Instructions générales Micro800 Programmable Controllers

Catalogues n° 2080-L50E, 2080-L70E, 2080-LC10, 2080-LC20, 2080-LC30, 2080-LC50, 2080-LC70

Publication Rockwell Automation 2080-RM001L-FR-E - Novembre 2023
Annule et remplace la publication 2080-RM001K-FR-E - Mars 2022

Rockwell Automation

Manuel de références

Instructions originales

Informations importantes pour l'utilisateur

Prenez connaissance de ce document ainsi que des documents répertoriés dans la section Ressources supplémentaires concernant l'installation, la configuration et l'utilisation de cet équipement avant d'installer, de configurer, d'utiliser ou d'entretenir ce produit. Les utilisateurs sont tenus de se familiariser avec les instructions d'installation et de câblage et de connaître les exigences de tous les codes, lois et normes en vigueur.

Les activités telles que l'installation, les réglages, la mise en service, l'utilisation, l'assemblage, le désassemblage et la maintenance doivent être effectuées par du personnel correctement formé dans les règles de l'art.

Si cet équipement est utilisé d'une manière non spécifiée par le fabricant, la protection qu'il fournit peut être amoindrie.

En aucun cas Rockwell Automation, Inc. ne sera responsable des dommages indirects ou consécutifs issus de l'utilisation ou de l'application de cet équipement.

Les exemples et diagrammes de ce manuel sont uniquement inclus à titre d'illustration. En raison des nombreuses variables et conditions associées à une installation particulière, Rockwell Automation, Inc. ne peut pas prendre la responsabilité de l'utilisation réelle à partir des exemples et des diagrammes.

Aucune responsabilité relative aux brevets n'est prise par Rockwell Automation, Inc. concernant l'utilisation des informations, des circuits, de l'équipement ou des logiciels décrits dans ce manuel.

La reproduction de tout ou partie de ce manuel sans la permission écrite de Rockwell Automation, Inc. est interdite.

Tout au long du manuel, si nécessaire, nous utilisons des notes pour vous faire part de remarques sur la sécurité.

AVERTISSEMENT : Identifie les informations sur les pratiques ou les circonstances susceptibles de provoquer une explosion dans un environnement dangereux, ce qui peut conduire à une lésion corporelle ou au décès, à des dégâts matériels ou à une perte économique.

ATTENTION : Identifie les informations sur les pratiques ou les circonstances susceptibles de conduire à une lésion corporelle ou au décès, à des dégâts matériels ou à une perte économique. Les attentions vous aident à reconnaître un risque, à éviter un risque et à en reconnaître la conséquence.

IMPORTANT Identifie des informations essentielles pour la bonne application et la bonne compréhension du produit.

Des étiquettes peuvent également être apposées sur ou dans un équipement pour présenter des précautions spécifiques.

RISQUE D'ÉLECTROCUTION : Des étiquettes peuvent être apposées sur ou dans un équipement, par exemple, un lecteur ou un moteur, pour avertir de la présence susceptible d'une tension dangereuse.

RISQUE DE BRÛLURE : Des étiquettes peuvent être apposées sur ou dans un équipement, par exemple, un lecteur ou un moteur, pour avertir de la possibilité par les surfaces d'atteindre des températures dangereuses.

RISQUE D'ARC ÉLECTRIQUE : Des étiquettes peuvent être apposées sur ou dans un équipement, par exemple, un centre de contrôle de moteur, pour avertir d'un risque d'arc électrique. Un arc électrique peut provoquer des blessures graves ou mortelles. Portez un équipement de protection individuel (EPI) approprié. Respectez TOUTES les exigences réglementaires concernant la sécurité au travail et les équipements de protection individuels (EPI).

Rockwell Automation reconnaît que certains des termes utilisés dans notre secteur d'activité et dans cette publication ne sont pas en corrélation avec le mouvement en faveur d'un langage inclusif dans la technologie. Nous collaborons de façon proactive avec nos pairs au sein de ce secteur d'activité afin de trouver des alternatives à de tels termes et d'apporter des modifications à nos produits et à notre contenu. Veuillez excuser l'utilisation de ces termes dans notre contenu pendant la mise en œuvre de ces modifications.

Table des matières

Préface

Dans ce manuel 15
Contrôleurs pris en charge 15
Ressources supplémentaires 15
Mentions légales 16

Chapitre 1

Recherche d'informations sur les instructions et les éléments de schéma à relais

Blocs d'instructions 19
- Opérateurs 19
- Fonctions 19
- Blocs de fonctions 21

Jeu d'instructions par ordre alphabétique 22

Chapitre 2

Langage à relais (LD)

Programme de schémas à relais (LD) 29
Environnement de développement des programmes LD 29
- Barre d'outils Instructions (LD) 30
  - Ajouter des éléments d'instruction à partir de la barre d'outils Instructions 30

Chapitre 3

Éléments de langage à relais (LD)

Ligne de programme 33
- Ajouter une ligne de programme à un programme LD 33
- Étiquettes de ligne de programme 35
- Commentaires de ligne 36

Branche 37
- Ajout d'une branche à un programme LD 38

Bloc d'instructions (LD) 40
Bobine 40
- Ajout d'éléments bobine 41
- Bobine directe 43
- Bobine inversée 43
- Bobine à impulsion sur front descendant 44
- Bobine à impulsion sur front montant 45
- Bobine d'enclenchement 45
- Bobine de déclenchement 46

Contact 47
- Ajouter un contact à un programme LD 47
- Contact normalement ouvert 50
- Contact normalement fermé 50
- Contact à impulsion sur front montant 51
- Contact à impulsion sur front descendant 51

Retour 52
saut 53
Blocs d'instruction dans les programmes LD 54
Utilisation de l'éditeur de programmation LD 55
Exemples de programme de schémas à relais (LD) 56
Raccourcis clavier LD 57

Chapitre 4

Instruction d'alarme

LIM_ALRM (alarme limite) 61

Chapitre 5

Instructions arithmétiques

ABS (valeur absolue) 65
ACOS (arc cosinus de source) 67
ACOS_LREAL (arc cosinus d'une valeur REAL longue) 68
Addition 70
ASIN (arc sinus) 71
ASIN_LREAL (arc sinus d'une valeur REAL longue) 73
ATAN (arc tangente) 74
ATAN_LREAL (arc tangente d'une valeur REAL longue) 75
COS (cosinus) 77
COS_LREAL (cosinus d'une valeur réelle longue) 78
Division 80
EXPT (exposant) 81
LOG (logarithme de base 10) 83
MOD (modulo) 84
MOV (déplacer) 86
Multiplication 87
Neg (négation) 89
POW (augmentation puissance) 89
RAND (valeur aléatoire) 91
SIN (sinus) 93
SIN_LREAL (sinus d'une valeur réelle longue) 94
SQRT (racine carrée) 96
Soustraction 97
TAN (tangente) 98
TAN_LREAL (tangente d'une valeur réelle longue) 100
TRUNC (troncation) 101

Chapitre 6

Instructions de port série ASCII

ABL (test ASCII pour la ligne de la mémoire tampon) 103
ACL (vidage de la mémoire tampon ASCII) 105
AHL (lignes de contrôle de flux ASCII) 107

ACB (caractères ASCII dans la mémoire tampon) 109
ARD (lecture ASCII) 111
ARL (ligne de lecture ASCII) 114
AWA (ajout en écriture ASCII) 116
AWT (écriture ASCII) 118
Détail des paramètres ASCII 120
Type de données ABLACB 120
Type de données ACL 121
Type de données AHL ChannelSts 121
Type de données AHLI 121
Type de données ARDARL 122
Type de données AWAAWT 122

Chapitre 7

Instructions binaires

AND_MASK (masque avec opérateur AND) 123
BSL (déplacement à gauche de bits) 125
BSR (déplacement à droite à droite de bits) 128
NOT_MASK (masque NOT de bit à bit) 131
OR_MASK (masque OR bit à bit) 132
ROL (pivoter à gauche) 134
ROR (pivoter à droite) 135
SHL (décalage à gauche) 137
SHR (décalage à droite) 139
XOR_MASK (masque OR exclusif) 140

Chapitre 8

Instructions booléennes

F_TRIG (détection d'un front descendant) 143
R_TRIG (détecteur de front montant) 145
RS (réinitialisation/définition) 146
OR 148
AND 148
XOR (OR exclusif) 149
NOT 150
SR (définition/réinitialisation) 150
TTABLE (tableau de vérité) 152
MUX8B (multiplexeur de 8 entrées BOOL) 154
MUX4B (multiplexeur de 4 entrées BOOL) 158

Chapitre 9

Instructions de communication

COM_IO_WDOG 161
MSG_CIPGENERIC (message générique du protocole CIP) 163
Type de données CIPAPPCFG 166

Type de données CIPCONTROLCFG 166
Type de données CIPSTATUS 167
Bits de statut CIPSTATUS 167
Codes d'erreur CIPSTATUS 168
Type de données CIPTARGETCFG 169
Chemin d'accès cible pour la messagerie CIP 170
Connexions de message CIP/EIP 170
Temporisateurs de message CIP 171
MSG_CIPSYMBOLIC (message symbolique du protocole CIP) 171
Type de données CIPSYMBOLICCFG 173
Syntaxe de l'écriture/lecture symbolique 174
Taille des paquets de données prise en charge pour la fonction Série CIP 175
MSG_MODBUS (message Modbus) 175
Type de données MODBUSLOCPARA 178
Processus d'exécution des messages (Ligne = TRUE) 180
Type de données MODBUSTARPARA 181
MSG_MODBUS2 (message MODBUS/TCP) 181
Type de données MODBUS2LOCPARA 184
Type de données MODBUS2TARPARA 185
MSG_PCCC (message de commandes de communication de contrôleur programmable) 187
Type de données CIPCONTROLCFG 189
Type de données PCCCCFG 190
Type de données TARGETCFG 191
Type de données CIPSTATUS 193
Bits de statut CIPSTATUS 193
Connexions de message CIP/EIP 194
Temps du message dépassé 195
Codes d'erreur MSG_PCCC 195
Exemple : Créez un programme de messagerie MSG_PCCC pour y écrire une valeur et lire une valeur à partir d'une adresse dans MicroLogix 198
Exemple d'écriture PCCC : Ajout d'un bloc de fonctions MSG_PCCC, de variables et d'un contact 198
Exemple de lecture PCCC : Ajout d'un bloc de fonctions PCCC_MSG, de variables et d'un contact 201
Processus d'exécution des messages et diagrammes de temporisation 205
Processus d'exécution de message (généralités) 205
Processus d'exécution des messages (Ligne = TRUE) 206

Diagramme de temporisation d'exécution des messages (Ligne = VRAI) 207
Processus d'exécution des messages (Ligne = FALSE) 208
Diagramme de temporisation d'exécution des messages (ligne = FALSE) 209
Processus d'exécution des messages (erreur) 210
Diagramme de temporisation d'exécution des messages (erreur) 211
Diagramme de temporisation d'exécution des messages (erreur de configuration PCCC) 211
Diagramme de temporisation d'exécution de message (message abandonné) 212
Utilisation des blocs de fonction de communication (message) 213
Configuration des valeurs de données d'objet pour les messages explicites (MSG_CIPGENERIC) 213
Exemple : création d'un programme de messagerie MSG_CIPGENERIC pour lire des données depuis un contrôleur . 215
Identification des valeurs initiales pour les variables d'entrée (MSG_CIPGENERIC) 216
Ajout de variables et d'un bloc fonctionnel MSG_CIPGENERIC 217
Configuration des valeurs initiales pour les variables 218
Ajout d'un contact et d'une bobine 220
Ajout d'un bloc fonctionnel COP, de variables et d'un contact (MSG_CIPGENERIC) 221
Vérification de la configuration IP sur le contrôleur B..... 222
Exemple : création d'un programme de messagerie MSG_CIPSYMBOLIC pour écrire une valeur dans une variable....223
Ajout d'un bloc fonctionnel COP, de variables et d'un contact (MSG_CIPSYMBOLIC) 224
Ajout d'un opérateur Égal à et d'une bobine..... 225
Ajout d'un bloc fonctionnel MSG_CIPSYMBOLIC, de variables et d'un contact 226
Vérification de la configuration IP sur le contrôleur B..... 229
Création d'une variable globale pour le contrôleur B..... 230
Examen de l'ensemble des résultats du programme..... 230
Exemple : configuration d'une communication Modbus pour lire à partir d'un variateur et écrire dans un variateur.....232
Prise en charge du protocole de communication 235
Canaux de communication intégrés 237

Chapitre 10

Instructions de comparaison

Équal 239

Supérieur à 242
Supérieur ou égal à 243
Inférieur à 244
Inférieur ou égal à 246
Non égal 247

Chapitre 11

Instructions de compteur

CTD (comptage décroissant) 249
CTU (comptage croissant) 251
CTUD (comptage croissant/décroissant) 252

Chapitre 12

Instructions de conversion des données

ANY_TO_BOOL 255
ANY_TO_BYTE 256
ANY_TO_DATE 257
ANY_TO_DINT 259
ANY_TO_DWORD 260
ANY_TO_INT 261
ANY_TO_LINT 262
ANY_TO_LREAL 263
ANY_TO_LWORD 264
ANY_TO_REAL 265
ANY_TO_SINT 266
ANY_TO_STRING 267
ANY_TO_TIME 268
ANY_TO_UDINT 269
ANY_TO_UINT 270
ANY_TO_ULINT 271
ANY_TO_USINT 272
ANY_TO_WORD 273

Chapitre 13

Instructions de manipulation des données

AVERAGE 275
COP 276
Copie vers un type de données différent 280
Exemple de tableau de chaîne COP 280
MIN (minimum) 282
MAX (maximum) 283

Chapitre 14

Instructions de compteur haute vitesse (HSC)

Qu'est-ce qu'un compteur haute vitesse ? 287
HSC (Comptage rapide) 288

Valeurs HSCCmd 290
Type de données HSCAPP 290
Type de données HSCSTS 294
Type de données PLS 301
Type de données HSCE_CHANNEL 302
Type de données HSCE_STS 303
Type de données PLS_HSCE 303
HSC_SET_STS (état défini du comptage rapide) 304
Utilisation des instructions de comptage rapide 306
Mise à jour des données d'application HSC 306
Boîte de dialogue Interruption par l'utilisateur de compteur haute vitesse (HSC) 307
Configuration d'une interruption par l'utilisateur du comptage rapide (HSC) 307
Ajout et configuration d'une interruption utilisateur du compteur haute vitesse (HSC) 308
Propriétés d'interruption HSC 308
Paramètres d'interruption HSC 310
Configuration d'un interrupteur à came rotatif (PLS) 311
Exemple : Création d'un programme de compteur haute vitesse (HSC) 312
Création d'une logique à relais et ajout de variables 313
Affectation de valeurs aux variables HSC 316
Affectation de valeurs aux variables et construction du programme 317
Test du programme et utilisation du compteur haute vitesse (HSC) 319
Add a Programmable Limit Switch (PLS) function 321
Exemple : Commutateur limite programmable (PLS) activé 322

