Langage à blocs fonction FBD

11

Objet de ce sous-chapitre

Ce chapitre décrit le langage à blocs fonction FBD conforme à la norme IEC 61131.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

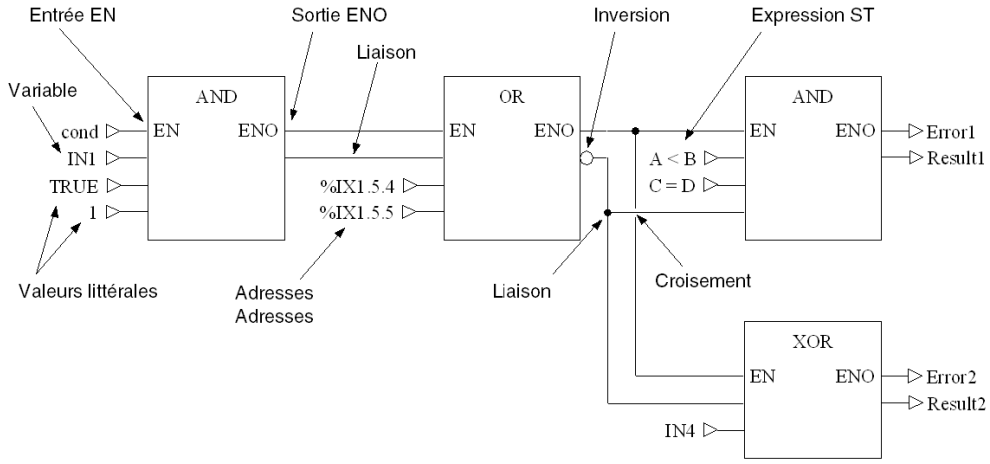
Informations générales sur le langage à blocs fonction FBD

Présentation

L'éditeur FBD permet la programmation graphique de blocs fonction conformément à la norme CEI 61131-3.

Représentation d'une section FBD

Représentation :



Objets

Les objets du langage FBD (diagramme de blocs fonctionnels) offrent des aides permettant de structurer une section en un ensemble de :

- EF et EFB (fonctions élémentaires *(voir page 330)* et blocs fonction élémentaires *(voir page 331)*),
- DFB (blocs fonction dérivés) *(voir page 332)*,
- procédures *(voir page 332)* et
- contrôles *(voir page 342)*.

Ces objets, regroupés sous l'abréviation générique FFB, peuvent être liés les uns aux autres par :

- des liaisons *(voir page 344)* ou
- des paramètres réels *(voir page 333)*.

La logique de la section peut être commentée par des objets texte (voir *Objet texte, page 346*).

Taille de la section

Une section FBD comprend une fenêtre incluant une seule page.

Cette page est placée sur une grille. Une unité de grille comprend 10 points de trame. Une unité de trame est l'espace le plus petit possible entre deux objets d'une section FBD.

Le langage FBD n'est pas basé sur les cellules les objets sont toutefois ajustés sur les points de trame.

Une section FBD peut être configurée en nombre de cellules (points de trame horizontaux et points de trame verticaux).

Conformité CEI

Pour la description de la conformité CEI du langage FBD, voir Conformité CEI *(voir page 657)*.

Fonctions élémentaires, blocs fonction élémentaires, blocs fonction dérivés et procédures (FFB)

Introduction

FFB est le terme générique pour :

- les fonctions élémentaires (EF) *(voir page 330)*
- les blocs fonction élémentaires (EFB) *(voir page 331)*
- les DFB (blocs fonction dérivés) *(voir page 332)*
- Procédure *(voir page 332)*

Fonction élémentaire

Les fonctions élémentaires (EF) n'ont pas d'état interne. Lorsque les valeurs d'entrée sont identiques, la valeur de sortie est la même à chaque exécution de la fonction. Par exemple, l'addition de deux valeurs donne toujours le même résultat.

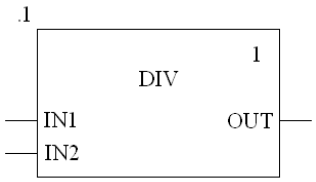
Une fonction élémentaire est représentée graphiquement comme un cadre avec des entrées et une sortie. Les entrées sont toujours représentées sur la gauche et la sortie toujours sur la droite du cadre.

Le nom de la fonction, c'est-à-dire le type de fonction, est affiché au centre du cadre.

Le numéro d'exécution *(voir page 347)* de la fonction apparaît à droite du type de fonction.

Le numéro de fonction est affiché au-dessus du cadre. Le numéro de fonction représente le numéro courant de la fonction dans la section actuelle. Les numéros de fonction ne peuvent pas être modifiés.

Fonction élémentaire



Pour certaines fonctions élémentaires, il est possible d'augmenter le nombre d'entrées.

Bloc fonction élémentaire

Les blocs fonction élémentaires (EFB) ont des états internes. Lorsque les valeurs d'entrée sont identiques, la valeur de sortie peut être différente pour toutes les exécutions de la fonction. Par exemple, pour un compteur, la valeur de sortie augmente.

Un bloc fonction élémentaire est représenté graphiquement sous forme de cadre avec des entrées et des sorties. Les entrées sont toujours représentées sur la gauche et les sorties toujours sur la droite du cadre.

Les blocs fonction peuvent avoir plusieurs sorties.

Le nom du bloc fonction, c'est-à-dire le type de bloc fonction, est affiché au centre du cadre.

Le numéro d'exécution *(voir page 347)* du bloc fonction apparaît à droite du type de bloc fonction.

Le nom d'instance est affiché au-dessus du cadre.

Le nom d'instance permet d'identifier précisément le bloc fonction dans un projet.

Le nom d'instance est généré automatiquement et présente la structure suivante : `FBI_n`

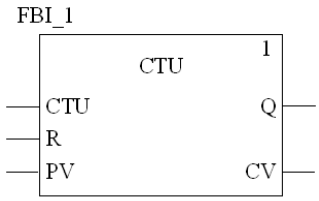
`FBI` = Instance de bloc fonction

`n` = numéro courant du bloc fonction au sein du projet

Vous pouvez modifier ces noms générés automatiquement pour rendre la vue d'ensemble plus claire. Le nom d'instance (32 caractères maximum) doit être unique dans tout le projet ; aucune distinction n'est faite ici entre majuscules et minuscules. Le nom d'instance doit respecter les conventions de noms générales.

