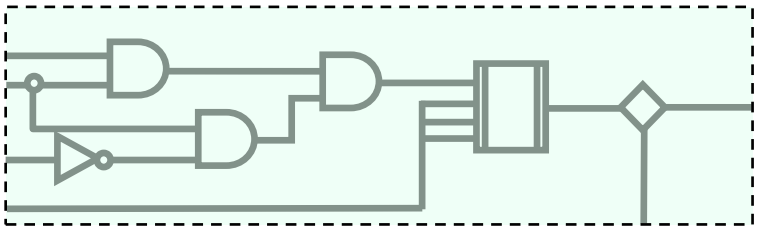
Zennio
Passion for KNX innovation



Fonctions logiques

Module de fonctions logiques-mathématiques

MANUEL DE L'UTILISATEUR

Édition du manuel: [0.3]_a

www.zennio.fr

Sommaire

ACTUALISATIONS DU DOCUMENT

Version	Modifications	Page(s)
[0.3]_a	Changements dans le programme: • Retard interne entre les envois simultanés. • Annulation d'envois périodiques ou retards des résultats antérieurs dans le cas de s'exécuter nouvellement la fonction.	-
	Annulation d'envois périodiques ou retards des résultats antérieurs dans le cas de s'exécuter nouvellement la fonction.	18, 19
	Correction des errata dans la table de conversions.	24
[0.2]_b	Note par rapport aux appels successifs à des fonctions avec résultats retardés.	18
	Les conversions à des valeurs de pourcentage d'un byte sont équivalentes aux conversions à des valeurs entiers d'un byte (c'est pour cela qu'une option spécifique ne se propose pas).	23, 24

1 INTRODUCTION

1.1 MODULE DE FONCTIONS LOGIQUES

Beaucoup des dispositifs Zennio (entre eux, toute la famille d'actionneurs ACTinBOX et MAXinBOX) incorporent un module de fonctions logiques, ce qui permet d'effectuer des opérations **mathématiques** ou en **logique binaire** avec des données en provenance du bus KNX ainsi que envoyer les résultats au moyen d'objets de communication de différentes tailles.

Les opérandes de ces fonctions peuvent être des types qui suivent:

- **Objets de communication** reçus à travers du bus KNX.
- **Variables Internes** avec des résultats partiels d'opérations précédentes.
- **Valeurs constants**, établis par paramètre sur ETS.

Il est possible de définir dix fonctions logiques différentes et indépendantes, chacune pourra en plus avoir jusqu'à quatre opérations successives, qui pourront partager des variables entre elles de façon que le résultat d'une soit l'entrée de la suivante.

Important: *Selon chaque dispositif, le nombre de fonctions logiques disponibles et la fonctionnalité changent légèrement, c'est pour ça que le manuel d'utilisateur de ce module a été personnalisé pour chaque dispositif. C'est pour cela, qu'il est recommandé d'utiliser les liens de téléchargement qui figurent sur la fiche du dispositif en particulier que vous voulez paramétrer, dans le site web de Zennio (www.zennio.com).*

2 CONFIGURATION

2.1 FONCTIONNEMENT GÉNÉRAL

Le module de fonctions logiques permet d'habiliter et paramétrer jusqu'à dix fonctions numériques indépendantes. Le fonctionnement de chacune de ces fonctions est divisée en quatre étapes:

- **Appel**: le premier pas pour que la fonction paramétrée s'exécute consiste à *l'appeler*. Pour ceci, il est possible de paramétrer un ou plusieurs objets de communication, de façon que chaque fois qu'un d'eux actualise sa valeur depuis le bus, se déclenchera automatiquement l'exécution de la fonction (toujours et quand la condition d'exécution s'accomplie).

- **Condition d'exécution**: après avoir activé un déclencheur assigné à une fonction logique il sera vérifié si la condition d'exécution s'accomplisse. Cette condition consiste à comparer les deux opérandes d'entrée. Dans le cas où la comparaison soit vrai, les opération seront exécutées. Si aucune condition d'exécution est paramétrée, la fonction sera exécutée à chaque appel.

- **Opérations**: le déclencheur de la fonction déchaînera, à la fois, l'exécution successive de jusqu'à quatre opérations mathématiques ou binaires. Pour chacune d'entre elles nous devrons paramétrer ce qui suit:

  - **Type d'opération**: action désirée (addition, soustraction, négation, etc.).

  - **Opérandes**: valeurs avec lesquels se fera l'opération. Pourront être des objets de communication d'entrée, variables internes où le résultat d'une opération précédente dont nous avons stocké le résultat, ou des constantes prédéfinies sur ETS.

  - **Résultat**: Variable interne où stocker le résultat de l'opération.

- **Résultat**: vous devrez sélectionner laquelle des variables internes contient le résultat global de la fonction. Sa valeur sera envoyée, à la fin de l'exécution de toutes les opérations, à travers de l'objet de communication qui correspond.

Sur la Figure 1 se montre un diagramme du fonctionnement général des fonctions logiques.