Chapitre 15

Instructions HSCE

HSCE 325
HSCE_CFG 328
HSCE_CFG_PLS 330
HSCE_READ_STS 332
HSCE_SET_STS 333
Codes d'erreur HSCE 335

Chapitre 16

Instructions d'entrée/sortie

LCD 340
LCD_BKLT_REM (rétro-éclairage LCD à distance) 342
LCD_REM (LCD à distance) 343

RHC (lecture de l'horloge haute vitesse) 346
RPC (lire la somme de contrôle du programme) 348
DLG (journal de données) 349
IIM (entrée immédiate) 350
IOM (sortie immédiate) 352
KEY_READ (lit des touches sur LCD) 355
KEY_READ_REM (lecture des touches pour un LCD à distance) 357
MM_INFO (informations du module mémoire) 359
Type de données MMINFO 361
MODULE_INFO 362
MODULE_INFO - Informations relatives au module enfichable et au module d'extension 366
Diagrammes de temporisation d'instructions MODULE_INFO 368
PLUGIN_INFO (informations sur le module enfichable) 373
PLUGIN_READ (lecture de module enfichable) 375
PLUGIN_RESET (réinitialisation du module enfichable) 377
PLUGIN_WRITE (écriture vers un module enfichable) 379
RCP (recette) 381
RTC_READ (lecture de l'horloge en temps réel) 383
RTC_SET (définir l'horloge en temps réel) 385
SYS_INFO 387
Type de données SYS_INFO 389
TRIMPOT_READ (lecture du potentiomètre d'ajustement) 390
Définition de l'ID de potentiomètre d'ajustement 391

Chapitre 17

Instructions d'interruption

STIS (sélectionner un démarrage temporisé) 393
UIC (effacer le bit de perte d'interruption) 395
UID (désactiver l'interruption) 396
UIE (activer l'interruption) 398
UIF (vider une interruption imminente) 400

Chapitre 18

Instructions de contrôle de mouvement

Règles générales propres aux blocs fonctionnels de contrôle du mouvement 404
Détails des paramètres des blocs fonctionnels de contrôle du mouvement 406
États des axes de contrôle du mouvement 406
Numéros des paramètres des blocs fonctionnels de contrôle du mouvement 408
Identifiants des erreurs propres aux blocs fonctionnels de contrôle du mouvement 409

Scénarios des erreurs au niveau de l'axe 411
Type de données AXIS_REF 411
Type de données FB_AXIS_REF 412
Variables d'axe 413
MC_AbortTrigger (déclencheur d'abandon du contrôle des mouvements) 413
MC_Halt (arrêt du contrôle des mouvements) 416
MC_Home (origine du contrôle des mouvements) 419
MC_MoveAbsolute (déplacement absolu du contrôle des mouvements) 422
MC_MoveRelative (déplacement relatif du contrôle des mouvements) 426
MC_MoveVelocity (vitesse de déplacement du contrôle des mouvements) 430
MC_Power (puissance du contrôle des mouvements) 435
MC_ReadActualPosition (contrôle des mouvements, lecture de la position réelle) 438
MC_ReadActualVelocity (contrôle des mouvements, lecture de la vitesse réelle) 442
MC_ReadAxisError (contrôle des mouvements, lecture de l'erreur de l'axe) 445
Codes d'erreur AxisErrorID 447
MC_ReadBoolParameter (contrôle des mouvements, lecture du paramètre BOOL) 449
MC_ReadParameter (contrôle des mouvements, lecture d'un paramètre) 451
MC_ReadStatus (contrôle des mouvements, lecture d'un état) 454
MC_Reset (contrôle des mouvements, réinitialisation) 458
MC_SetPosition (contrôle des mouvements, définition de position) 461
MC_Stop (contrôle des mouvements, arrêt) 463
MC_TouchProbe (contrôle des mouvements, sonde tactile) 466
MC_WriteBoolParameter (contrôle des mouvements, écriture du paramètre BOOL) 470
MC_WriteParameter (contrôle des mouvements, écriture du paramètre) 473

Chapitre 19

Instructions de contrôle de traitement

DERIVATE 477
FFL (chargement FIFO) 479
Diagrammes de temporisation d'instruction FFL et FFU 483
FFU (déchargement FIFO) 487
HYSTER (hystérésis) 492

INTEGRAL 494
AND 499
LFL (chargement LIFO) 500
LFU (déchargement LIFO) 502
PWM (Pulse Width Modulation) 505
SCALER (échelle) 507
STACKINT (pile d'entiers) 509
SCL 512
Exemples de diagrammes de temporisation d'instructions SCL 515
TND (arrêt du programme en cours) 518
LIMITE (test limite) 519

Instructions de contrôle de programme

Chapitre 20

AFI (toujours faux) 523
NOP (aucune opération) 523
SUS (attente) 523

Instructions de dérivé intégral proportionnel (PID)

Chapitre 21

Qu'est-ce que le contrôle de dérivé intégral proportionnel (PID) ? 527
IPIDCONTROLLER (contrôleur de dérivé intégral proportionnel) 527
Type de données GAIN_PID 531
Type de données AT_Param 532
Mise en œuvre du contrôle PID par le bloc fonctionnel IPIDController 532
Opération du bloc fonctionnel IPIDController 533
Utilisation de l'instruction de dérivé intégral proportionnel 534
Utilisation du réglage automatique avec le bloc de fonctions IPIDController 537
Réglage automatique de systèmes primaires et secondaires 537
Configuration du réglage automatique 538
Utilisation d'un bloc d'instruction programmable en Interruption temporisée (STI) avec la fonction de réglage automatique 544
Exemple : IPIDController avec réglage automatique 544
Exemple : Création d'une boucle de rétroaction pour la valeur manipulée 545
Exemple : Ajout d'un UDFB à un programme PID 546
Exemple : Création d'un programme IPIDController pour contrôler la température 547
Exemple : Création d'un programme IPIDController pour contrôler le niveau d'alimentation en eau 549
PID (dérivé intégral proportionnel) 552

Machine d'état d'instruction PID....556
Diagrammes de temporisation d'instructions PID....556

Chapitre 22

Instructions d'horloge en temps réel (RTC)

RTC_READ (lecture de l'horloge en temps réel) 561
RTC_SET (définir l'horloge en temps réel) 563

Chapitre 23

Instructions de socket

SOCKET_ACCEPT 568
SOCKET_CREATE 571
SOCKET_DELETE....576
SOCKET_DELETEALL....578
SOCKET_INFO 580
SOCKET_OPEN....586
SOCKET_READ 590
SOCKET_WRITE....594
Type de données SOCKADDR_CFG....598
Type de données SOCK_STATUS....598
Codes d'erreur du socket 598
Bits d'état d'instruction de socket.... 602
Diagrammes de temporisation d'instructions de socket 603
Diagrammes de transaction d'instructions de socket....610
Transactions pour un client TCP....610
Transactions pour un serveur TCP 610
Transactions pour UDP avec une connexion ouverte 610
Transactions pour UDP sans connexion ouverte 611
Diagrammes d'états de machine pour TCP.... 611
Diagrammes de machine à état pour UDP.... 613

Chapitre 24

Instructions de manipulation des chaînes

ASCII....615
CHAR (code ASCII pour un caractère de chaîne)....617
DELETE (supprimer une sous-chaîne) 618
FIND (recherche une sous-chaîne) 620
INSERT (insérer une chaîne).... 622
LEFT (extrait la partie gauche d'une chaîne).... 624
MID (extraction de la partie du centre d'une chaîne) 626
MLEN (longueur de la chaîne).... 628
RIGHT (extrait la partie droite d'une chaîne) 629
REPLACE (remplacer une sous-chaîne) 631

Chapitre 25

Instructions de temporisateur

Configuration de l'instruction de temporisateur 635
TOF (temporisateur de retard inactif) 635
TOF (temporisateur de retard actif) 637
TONOFF (retard, actif, inactif) 640
TP (temporisation d'impulsion) 642
RTO (temporisateur rémanent, retard actif) 644
DOY (vérifier l'année pour l'horloge temps réel) 646
TDF (différence de temps) 648
TOW (vérifier la semaine pour l'horloge temps réel) 650

Indice

Dans ce manuel

Ce guide fournit des informations de référence concernant le jeu d'instructions disponible pour le développement des programmes à utiliser dans les systèmes de contrôle Micro800®. Le jeu d'instructions inclut la prise en charge des langages de programmation texte structuré (ST), logique à relais (LD) et diagramme de bloc fonctionnel (FBD). De plus, les éléments de logique à relais pris en charge dans l'environnement de développement Connected Components Workbench™ sont définis.

Contrôleurs pris en charge

Connected Components Workbench inclut la prise en charge de la configuration des contrôleurs Micro800 suivants.

- 2080-LC10-12AWA
- 2080-LC10-12DWD
- 2080-LC10-12QBB
- 2080-LC10-12QWB
- 2080-LC20-20AWB
- 2080-LC20-20QBB
- 2080-LC20-20QWB
- 2080-LC30-10QVB
- 2080-LC30-10QWB
- 2080-LC30-16AWB
- 2080-LC30-16QVB
- 2080-LC30-16QWB
- 2080-LC30-24QBB
- 2080-LC30-24QVB
- 2080-LC30-24QWB
- 2080-LC30-48AWB
- 2080-LC30-48QBB
- 2080-LC30-48QVB
- 2080-LC30-48QWB
- 2080-LC50-24AWB
- 2080-LC50-24QBB
- 2080-LC50-24QVB
- 2080-LC50-24QWB
- 2080-LC50-48AWB
- 2080-LC50-48QBB
- 2080-LC50-48QVB
- 2080-LC50-48QWB
- 2080-LC50-48QWB-SIM
- 2080-L50E-24AWB
- 2080-L50E-24QBB
- 2080-L50E-24QVB
- 2080-L50E-24QWB
- 2080-L50E-48AWB
- 2080-L50E-48QBB
- 2080-L50E-48QVB
- 2080-L50E-48QWB
- 2080-L70E-24AWB
- 2080-L70E-24QBB
- 2080-L70E-24QBBN
- 2080-L70E-24QWB
- 2080-L70E-24QWBN
- 2080-LC70-24AWB
- 2080-LC70-24QBB
- 2080-LC70-24QWB

Ressources supplémentaires

Ces documents contiennent des informations complémentaires concernant les produitsRockwell Automation® associés.

Ressource	Description

Ressource	Description
Industrial Automation Wiring and Grounding Guidelines (Directives concernant le câblage et la mise à la terre dans l'automatisation industrielle), publication 1770-4.1 disponible à la page http://literature.rockwellautomation.com/idc/groups/literature/documents/in/1770-in041_-en-p.pdf.	Fournit des indications générales sur l'installation d'un système industriel Rockwell Automation.
Site web des certifications de produit, http://www.ab.com	Fournit des déclarations de conformité, des certificats et d'autres informations de certification.
Guide de migration des contrôleurs MicroLogix aux contrôleurs Micro800, disponible à l'adresse https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/2080-rm002_-en-e.pdf	Indique comment utiliser les outils logiciels pour sélectionner un contrôleur Micro800 approprié, et comment convertir les programmes MicroLogix pour qu'ils fonctionnent avec le contrôleur Micro800.
Démarrage rapide du pack de démarrage des contrôleurs Micro800, disponible à l'adresse https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/2080-qs004_-en-e.pdf	Indique comment utiliser un contrôleur Micro800 avec un terminal PanelView 800.
Configuration des contrôleurs Micro800 sur FactoryTalk Gateway, disponible à l'adresse https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/2080-qs005_-en-e.pdf	Fournit des instructions de démarrage rapide pour la configuration d'un contrôleur Micro800 sur FactoryTalk Linx Gateway.

Vous pouvez afficher ou télécharger des publications à l'adresse http://www.rockwellautomation.com/literature. Pour commander des exemplaires papier de la documentation technique, contactez votre distributeur ou votre représentant commercial local Rockwell Automation.

Mentions légales

Rockwell Automation publie des avis juridiques, tels que des politiques de confidentialité, des accords de licence, des divulgations de marques et d'autres conditions sur la page Mentions légales du site Web Rockwell Automation.

Accord sur les logiciels et services cloud

Consultez et acceptez l'accord sur les logiciels et services cloud Rockwell Automation ici.

Licences de logiciels open source

Le logiciel inclus dans ce produit contient des éléments sous copyright octroyés sous licence en tant qu'éléments open source.

Vous pouvez consulter la liste complète de tous les logiciels open source utilisés dans ce produit et les licences correspondantes en ouvrant le fichier oss_license.txt situé dans le dossier OPENSOURCE de votre produit sur votre disque dur. Ce fichier est organisé selon les sections suivantes :

- Composants
  Comprend le nom du composant open source, son numéro de version et le type de licence.

- Texte de copyright
  Comprend le nom du composant open source, son numéro de version et la déclaration de droits d'auteurs.
- Licences
  Comprend le nom de la licence, la liste des composants open source qui exécutent la licence et les conditions de la licence.

L'emplacement par défaut de ce fichier est :

C:\Program Files (x86)\Common Files\Rockwell\Help\<*product name*>\Release Notes\OPENSOURCE\oss_licenses.txt.

Vous pouvez obtenir le code source correspondant des packages open source inclus dans ce produit sur le ou les sites Web respectifs des projets. Vous pouvez également obtenir le code source complet correspondant en contactant Rockwell Automation via notre formulaire de **contact** sur le site Web de Rockwell Automation : http://www.rockwellautomation.com/global/about-us/contact/contact.page. Veuillez inclure « Open Source » dans le texte de la demande.

Licences logicielles commerciales

Le tableau suivant répertorie les composants logiciels sous licence commerciale dans Connected Components Workbench.

Composant	Copyright
Actipro v17.1.0651	Copyright (c) 2002-2013 Actipro Software LLC
ag-grid v21.0.1	Copyright (c) 2015-2020 AG GRID LTD

Recherche d'informations sur les instructions et les éléments de schéma à relais

Connected Components Workbench contient un ensemble complet d'instructions avec des structures et des tableaux, des environnements de développement pour des programmes de logique de schéma à relais, de texte structuré, de Diagramme de Bloc Fonction et de bloc de fonctions défini par l'utilisateur.

De plus, Connected Components Workbench comprend des outils de configuration d'interface utilisateur pour les contrôleurs Micro800, les variateurs PowerFlex®, les appareils de relais de sécurité, les terminaux graphiques PanelView™ Component et les options de connectivité en série ou en réseau.

Pour obtenir plus d'informations sur une instruction en particulier, notamment sa description, le détail de ses paramètres et des exemples de langage, recherchez l'instruction dans la table des matières ou dans les rubriques de référence suivantes.

- Référence aux instructions par ordre alphabétique sur la page 22

Pour obtenir une description des éléments de schéma à relais utilisés dans Connected Components Workbench, consultez la section suivante :

- Références du langage de schémas à relais (LD) sur la page 29

Blocs d'instructions

Le jeu d'instructions de Connected Components Workbench comprend des blocs d'instruction conformes à la norme CEI 61131-3 pour les contrôleurs Micro800. Les blocs d'instruction incluent collectivement des opérateurs sur la page 19, des fonctions sur la page 19 et des blocs fonctionnels sur la page 21.

Opérateurs

Le jeu d'instructions de Connected Components Workbench comprend des blocs d'instruction conformes à la norme CEI 61131-3 pour les contrôleurs Micro800 ; des opérateurs sont inclus dans le jeu d'instructions Connected Components Workbench.