NOTE : conformément à la norme CEI 61131-3, les noms d'instance doivent commencer par une lettre. Si vous voulez également utiliser des chiffres comme premier caractère, vous devez activer cette fonction.

Bloc fonction élémentaire

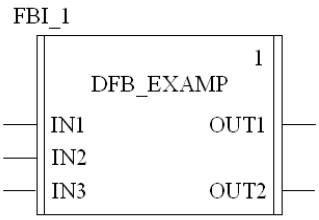


DFB

Les blocs fonction dérivés (DFB) ont les mêmes caractéristiques que les blocs fonction élémentaires. Ils sont cependant créés par l'utilisateur dans les langages FBD, LD, IL et/ou ST.

L'unique différence par rapport aux blocs fonction élémentaires est que le bloc fonction dérivé est représenté graphiquement sous forme de cadre avec deux lignes verticales.

Bloc fonction dérivé



Procédure

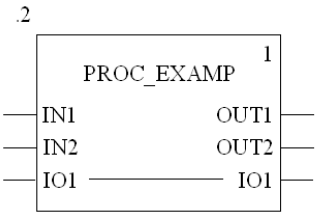
Techniquement, les procédures sont des fonctions.

L'unique différence par rapport aux fonctions élémentaires est que les procédures peuvent comprendre plus d'une sortie et qu'elles prennent en charge le type de données `VAR_IN_OUT`.

Les procédures sont une extension de la norme CEI 61131-3 et doivent être activées de manière explicite.

Il n'y a pas de différence physique entre les procédures et les fonctions élémentaires.

Procédure

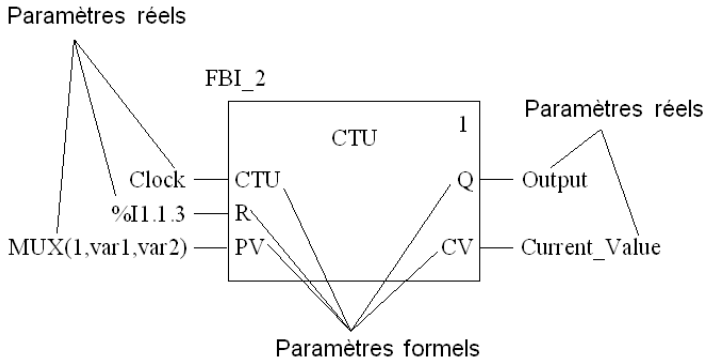


Paramètres

Pour importer des valeurs dans le FFB ou exporter des valeurs du FFB, des entrées et des sorties sont nécessaires. Elles sont appelées paramètres formels.

Les paramètres formels sont liés à des objets qui comprennent les états courants du traitement. Ces états sont appelés paramètres réels.

Paramètres formels et réels :



Durant l'exécution du programme, les valeurs sont transmises, par le biais des paramètres réels, du processus au FFB, et renvoyées à nouveau à la sortie après le traitement.

Seul un objet (paramètre réel) du type de données suivant peut être relié aux entrées FFB :

- variable
- adresse
- libellé
- expression ST *(voir page 509)*
  Les expressions ST des entrées FFB représentent une extension de la norme CEI 61131-3 et doivent être activées de manière explicite.
- Liaison

Les combinaisons d'objets (paramètres réels) suivantes peuvent être reliées aux sorties FFB :

- une variable
- une variable et une ou plusieurs liaisons (non valable pour les sorties VAR_IN_OUT *(voir page 339)*)
- une adresse,
- une adresse et une ou plusieurs liaisons (non valable pour les sorties VAR_IN_OUT *(voir page 339)*)
- une ou plusieurs liaisons (non valable pour les sorties VAR_IN_OUT *(voir page 339)*)

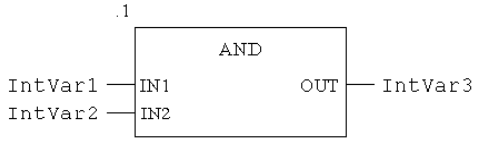
Le type des données de l'objet à relier doit correspondre au type des données de l'entrée/la sortie FFB. On choisira un type de données adapté pour le bloc fonction, si tous les paramètres réels sont constitués de valeurs littérales.

Exception : pour les entrées/sorties génériques FFB de type de données ANY_BIT, des objets de type de données INT ou DINT (pas UINT ni UDINT) peuvent être reliés.

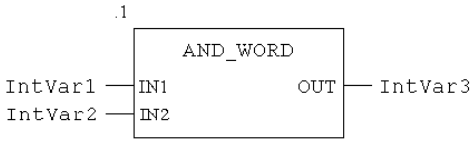
Il s'agit d'une extension de la norme CEI 61131-3 et doit donc être activée de manière explicite.

Exemple :

Autorisé :



Non autorisé :



(Dans ce cas AND_INT doit être utilisé.)

Il n'est en principe pas nécessaire d'affecter un paramètre réel à tous les paramètres formels. Cependant, cela n'est pas valable pour les broches inversées. Un paramètre réel doit toujours leur être affecté. Cela est également impératif pour certains types de paramètre formel. Pour connaître les types concernés, veuillez vous reporter au tableau suivant.

Tableau des types de paramètre formel :

Type de paramètre	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
EFB : Entrée	-	+	+	+	/	+	/	+
EFB : VAR_IN_OUT	+	+	+	+	+	+	/	+
EFB : Sortie	-	-	+	+	+	-	/	+
DFB : Entrée	-	+	+	+	/	+	/	+
DFB : VAR_IN_OUT	+	+	+	+	+	+	/	+
DFB : Sortie	-	-	+	/	/	-	/	+
EF : Entrée	-	-	+	+	+	+	+	+
EF : VAR_IN_OUT	+	+	+	+	+	+	/	+
EF : Sortie	-	-	-	-	-	-	/	-
Procédure : Entrée	-	-	+	+	+	+	+	+

Type de paramètre	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
Procédure : VAR_IN_OUT	+	+	+	+	+	+	/	+
Procédure : Sortie	-	-	-	-	-	-	/	+
+ paramètre réel impératif								
- Paramètre réel non impératif								
/ Non applicable								

Les FFB utilisant aux entrées des paramètres réels, auxquels aucune valeur n'a encore été affectée, fonctionnent avec les valeurs initiales de ces paramètres réels.