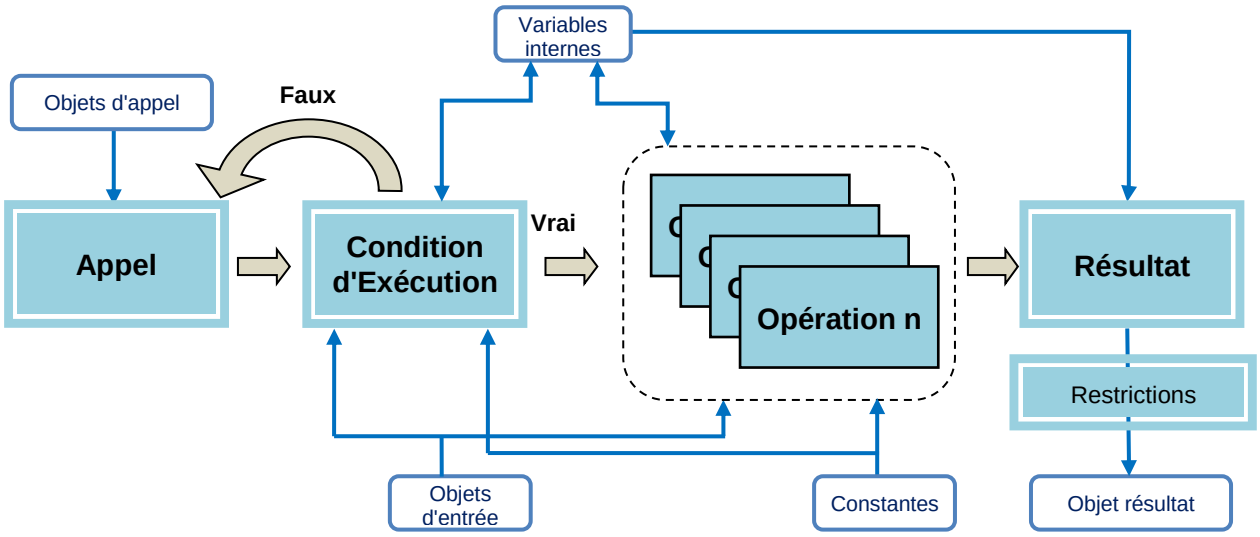


Figure 1. Diagramme de fonctionnement des fonctions logiques.

2.2 DÉCLENCHEURS

Pour chaque fonction, vous disposerez de jusqu'à huit objets d'appel (d'un bit, un byte deux bytes ou quatre bytes), chacun déclenchera la fonction automatiquement chaque fois qu'il recevra une valeur depuis le bus. Ces objets ne doivent pas forcement coïncider avec les objets utilisés comme opérandes.

Notez que l'exécution des opérations une fois déclenché la fonction dépendra du résultat d'évaluer la condition d'exécution, s'il y est.

2.3 CONDITION D'EXÉCUTION

Une fois que le déclencheur est activé, le pas suivant est de vérifier si la situation actuelle accomplie les conditions nécessaires pour que le calcul des opérations se réalise.

Si aucune condition d'exécution a été paramétrée, les opérations s'effectueront directement. Par contre, si elle est habilitée, les opérations se feront seulement si la comparaison de **deux opérandes d'entrée** s'accomplisse, qui peuvent être des valeurs constantes, objets de communication ou variables internes.

Les opérations de comparaison disponibles sont (en fonction de la taille de comparaison choisie): égal à; non égal à; plus grand que; plus grand ou égal que; plus petit que; plus petit ou égal que.

2.4 OPÉRATIONS

Comme il a déjà été expliqué, chaque fonction logique consiste à l'exécution de jusqu'à quatre opérations consécutives qui peuvent être des types qui suivent:

- **Logique**: ID, NOT, AND, OR, XOR, NAND, NOR y NXOR.
- **Arithmétique**: Oui, addition, soustraction, multiplication, division, maximum et minimum.
- **Comparaison**: plus grand, plus grand ou égal, plus petit, plus petit ou égal, égal, inégal.
- **Conversion**: opérations de conversion de la taille (*cast*) d'un opérande en particulier. Par exemple: d'un bit à un byte.

Le module de fonctions logiques peut opérer avec les rangs de valeurs (que ce soit des objets, variables internes avec résultats intermédiaires préalables, ou constantes établies para paramètre sur ETS):

- Binaires: **0** et **1**.
- Entiers sans signe (un byte): **0 – 255**.
- Valeurs de pourcentage () **0 – 100.**
- Entiers sans signe (deux bytes): **0 – 65535**.
- Entiers avec signe (deux bytes): **-32768 – 32767**.
- Valeurs en virgule flotante (deux bytes): **-671088,64 – 671760,96**.
- Entiers avec signe (quatre bytes): **-2147483648 – 2147483647**.

Pour plus d'information sur ces opérations, sur la conversion entre tailles et sur le tronqué de valeurs consultez l'*ANNEXE I: Liste d'opÉrations*.

2.5 OBJETS D'ENTRÉE

De multiples objets spécifiques pourront être activés pour les utiliser avec les fonctions:

- Jusqu'à 32 objets d'un bit,
- Jusqu'à 16 objets d'un byte,
- Jusqu'à 16 objets de deux bytes.
- Jusqu'à 8 objets de quatre bytes.

Les valeurs de ces objets pourront intervenir, par exemple, comme opérandes dans les opérations des fonctions qui sont habilitées.

2.6 VARIABLES INTERNES

De la même façon, l'intégrateur pourra disposer de:

- 32 variables internes d'un bit (b1, …, b32),
- 16 variables internes d'un byte (n1, …, n16),
- 16 variables internes de deux bytes (x1, …, x8).
- 8 variables internes de quatre bytes (y1, …, y8).

Toutes celles-ci permettront le stockage temporel des résultats intermédiaires, qui à son tour pourront s'employer comme valeurs d'entrée d'opérations suivantes.

2.7 OBJET DE RÉSULTAT

Chaque fonction logique habilité disposera d'un objet spécifique (qui sera de un bit, un byte, deux bytes ou quatre bytes, en fonction de la configuration de la fonction) à travers de laquelle s'enverra au bus la valeur de la variable interne qui se sélectionne par paramètre comme résultat de la séquence d'opérations qui conforment la fonction.