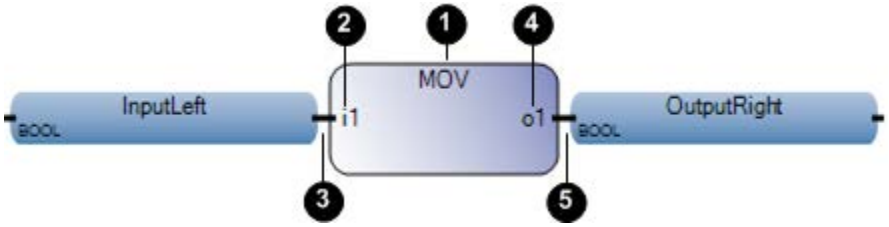
Un opérateur est une opération logique de base telle qu'une opération arithmétique, booléenne, de comparateur ou une conversion de données.

Fonctions

Les fonctions ont un ou plusieurs paramètres d'entrée et un paramètre de sortie.

Format des blocs d'instructions

Un bloc d'instructions est représenté par un rectangle unique et possède un nombre fixe de points de connexion d'entrée et de points de connexion de sortie. Un bloc d'instructions élémentaire n'a qu'une seule fonction.



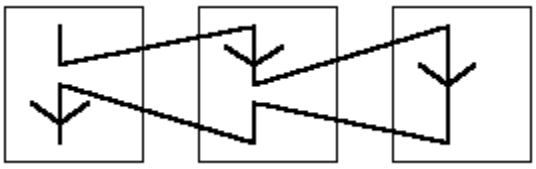
N° d'élément	Élément	Description
1	Nom de bloc	Le nom de la fonction à exécuter par le bloc d'instructions est écrit à l'intérieur de sa forme rectangulaire (en haut).
2	Entrée	Chaque entrée d'un bloc d'instructions est étiquetée et a un type défini.
3	Connexion d'entrée	Les entrées sont connectées au bord gauche.
4	Sortie	Chaque sortie d'un bloc d'instructions est étiquetée et a un type défini.
5	Connexion de sortie	Les sorties sont connectées au bord droit.

Appel d'une fonction

Connected Components Workbench ne prend pas en charge les appels de fonction récursive. Lorsqu'une fonction de la section Fonctions est appelée par elle-même ou une de ses fonctions appelées, une erreur d'exécution se produit. En outre, les fonctions ne stockent pas les valeurs locales de leurs variables locales. Comme les fonctions ne sont pas instanciées, elles ne peuvent pas appeler des blocs de fonctions.

- Une fonction peut être appelée par un programme, une fonction ou un bloc fonctionnel.
- Tout programme d'une section peut appeler une ou plusieurs fonctions. Une fonction peut avoir des variables locales.
- Une fonction ne dispose d'aucune instance, ce qui signifie que les données locales ne sont pas stockées et sont généralement perdues entre un appel et un autre.
- Du fait que l'exécution d'une fonction dépend de son programme parent, l'exécution de ce programme parent est suspendue jusqu'à ce que la fonction se termine.

Programme principal **Fonction** **Fonction**



Définition des noms de paramètre et de fonction

L'interface d'une fonction doit être explicitement définie avec un type et un nom unique pour chacun de ses paramètres d'appel (entrée) ou de retour (sortie). Une fonction a un paramètre de retour. La valeur du paramètre de retour d'un bloc fonctionnel diffère d'un langage de programmation à l'autre (FBD, LD, ST).

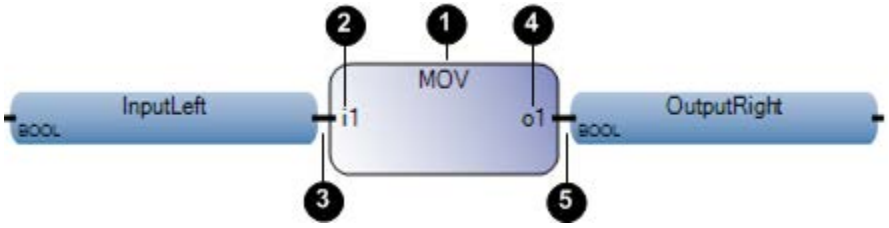
Les noms de fonction et les noms des paramètres de fonction peuvent comporter jusqu'à 128 caractères. Les noms des paramètres de fonction peuvent commencer avec une lettre ou un trait de soulignement suivi de lettres, chiffres et traits de soulignement simples.

Blocs de fonctions

Un bloc fonctionnel est un bloc d'instructions disposant de paramètres d'entrée et de sortie, travaillant sur des données internes (paramètres). Il peut être écrit en langage de texte structuré, de logique à relais ou de diagramme de bloc fonctionnel.

Format des blocs d'instructions

Un bloc d'instructions est représenté par un rectangle unique et possède un nombre fixe de points de connexion d'entrée et de points de connexion de sortie. Un bloc d'instructions élémentaire n'a qu'une seule fonction.



N° d'élément	Élément	Description
1	Nom de bloc	Le nom de la fonction à exécuter par le bloc d'instructions est écrit à l'intérieur de sa forme rectangulaire (en haut).
2	Entrée	Chaque entrée d'un bloc d'instructions est étiquetée et a un type défini.
3	Connexion d'entrée	Les entrées sont connectées au bord gauche.
4	Sortie	Chaque sortie d'un bloc d'instructions est étiquetée et a un type défini.
5	Connexion de sortie	Les sorties sont connectées au bord droit.

Appel d'un bloc fonctionnel

Lorsqu'un bloc fonctionnel est appelé dans un programme, une instance du bloc est en réalité appelée. Cette instance utilise le même code, mais les paramètres d'entrée et de sortie sont instanciés, c'est-à-dire que les variables

locales du bloc fonctionnel sont copiées pour chaque instance. Les valeurs des variables d'une instance de bloc fonctionnel sont stockées d'un cycle à l'autre.

Un bloc fonctionnel peut être appelé par un programme ou un autre bloc fonctionnel. Les fonctions n'étant pas instanciées, elles ne peuvent appeler des blocs fonctionnels.

Définition des noms de paramètre et de bloc fonctionnel

L'interface d'un bloc fonctionnel doit être explicitement définie avec un type et un nom unique pour chacun de ses paramètres d'appel (entrée) ou de retour (sortie). Les blocs fonctionnels peuvent avoir plus d'un paramètre de sortie. La valeur du paramètre de retour d'un bloc fonctionnel diffère d'un langage de programmation à l'autre (FBD, LD, ST).

Les noms de bloc fonctionnel et les noms des paramètres de bloc fonctionnel peuvent comporter jusqu'à 128 caractères. Les noms des paramètres de bloc de fonctions peuvent commencer avec une lettre ou un trait de soulignement suivi de lettres, chiffres et traits de soulignement simples.

Jeu d'instructions par ordre alphabétique

Le tableau suivant répertorie par ordre alphabétique les instructions du contrôleur Micro800 disponibles dans Connected Components Workbench ainsi que les instructions mappées dans le thème Logix.

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
- sur la page 97	SUB	Arithmétique	Opérateur	Soustrait une valeur entière, réelle ou de temps d'une autre valeur entière, réelle ou de temps.
* sur la page 87	MUL	Arithmétique	Opérateur	Multiplie plusieurs valeurs entières ou réelles.
/ sur la page 80	DIV	Arithmétique	Opérateur	Divise deux valeurs entières ou réelles.
+ sur la page 70	ADD	Arithmétique	Opérateur	Ajoute plusieurs valeurs entières, réelles, de temps ou de chaîne.
< sur la page 244	LES	Comparer	Opérateur	Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est inférieure à la seconde.
<= sur la page 246	LEQ	Comparer	Opérateur	Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est inférieure ou égale à la seconde.
<> sur la page 247	NEQ	Comparer	Opérateur	Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est égale ou non à la seconde.
= sur la page 239	EQU	Comparer	Opérateur	Teste si une valeur est égale à une autre.
> sur la page 242	GRT	Comparer	Opérateur	Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est supérieure à la seconde.
>= sur la page 243	GEQ	Comparer	Opérateur	Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est égale ou supérieure à la seconde.
ABL sur la page 103	ABL	Communications	Bloc de fonctions	Compte le nombre de caractères dans la mémoire tampon, jusqu'au caractère de fin de ligne inclus.
ABS sur la page 65	ABS	Arithmétique	Fonction	Renvoie la valeur absolue d'une valeur REAL.

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
ACB sur la page 109	ACB	Communications	Bloc de fonctions	Compte le nombre total de caractères dans la mémoire tampon.
ACL sur la page 105	ACL	Communications	Bloc de fonctions	Vide les mémoires tampons de réception et de transmission.
ACOS sur la page 67	ACS	Arithmétique	Fonction	Calcule l'arc cosinus d'une valeur REAL.
ACOS_LREAL sur la page 68	ACOS_LREAL	Arithmétique	Fonction	Calcule l'arc cosinus d'une valeur REAL longue.
AFI sur la page 523	AFI	Contrôle de programme	Fonction	Désactive temporairement une ligne de programme lors du débogage.
AHL sur la page 107	AHL	Communications	Bloc de fonctions	Définit ou réinitialise les lignes de contrôle de flux avec le modem.
AND sur la page 148	AND	Opérations booléennes	Opérateur	Effectue une opération booléenne AND (ET) entre deux ou plusieurs valeurs.
AND_MASK sur la page 123	AND_MASK	Opérations binaires	Fonction	Effectue un AND bit à bit entre deux valeurs entières.
ANY_TO_BOOL sur la page 255	ANY_TO_BOOL	Conversion de données	Fonction	Convertit une valeur non booléenne en valeur booléenne.
ANY_TO_BYTE sur la page 256	ANY_TO_BYTE	Conversion de données	Fonction	Convertit une valeur en valeur BYTE.
ANY_TO_DATE sur la page 257	ANY_TO_DATE	Conversion de données	Fonction	Convertit un type de données STRING, INTEGER, REAL ou TIME en type de données DATE.
ANY_TO_DINT sur la page 259	STOD	Conversion de données	Fonction	Convertit une valeur en entier double.
ANY_TO_DWORD sur la page 260	ANY_TO_DWORD	Conversion de données	Fonction	Convertit une valeur en valeur WORD double.
ANY_TO_INT sur la page 261	ACI	Conversion de données	Fonction	Convertit une valeur en entier.
ANY_TO_LINT sur la page 262	ANY_TO_LINT	Conversion de données	Fonction	Convertit une valeur en entier long.
ANY_TO_LREAL sur la page 263	ANY_TO_LREAL	Conversion de données	Fonction	Convertit une valeur en valeur REAL longue.
ANY_TO_LWORD sur la page 264	ANY_TO_LWORD	Conversion de données	Fonction	Convertit une valeur en valeur WORD longue.
ANY_TO_REAL sur la page 265	STOR	Conversion de données	Fonction	Convertit une valeur en valeur REAL.
ANY_TO_SINT sur la page 266	ANY_TO_SINT	Conversion de données	Fonction	Convertit une valeur en entier court.
ANY_TO_STRING sur la page 267	DTOS	Conversion de données	Fonction	Convertit une valeur en valeur STRING.
ANY_TO_TIME sur la page 268	ANY_TO_TIME	Conversion de données	Fonction	Convertit une valeur en type de données TIME.
ANY_TO_UDINT sur la page 269	ANY_TO_UDINT	Conversion de données	Fonction	Convertit une valeur en entier double non signé.
ANY_TO_UINT sur la page 270	ANY_TO_UINT	Conversion de données	Fonction	Convertit une valeur en entier non signé.
ANY_TO_ULINT sur la page 271	ANY_TO_ULINT	Conversion de données	Fonction	Convertit une valeur en entier long non signé.
ANY_TO_USINT sur la page 272	ANY_TO_USINT	Conversion de données	Fonction	Convertit une valeur en entier court non signé.
ANY_TO_WORD sur la page 273	ANY_TO_WORD	Conversion de données	Fonction	Convertit une valeur en valeur WORD.
ARD sur la page 111	ARD	Communications	Bloc de fonctions	Lit les caractères de la mémoire tampon d'entrée et les place dans une chaîne.
ARL sur la page 114	ARL	Communications	Bloc de fonctions	Lit une ligne de caractères à partir de la mémoire tampon d'entrée et insère ces caractères dans une chaîne.
ASCII sur la page 615	ASCII	Manipulation de chaîne de caractères	Fonction	Renvoie le code ASCII des caractères d'une chaîne. Caractère -> code ASCII.
ASIN sur la page 71	ASN	Arithmétique	Fonction	Calcule l'arc sinus d'une valeur REAL.
ASIN_LREAL sur la page 73	ASN_LREAL	Arithmétique	Fonction	Calcule l'arc sinus d'une valeur REAL longue.
ATAN sur la page 74	ATN	Arithmétique	Fonction	Calcule l'arc tangente d'une valeur REAL.
ATAN_LREAL sur la page 75	ATAN_LREAL	Arithmétique	Fonction	Calcule l'arc tangente d'une valeur REAL longue.
AVERAGE sur la page 275	AVE	Manipulation de données	Bloc de fonctions	Calcule une moyenne mobile sur un nombre d'échantillons définis.
AWA sur la page 116	AWA	Communications	Fonction	Écrit une chaîne avec deux caractères (configurés par l'utilisateur) ajoutés à un appareil externe.
AWT sur la page 118	AWT	Communications	Fonction	Écrit les caractères provenant d'une chaîne source vers un appareil externe.
BSL sur la page 125	BSL	Opérations binaires	Bloc de fonctions	Déplace un bit dans un élément du tableau vers la gauche.