Si aucune valeur n'est affectée à un paramètre formel, la valeur initiale est utilisée pendant l'exécution du bloc fonction. Si aucune valeur initiale n'est définie, la valeur par défaut (0) est utilisée.

Si aucune valeur n'est affectée à un paramètre formel et que le bloc fonction/DFB a été instancié à plusieurs reprises, les instances appelées par la suite travaillent avec l'ancienne valeur.

Variables publiques

Certains blocs fonction disposent non seulement d'entrées et de sorties, mais également de variables publiques.

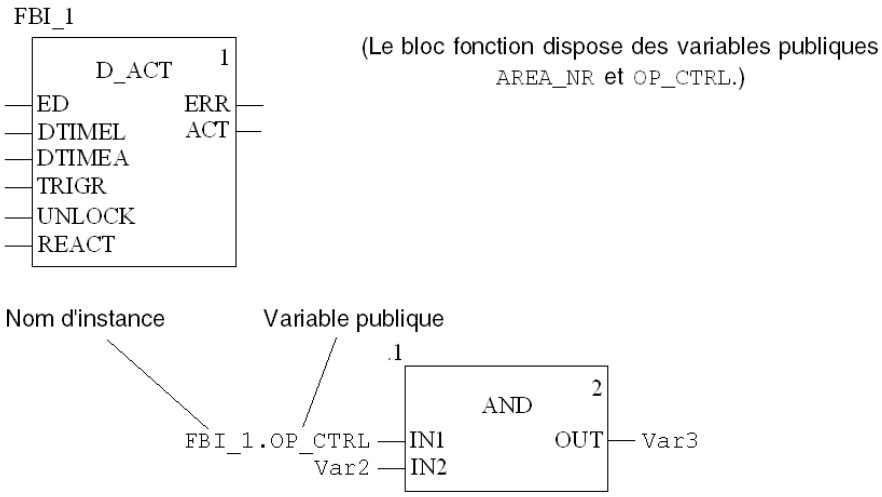
Ces variables permettent de transmettre des valeurs statiques (valeurs non influencées par le procédé) au bloc fonction. Elles sont donc utilisées lors du paramétrage du bloc fonction.

Les variables publiques sont une extension de la norme CEI 61131-3.

Les valeurs sont affectées aux variables publiques via leur valeur initiale.

Les valeurs des variables publiques sont ensuite lues à partir du nom d'instance du bloc fonction et du nom de la variable publique.

Exemple :



Variables privées

Certains blocs fonction disposent non seulement d'entrées, de sorties et de variables publiques, mais également de variables privées.

A l'instar des variables publiques, ces variables permettent de transmettre des valeurs statistiques (valeurs non influencées par le procédé) au bloc fonction.

Le programme utilisateur ne peut pas accéder à ces variables. Seule la table d'animation en a la capacité.

NOTE : les DFB imbriqués sont déclarés comme des variables privées du DFB parent. Ainsi, leurs variables ne sont pas accessibles via la programmation, mais via la table d'animation.

Les variables privées sont une extension de la norme CEI 61131-3.

Remarques sur la programmation

Veuillez tenir compte des remarques qui suivent sur la programmation :

- Les FFB ne sont exécutées que lorsque l'entrée EN = 1 ou lorsque l'entrée `EN` est désactivée (voir aussi EN et ENO *(voir page 337)*).
- Les entrées et sorties booléennes peuvent être inversées.
- Des conditions particulières s'appliquent lors de l'utilisation de variables VAR_IN_OUT *(voir page 339)*.
- Les instances de DFB ou bloc fonction peuvent être appelées à plusieurs reprises (voir aussi Appel multiple d'une instance de bloc fonction *(voir page 337)*).

Appel multiple d'une instance de bloc fonction

Les instances de DFB ou bloc fonction peuvent être appelées à plusieurs reprises, à l'exception des instances d'EFB de communication et blocs fonction/DFB ayant une sortie ANY mais pas d'entrée ANY, qui ne peuvent être appelées qu'une seule fois.

L'appel multiple de la même instance de DFB/bloc fonction est par exemple utile dans les cas suivants :

- si le bloc fonction/DFB ne comporte aucune valeur interne ou si celle-ci n'est plus nécessaire pour un traitement ultérieur.
  Dans ce cas, l'appel multiple de la même instance de DFB/bloc fonction permet d'économiser de l'espace mémoire, car le code du bloc fonction/DFB n'est alors chargé qu'une seule fois.
  Le bloc fonction/DFB est pour ainsi dire traité comme une "fonction".
- si le bloc fonction/DFB comprend des valeurs internes et que celles-ci doivent être influencées à différents endroits du programme, la valeur d'un compteur, par exemple, doit être augmentée à différents endroits du programme.
  Dans ce cas, l'appel multiple de la même instance de bloc fonction/DFB permet d'économiser la mémoire des résultats intermédiaires pour un traitement ultérieur à un autre endroit du programme.

EN et ENO

Une entrée EN et une sortie ENO peuvent être configurées pour tous les FFB.

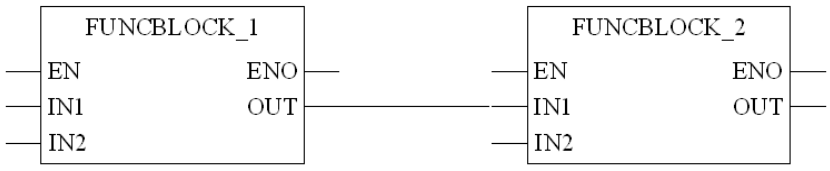
Si la valeur de EN est déjà réglé sur 0, lors de l'appel de FFB, les algorithmes définis par FFB ne sont pas exécutés et ENO est réglé sur 0.

Si la valeur de EN est déjà réglé sur 1, lors de l'appel de FFB, les algorithmes définis par FFB sont exécutés. Après l'exécution sans erreur de ces algorithmes, la valeur de ENO est réglée sur 1. Si une erreur se produit durant l'exécution de ces algorithmes, ENO est réglé sur 0.

Si aucune valeur n'est attribuée à la broche EN à l'appel du FFB, l'algorithme défini par ce dernier est exécuté (comme lorsque EN a la valeur « 1 »). Reportez-vous à la section *Maintenir les liens de sortie sur les EF désactivés (voir Unity Pro, Modes de marche,)*.

