN** dans un mot signé de 16 bits et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères en base décimale attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis.

---

### Limite

Cette opération est correcte dans la limite des bornes de valeurs du INT.

---

### Exemple ST

```
A:= STRING_TO_INT(B);
```

---



### 12.78 STRING\_TO\_DINT

*Convertit une chaîne de caractères en un mot signé de 32 bits*

---

#### Paramètres

entrées			
IN	STRING	Chaîne de 11 caractères	
sorties			
OUT	WORD	Résultat en sortie dans un mot signé de 32 bits	

---

#### Description

Cette opération convertit la variable **IN** dans un mot signé de 32 bits et stocke le résultat dans la variable **OUT**.  
La chaîne de caractères en base décimale attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis.

---

#### Limite

Cette opération est correcte dans la limite des bornes de valeurs du DINT.

---

#### Exemple ST

```
A:= STRING_TO_DINT(B);
```

---





12.79

STRING\_TO\_REAL

Convertit une chaîne de caractères en un réel

Paramètres	
entrées	
IN	STRING Chaîne de 15 caractères
sorties	
OUT	WORD Résultat en sortie dans un réel
Description	<p>Cette opération convertit la variable <b>IN</b> dans un réel et stocke le résultat dans la variable <b>OUT</b>.</p> <p>La chaîne de caractères en base décimale attendue est justifiée à gauche. Le signe plus et les zéros de têtes peuvent être omis.</p>
Limite	<p>Cette opération est correcte dans la limite des bornes de valeurs du REAL.</p>
Exemple ST	<pre>A:= STRING_TO_REAL(B);</pre>



### 12.80 DT\_TO\_DATE

*Extrait la date d'une variable de type DATE\_AND\_TIME dans une variable de type DATE*

---

#### Paramètres

entrées					
IN	DT	Variable	de	type	DATE_AND_TIME à
convertir					
sorties					
OUT	DATE	Résultat	en	sortie	dans une variable de type
DATE					

---

#### Description

Cette opération convertit la variable **IN** de type DATE\_AND\_TIME en type DATE et place le résultats dans la variable **OUT**.

---

#### Exemple ST

A:= DT\_TO\_DATE(B);

---



12.81 DT\_TO\_TOD

Extrait l'heure d'une variable de type DATE\_AND\_TIME dans une variable de type TOD

Paramètres			
entrées	IN	DT	Variable de type DATE_AND_TIME à convertir
sorties	OUT	TOD	Résultat en sortie dans une variable de type TOD
Description			
Cette opération convertit la variable <b>IN</b> de type DATE_AND_TIME en type TOD et place le résultats dans la variable <b>OUT</b> .			
Exemple ST			
A:= DT_TO_TOD(B);			