L'intégrateur pourra configurer si l'envoi de cet objet au bus s'effectue une fois chaque que la fonction s'exécute, ou bien périodiquement, ou bien seulement lorsque la fonction apporte un résultat différent de celui de la précédent exécution.

Du même mode, pourront se restreindre les résultats qui s'envoient, de tel mode que seulement se notifie le résultat au bus lorsqu'il se trouve dans une échelle déterminée. Et pour finir, il est aussi possible de configurer sur ETS un retard pour l'envoi du résultat.

3 PARAMÉTRAGE ETS

3.1 ÉCRAN GÉNÉRAL

L'onglet général des paramètres du module de fonctions logiques contient les options qui se montrent ci-dessous. Tenez en compte que cette figure se réfère au dispositif MAXinBOX 66 et qu'il peut y avoir de légères différences entre d'autres dispositifs.

Note: *cela peut-être parce que les onglets de paramètres du module de fonctions logiques ne se montrent pas par défaut sur ETS et qu'il soit nécessaire d'habiliter ce module depuis l'onglet Général du propre dispositif.* S'il vous plaît, consultez le manuel de l'utilisateur du dispositif spécifique pour plus de détails.

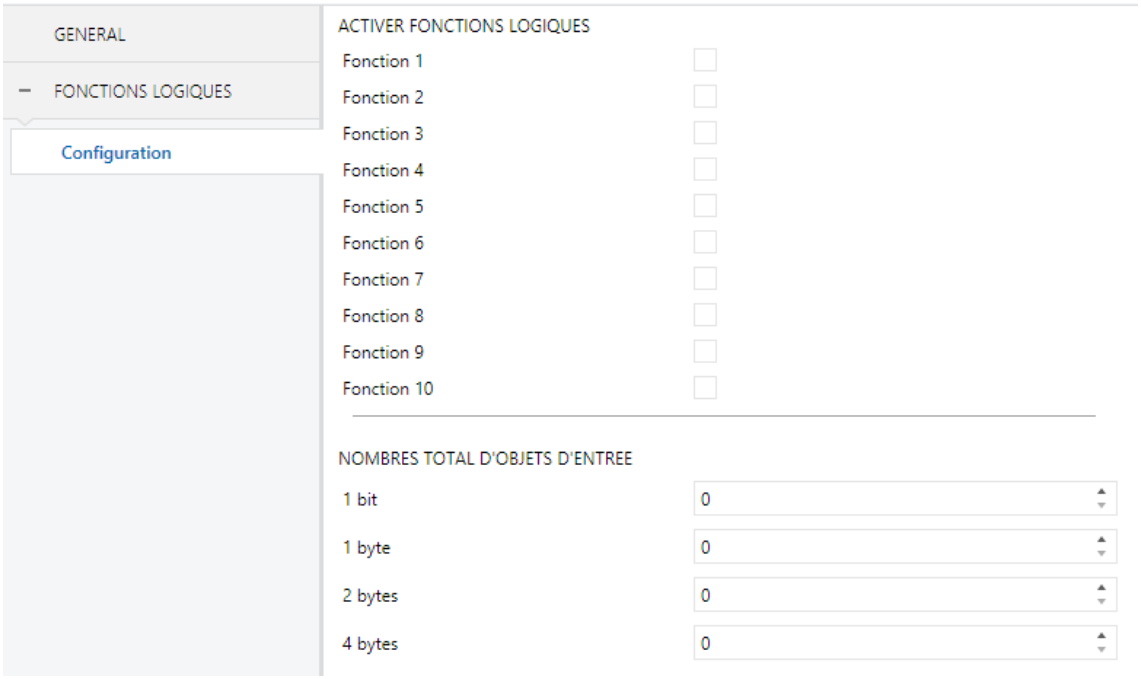


Figure 2. Onglet générale du module de fonctions logiques:

Comme on peut le voir, aucune des dix fonctions apparaît active par défaut. À mesure que s'habilitent, viendront apparaître des onglets additionnels dans la liste sur la gauche.

De plus, dans cet onglet se détermine le nombre d'objets d'entrées nécessaires de chaque type pour l'ensemble de toutes les fonctions logiques qui vont être utilisées. Comme déjà vu dans la section 2.5, pourront s'habiliter jusqu'à 32 objets de un bit, 16 de un byte, 16 de deux bytes ou 8 de quatre bytes.

Dans les suivantes sections s'explique la finalité de chaque onglet et des paramètres qu'il contient.

3.2 FONCTION N

Pour chaque fonction habilité (voir 3.1) se montrera dans la liste de la gauche un onglet spécifique, qui à leur tour se divise en quatre autres.