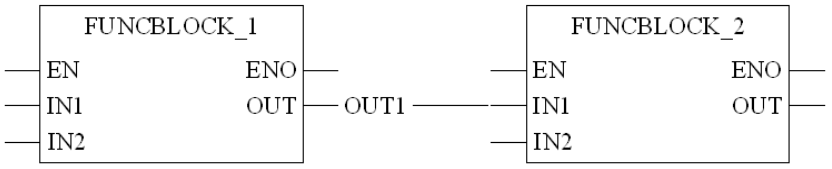
Si ENO est réglé sur 0 (car EN = 0 ou en raison d'une erreur d'exécution) :

- Blocs fonction
  - Traitement EN/ENO pour les blocs fonction ayant (seulement) une liaison comme paramètre de sortie :



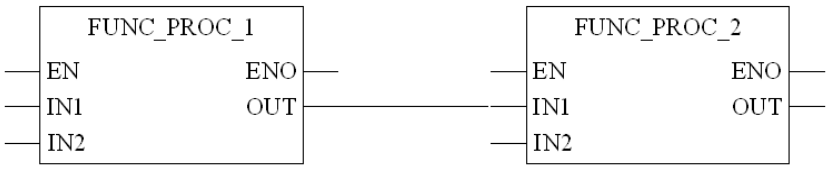
Lorsque EN est réglé sur 0 par FUNCBLOCK_1, la liaison à la sortie OUT de FUNCBLOCK_1 conserve l'ancien statut qu'elle avait lors du dernier cycle correct.

- Traitement EN/ENO pour les blocs fonction ayant une variable et une liaison comme paramètre de sortie :



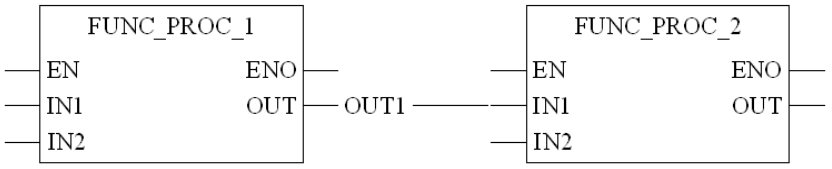
Lorsque EN est réglé sur 0 par FUNCBLOCK_1, la liaison à la sortie OUT de FUNCBLOCK_1 conserve l'ancien statut qu'elle avait lors du dernier cycle correct. La variable OUT1 située sur la même broche conserve son ancien statut ou peut être modifiée depuis l'extérieur sans avoir d'influence sur la liaison. La variable et la liaison sont enregistrées indépendamment l'une de l'autre.

- Fonctions/Procédures
  Selon la définition CEI 61131-3, les sorties de fonctions désactivées (entrée EN mise à "0") sont indéfinies. (Le même principe s'applique aux procédures.)
  Voici, néanmoins, une explication des états des sorties dans un tel cas :
  - Traitement EN/ENO pour les fonctions/procédures ayant (seulement) une liaison comme paramètre de sortie :



Si EN de FUNC_PROC_1 est réglé sur 0, la valeur de la liaison sur la sortie OUT de FUNC_PROC_1 dépend du paramètre de projet **Maintenir les liens de sortie sur les EF désactivés**, disponible depuis Unity Pro 4.1.
Si ce paramètre de projet est réglé sur 0, la valeur de la liaison est réglée sur 0.
Si ce paramètre de projet est réglé sur 1, la liaison conserve la valeur qu'elle avait lors du dernier cycle exécuté correctement.
Consultez la section *Maintenir les liens de sortie sur les EF désactivés (voir Unity Pro, Modes de marche,)*.

  - Traitement EN/ENO pour les fonctions/procédures ayant une variable et une liaison comme paramètre de sortie :



Si EN de FUNC_PROC_1 est réglé sur 0, la valeur de la liaison sur la sortie OUT de FUNC_PROC_1 dépend du paramètre de projet **Maintenir les liens de sortie sur les EF désactivés**, disponible depuis Unity Pro 4.1.
Si ce paramètre de projet est réglé sur 0, la valeur de la liaison est réglée sur 0.
Si ce paramètre de projet est réglé sur 1, la liaison conserve la valeur qu'elle avait lors du dernier cycle exécuté correctement.
Consultez la section *Maintenir les liens de sortie sur les EF désactivés (voir Unity Pro, Modes de marche,)*.
La variable OUT1 située sur la même broche conserve son ancien statut ou peut être modifiée depuis l'extérieur sans avoir d'influence sur la liaison. La variable et la liaison sont enregistrées indépendamment l'une de l'autre.

Le comportement aux sorties des FFB est indépendant du fait que les FFB soient appelés sans EN/ENO ou avec EN = 1.

NOTE : pour les blocs fonction désactivés (EN = 0) équipés d'une fonction d'horloge interne (par exemple, le bloc fonction DELAY), le temps semble continuer de s'écouler, car il est calculé à l'aide d'une horloge système et est, par conséquent, indépendant du cycle du programme et de la libération du bloc.

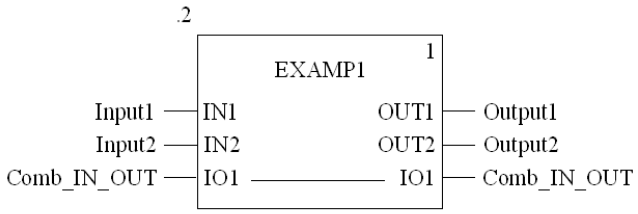
Variable VAR_IN_OUT

Les FFB sont souvent utilisés pour lire une variable à l'entrée (variables d'entrée), pour la traiter et pour transmettre de nouveau les valeurs modifiées de cette **même** variable (variables de sortie).

Ce cas exceptionnel d'une variable d'entrée/de sortie est également appelé variable VAR_IN_OUT.

Dans le FFB, une ligne indique que les variables d'entrée et de sortie sont liées l'une à l'autre.