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
BSR sur la page 128	BSR	Opérations binaires	Bloc de fonctions	Déplace un bit dans un élément du tableau vers la droite.
CHAR sur la page 617	CHAR	Manipulation de chaîne de caractères	Fonction	Renvoie une chaîne d'un caractère pour un code ASCII. Code ASCII -> caractère.
COM_IO_WDOG sur la page 161	COM_IO_WDOG	Communications	Bloc de fonctions	Surveille les communications vers le contrôleur.
COP sur la page 276	COP	Conversion de données	Bloc de fonctions	Copie les données binaires de l'élément source dans l'élément de destination.
COS sur la page 77	COS	Arithmétique	Fonction	Calcule le cosinus d'une valeur REAL.
COS_LREAL sur la page 78	COS_LREAL		Fonction	Calcule le cosinus d'une valeur REAL longue.
CTD sur la page 249	CTD	Compteurs	Fonction	Décompte des entiers, 1 par 1, à partir d'une valeur donnée jusqu'à 0.
CTU sur la page 251	CTU	Compteurs	Fonction	Décompte des entiers, 1 par 1, de 0 jusqu'à une valeur donnée.
CTUD sur la page 252	CTUD	Compteurs	Fonction	Décompte des entiers, 1 par 1, de 0 jusqu'à une valeur donnée ou d'une valeur donnée jusqu'à 0.
DELETE sur la page 618	DELETE	Manipulation de chaîne de caractères	Fonction	Supprime des caractères dans une chaîne.
DERIVATE sur la page 477	DERIVATE	Contrôle de procédés	Bloc de fonctions	Différencie une valeur réelle pendant un temps de cycle défini.
DLG sur la page 349	DLG	Entrée/Sortie	Bloc de fonctions	Écrit des valeurs de variables depuis le moteur d'exécution dans un fichier de journalisation des données sur une carte SD.
DOY sur la page 646	DOY	Heure	Fonction	Active une sortie si la valeur de l'horloge temps réel est comprise dans la plage de valeurs propre au paramétrage de l'année.
EXPT sur la page 81	EXPT	Arithmétique	Fonction	Calcule la valeur réelle d'un nombre de base à la puissance de l'exposant de type entier.
F_TRIG sur la page 143	OSF	Opérations booléennes	Bloc de fonctions	Détecte un front descendant d'une variable booléenne.
FFL sur la page 487	FFL	Contrôle de procédés	Bloc de fonctions	Charge des données de 8 bits, 16 bits, 32 bits ou 64 bits dans un tableau créé par l'utilisateur appelé une pile FIFO.
FFU sur la page 487	FFU	Contrôle de procédés	Bloc de fonctions	Décharge des données 8 bits, 16 bits, 32 bits ou 64 bits d'un tableau créé par l'utilisateur et appelé pile FIFO (First In First Out, premier entré, premier sorti) dans le même ordre que les données ont été chargées avec l'instruction FFL.
FIND sur la page 620	FIND	Manipulation de chaîne de caractères	Fonction	Localise et communique la position de sous-chaînes au sein de chaînes.
HSC sur la page 288	HSC	Entrée/Sortie	Bloc de fonctions	HSC applique les valeurs hautes prédéfinies, les valeurs basses prédéfinies et les valeurs de source de sortie au compteur haute vitesse.
HSC_SET_STS sur la page 304	HSC_SET_STS	Entrée/Sortie	Bloc de fonctions	HSC_SET_STS définit ou réinitialise manuellement l'état de décompte HSC.
HSCE sur la page 325	HSCE	Entrée/Sortie	Bloc de fonctions	HSCE démarre, arrête et lit la valeur de l'accumulateur.
HSCE_CFG sur la page 328	HSCE_CFG	Entrée/Sortie	Bloc de fonctions	HSCE_CFG est la configuration de comptage rapide.
HSCE_CFG_PLS sur la page 330	HSCE_CFG_PLS	Entrée/Sortie	Bloc de fonctions	HSCE_CFG_PLS est la configuration PLS de comptage rapide.
HSCE_READ_STS sur la page 332	HSCE_READ_STS	Entrée/Sortie	Bloc de fonctions	HSCE_READ_STS lit l'état du comptage rapide.
HSCE_SET_STS sur la page 333	HSCE_SET_STS	Entrée/Sortie	Bloc de fonctions	HSCE_SET_STS définit/réinitialise manuellement l'état de comptage rapide.
HYSTER sur la page 492	HYSTER	Contrôle de procédés	Bloc de fonctions	Hystérésis de type booléen sur la différence entre des réels.
IIM sur la page 350	IIM	Entrée/Sortie	Bloc de fonctions	Met à jour les entrées avant la scrutation de sortie normale.
INSERT sur la page 622	INSERT	Manipulation de chaîne de caractères	Fonction	Insère des sous-chaînes à des positions définies par l'utilisateur dans des chaînes.
INTEGRAL sur la page 494	INTEGRAL	Contrôle de procédés	Bloc de fonctions	Intègre une valeur réelle pendant le temps de cycle défini.
IOM sur la page 352	IOM	Entrée/Sortie	Bloc de fonctions	Met à jour les sorties avant la scrutation de sortie normale.

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
IPIDCONTROLLER sur la page 527	IPIDCONTROLLER	Contrôle de procédés	Bloc de fonctions	Configure et contrôle les entrées et sorties utilisées pour la logique de dérivé intégral proportionnel (PID).
KEY_READ sur la page 355		Entrée/Sortie	Bloc de fonctions	Micro810 uniquement. Lit l'état des touches sur le module LCD facultatif lorsque l'affichage utilisateur est actif.
KEY_READ_REM sur la page 357	KEY_READ_REM	Entrée/Sortie	Bloc de fonctions	Micro820 uniquement. Lit l'état des touches sur le module LCD à distance facultatif lorsque l'affichage utilisateur est actif.
LCD sur la page 340		Entrée/Sortie	Fonction	Micro810 uniquement. Affiche une chaîne ou un nombre sur un écran LCD.
LCD_BKLT_REM sur la page 342	LCD_BKLT_REM	Entrée/Sortie	Fonction	Définit les paramètres de rétro-éclairage du LCD à distance dans un programme utilisateur.
LCD_REM sur la page 343	LCD_REM	Entrée/Sortie	Fonction	Affiche les messages définis par l'utilisateur pour le LCD à distance.
LEFT sur la page 624	LEFT	Manipulation de chaîne de caractères	Fonction	Extrait les caractères de la partie gauche d'une chaîne.
LFL (chargement LIFO) sur la page 500	LFL	Contrôle de procédés	Bloc de fonctions	Charge des données de 8 bits, 16 bits, 32 bits ou 64 bits dans un tableau créé par l'utilisateur appelé une pile LIFO.
LFU (déchargement LIFO) sur la page 502	LFU	Contrôle de procédés	Bloc de fonctions	Décharge des données 8 bits, 16 bits, 32 bits ou 64 bits d'un tableau créé par l'utilisateur et appelé pile LIFO (Last In First Out, dernier entré, premier sorti) dans le même ordre que les données ont été chargées avec l'instruction LFL.
LIM_ALRM sur la page 61	LIM	Alarme	Bloc de fonctions	Alarme avec hystérésis sur une valeur réelle pour des limites haute et basse.
LIMIT sur la page 519	LIMIT	Contrôle de procédés	Fonction	Restreint des valeurs entières à un intervalle donné.
LOG sur la page 83	LOG	Arithmétique	Fonction	Calcule le logarithme (base 10) d'une valeur REAL.
MAX sur la page 283	MAX	Manipulation de données	Fonction	Calcule la valeur maximale de deux entiers.
MC_AbortTrigger sur la page 413	MC_AbortTrigger	Mouvement	Bloc de fonctions	Interrompt les blocs de fonctions de mouvement connectés à des événements de déclenchement.
MC_Halt sur la page 416	MC_Halt	Mouvement	Bloc de fonctions	Commande un arrêt contrôlé du mouvement dans des conditions de fonctionnement normales.
MC_Home sur la page 419	MAH	Mouvement	Bloc de fonctions	Commande l'axe pour effectuer une séquence *<rechercher origine>*.
MC_MoveAbsolute sur la page 422	MAM	Mouvement	Bloc de fonctions	Commande un mouvement contrôlé vers une position absolue spécifiée.
MC_MoveRelative sur la page 426	MC_MoveRelative	Mouvement	Bloc de fonctions	Commande un mouvement contrôlé d'une distance spécifique relative à la position actuelle au moment de l'exécution.
MC_MoveVelocity sur la page 430	MCD	Mouvement	Bloc de fonctions	Commande un mouvement contrôlé sans fin à une vélocité spécifiée.
MC_Power sur la page 435	MSO	Mouvement	Bloc de fonctions	Contrôle la phase d'alimentation ON ou OFF.
MC_ReadActualPosition sur la page 442	MC_ReadActualPosition	Mouvement	Bloc de fonctions	Renvoie la position réelle de l'axe retour codeur.
MC_ReadActualVelocity sur la page 442	MC_ReadActualVelocity	Mouvement	Bloc de fonctions	Renvoie la vitesse réelle de l'axe retour codeur.
MC_ReadAxisError sur la page 445	MC_ReadAxisError	Mouvement	Bloc de fonctions	Lit les erreurs de l'axe non liées aux blocs d'instruction de contrôle des mouvements.
MC_ReadBoolParameter sur la page 449	MC_ReadBoolParameter	Mouvement	Bloc de fonctions	Renvoie la valeur d'un paramètre de type BOOL spécifique à un fournisseur.
MC_ReadParameter sur la page 451	MC_ReadParameter	Mouvement	Bloc de fonctions	Renvoie la valeur d'un paramètre de type REAL spécifique à un fournisseur.

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
MC_ReadStatus sur la page 454	MC_ReadStatus	Mouvement	Bloc de fonctions	Renvoie l'état de l'axe en fonction du mouvement actuellement en cours.
MC_Reset sur la page 458	MAFR	Mouvement	Bloc de fonctions	Fait passer l'axe de l'état Arrêt suite à une erreur à l'état Immobilisation en réinitialisant toutes les erreurs propres à l'axe.
MC_SetPosition sur la page 461	MRP	Mouvement	Bloc de fonctions	Bascule le système de coordonnées d'un axe par la manipulation de la position réelle.
MC_Stop sur la page 463	MAS	Mouvement	Bloc de fonctions	Commande un arrêt contrôlé du mouvement et fait passer l'axe à l'état Arrêt.
MC_TouchProbe sur la page 466	MC_TouchProbe	Mouvement	Bloc de fonctions	Enregistre la position de l'axe lors du déclenchement d'un événement.
MC_WriteBoolParameter sur la page 470	MC_WriteBoolParameter	Mouvement	Bloc de fonctions	Modifie la valeur d'un paramètre de type BOOL spécifique à un fournisseur.
MC_WriteParameter sur la page 473	MC_WriteParameter	Mouvement	Bloc de fonctions	Modifie la valeur d'un paramètre de type REAL spécifique à un fournisseur.
MID sur la page 626	MID	Manipulation de chaîne de caractères	Fonction	Extrait les caractères de la partie centrale d'une chaîne.
MIN sur la page 282	MIN	Manipulation de données	Fonction	Calcule la valeur minimale de deux entiers.
MLEN sur la page 628	MLEN	Manipulation de chaîne de caractères	Fonction	Calcule la longueur d'une chaîne.
MM_INFO sur la page 359	MM_INFO	Entrée/Sortie	Bloc de fonctions	Lit les informations d'en-tête du module mémoire.
MOD sur la page 84	MOD	Arithmétique	Fonction	Effectue un calcul du modulo sur les valeurs entières.
MODULE_INFO sur la page 362	MODULE_INFO	Entrée/Sortie	Bloc de fonctions	Lit les informations de module à partir d'un module enfichable ou d'un module d'extension.
MOV sur la page 86	MOV	Arithmétique	Opérateur	Affecte la valeur d'entrée à la sortie.
MSG_CIPGENERIC sur la page 163	MSG	Communications	Fonction	Envoie un message explicite générique CIP.
MSG_CIPSYMBOLIC sur la page 171	MSG_CIPSYMBOLIC	Communications	Fonction	Envoie un message explicite symbolique CIP.
MSG_MODBUS sur la page 175	MSG_MODBUS	Communications	Fonction	Envoie un message Modbus.
MSG_MODBUS2 sur la page 181	MSG_MODBUS2	Communications	Fonction	Envoie un message MODBUS/TCP via un canal Ethernet.
MSG_PCCC sur la page 187	MSG_PCCC	Communications	Fonction	Envoie un message PCCC via un canal Ethernet ou un port série.
MUX4B sur la page 158	MUX4B	Booléen	Fonction	Multiplexeur entre quatre entrées BOOL, une valeur BOOL de sortie.
MUX8B sur la page 154	MUX8B	Booléen	Fonction	Multiplexeur entre huit entrées BOOL, une valeur BOOL de sortie.
Neg sur la page 89	NEG	Arithmétique	Opérateur	Convertit une valeur en valeur inversée.
NOP sur la page 523	NOP	Contrôle de programme	Fonction	Fonctionne comme un paramètre fictif.
NOT sur la page 150	NOT	Opérations booléennes	Opérateur	Convertit des valeurs booléennes en valeurs inversées.
NOT_MASK sur la page 131	NOT_MASK	Opérations binaires	Fonction	Masque de négation bit à bit entier, inverse une valeur de paramètre.
OR sur la page 148	OR	Opérations booléennes	Opérateur	Opérateur booléen OR (OU) entre deux valeurs ou plus.
OR_MASK sur la page 132	OR_MASK	Opérations binaires	Fonction	Masque avec opérateur OR bit à bit entier, active les bits.
PID sur la page 552	PID	Contrôle de procédés	Bloc de fonctions	Une instruction de sortie qui contrôle les propriétés physiques telles que la température, la pression, le niveau de liquide ou le débit utilisant les boucles de processus.
PLUGIN_INFO sur la page 373	PLUGIN_INFO	Entrée/Sortie	Bloc de fonctions	Récupère les informations du module à partir d'un module enfichable générique (à l'exclusion du module mémoire).
PLUGIN_READ sur la page 375	PLUGIN_READ	Entrée/Sortie	Bloc de fonctions	Lit les données à partir d'un module enfichable générique (à l'exclusion du module mémoire).
PLUGIN_RESET sur la page 377	PLUGIN_RESET	Entrée/Sortie	Bloc de fonctions	Réinitialise un module enfichable générique, réinitialisation matérielle (à l'exclusion du module mémoire).

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
PLUGIN_WRITE sur la page 379	PLUGIN_WRITE	Entrée/Sortie	Bloc de fonctions	Écrit les données sur un module enfichable générique (à l'exclusion du module mémoire).
POW sur la page 89	XPY	Arithmétique	Fonction	Calcule la valeur d'un nombre réel à la puissance de l'exposant de type REAL.
PWM sur la page 505	PWM	Contrôle de procédés	Bloc de fonctions	Active ou désactive la sortie PWM (Pulse Width Modulation) pour un canal PWM configuré.
R_TRIG sur la page 145	OSR	Opérations booléennes	Bloc de fonctions	Détecte un front montant d'une variable booléenne.
RAND sur la page 91	RAND	Arithmétique	Fonction	Calcule des valeurs d'entiers aléatoires à partir d'une plage définie.
RCP sur la page 381	RCP	Entrée/Sortie	Bloc de fonctions	Lit et écrit les données de recette depuis et vers une carte mémoire SD.
REPLACE sur la page 631	REPLACE	Manipulation de chaîne de caractères	Fonction	Remplace les parties d'une chaîne par de nouveaux jeux de caractères.
RHC sur la page 346	RHC	Entrée/Sortie	Fonction	Lit l'horloge haute vitesse.
RIGHT sur la page 629	RIGHT	Manipulation de chaîne de caractères	Fonction	Extrait les caractères de la partie droite d'une chaîne.
ROL sur la page 134	ROL	Opérations binaires	Fonction	Pour des entiers 32 bits, fait pivoter les bits d'entiers vers la gauche.
ROR sur la page 135	ROR	Opérations binaires	Fonction	Pour des entiers 32 bits, fait pivoter les bits d'entiers vers la droite.
RPC sur la page 348	RPC	Entrée/Sortie	Fonction	Lecture de la somme de contrôle (checksum) du programme utilisateur.
RS sur la page 146	RS	Opérations booléennes	Bloc de fonctions	Réinitialise le bistable dominant.
RTC_READ sur la page 383	RTC_READ	Entrée/Sortie	Bloc de fonctions	Lit les informations du module d'horloge temps réel (RTC).
RTC_SET sur la page 385	RTC_SET	Entrée/Sortie	Bloc de fonctions	Définit les données RTC (horloge temps réel) pour les informations du module RTC.
RTO sur la page 644	RTO	Heure	Bloc de fonctions	Temporisation retenue. Incrémente un temporisateur interne lorsque son entrée est active, mais ne le réinitialise pas lorsque son entrée devient inactive.
SCALER sur la page 507	SCP	Contrôle de procédés	Bloc de fonctions	Met la valeur d'entrée à l'échelle en fonction de la plage de sortie.
SCL sur la page 512	SCL	Contrôle de procédés	Bloc de fonctions	Convertit une valeur d'entrée sans échelle en une valeur de virgule flottante dans les unités Ingénieur.
SHL sur la page 137	SHL	Opérations binaires	Fonction	Pour des entiers 32 bits, déplace les entiers vers la gauche et place 0 dans le bit de poids faible.
SHR sur la page 139	SHR	Opérations binaires	Fonction	Pour des entiers 32 bits, déplace les entiers vers la droite et place 0 dans le bit de poids fort.
SIN sur la page 93	SIN	Arithmétique	Fonction	Calcule le sinus d'une valeur REAL.
SIN_LREAL sur la page 94	SIN_LREAL	Arithmétique	Fonction	Calcule le sinus d'une valeur REAL longue.
SOCKET_ACCEPT sur la page 568	SOCKET_ACCEPT	Communications	Bloc de fonctions	Accepte une demande de connexion TCP à partir d'une destination à distance et renvoie une instance de socket utilisée pour envoyer et recevoir des données sur la connexion nouvellement créée.
SOCKET_CREATE sur la page 571	SOCKET_CREATE	Communications	Bloc de fonctions	Crée une instance du socket et renvoie un numéro d'instance qui est utilisé comme entrée dans toutes les opérations de socket suivantes.
SOCKET_DELETE sur la page 576	SOCKET_DELETE	Communications	Bloc de fonctions	Supprime une instance de socket créée.
SOCKET_DELETEALL sur la page 578	SOCKET_DELETEALL	Communications	Bloc de fonctions	Supprime toutes les instances de socket créées.
SOCKET_INFO sur la page 580	SOCKET_INFO	Communications	Bloc de fonctions	Renvoie des informations sur le socket, comme des codes d'erreur et des états d'exécution.