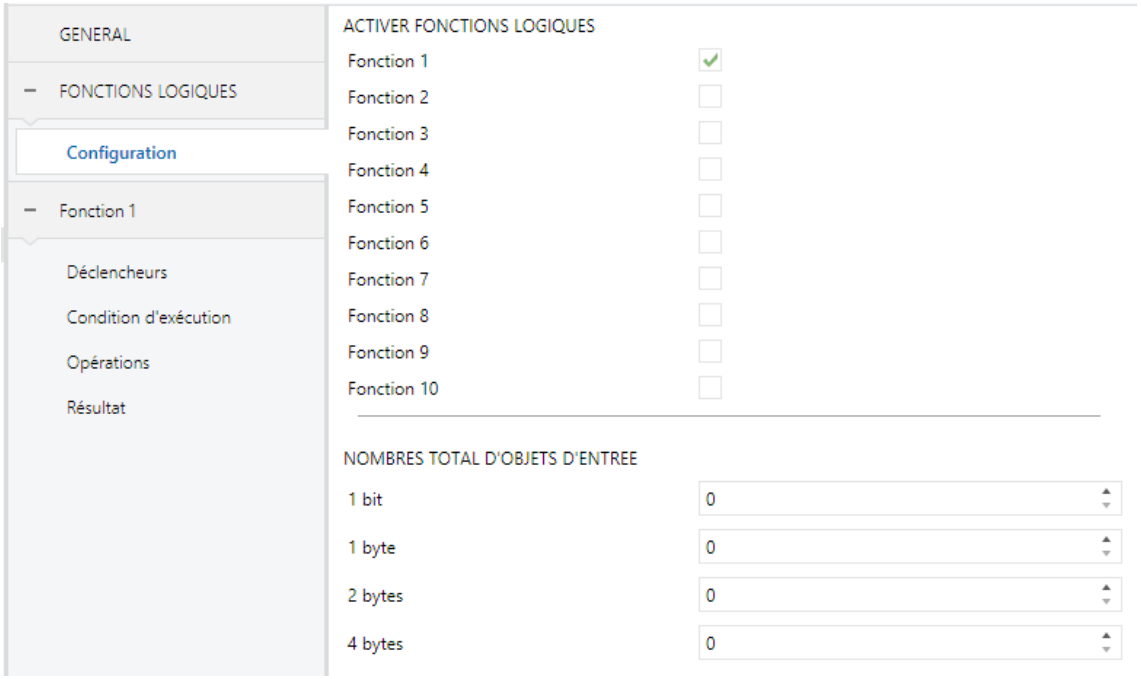


Figure 3. Fonction 1 habilité.

3.2.1 DÉCLENCHEURS

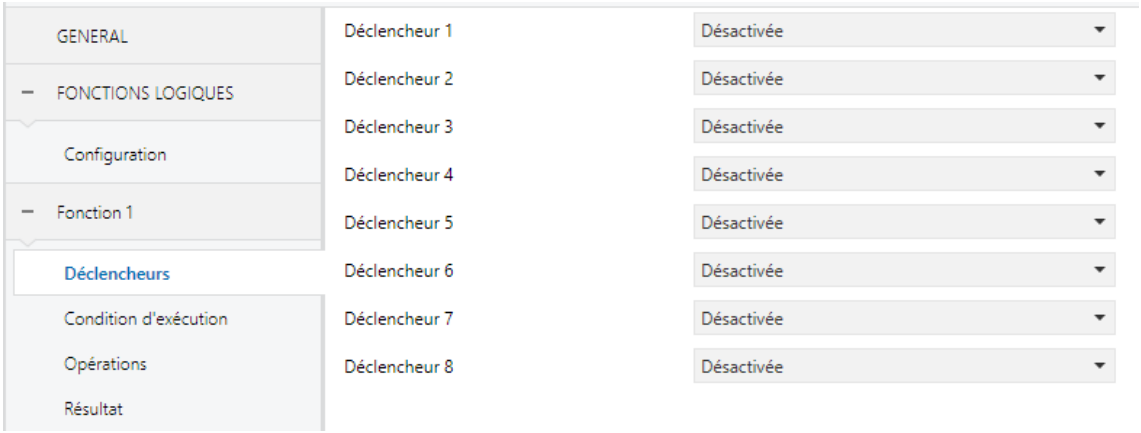


Figure 4. Onglet "Déclencheurs"

Depuis cette section on pourra sélectionner jusqu'à **huit déclencheurs** qui agiront comme objets d'appel. Les objets d'appel seront les responsables de déclencher l'exécution de la fonction chaque fois qu'une d'entre elles reçoit quelque valeur du bus, toujours et lorsque se remplie la condition d'exécution. Également, si s'utilise un même objet comme objet d'appel de plusieurs fonctions, toutes se déclencheront consécutivement lorsque s'écrit quelque valeur dans l'objet.

Note: Les objets que l'on désire utiliser comme déclencheurs devront avoir été habilité auparavant (voir *3.1*).

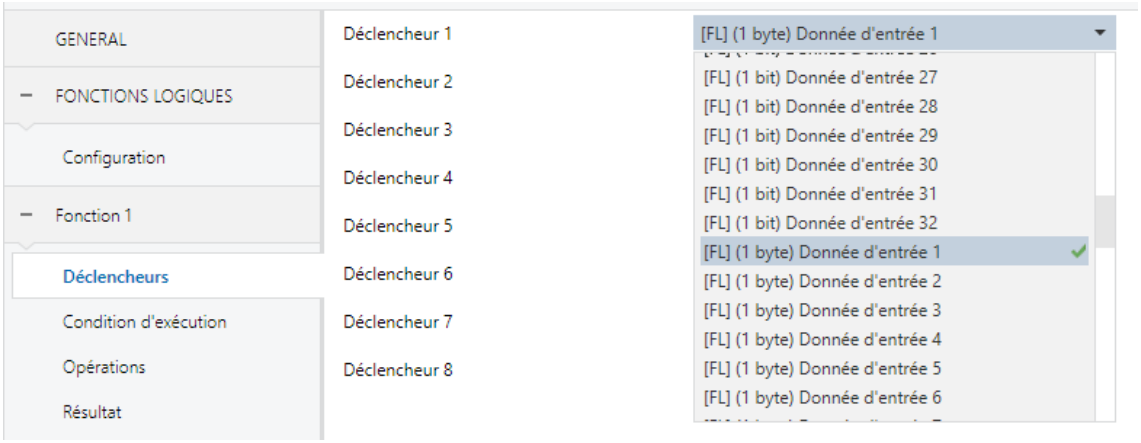


Figure 5. Sélection des objets déclencheurs

3.2.2 CONDITION D'EXÉCUTION

Depuis cette section on peut configurer la condition d'exécution au moyen des paramètres suivants:

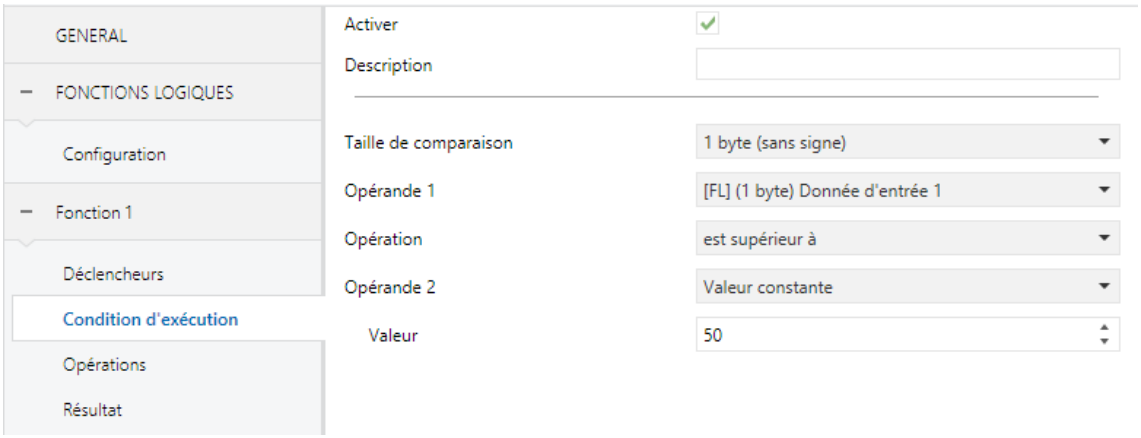


Figure 6. Onglet "Condition d'exécution"

- **Habiliter:** établie si on désire ou non configurer une condition d'exécution.
- **Description:** permet de spécifier une brève description (jusqu'à cent caractères). Ce champ ne possède aucune implication pratique; simplement facilite à l'intégrateur l'identification de la fonction.
- **Taille de comparaison:** taille des opérandes d'entrée: 1 bit, 1 byte (sans signe), 1 byte (pourcentage), 2 bytes (sans signe), 2 bytes (avec signe), 2 bytes (flottante), 4 bytes (avec signe).
- **Opérande 1:** Origine de la valeur: constante, objet ou variable interne. Dans le cas de sélectionner "Valeur constante", se montrera un nouveau champ ("**Valeur**") au moyen de laquelle spécifier la valeur constante désirée.
- **Opération:** la liste dépendra du champ **Taille de comparaison.** D'être de un bit, se montreront seulement les options "est égal à" et "n'est pas égal à". Si elle est supérieure à 1 bit, se montreront toutes les possibilités (voir chapitre 2.4).
- **Opérande 2:** Origine de la valeur: constante, objet ou variable interne. Dans le cas de sélectionner "Valeur constante", se montrera un nouveau champ ("**Valeur**") au moyen de laquelle spécifier la valeur constante désirée.

3.2.3 OPÉRATIONS

Le nombre maximum d'opérations par fonction logique est de quatre. Celles-ci s'activent individuellement et s'exécuteront de forme séquentielle. Si il y a quelque opération désactivée au milieu, elle sera ignorée.

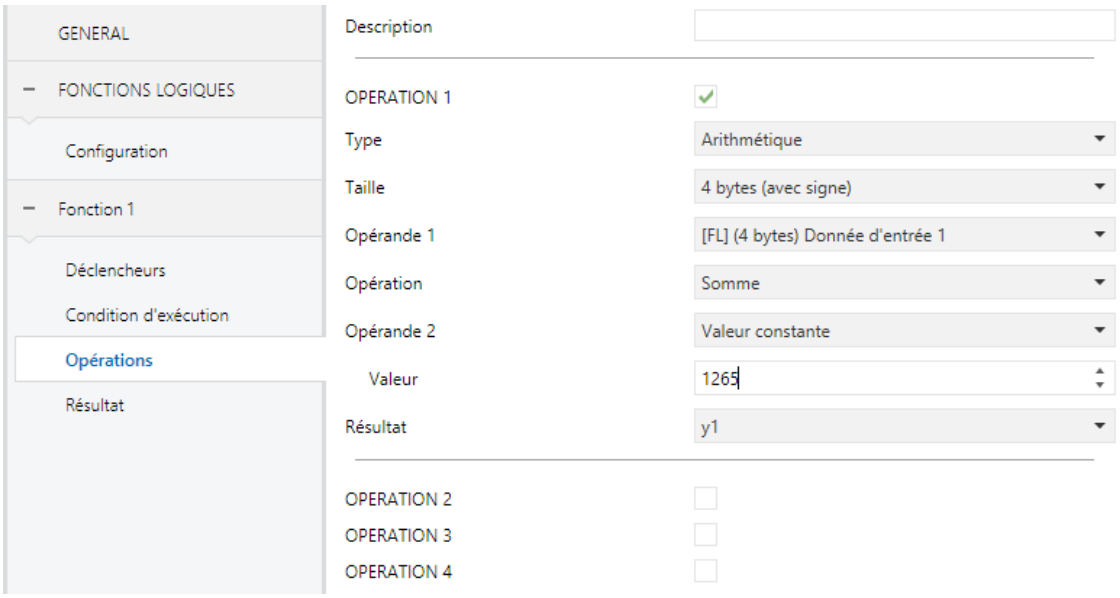


Figure 7. Onglet "Opérations".