Variable VAR_IN_OUT



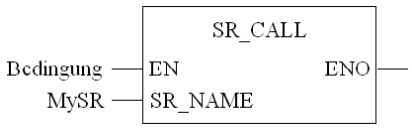
Il convient de noter les particularités suivantes en cas d'utilisation de FFB avec des variables VAR_IN_OUT :

- une variable doit être affectée à toutes les entrées VAR_IN_OUT.
- les liaisons graphiques permettent uniquement de relier des sorties VAR_IN_OUT à des entrées VAR_IN_OUT.
- seule une liaison graphique unique peut être reliée à une entrée/sortie VAR_IN_OUT.
- une combinaison de variables/d'adresses et de liaisons graphiques n'est pas possible pour les sorties VAR_IN_OUT.
- il est interdit de relier des valeurs littérales ou des constantes à des entrées/sorties VAR_IN_OUT.
- il est interdit d'utiliser des négations au niveau des entrées/sorties VAR_IN_OUT.
- des variables/composantes de variables différentes peuvent être reliées à l'entrée VAR_IN_OUT et à la sortie VAR_IN_OUT. Dans un tel cas, la valeur de la variable/composante de variable à l'entrée est copiée dans la variable/composante de variable à la sortie.

Appels de sous-programme

Appel d'un sous-programme

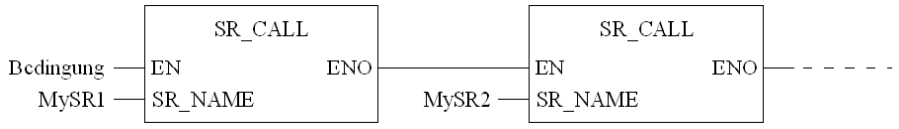
En FBD, les sous-programmes sont appelés à l'aide du bloc ci-dessous.



Si 1 est l'état de `EN`, le sous-programme correspondant (nom des variables à `SR_Name`) est appelé.

Pour ce type de bloc, la sortie `ENO` ne sert pas à afficher l'état d'erreur. Pour ce type de bloc, la sortie `ENO` est toujours 1 et permet d'appeler simultanément plusieurs sous-programmes.

La construction suivante permet d'appeler simultanément plusieurs sous-programmes.



Le sous-programme à appeler doit se trouver dans la même tâche que la section FBD appelante.

Il est possible d'appeler des sous-programmes au sein de sous-programmes.

Les appels de sous-programmes sont un complément de la norme CEI 61131-3 et doivent être activés de manière explicite.

Dans les sections d'actions SFC, les appels de sous-programmes ne sont autorisés que si le mode Multitoken a été activé.

Contrôles

Introduction

Les éléments de commande servent à l'exécution de sauts au sein d'une section FBD et au retour prématuré dans le programme principal depuis un sous-programme (SRx) ou un bloc fonction dérivé (DFB).

Contrôles

Les contrôles suivants sont disponibles.

Désignation	Représentation	Description
Jump		Si l'état de la liaison gauche est 1, un saut est exécuté jusqu'à l'étiquette (dans la section courante). Pour générer un saut conditionnel, l'objet saut est lié à une sortie FFB booléenne. Pour générer un saut inconditionnel, la valeur 1 est affectée à l'objet saut via la fonction `AND`.
Libellé	`LABEL:`	Les repères (destinations de saut) sont représentés comme du texte avec deux-points à la fin. Le texte est limité à 32 caractères et doit être unique dans l'ensemble de la section. Le texte doit respecter les conventions de nommage générales. Les étiquettes de saut ne peuvent être placées qu'entre les deux premiers points de trame sur la marge gauche de la section. **Note :** Les étiquettes de saut ne doivent "couper" aucun réseau, c'est-à-dire qu'une ligne imaginaire entre l'étiquette de saut et la marge droite de la section ne doit être coupée par aucun objet. Cela est également valable pour les liaisons.

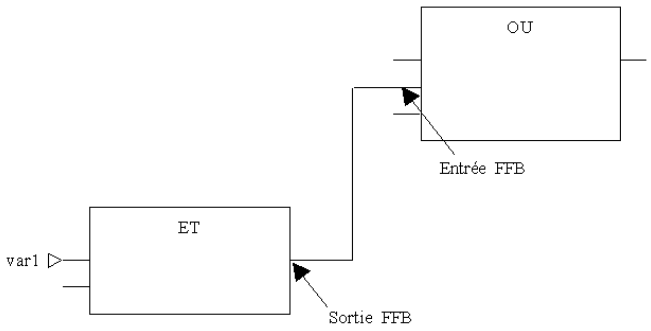
Désignation	Représentation	Description

Return		Des objets `RETURN` ne peuvent pas être utilisés dans le programme principal.
		● Dans un DFB, un objet `RETURN` force le retour au programme qui a appelé le DFB.
		  ● Le reste de la section de DFB contenant l'objet `RETURN` n'est pas exécuté.
		  ● Les sections suivantes du DFB ne sont pas exécutées.
		  Le programme qui a appelé le DFB sera exécuté après retour du DFB.
		  Si le DFB est appelé par un autre DFB, le DFB appelant sera exécuté après le retour.
		● Dans un SR, un objet `RETURN` force le retour au programme qui a appelé le SR.
		  ● Le reste du SR contenant l'objet `RETURN` n'est pas exécuté.
		  Le programme qui a appelé le SR sera exécuté après retour du SR.

Liaison

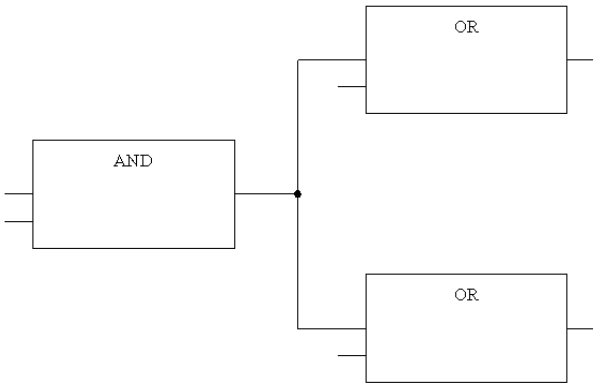
Description

Les liaisons sont des connexions verticales et horizontales entre les FFB.

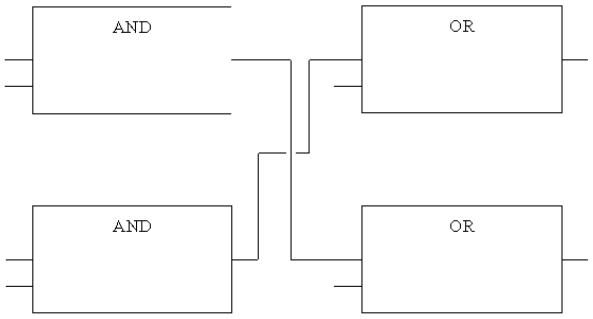


Représentation

Les points de liaison sont marqués par un cercle rempli.