Instruction	Instruction mappée (thème Logix)	Catégorie	Type	Description
SOCKET_OPEN sur la page 586	SOCKET_OPEN	Communications	Bloc de fonctions	Ouvre la connexion pour l'adresse de destination spécifiée pour les connexions TCP (Transmission Control Protocol). Pour les connexions UDP (User Datagram Protocol), associe une adresse IP de destination et un numéro de port au socket spécifié.
SOCKET_READ sur la page 590	SOCKET_READ	Communications	Bloc de fonctions	Lit des données sur un socket.
SOCKET_WRITE sur la page 594	SOCKET_WRITE	Communications	Bloc de fonctions	Envoie des données sur un socket.
SQRT sur la page 96	SQR	Arithmétique	Fonction	Calcule la racine carrée d'une valeur REAL.
SR sur la page 150	SR	Opérations booléennes	Bloc de fonctions	Définit le bistable dominant.
STACKINT sur la page 509	STACKINT	Contrôle de procédés	Bloc de fonctions	Gère la pile d'entiers.
STIS sur la page 393	STS	Interruptions	Fonction	Démarre le temporisateur STI à partir du programme de contrôle plutôt que manuellement.
SUS sur la page 523	SUS	Contrôle de programme	Bloc de fonctions	Suspend l'exécution du contrôleur <800>.
SYS_INFO sur la page 387	SYS_INFO	Entrée/Sortie	Bloc de fonctions	Lit le bloc de données d'état pour le contrôleur Micro800.
TAN sur la page 98	TAN	Arithmétique	Fonction	Calcule la tangente d'une valeur REAL.
TAN_LREAL sur la page 100	TAN_LREAL	Arithmétique	Fonction	Calcule la tangente d'une valeur REAL longue.
TDF sur la page 648	TDF	Heure	Fonction	Calcule la différence de temps entre TimeA et TimeB.
TND sur la page 518	TND	Contrôle de procédés	Fonction	Arrête le cycle actuel de la scrutation du programme utilisateur.
TOF sur la page 635	TOF	Heure	Bloc de fonctions	Temporisation interne sur front descendant. Incrémente un temporisateur interne jusqu'à une valeur donnée.
TON sur la page 637	TON	Heure	Bloc de fonctions	Temporisation interne sur front montant. Incrémente un temporisateur interne jusqu'à une valeur donnée.
TONOFF sur la page 640	TONOFF	Heure	Bloc de fonctions	Retard dans l'activation d'une sortie sur une ligne Vrai, puis retard dans la désactivation de la sortie sur la ligne Faux.
TOW sur la page 650	TOW	Heure	Fonction	Active une sortie si la valeur de l'horloge temps réel est comprise dans la plage de valeurs propre au paramétrage de la semaine.
TP sur la page 642	TP	Heure	Bloc de fonctions	Temporisation interne avec comptage. Dans le cas d'un front montant, incrémente un temporisateur interne jusqu'à une valeur donnée.
TRIMPOT_READ sur la page 390	TRIMPOT_READ	Entrée/Sortie	Bloc de fonctions	Lit la valeur de potentiomètre d'ajustement à partir d'un potentiomètre d'ajustement spécifique.
TRUNC sur la page 101	TRN	Arithmétique	Fonction	Tronque les valeurs réelles en ne gardant que l'entier.
TTABLE sur la page 152	TTABLE	Booléen	Fonction	Renvoie la valeur de la sortie déterminée en fonction de la combinaison des entrées.
UIC sur la page 395	UIC	Interruptions	Fonction	Efface le bit de perte de l'interruption par l'utilisateur sélectionnée.
UID sur la page 396	UID	Interruptions	Fonction	Désactive une interruption par l'utilisateur spécifique.
UIE sur la page 398	UIE	Interruptions	Fonction	Active une entrée utilisateur spécifique.
UIF sur la page 400	UIF	Interruptions	Fonction	Vide ou supprime une entrée utilisateur en attente.
XOR sur la page 149	XOR	Opérations booléennes	Opérateur	Opérateur OR (OU) exclusif booléen entre deux valeurs.
XOR_MASK sur la page 140	XOR_MASK	Opérations binaires	Fonction	Masque avec opérateur OR exclusif bit à bit entier, renvoie les valeurs de bits inversées.

Langage à relais (LD)

Une logique à relais (LD) est une représentation graphique d'équations booléennes, combinant des contacts (arguments d'entrée) à des bobines (résultats de sortie). À travers des symboles graphiques dans un diagramme de programme (organisé comme un diagramme de câblage de logique à relais), le langage LD décrit des tests et des modifications de données booléennes.

Les symboles graphiques LD sont organisés au sein du diagramme comme un diagramme de contacts électriques. Le terme « relais » provient du concept de lignes connectées à des rails d'alimentation verticaux aux deux extrémités, chaque ligne représentant un circuit individuel.

Connected Components Workbench fournit un éditeur de programmation LD et prend en charge les éléments sur la page 33 et les instructions fournies uniquement avec le logiciel Connected Components Workbench.

Programme de schémas à relais (LD)

Un ladder diagram (LD) è una rappresentazione grafica di equazioni Booleana che combinano i contatti (argomenti ingresso) con le bobine (risultati uscita). Utilizzando simboli grafici su un diagramma di programma (organizzato come ladder diagram di cablaggio relè), il linguaggio LD descrive prove e modifiche dei dati Booleana.

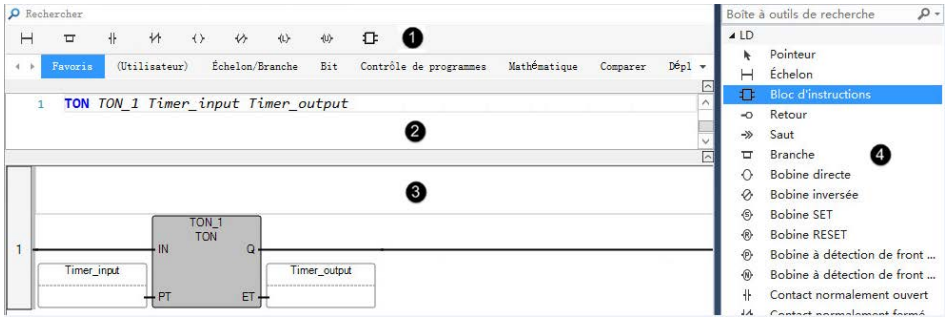
Il linguaggio LD utilizza simboli grafici su un diagramma di programma, organizzato come diagramma ladder del cablaggio del relè, per descrivere le prove e le modifiche dei dati Booleani.

Connected Components Workbench fornisce un editor di lingue Diagramma ladder sur la page 55 che supporta solo gli elementi e le istruzioni forniti con il software Connected Components Workbench.

Environnement de développement des programmes LD

L'éditeur de programmation d'un programme de schémas à relais (LD) dans lequel un Programme Unitaire (POU) LD est développé.

L'image suivante représente les principales zones de l'environnement de développement du programme LD.



N°	Nom	Description
1	Barre d'outils Instructions sur la page 30	Sélectionnez rapidement un élément d'instruction et envoyez-le dans l'éditeur graphique LD, ou cliquez une fois pour l'ajouter dans l'éditeur de texte LD.
2	Éditeur de texte LD	Édite la logique à l'aide des mnémoniques des instructions ASCII.
3	Éditeur graphique LD	Édite la logique à l'aide des éléments d'instruction graphiques.
4	Boîte à outils LD	Ajoute des éléments à l'éditeur graphique LD.

Barre d'outils Instructions (LD)

La barre d'outils Instructions est le nom courant du panneau secondaire dans le panneau de l'éditeur de programmation, qui fonctionne comme une barre d'outils et permet d'ajouter des éléments de langage sur la page 30 (par exemple, des instructions) à l'espace de travail de l'éditeur de programmation. Elle est complémentaire à la boîte à outils générale de Workbench.

Élément	Description
Onglets	Répertorie les éléments d'instruction par catégorie. Cliquez sur une catégorie pour voir les instructions de cette catégorie. Les touches fléchées peuvent également être utilisées pour sélectionner différentes catégories. Cliquez sur les flèches à gauche des onglets pour parcourir les catégories. Cliquez sur la flèche vers le bas à droite des onglets pour voir une liste de toutes les catégories.
Instructions	Répertorie toutes les instructions qui correspondent à la catégorie d'onglet sélectionnée. Il est également possible d'épingler des éléments d'instruction au début d'une liste de catégorie. Cliquez sur les flèches à gauche des instructions pour parcourir ces instructions.
Recherche	Filtre immédiatement les instructions par catégorie, dès que vous saisissez des noms ou des mots clés descriptifs. Cliquez sur le nom d'une catégorie pour voir les instructions filtrées de cette catégorie.
Favoris	Personnalise une liste d'instructions qui peut être rapidement trouvée et ajoutée à votre programme.

Ajouter des éléments d'instruction à partir de la barre d'outils Instructions

Utilisez la barre d'outils Instructions sur la page 30 pour rechercher, parcourir et ajouter des instructions à l'espace de travail de l'éditeur de programmation de langage à relais (Ladder Diagram, LD).

Pour ajouter un élément d'instruction de la Barre d'outils Instructions à l'éditeur de programmation

1. Cliquez sur l'onglet Catégorie qui inclut l'instruction à ajouter. Vous pouvez parcourir les onglets à l'aide des touches fléchées, puis les réorganiser par glisser-déposer.
2. Sélectionnez l'instruction par glisser-déposer ou cliquez sur celle-ci.
3. (Facultatif) Pour trouver rapidement une instruction, cliquez sur le champ Recherche, puis recherchez l'élément d'instruction en saisissant son nom ou un mot clé. Pour quitter la recherche et activer la navigation avec les touches fléchées, appuyez sur la touche **Échap**.
4. (Facultatif) Cliquez avec le bouton droit sur l'instruction et sélectionnez **Ajouter aux favoris** pour l'ajouter à l'onglet **Favoris** ou **Retirer des favoris** pour l'en retirer. Le paramètre Favoris sera automatiquement enregistré.

Éléments de langage à relais (LD)

Les éléments du langage à relais (LD) sont les composants utilisés pour assembler un programme de schémas à relais. Tous les éléments répertoriés dans le tableau suivant peuvent être ajoutés à un programme de schémas à relais dans Connected Components Workbench.

Élément	Description
Ligne de programme sur la page 33	Représente un groupe d'éléments de circuit entraînant l'activation d'une bobine.
Bloc d'instructions (LD) sur la page 40	Les instructions incluent des opérateurs, des fonctions et des blocs fonctionnels, dont des blocs fonctionnels définis par l'utilisateur.
Branche sur la page 37	Deux instructions ou plus en parallèle.
Bobine sur la page 40	Représente l'affectation de sorties ou de variables internes. Dans un programme de LD, une bobine représente une action.
Contact sur la page 47	Représente la valeur ou la fonction d'une entrée ou d'une variable interne.
Retour sur la page 52	Représente la fin conditionnelle d'une sortie de diagramme.
saut sur la page 53	Représente la logique conditionnelle et inconditionnelle du programme LD qui contrôle l'exécution des diagrammes.

Ligne de programme

Les lignes de programme sont les composants graphiques des programmes LD qui représentent un groupe d'éléments de circuit entraînant l'activation d'une bobine. Utilisez des étiquettes pour identifier les lignes de programme dans le diagramme. Les étiquettes, avec les sauts, contrôlent l'exécution d'un diagramme. Les commentaires sont du texte en format libre que vous pouvez ajouter au-dessus de la ligne de programme à des fins de documentation.

Ajouter une ligne de programme à un programme LD

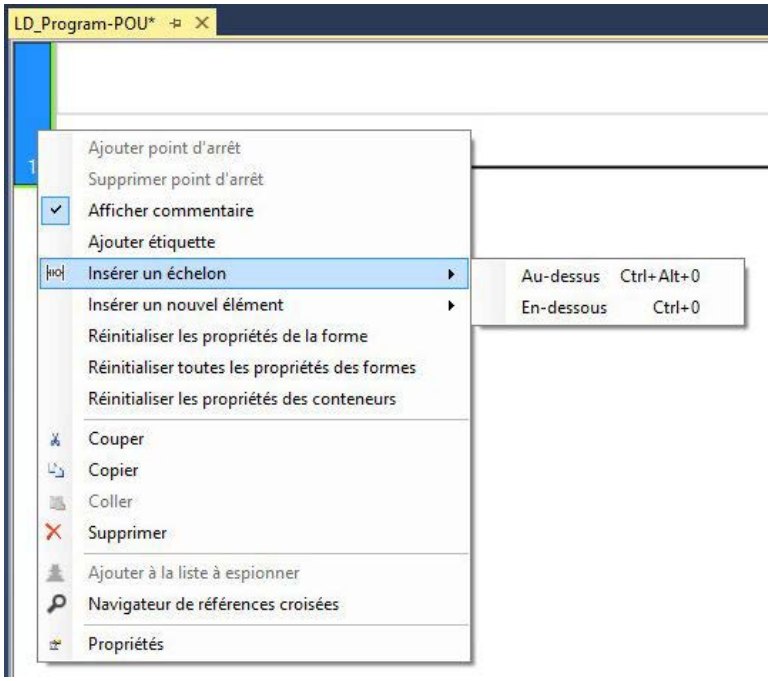
Les lignes de programme sur la page 33 sont les composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent un groupe d'éléments de circuit entraînant l'activation d'une bobine.