Dans chaque opération, il est possible de configurer les paramètres suivants:

- **Description:** permet de spécifier une brève description (jusqu'à cent caractères) de la fonction logique. Ce champ ne possède aucune implication pratique; simplement facilite à l'intégrateur l'identification de la fonction.

- **Opération "i"**: permet d'habiliter ou déshabiliter l'opération "i" (1-4). Chaque opération habilité montrera à son tour les paramètres suivants:

  - **Type:** permet de sélectionner le type d'opération (logique, arithmétique, de comparaison ou de conversion). Sauf si se sélectionne "logique", se montrera un paramètre additionnel ("**Taille**") pour la sélection de la taille du résultat: 1 bit, 1 byte (sans signe), 1 byte (pourcentage), 2 bytes (sans signe), 2 bytes (avec signe), 2 bytes (flottante) et 4 bytes (avec signe).

  - **Opération**: permet de sélectionner l'action qu'exécutera l'opération "i". En fonction du type d'opération sélectionnée (logique, arithmétique, de comparaison ou de conversion), ce paramètre montrera des options ou autres. Pour plus d'information, voir l'*ANNEXE I: Liste* d'opÉrations.

- **Opérande "j"**: suivant l'option sélectionnée dans le paramètre précédent, s'habiliteront un ou plusieurs paramètres de nom "Opérandes" de l'opération. Ceux-ci peuvent être objets de communication, variables internes ou valeurs constantes. Regardez la section 2.4 et les suivantes.

- **Résultat de l'opération**: permet de sélectionner la variable interne dans laquelle se gardera le résultat de l'opération. Ce résultat intermédiaire pourra être utilisé comme opérande d'entrée dans les opérations successives ou comme résultat final de la fonction, si se désire.

Note: *les variables internes sont communes pour toutes les fonctions logiques.* Par exemple, si la fonction 1 garde un résultat intermédiaire dans la variable interne "n1" et ensuite la fonction "2" utilise cette variable interne comme donnée d'entrée, la valeur lue sera celle qu'a écrite la fonction 1.

3.2.4 RÉSULTAT

Depuis cette section on pourra sélectionner quelle variable interne est celle qui détermine le résultat final de la fonction, de tel mode qu'après exécuter les opérations qui conforment la fonction on pourra consulter la valeur de cette variable et l'envoyer au bus à travers de l'objet "**[FL] Fonction n - Résultat**".

Figure 8. Onglet "Résultat".

- **Taille:** établie la taille du résultat de la fonction. Les options sont: 1 bit, 1 byte (sans signe), 1 byte (pourcentage), 2 bytes (sans signe), 2 bytes (avec signe), 2 bytes (flottante) ou 4 bytes (avec signe).

- **Valeur:** détermine la variable interne dont la valeur s'enverra au bus, à travers de l'objet du résultat de la fonction, à la fin des opérations.

- **Condition d'envoi:** établie les restrictions sur les résultats qui s'envoient au bus, de tel mode que seulement s'envoient celles qui remplissent cette restriction. Les options disponibles sont:

  - **Pour résultat de taille de 1 bit:**

    - Sans Restrictions,
    - Est égal à,
    - N'est pas égal à,

  - **Pour résultats d'autres tailles (1 byte, 2 bytes, 4 bytes):**

    - Sans Restrictions,
    - Est égal à,
    - N'est pas égal à,
    - Est supérieur à,
    - Est supérieur ou égal à,
    - Est inférieur à,
    - Est inférieur ou égal à,

    Lorsque se choisie une restriction, il faudra aussi remplir le paramètre "**Valeur"**. Le rang de ce paramètre dépend de la Taille choisie (voir section 2.4).

- **Mode d'envoi**: dans ce champ se définie sous quelle condition s'enverra le résultat au bus KNX, une fois que les restrictions précédentes sont satisfaites.

  - Toujours.

  - Résultat différent au précédent: il s'enverra seulement au bus KNX le résultat de la fonction lorsque celui-ci est différent à celui de l'exécution précédente.

- Envoi périodique: l'objet du résultat se renverra au bus de façon répétée chaque certain temps à partir de l'exécution de la fonction pour la première fois, comme établie dans le paramètre "Mode d'envoi". L'échelle de ce paramètre dépend de l'unité de temps choisie: 10 à 600 dixièmes de seconde, 1 à 3600 secondes, 1 à 1440 minutes ou 1 à 24 heures.

  Tenez en compte que renvoyer le résultat périodiquement **n'implique pas que se calcul nouvellement** devant chaque renvoi, pour ce que les changements dans les objets d'entrée n'altèreront pas la valeur qui s'envoie au bus; pour cela il est nécessaire revenir à déclencher la fonction. Notez en plus qu'à se déclencher nouvellement la fonction, s'arrêtera l'envoi périodique ou précédent et se réinitialisera le compteur de temps de période, inclus bien que le nouveau résultat ne remplisse pas les conditions pour être envoyé au bus (dans tel cas, le précédent envoi périodique s'interrompra également).

- **Retard:** établie un temps de retard pour l'envoi du résultat après l'exécution de la fonction. Si se désire un envoi immédiat, il devra s'établir la valeur "0".

  L'échelle de ce paramètre dépend de l'unité de temps choisie: 0 à 600 dixièmes de seconde, 0 à 3600 secondes, 0 à 1440 minutes ou 0 à 24 heures.