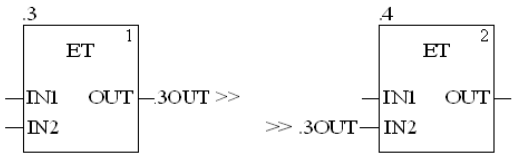
Les croisements sont représentés par une liaison interrompue.



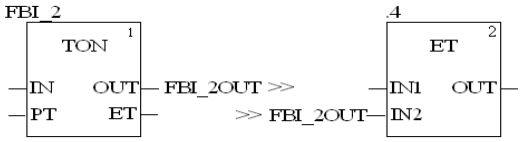
Remarques sur la programmation

Veuillez observer les remarques qui suivent sur la programmation :

- Les liaisons peuvent être utilisées pour chaque type de données.
- Les types de données respectifs des entrées/sorties à relier doivent correspondre les uns aux autres.
- Plusieurs liaisons peuvent être reliées à une sortie FFB, une seule cependant avec une entrée FFB.
- Seules des entrées et sorties peuvent être reliées ensemble. La liaison de plusieurs sorties n'est pas possible. Cela signifie qu'aucun lien OU via des liaisons n'est possible dans FBD. Il faut toujours utiliser une fonction `OU`.
- Le chevauchement des liaisons avec d'autres objets est admis.
- les boucles ne peuvent pas être configurées par le biais de liaisons, étant donné que, dans ce cas, l'ordre d'exécution dans la section ne peut pas être défini de façon unique. Les boucles doivent être résolues par le biais de paramètres réels (voir *Configuration de boucles, page 353*).
- Afin d'éviter le croisement de liaisons, ces dernières peuvent également être représentées sous la forme de connecteurs.
  A cette occasion, la source et la cible de la liaison sont caractérisées par un nom unique au sein de la section.
  Suivant le type d'objet source, le nom du connecteur est formé comme suit :
  - pour les fonctions : "numéro de fonction/paramètre formel" de la source de la liaison.



  - pour les blocs fonction : "nom d'instance/paramètre formel" de la source de la liaison.



Objet texte

Description

Dans le langage blocs fonctions FBD, les textes peuvent être placés sous forme d'objets texte. La taille de ces objets texte est fonction de la longueur du texte. Selon la longueur du texte, la taille de l'objet peut être agrandie, dans les sens vertical et horizontal, d'unités de grille supplémentaires. Les objets texte ne doivent pas se chevaucher avec des FFB, le chevauchement avec des liaisons est toutefois admis.

Ordre d'exécution des FFB

Introduction

L'ordre d'exécution est défini par la position des FFB dans la section (exécution de gauche à droite et de haut en bas). Lorsque, par la suite, les FFB sont liés à des liaisons graphiques, l'ordre d'exécution est alors déterminé par le flux de signaux.

Le numéro d'exécution (numéro figurant dans le coin supérieur droit du cadre de FFB) indique l'ordre d'exécution.

Ordre d'exécution des réseaux

Les règles suivantes s'appliquent à l'ordre d'exécution des réseaux :

- l'exécution d'une section a lieu réseau pour réseau via la liaison des FFB du haut vers le bas.
- les boucles ne peuvent pas être configurées par le biais de liaisons, étant donné que, dans ce cas, l'ordre d'exécution ne peut pas être défini de façon unique. Les boucles doivent être résolues par le biais de paramètres réels (voir *Configuration de boucles, page 353*).
- L'ordre d'exécution des réseaux qui ne sont pas reliés entre eux par des liaisons est défini par l'ordre graphique (de la partie supérieure droite vers la partie inférieure gauche). Vous pouvez influencer l'ordre d'exécution (voir *Modification de l'ordre d'exécution, page 349*).
- Le calcul d'un réseau doit être terminé entièrement avant que ne commence le calcul d'un autre réseau qui utilise les sorties du réseau précédent.
- Aucun élément d'un réseau n'est considéré comme calculé avant que l'état de toutes les entrées de cet élément n'ait été calculé.
- Le calcul d'un réseau est considéré comme terminé lorsque toutes les sorties de ce réseau sont calculées.

Flux de signaux dans un réseau

Les règles suivantes s'appliquent à l'ordre d'exécution au sein d'un réseau :

- Un FFB n'est calculé que lors tous les éléments (sorties FFB, etc) qui sont reliés à ses entrées sont calculés.
- L'ordre d'exécution des FFB reliés à différentes sorties du même FFB va du haut vers le bas.
- L'ordre d'exécution des FFB n'est pas influencé par leur position au sein du réseau.
  Cela ne s'applique pas lorsque plusieurs FFB sont reliés à la même sortie du FFB "à appeler". Dans ce cas l'ordre d'exécution est défini par l'ordre graphique (du haut vers le bas).

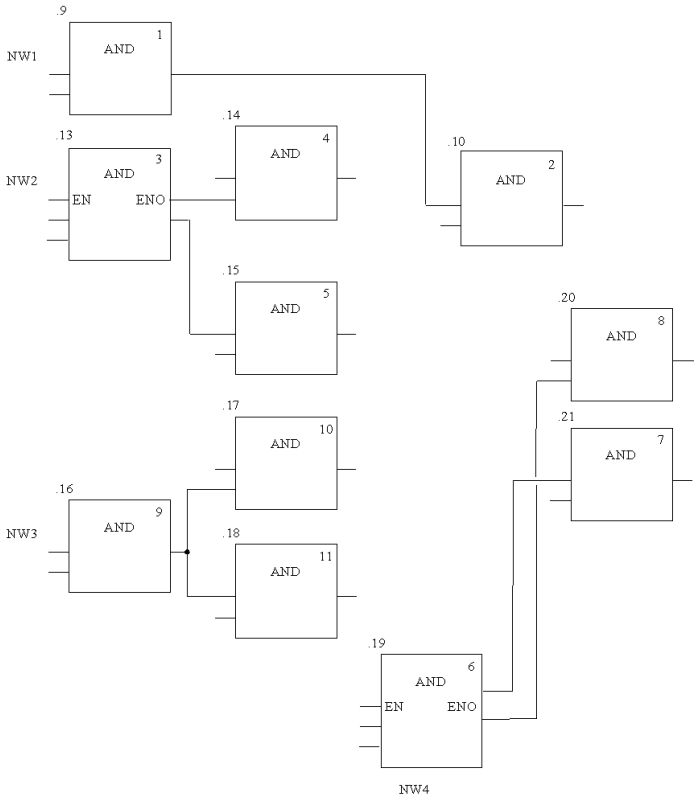
Priorités

Priorités lors de la détermination du flux de signaux au sein d'une section.