Dans Connected Components Workbench, vous pouvez ajouter une ligne de programme à un programme de schémas à relais (Ladder Diagram, LD) à partir des éléments suivants :

- Éditeur de programmation LD
- Éditeur multi langages situé dans le menu Outils
- Boîte à outils LD

Pour ajouter un élément de ligne de programme à un programme de schémas à relais

1. Dans l'éditeur de programmation LD, cliquez avec le bouton droit de la souris sur une ligne de programme existante, puis :
   - Cliquez sur **Copier**, puis cliquez sur **Coller** pour insérer une copie de la ligne de programme dans l'éditeur de programmation.
   - Cliquez sur **Insérer une ligne de programme**, puis effectuez l'une des opérations suivantes :
     - Cliquez sur **Au-dessus** pour ajouter la ligne de programme au-dessus de la ligne de programme sélectionnée.
     - Cliquez sur **En dessous** pour ajouter la ligne de programme en dessous de la ligne de programme sélectionnée.



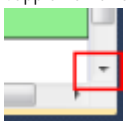
- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis appuyez sur :
  - **CTRL+ALT+0** : pour ajouter la ligne au-dessus de la ligne de programme sélectionnée.
  - **CTRL+0** : pour ajouter la ligne en dessous de la ligne de programme sélectionnée.
- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis appuyez sur :
  - Cliquez sur **Outils** > **Éditeur multi langages** > **Insérer une ligne en dessous** pour ajouter la ligne de programme en dessous de la ligne de programme sélectionnée.
  - Cliquez sur **Outils** > **Éditeur multi langages** > **Insérer une ligne au-dessus** pour ajouter la ligne de programme au-dessus de la ligne de programme sélectionnée.

1. (facultatif) Ouvrez la Boîte à outils LD pour afficher l'élément de ligne de programme :
   - Pour insérer une ligne de programme en dessous d'une ligne de programme existante, sélectionnez la ligne de programme dans l'éditeur de programmation LD, puis double-cliquez sur **Ligne de programme** dans la boîte à outils LD.

     Si un élément n'est pas sélectionné avant de double-cliquer sur **Ligne de programme** dans la boîte à outils LD, la ligne de programme est insérée sous la dernière ligne de programme dans l'éditeur de programmation LD.
   - Sélectionnez **Ligne de programme**, puis faites glisser l'élément dans l'éditeur de programmation LD.

     Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour ajouter l'élément.

     Conseil : si le langage à relais contient plus de 355 lignes de programme, utilisez le triangle bas plutôt que la barre de défilement pour afficher les lignes de programme supplémentaires.

Étiquettes de ligne de programme

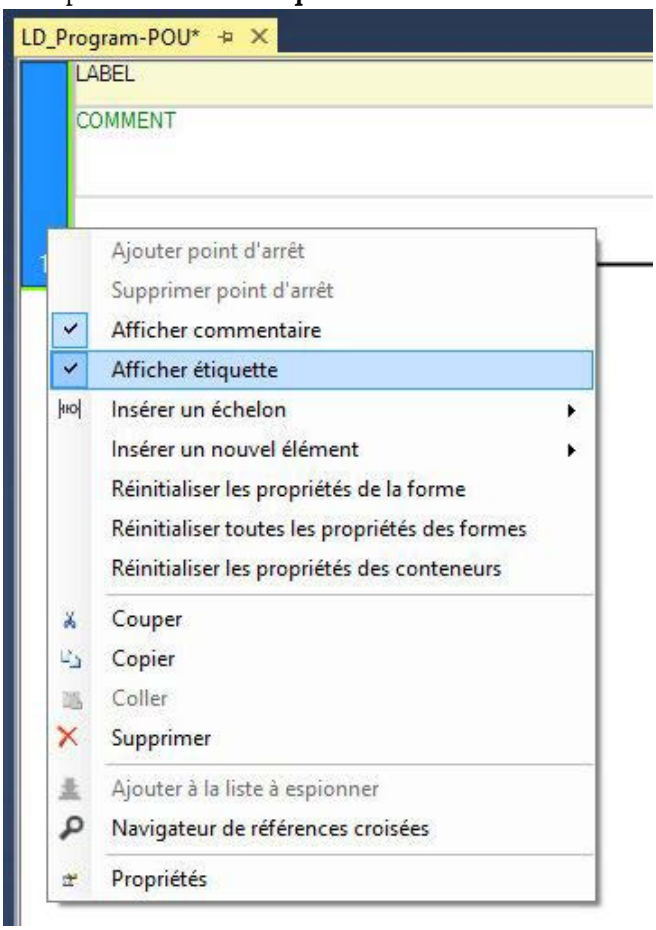
Les commentaires sont des ajouts facultatifs à chaque ligne de programme de l'éditeur de programmation LD.

Les étiquettes peuvent avoir un nombre illimité de caractères et doivent commencer par une lettre ou un trait de soulignement, suivi de lettres, chiffres ou traits de soulignement. Les étiquettes ne peuvent comporter d'espaces ou de caractères spéciaux (par exemple +, - ou \).

Pour ajouter une étiquette pour une ligne de programme dans l'éditeur de programmation LD

1. Pour ajouter une étiquette à une ligne de programme, cliquez avec le bouton droit de la souris sur la ligne de programme afin d'ouvrir le menu de l'éditeur de programmation LD, puis sélectionnez **Afficher l'étiquette**.

La ligne de programme est mise à jour pour inclure l'étiquette, et le menu de l'éditeur de programmation LD affiche une coche en regard de l'option **Afficher l'étiquette**.



2. Sélectionnez l'**étiquette**, puis saisissez une description.
3. (facultatif) Pour supprimer l'étiquette, cliquez sur **Afficher l'étiquette** dans le menu de l'éditeur de programmation LD.

Commentaires de ligne

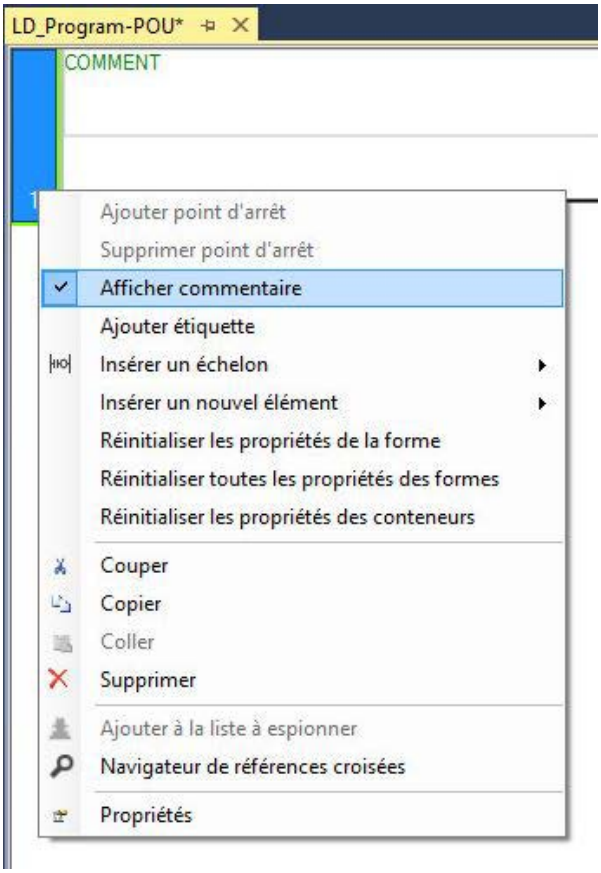
Les commentaires sont facultatifs pour chaque ligne de programme de l'Éditeur de programmation LD. Par défaut, un commentaire est inclus lorsque vous ajoutez un élément de ligne de programme dans l'éditeur de programmation LD sur la page 33.

Les commentaires sont :

- Saisis dans l'espace au-dessus de la ligne de programme.
- Enregistrés au format RTF.
- Stockés dans le contrôleur.

Pour ajouter ou supprimer un commentaire pour une ligne de programme dans l'Éditeur de programmation LD

1. Pour supprimer un commentaire, cliquez avec le bouton droit de la souris sur la ligne de programme pour ouvrir le menu de l'Éditeur de programmation LD.

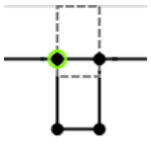


2. Dans le menu de l'Éditeur de programmation LD, cliquez sur **Afficher le commentaire**. Le commentaire est retiré de la ligne de programme et la marque en regard de l'option **Afficher le commentaire** dans le menu de l'Éditeur de programmation LD est supprimée.
3. (facultatif) Pour ajouter un commentaire à une ligne de programme, cliquez sur **Afficher le commentaire** dans le menu de l'Éditeur de programmation LD.

Branche

Les branches créent un autre routage pour les connexions. Vous pouvez ajouter des branches parallèles à des éléments sur une ligne de programme à l'aide de l'éditeur de programmation de langage à relais (Ladder Diagram, LD) sur la page 38.

Exemple de branche



Ajout d'une branche à un programme LD

Les branches sur la page 37 sont des composants graphiques des programmes de schémas à relais (Ladder Diagram, LD) qui créent un routage alternatif pour les connexions et peuvent inclure des branches parallèles.

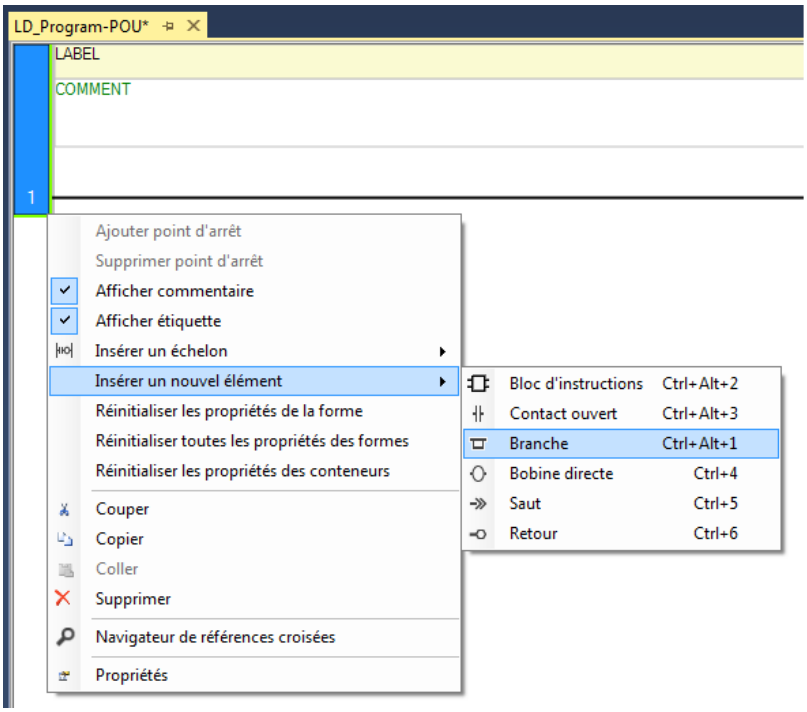
Dans Connected Components Workbench, vous pouvez ajouter une branche à un programme de schémas à relais (Ladder Diagram, LD) à partir des éléments suivants :

- Éditeur de programmation de langage à relais (LD)
- Éditeur multi langages situé dans le menu Outils
- Boîte à outils LD
- Barre d'outils Instructions (LD)

Pour ajouter une branche à un programme de schémas à relais

1. Dans l'éditeur de programmation LD, vérifiez qu'une ligne de programme est définie pour la branche dans le programme LD. Effectuez ensuite l'une des opérations suivantes :
   - Cliquez avec le bouton droit sur une ligne de programme ou un élément dans l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Branche**.

     Si le **Sélecteur de variable** s'ouvre, sélectionnez une variable ou cliquez sur **OK** sans sélectionner de variable pour ajouter l'élément **Branche**.



   - Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis appuyez sur :

     CTRL+ALT+1 pour ajouter la branche à gauche de la ligne de programme ou de l'élément sélectionné(e).

CTRL+1 pour ajouter la branche à droite de la ligne de programme ou de l'élément sélectionné(e).

- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis effectuez l'une des opérations suivantes :

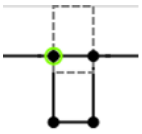
  Cliquez sur **Outils** > **Éditeur multi langages** > **Insérer une branche avant** pour ajouter la branche à gauche de l'élément sélectionné.
  Cliquez sur **Outils** > **Éditeur multi langages** > **Insérer une branche après** pour ajouter la branche à droite de l'élément sélectionné.

- Ouvrez la Boîte à outils LD pour afficher l'élément de branche, puis :

  Double-cliquez sur l'élément de branche pour l'ajouter à l'éditeur de programmation LD.
  Faites glisser l'élément de branche dans l'éditeur de programmation LD et placez-le sur la ligne de programme.
  Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour ajouter l'élément.

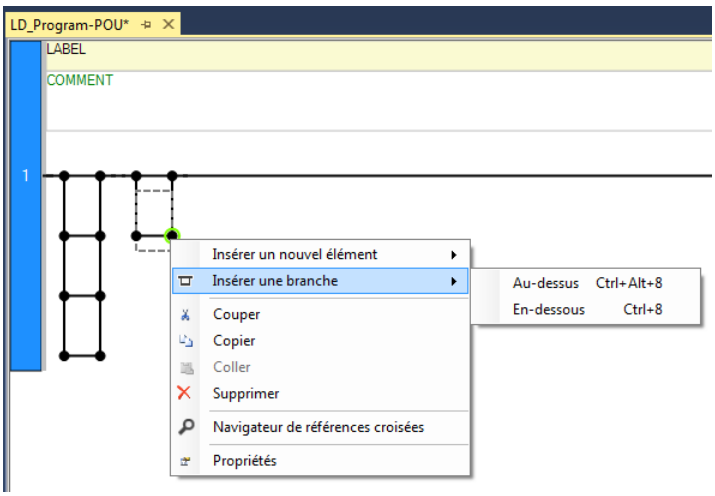
1. (facultatif) Pour insérer une branche parallèle :

   a. Dans l'éditeur de programmation LD, cliquez avec le bouton droit sur la branche pour ouvrir le menu de l'éditeur de programmation LD.



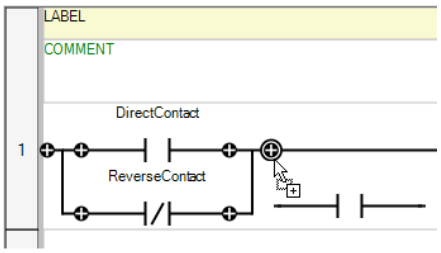
   b. Dans le menu de l'éditeur de programmation LD, sélectionnez **Insérer une branche**, puis cliquez sur :

   Au-dessus pour ajouter une branche au-dessus de la branche sélectionnée.
   En dessous pour ajouter la branche en dessous de la branche sélectionnée.



2. (facultatif) Pour déplacer un élément de branche vers un autre emplacement dans un programme LD, sélectionnez l'élément et faites-le glisser vers un nouvel emplacement dans le programme LD.

   Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour insérer l'élément dans l'emplacement cible.