  Tel comme il a été expliqué, le retard s'applique au moment d'envoyer l'objet au bus: la fonction en soit s'exécute immédiatement après l'appel, indépendamment si le résultat s'envoi ou non en différé, pour ce que celui-ci ne sera pas affecté par les changements de valeur des opérandes qui ont lieu pendant le temps de retard.

  Note: *Dans le cas de recevoir un nouvel ordre de déclenchement d'une fonction dont les résultats n'ont pas encore été envoyé (dû au retard), celle-ci s'exécutera de nouveau, évitant le résultat d'envoi et en réinitialisant la temporisation.* Dans le cas où le nouveau résultat ne remplisse pas les conditions pour être envoyé au bus, l'envoi pendant s'annulera également.

ANNEXE I: LISTE D'OPÉRATIONS

OPÉRATIONS DE LOGIQUE BOOLÉENNE (1 BIT)

- **OUI (Identité)**

Opérande 1	*Opérande 2*	*Résultat*
0	-	0
1	-	1

- **ET (produit logique ou conjonction)**

Opérande 1	*Opérande 2*	*Résultat*
0	0	0
0	1	0
1	0	0
1	1	1

- **OU (somme logique ou disjonction)**

Opérande 1	*Opérande 2*	*Résultat*
0	0	0
0	1	1
1	0	1
1	1	1

- **OU exclusif (OU exclusif)**

Opérande 1	*Opérande 2*	*Résultat*
0	0	0
0	1	1
1	0	1
1	1	0

- **NON (négation)**

Opérande 1	*Opérande 2*	*Résultat*
0	-	1
1	-	0

- **NON ET (ET refusé)**

Opérande 1	*Opérande 2*	*Résultat*
0	0	1
0	1	1
1	0	1
1	1	0

- **NON OU (OU refusé)**

Opérande 1	*Opérande 2*	*Résultat*
0	0	1
0	1	0
1	0	0
1	1	0

- **NON-OU EXCLUSIF (OU EXCLUSIF refusé)**

Opérande 1	*Opérande 2*	*Résultat*
0	0	1
0	1	0
1	0	0
1	1	1

OPÉRATIONS ARITHMÉTIQUES

- **OUI (Identité)**

Opérande 1	*Opérande 2*	*Résultat*
v1	-	*v1*

- **ADDITION**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*v1 + v2*

- **SOUSTRACTION**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*v1 - v2*

- **MULTIPLICATION**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*v1 * v2*

- **DIVISION**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*v1 / v2*

- **MAXIMUM**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*max(v1, v2)*

- **MINIMUM**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*min(v1, v2)*

Note: Il est recommandé de lire la section *0*plusieurs observations pour plus d'information sur déterminés cas spécifiques et débordements.

OPÉRATIONS DE COMPARAISON

- **SUPÉRIEUR**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*1 ↔ v1 > v2.* *0 en autre cas.*

- **SUPÉRIEUR OU ÉGAL**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*1 ↔ v1 ≥ v2.* *0 en autre cas.*

- **INFÉRIEUR**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*1 ↔ v1 < v2.* *0 en autre cas.*

- **INFÉRIEUR OU ÉGAL**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*1 ↔ v1 ≤ v2.* *0 en autre cas.*

- **ÉGAL**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*1 ↔ v1 = v2.* *0 en autre cas.*

- **DIFFÉRENT**

Opérande 1	*Opérande 2*	*Résultat*
v1	*v2*	*1 ↔ v1 ≠ v2.* *0 en autre cas.*

Note: Il est recommandé de lire la section *0*plusieurs observationspour plus d'information sur déterminés cas spécifiques et débordements.

OPÉRATIONS DE CONVERSION

Opérande de un bit:

	1 Byte (Sans signe)	*2 Bytes (sans signe)*	*2 Bytes (avec signe)*	*2 Bytes (flottante)*	*4 Bytes (avec signe)*
0	*0*	*0*	*0*	*0,00*	*0*
1	*1*	*1*	*1*	*1,00*	*1*

Opérande de un byte sans signe

	1 Bit	*2 Bytes (Sans signe)*	*2 Bytes (avec signe)*	*2 Bytes (flottante)*	*4 Bytes (avec signe)*
0	*0*	*0*	*0*	*0,00*	*0*
10	*1*	*10*	*10*	*10,00*	*10*
180	*1*	*180*	*180*	*180,00*	*180*
255	*1*	*255*	*255*	*255,00*	*255*

Opérande de un byte type pourcentage

	1 Bit	1 Byte (Sans signe)	2 Bytes (Sans signe)	2 Bytes (avec signe)	2 Bytes (flottante)	4 Bytes (avec signe)
0	0	0	0	0	0,00	0
10	1	10	10	10	10,00	10
80	1	80	80	80	80,00	80
100	1	100	100	100	100,00	100

Opérande de deux bytes sans signe

	1 Bit	1 Byte (Sans signe)	2 Bytes (avec signe)	2 Bytes (flottante)	4 Bytes (avec signe)
0	0	0	0	0,00	0
10	1	10	10	10,00	10
500	1	255	500	500,00	500
65535	1	255	32767	65535,00	65535

Opérande de deux bytes avec signe

	1 Bit	1 Byte (Sans signe)	2 Bytes (sans signe)	2 Bytes (flottante)	4 Bytes (avec signe)
-32768	0	0	0	-32768,00	-32768
-12000	0	0	0	-12000,00	-12000
0	0	0	0	0,00	0
12345	1	255	12345	12345,00	12345