Priorité	Règle	Description
1	Lien	Les liaisons ont la priorité la plus élevée lors de la détermination du flux de signaux au sein d'une section FBD.
2	Définition utilisateur	Intervention de l'utilisateur sur l'ordre d'exécution.
3	Réseau par réseau	Le calcul d'un réseau doit être complètement terminé avant que le calcul du réseau suivant puisse commencer.
4	Ordre des sorties	Les FFB qui sont reliés aux sorties du même FFB "à appeler" sont calculés du haut vers le bas.
5	Rung par rung	Priorité la moins élevée. (Cela ne s'applique que si aucune autre règle n'intervient.)

Exemple

Exemple d'ordre d'exécution des objets dans une section FBD :



Modification de l'ordre d'exécution

Introduction

L'ordre d'exécution des réseaux et l'ordre d'exécution des objets au sein d'un réseau sont définis par une série de règles *(voir page 348)*.

Dans certains cas, il est nécessaire de modifier l'ordre d'exécution proposé par le système.

Pour définir/modifier l'ordre d'exécution des réseaux, vous disposez des possibilités suivantes :

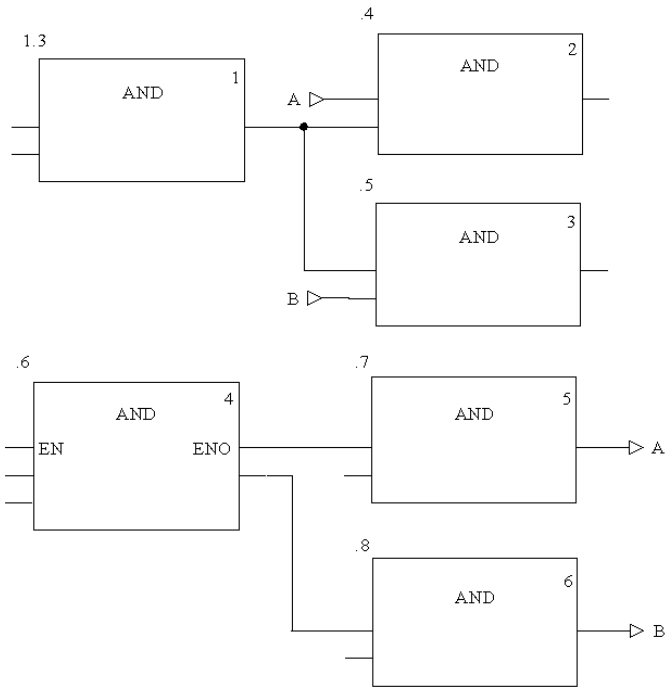
- utilisation de liaisons au lieu des paramètres réels,
- position des réseaux,
- détermination explicite de l'ordre d'exécution.

Pour définir/modifier l'ordre d'exécution des réseaux, vous disposez des possibilités suivantes :

- position des FFB.

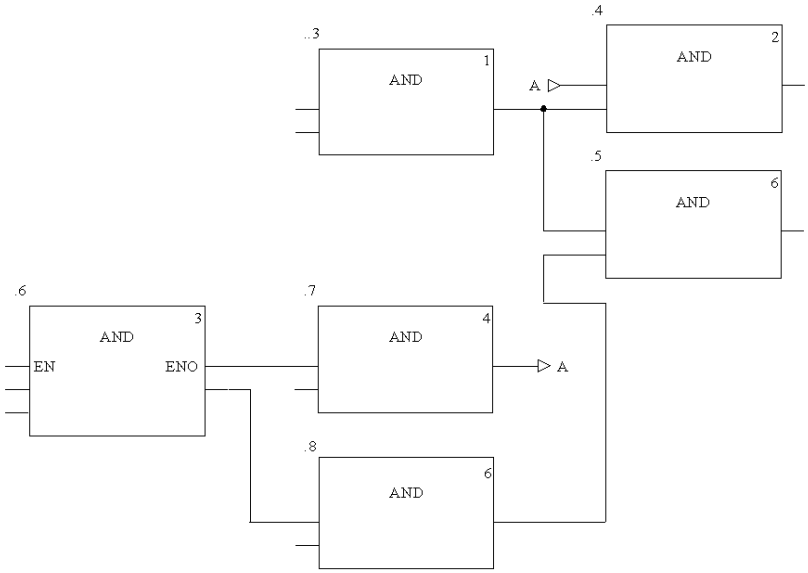
Situation initiale

Le schéma suivant représente deux réseaux dont l'ordre d'exécution est déterminé uniquement par leur position au sein de la section, même si les blocs `.4/.5` et `.7/.8` nécessitent un ordre d'exécution différent.



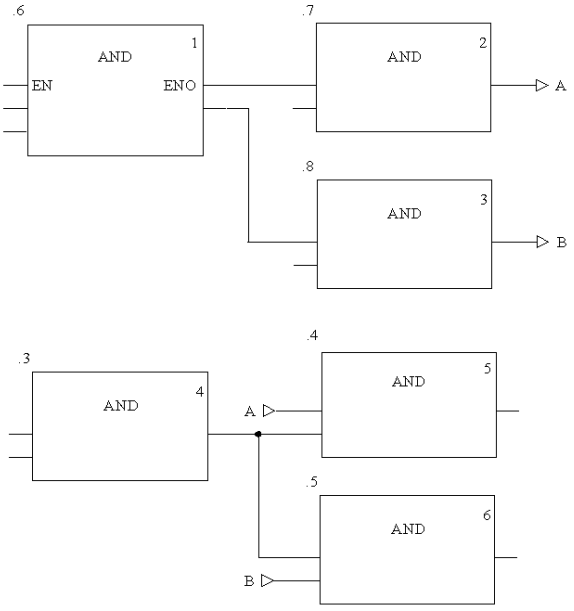
Liaison au lieu des paramètres réels

Lorsque l'on utilise une liaison au lieu d'une variable, les deux réseaux sont exécutés dans l'ordre correct (voir également *Situation initiale, page 349*).