Bloc d'instructions (LD)

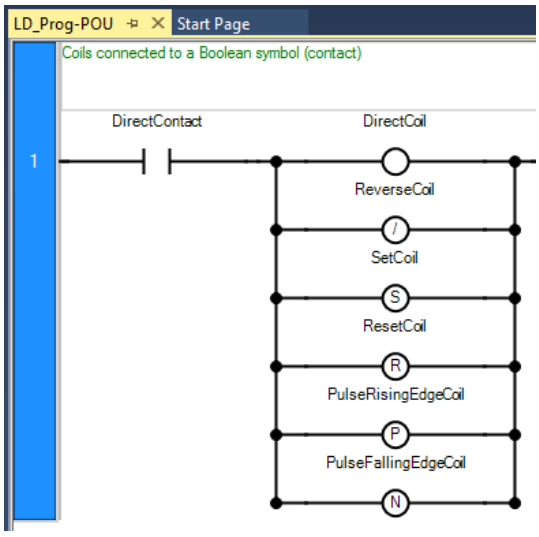
Un élément de bloc d'instruction de langage à relais est un élément fonctionnel conforme à la norme CEI 61131-3 dans un programme de schémas à relais pouvant être un bloc de fonctions, une fonction, un bloc de fonctions défini par l'utilisateur ou un opérateur.

Bobine

Les bobines sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne. Dans les programmes de schémas à relais (LD), une bobine représente une action. Une bobine doit être connectée à gauche à un symbole booléen, tel qu'un contact ou la sortie booléenne d'un bloc d'instruction. Les bobines peuvent uniquement être ajoutées à une ligne de programme définie dans l'éditeur de programmation du langage à relais. La définition de la bobine peut être modifiée après que la bobine est ajoutée à la ligne de programme.

L'exemple suivant illustre les types d'élément bobine disponibles pour les programmes LD.

Exemple : Bobines



Ajout d'éléments bobine

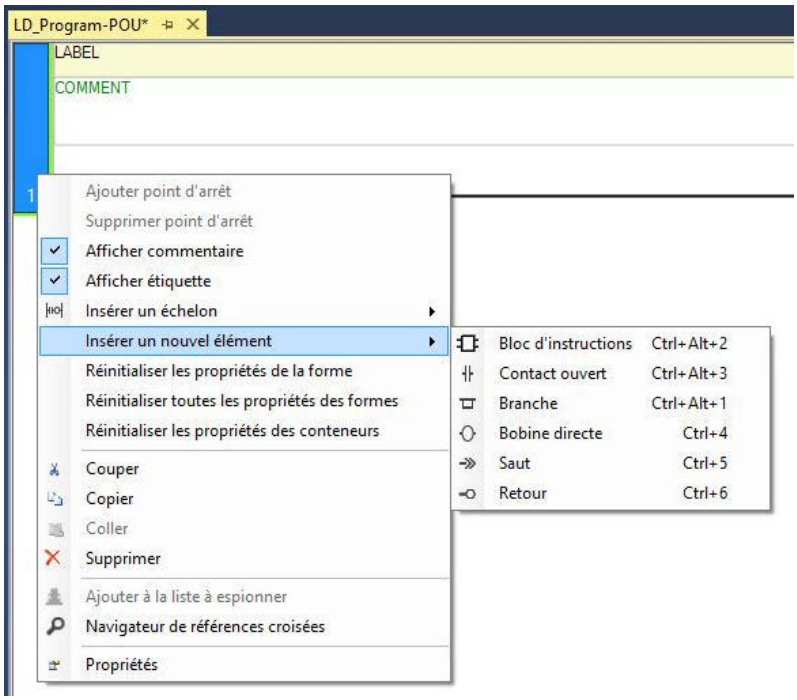
Les bobines sont des composants graphiques de programmes LD qui représentent une action prise, une sortie ou une variable interne.

Dans Connected Components Workbench, vous pouvez ajouter une bobine à un programme de schémas à relais (Ladder Diagram, LD) à partir des éléments suivants :

- Éditeur de programmation de langage à relais (LD)
- Éditeur multi langages situé dans le menu Outils
- Boîte à outils LD
- Barre d'outils Instructions (LD)

1. Pour ajouter un élément bobine à un programme de schémas à relais : dans l'éditeur de programmation LD, vérifiez qu'une ligne de programme est définie pour la bobine dans le programme LD. Effectuez ensuite l'une des opérations suivantes :
   - Cliquez avec le bouton droit de la souris sur une ligne de programme ou un élément dans l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Bobine directe**.

     Si le **Sélecteur de variable** s'ouvre, sélectionnez une variable ou cliquez sur **OK** sans sélectionner de variable pour ajouter l'élément **Bobine directe**.



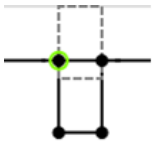
- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis appuyez sur **CTRL+4** pour ajouter la **Bobine directe** à droite de la ligne de programme.
- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis cliquez sur **Outils** > **Éditeur multi langages** > **Insérer une bobine** pour ajouter la bobine à droite de la ligne de programme.

- Ouvrez la Boîte à outils LD pour afficher les éléments bobines (**Bobine directe**, **Bobine inversée**, **Bobine d'enclenchement**, **Bobine de déclenchement, Bobine du front montant de l'impulsion**, **Bobine du front descendant de l'impulsion**), puis effectuez l'une des opérations suivantes :

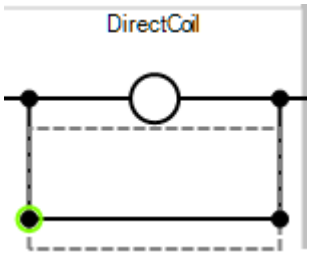
  Double-cliquez sur l'élément bobine pour l'ajouter à l'éditeur de programmation LD.
  Faites glisser l'élément bobine dans l'éditeur de programmation LD et placez-le sur la ligne de programme.
  Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour ajouter l'élément.

1. (facultatif) Pour insérer une bobine parallèle

   a. Dans l'éditeur de programmation LD, vérifiez que le programme LD contient une branche définie, puis cliquez avec le bouton droit de la souris sur le niveau supérieur de la branche sur la page 37 pour ouvrir le menu de l'éditeur de programmation LD.

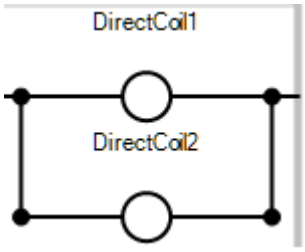
   

   b. Dans le menu de l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Bobine directe**. L'élément est inséré sur le niveau supérieur de la branche.

   c. Cliquez avec le bouton droit de la souris sur le niveau inférieur de la branche pour ouvrir le menu de l'éditeur de programmation LD.

   d. Dans le menu de l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Bobine directe**. L'élément est inséré sur le niveau inférieur de la branche.

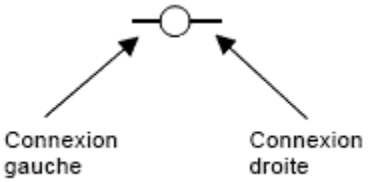
2. (facultatif) Pour modifier le type de bobine, sélectionnez la bobine dans l'éditeur de programmation, puis appuyez sur la **barre d'espace** jusqu'à ce que le type de bobine s'affiche dans l'éditeur de programmation.

   Chaque fois que vous appuyez sur la barre d'espace, les types de bobine suivants défilent : Directe, Inversée, Enclenchement, Déclenchement, Front montant de l'impulsion et Front descendant de l'impulsion.

Bobine directe

Les bobines sur la page 40 sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne.

Une bobine directe prend en charge une sortie booléenne d'état booléen d'une ligne de connexion.



L'état booléen de la connexion gauche est affecté à la variable associée. L'état de la connexion gauche se propage à la connexion droite. La connexion droite doit être reliée au rail d'alimentation vertical droit (sauf si vous disposez de bobines parallèles, auquel cas seule la bobine supérieure doit être connectée au rail d'alimentation vertical droit).

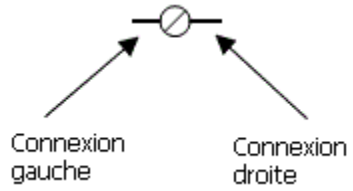
La variable booléenne associée doit être une sortie ou elle doit être définie par l'utilisateur.

Exemple de bobine directe

Bobine inversée

Les bobines sur la page 40 sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne.

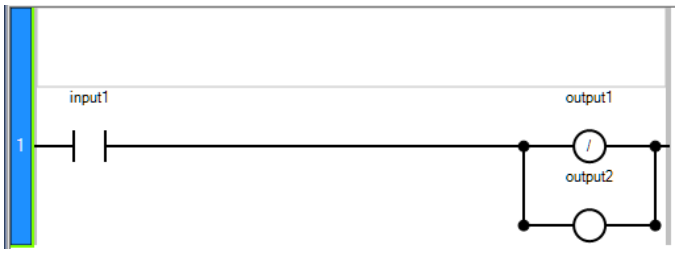
Une bobine inversée prend en charge une sortie booléenne conformément à l'inversion booléenne de l'état d'une ligne de connexion.



L'inversion booléenne de l'état de la connexion gauche est affectée à la variable associée. L'état de la connexion gauche se propage à la connexion droite. La connexion droite doit être reliée au rail d'alimentation vertical droit (sauf si vous disposez de bobines parallèles, auquel cas seule la bobine supérieure doit être connectée au rail d'alimentation vertical droit).

La variable booléenne associée doit être une sortie ou elle doit être définie par l'utilisateur.

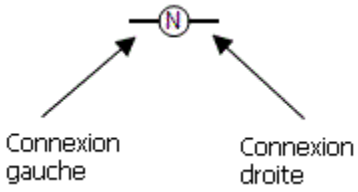
Exemple de bobine inversée



Bobine à impulsion sur front descendant

Les bobines sur la page 40 sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne.

Les bobines à impulsion sur front descendant sur la page 45 (ou négatives) prennent en charge une sortie booléenne d'état booléen d'une ligne de connexion.



La variable associée est sur VRAI lorsque l'état booléen de la connexion gauche passe de VRAI à FAUX. La variable de sortie est réinitialisée sur FAUX dans tous les autres cas. L'état de la connexion gauche se propage à la connexion droite. La connexion droite doit être reliée au rail d'alimentation vertical droit (sauf si vous disposez de bobines parallèles, auquel cas seule la bobine supérieure doit être connectée au rail d'alimentation vertical droit).

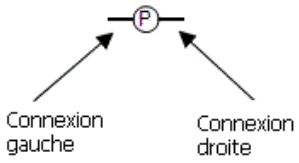
La variable booléenne associée doit être une sortie ou elle doit être définie par l'utilisateur.

Exemple de bobine du front descendant de l'impulsion

Bobine à impulsion sur front montant

Les bobines sur la page 40 sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne.

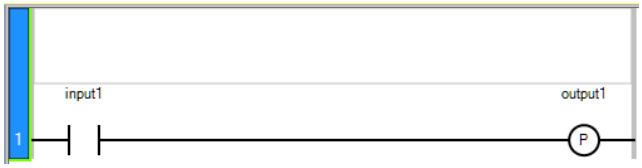
Les bobines à détection de front positif (ou positives) prennent en charge une sortie booléenne d'état booléen d'une ligne de connexion.



La variable associée est sur VRAI lorsque l'état booléen de la connexion gauche passe de FAUX à VRAI. La variable de sortie est réinitialisée sur FAUX dans tous les autres cas. L'état de la connexion gauche se propage à la connexion droite. La connexion droite doit être reliée au rail d'alimentation vertical droit (sauf si vous disposez de bobines parallèles, auquel cas seule la bobine supérieure doit être connectée au rail d'alimentation vertical droit).

La variable booléenne associée doit être de type sortie ou définie par l'utilisateur.

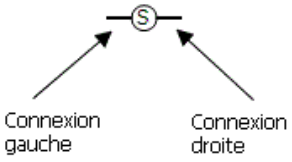
Exemple de bobine du front montant de l'impulsion



Bobine d'enclenchement

Les bobines sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne. Dans les programmes de schémas à relais (LD), une bobine représente une action.

Les bobines SET prennent en charge une sortie booléenne d'état booléen d'une ligne de connexion.



La variable associée est définie sur VRAI lorsque l'état booléen de la connexion gauche devient VRAI. La variable de sortie conserve cette valeur jusqu'à ce qu'un ordre inverse soit donné par une bobine de déclenchement. L'état de la connexion gauche se propage à la connexion droite. La connexion droite doit être reliée au rail d'alimentation vertical droit (sauf si vous disposez de

bobines parallèles, auquel cas seule la bobine supérieure doit être connectée au rail d'alimentation vertical droit).

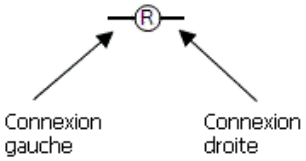
La variable booléenne associée doit être une sortie ou elle doit être définie par l'utilisateur.

Exemple de bobine d'enclenchement

Bobine de déclenchement

Les bobines sur la page 40 sont des composants graphiques d'un programme de langage à relais (Ladder Diagram, LD) qui représentent l'affectation d'une sortie ou d'une variable interne.

Les bobines RESET prennent en charge une sortie booléenne d'état booléen d'une ligne de connexion.



La variable associée est réinitialisée sur FAUX lorsque l'état booléen de la connexion gauche devient VRAI. La variable de sortie conserve cette valeur jusqu'à ce qu'un ordre inverse soit donné par une bobine d'enclenchement. L'état de la connexion gauche se propage à la connexion droite. La connexion droite doit être reliée au rail d'alimentation vertical droit (sauf si vous disposez de bobines parallèles, auquel cas seule la bobine supérieure doit être connectée au rail d'alimentation vertical droit).

La variable booléenne associée doit être de type sortie ou définie par l'utilisateur.

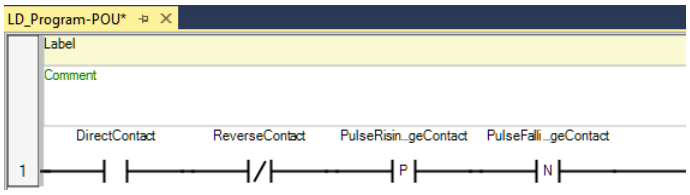
Exemple de bobine de déclenchement

Contact

Les contacts sont des composants graphiques des programmes de langage à relais (Ladder Diagram, LD). En fonction du type de contact, il représente la valeur ou la fonction d'une entrée ou d'une variable interne. Les contacts peuvent uniquement être ajoutés à une ligne définie dans l'éditeur de langue LD. Une fois le contact ajouté, sa définition peut être modifiée.

L'exemple suivant illustre les types d'élément contact disponibles pour les programmes LD.

Exemple : Contacts



Ajouter un contact à un programme LD

Les contacts sont des composants graphiques d'un programme de schémas à relais. En fonction du type de contact, il représente la valeur ou la fonction d'une entrée ou d'une variable interne. Les contacts peuvent uniquement être ajoutés à une ligne définie dans l'éditeur de langue LD.