Opérande de deux bytes avec virgule flottante

	1 Bit	1 Byte (Sans signe)	2 Bytes (sans signe)	2 Bytes (avec signe)	4 Bytes (avec signe)
-671088,64	0	0	0	-32768	-671088
-321654,98	0	0	0	-32768	-321654
0,00	0	0	0	0	0
12345,67	1	255	12345	12345	12345

Opérande de quatre bytes avec signe

	1 Bit	1 Byte (Sans signe)	2 Bytes (sans signe)	2 Bytes (avec signe)	2 Bytes (flottante)
-2147483648	0	0	0	-32768	0xF800 (*)
-1	0	0	0	-1	-1,00
0	0	0	0	0	0,00
123456	1	255	65535	32767	123456,00
2147483647	1	255	65535	32767	0x7FFF (*)

(*) Les valeurs minimum et maximum de deux bytes avec virgule flottante permises dans le standard KNX s'opèrent respectivement, comme -infini et +infini, comme s'indique dans Plusieurs plusieurs observations.

PLUSIEURS OBSERVATIONS

Comme il a déjà été expliqué, le module de fonctions logiques peut travailler avec les types de données suivantes:

- Binaires: **0** et **1**.
- entier sans signe.
  - Un byte: **0 – 255**.
  - Deux bytes: **0 – 65535**
- Valeurs de pourcentage (un byte): **0 – 100**.
- entiers avec signe.
  - Deux bytes: **-32768 – 32767**.
  - Quatre bytes: **-2147483648 – 2147483647**.
- Virgule flottante (2 bytes): **-671088,64 – 671760,96**.

Ces opérandes pourront être objets de communication, variables internes ou aller garder les résultats intermédiaires, ou inclus, selon quelles opérations, constantes numériques établies par paramètre dans ETS.

D'autre part, il est important de tenir présente les observations suivantes:

- Les **débordements dans les opérations mathématiques** se répondent en rendant la valeur de la limite qui a été dépassé. Par exemple; une somme de taille de un byte entre les valeurs 250 et 10 rendra 255 à être 255 la limite de l'échelle permise pour un byte.

- **La perte de tension** sur le bus ne suppose pas la perte des valeurs que possédaient les objets ni les variables internes: au retour de la tension, se récupèreront les valeurs précédentes.

- Les opérations de **division entre zéro** ne rendent aucun résultat; si quelque fonction contienne un opération de division entre zéro, s'interrompra au terme de la précédente opération.

- **La multiplication et la division de valeurs de pourcentage** entre elles, oui s'effectue conforme aux exemples suivants:

  - Deux constantes:

    La multiplication et la division de deux constantes entre elles, oui se fait en assumant qu'une constante est un entier. Par exemple; si l'opérande 1 se fixe en "25" et l'opérande 2 dans "2", le résultat est "50%" (notez que le résultat de cette opération se connait d'avance).

  - Une constante et un objet:

    Cette option se destine dans les cas où l'on désire, par exemple, que la position d'un volet soit toujours "k" fois (étant "k" une constante) celle d'une autre. Par exemple, si ce lui assigne à "k" la valeur "3" et la position du deuxième volet est "30%", le premier bougera de 90%.

  - Deux objets:

    - Si se désire, comme dans le cas juste auparavant, qu'un volet B acquiert "k" fois la position d'un autre volet A, étant "k" dans ce cas un objet entier d'un byte sans signe (et non une constante), il sera nécessaire de convertir "k" en une variable interne, pour que après elle soit multipliée par l'objet d'état de B.

    - Dans la même situation, si "k%" est la valeur d'un objet de un byte type pourcentage (et non une constante), pour que B acquiert un "k%" dans la position A (ex.: 50% de 40% cela est, 20%),sera nécessaire de convertir les deux en variables de deux bytes virgule flottante, les multiplier et finalement convertir le résultat en une valeur de pourcentage d'un byte.

- Les limites supérieures (plus positive) et inférieure (plus négative) de l'échelle de valeurs en virgule flottante permises par KNX (cela est,-671088,64 représenté comme 0xF800, et +670760.96 représenté comme 0x7FFF) s'opèrent comme **+∞ (plus infini) et –∞ (moins infini)** respectivement, comme dans les exemples suivants:)
  - (670760,96) * (-671088,64) = (-671088,64)
  - (-671088,64) * (-671088,64) = (670760,96)
  - (670760,96) * (34,00) = (670760,96)
  - (670760,96) * (-34,80) = (-671088,64)
  - (670760,96) / (65,20) = (670760,96)
  - (670760,96) / (-341,12) = (-671088,64)
  - (120000,00) / (670760,96) = (0,00)
  - (-671088,64) / (670760,96) = (-1,00)
  - (-671088,64) / (3500,66) = (-671088,64)
- Dans le cas de paramétrer une **période d'envoi** du résultat et, en plus, un **temps de retard**, celui-ci s'appliquera seulement devant le premier envoi au bus, à la fin des opérations. Passé le retard il s'enverra le résultat au bus et commencera à compter la période de renvoi. Dans les envois successifs il ne s'appliquera pas le retard, sauf si se déclenche de nouveau une fonction et se génère un nouveau résultat qui doit s'envoyer au bus, auquel cas, se recommencera à appliquer le retard dans le premier envoi.

Venez nous poser vos questions sur les dispositifs Zennio à:

http://support.zennio.com

Zennio Avance y Tecnología S.L.
C/ Río Jarama, 132. Nave P-8.11
45007 Toledo (Spain).

Tel. +34 925 232 002.
Tel. 01 76 54 09 27

www.zennio.fr
info@zennio.fr