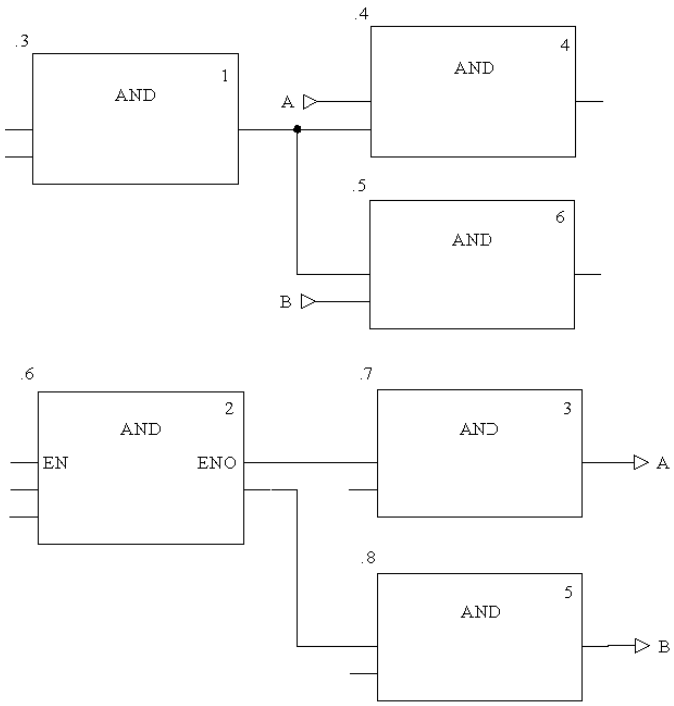
Position des réseaux

Vous pouvez obtenir l'ordre d'exécution correct en modifiant les positions des réseaux dans la section (voir également *Situation initiale, page 349*).



Détermination explicite

Vous pouvez obtenir l'ordre d'exécution correct en modifiant de manière explicite l'ordre d'exécution d'un FFB. Pour les FFB dont l'ordre d'exécution a été modifié de manière explicite, le numéro d'exécution s'affiche dans un champ noir (voir aussi *Situation initiale, page 349*).



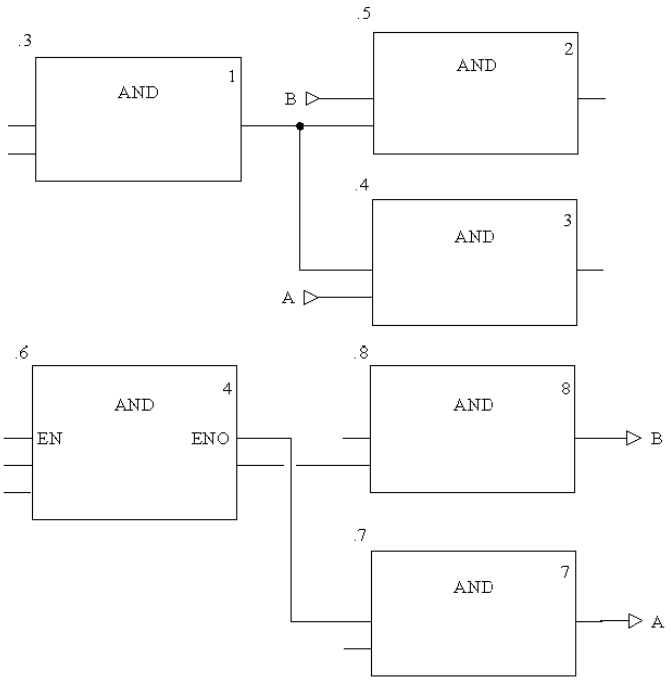
NOTE : Le système n'autorise qu'une seule référence par instance. Ainsi, l'instance ".7"par exemple ne peut être référencée qu'une fois.

Positions des FFB

La position des FFB n'a alors une influence sur l'ordre d'exécution que si plusieurs FFB sont reliés à la même sortie du FFB "à appeler" (voir également *Situation initiale, page 349*).

Les positions des blocs `.4` et `.5` sont permutées dans le premier réseau. Dans ce cas (source commune des deux entrées de bloc), l'ordre d'exécution des deux blocs est également permuté (traitement du haut vers le bas).

Les positions des blocs `.7` et `.8` sont permutées dans le deuxième réseau. Dans ce cas (source différente des entrées de bloc) l'ordre d'exécution des deux blocs n'est pas permuté (traitement dans l'ordre des sorties de bloc à appeler).

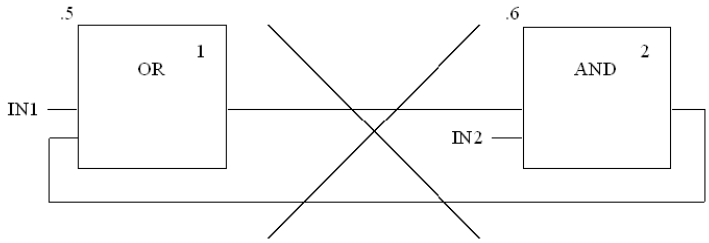


Configuration de boucles

Boucles non permises

La configuration de boucles exclusivement par le biais de liaisons n'est pas permise, étant donné que, dans ce cas, une détermination unique du flux de signaux n'est pas possible (la sortie d'un FFB est l'entrée du FFB suivant, et la sortie de celui-ci est à son tour l'entrée du premier).

Boucles non permises par le biais de liaisons



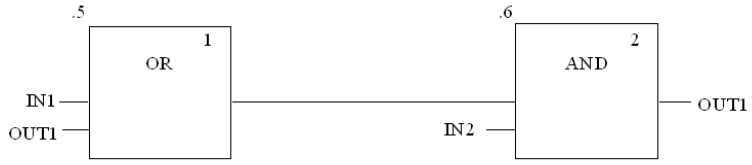
Résolution par le biais d'un paramètre réel

Une telle logique doit être résolue par le biais de variables de réaction, afin que le flux de signaux puisse être défini de façon unique.

Les variables de réaction doivent être initialisées. La valeur initiale est utilisée lors de la première exécution de la logique. Une fois la première exécution effectuée, la valeur initiale est remplacée par la valeur actuelle.

Respectez pour les deux variantes l'ordre d'exécution (numéro entre parenthèses après le nom d'instance) des deux blocs.

Boucle résolue par le biais d'un paramètre réel : Variante 1



Boucle résolue par le biais d'un paramètre réel : Variante 2