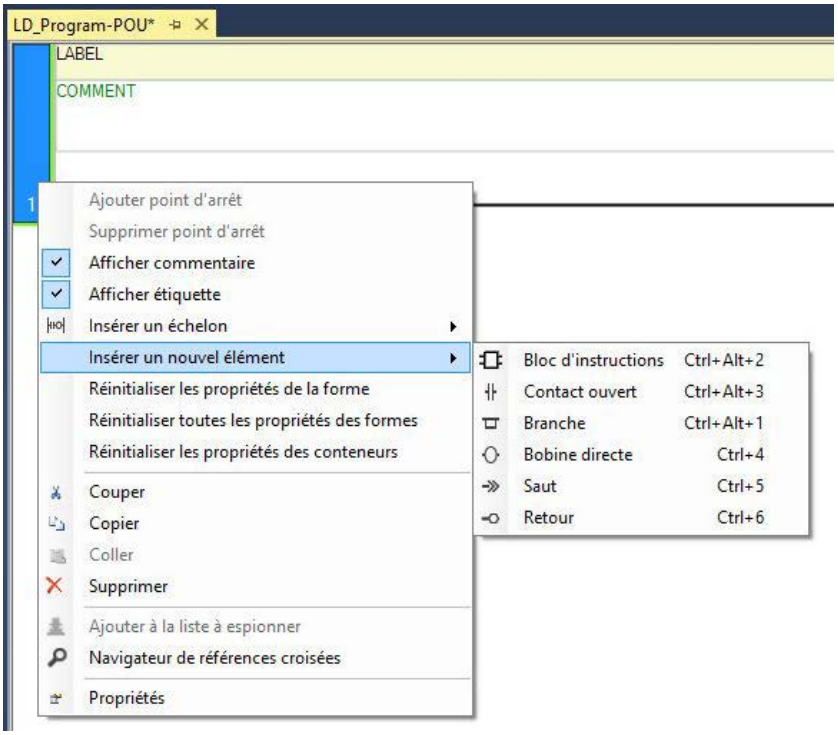
Dans Connected Components Workbench, vous pouvez ajouter un contact à un programme de schémas à relais (Ladder Diagram, LD) à partir des éléments suivants :

- Éditeur de programmation de langage à relais (LD)
- Éditeur multi langages situé dans le menu Outils
- Boîte à outils LD
- Barre d'outils Instructions (LD)

Pour ajouter un élément de contact à un programme de schémas à relais

1. Dans l'éditeur de programmation LD, vérifiez qu'une ligne de programme est définie pour le contact dans le programme LD. Effectuez ensuite l'une des opérations suivantes :
   - Cliquez avec le bouton droit de la souris sur une ligne de programme ou un élément dans l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Contact normalement ouvert**.

Si le **Sélecteur de variable** s'ouvre, sélectionnez une variable ou cliquez sur **OK** sans sélectionner de variable pour ajouter l'élément de **Contact normalement ouvert**.



- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis appuyez sur :

  CTRL+ALT+3 pour ajouter l'élément de contact normalement ouvert à gauche de la ligne de programme ou de l'élément sélectionné.
  CTRL+3 pour ajouter l'élément de contact normalement ouvert à droite de la ligne de programme ou de l'élément sélectionné.

- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis effectuez l'une des opérations suivantes :

  Cliquez sur **Outils** > **Éditeur multi langages** > **Insérer un contact avant** pour ajouter le contact à gauche de la ligne de programme ou de l'élément sélectionné.
  Cliquez sur **Outils** > **Éditeur multi langages** > **Insérer un contact après** pour ajouter le contact à droite de la ligne de programme ou de l'élément sélectionné.

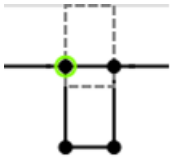
- Ouvrez la Boîte à outils LD pour afficher les éléments de contact (**Contact normalement ouvert**, **Contact normalement fermé**, **Contact du front montant de l'impulsion**, **Contact du front descendant de l'impulsion**), puis effectuez l'une des opérations suivantes :

  Double-cliquez sur l'élément de contact pour l'ajouter à l'éditeur de programmation LD.
  Faites glisser l'élément de contact dans l'éditeur de programmation LD et placez-le sur la ligne de programme.

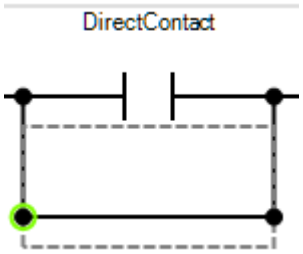
Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour ajouter l'élément.

1. (facultatif) Pour insérer un contact parallèle :

   a. Dans l'éditeur de programmation LD, vérifiez que le programme LD contient une branche sur la page 37 définie, puis cliquez avec le bouton droit de la souris sur le niveau supérieur de la branche pour ouvrir le menu de l'éditeur de programmation LD.

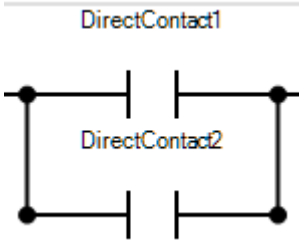


   b. Dans le menu de l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Contact normalement ouvert**. L'élément est inséré sur le niveau supérieur de la branche.



   c. Cliquez avec le bouton droit de la souris sur le niveau inférieur de la branche pour ouvrir le menu de l'éditeur de programmation LD.

   d. Dans le menu de l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Contact normalement ouvert**. L'élément est inséré sur le niveau inférieur de la branche.



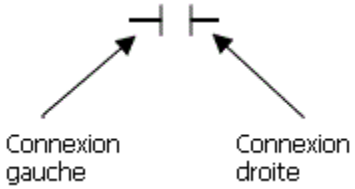
2. (facultatif) Pour modifier le type de contact, sélectionnez le contact dans l'éditeur de programmation, puis appuyez sur la **barre d'espace** jusqu'à ce que le type de contact s'affiche dans l'éditeur de programmation.

   Chaque fois que vous appuyez sur la barre d'espace, les types de contact suivants défilent : Normalement ouvert, Normalement fermé, Front montant de l'impulsion et Front descendant de l'impulsion.

Contact normalement ouvert

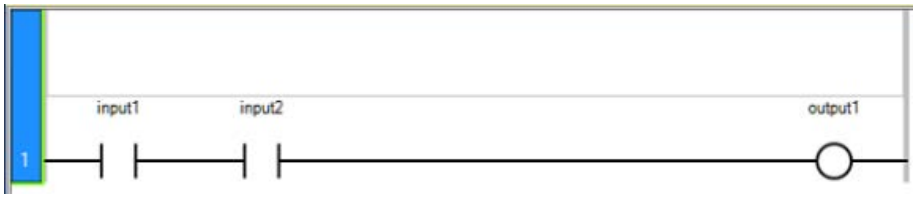
Les contacts sur la page 47 sont des composants graphiques des programmes de langage à relais (Ladder Diagram, LD).

Les contacts normalement ouverts prennent en charge une opération booléenne entre un état de ligne de connexion et une variable booléenne.



L'état de la ligne de connexion à droite du contact est le AND logique entre l'état de la ligne de connexion gauche et la valeur de la variable associée au contact.

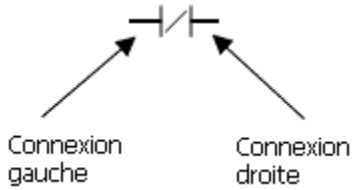
Exemple de contact direct



Contact normalement fermé

Les contacts sur la page 47 sont des composants graphiques des programmes de langage à relais (Ladder Diagram, LD).

Les contacts normalement fermés prennent en charge une opération booléenne entre un état de ligne de connexion et l'inversion booléenne d'une variable booléenne.



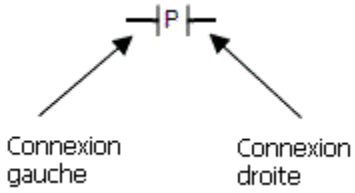
L'état de la ligne de connexion à droite du contact est le AND logique entre l'état de la ligne de connexion gauche et l'inversion booléenne de la valeur de la variable associée au contact.

Exemple de contact normalement fermé

Contact à impulsion sur front montant

Les contacts sur la page 47 sont des composants graphiques des programmes de langage à relais (Ladder Diagram, LD).

Les contacts à impulsion sur front montant (ou positifs) prennent en charge une opération booléenne entre un état de ligne de connexion et le front montant d'une variable booléenne.



L'état de la ligne de connexion à droite du contact est VRAI lorsque l'état de la ligne de connexion à gauche est VRAI, et l'état de la variable associée passe de FAUX à VRAI. L'état est réinitialisé sur FAUX dans tous les autres cas.

Exemple de contact du front montant de l'impulsion

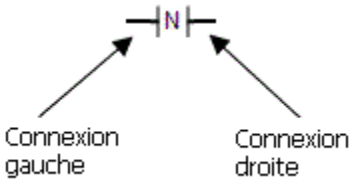
Recommandation : limiter l'utilisation des variables de sortie avec des contacts à impulsion sur front montant/descendant

Nous recommandons de ne pas utiliser de sorties ou de variables avec un contact du front montant de l'impulsion sur la page 51 (positif) ou un contact du front descendant de l'impulsion sur la page 51 (négatif). Ces contacts sont destinés à des entrées physiques dans une logique à relais. Pour détecter le front d'une variable ou d'une sortie, nous recommandons d'utiliser le bloc de fonctions R_TRIG/F_TRIG, qui est pris en charge et fonctionne dans n'importe quel langage et n'importe où dans votre programme.

Contact à impulsion sur front descendant

Les contacts sont des composants graphiques des programmes de langage à relais (Ladder Diagram, LD).

Les contacts à impulsion sur front descendant (ou négatifs) prennent en charge une opération booléenne entre un état de ligne de connexion et le front descendant d'une variable booléenne.



L'état de la ligne de connexion à droite du contact est VRAI lorsque l'état de la ligne de connexion à gauche est VRAI, et l'état de la variable associée passe de VRAI à FAUX. L'état est réinitialisé sur FAUX dans tous les autres cas.

Exemple de contact du front descendant de l'impulsion

Recommandation : limiter l'utilisation des variables de sortie avec des contacts à impulsion sur front montant/descendant

Nous recommandons de ne pas utiliser de sorties ou de variables avec un contact du front montant de l'impulsion sur la page 51 (positif) ou un contact du front descendant de l'impulsion sur la page 51 (négatif). Ces contacts sont destinés à des entrées physiques dans une logique à relais. Pour détecter le front d'une variable ou d'une sortie, nous recommandons d'utiliser le bloc de fonctions R_TRIG/F_TRIG, qui est pris en charge et fonctionne dans n'importe quel langage et n'importe où dans votre programme.

Retour

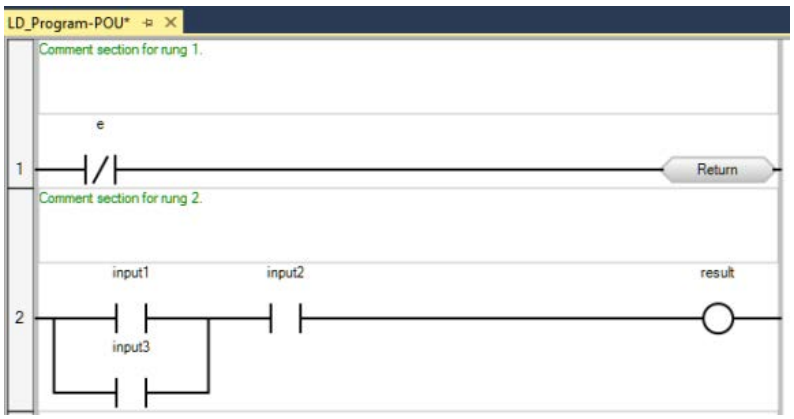
Les retours sont des sorties représentant une fin conditionnelle d'un programme de schémas à relais (LD).

vous ne pouvez placer de connexions à droite d'un élément de retour.

Lorsque la ligne de connexion gauche correspond à l'état booléen TRUE, le diagramme se termine sans exécuter les instructions situées sur les lignes suivantes du diagramme.

Lorsque le diagramme de LD est une fonction, son nom est associé à une bobine de sortie pour initialiser la valeur de retour (ramenée au diagramme d'appel).

Exemple de retour



Pour insérer un retour dans un programme de schémas à relais

Effectuez l'une des opérations suivantes :

- Cliquez avec le bouton droit de la souris sur une ligne de programme ou un élément dans l'éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Retour**.
- Sélectionnez une ligne de programme ou un élément dans l'Éditeur de programmation LD, puis appuyez sur **CTRL+6**.
- Sélectionnez une ligne de programme ou un élément dans l'éditeur de programmation LD, puis cliquez sur **Outils** > **Éditeur multi langages** > **Insérer un retour**.
- Ouvrez la boîte à outils LD pour afficher l'élément de retour, puis :

  Double-cliquez sur l'élément de retour pour l'ajouter à l'éditeur de programmation LD.
  Faites glisser l'élément de retour dans l'éditeur de programmation LD et placez-le sur la ligne de programme.
  Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour ajouter l'élément.

saut

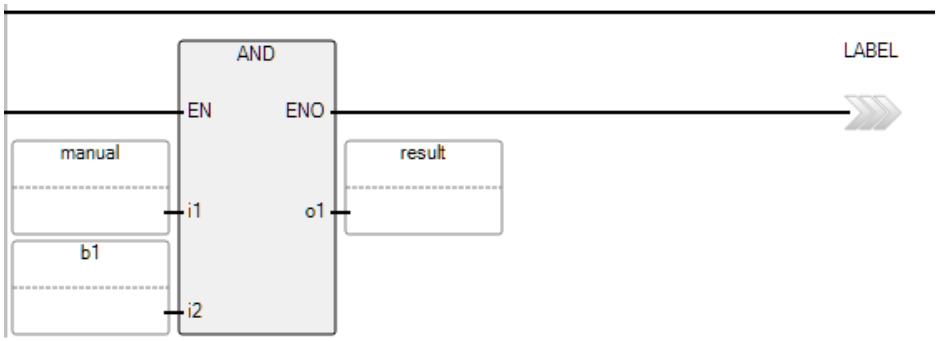
Les sauts sont des éléments conditionnels ou inconditionnels contrôlant l'exécution de programmes de schémas à relais (LD).

Notation de saut

La notation suivante indique un saut vers une étiquette :

>>LABEL : saut vers une étiquette, où le nom d'étiquette est « LABEL »

Exemple de saut



Pour insérer un saut

Effectuez l'une des opérations suivantes :

- Cliquez avec le bouton droit de la souris sur une ligne de programme ou un élément dans l'Éditeur de programmation LD, sélectionnez **Insérer des éléments de schéma à relais**, puis cliquez sur **Saut**.
- Sélectionnez une ligne de programme ou un élément dans l'Éditeur de programmation LD, puis appuyez sur **CTRL+5**.
- Sélectionnez une ligne de programme ou un élément dans l'Éditeur de programmation LD, puis cliquez sur **Outils** > **Éditeur multi langages** > **Insérer un saut**.
- Ouvrez la boîte à outils LD pour afficher l'élément de saut, puis :

  Double-cliquez sur l'élément de saut pour l'ajouter à l'Éditeur de programmation LD.
  Faites glisser l'élément de saut dans l'Éditeur de programmation LD et placez-le sur la ligne de programme.
  Un signe plus (+) s'affiche dans l'éditeur de programmation LD pour afficher une cible valide. Relâchez le bouton de la souris pour ajouter l'élément.

Blocs d'instruction dans les programmes LD

Le jeu d'instructions de Connected Components Workbench comprend des blocs d'instruction conformes à la norme CEI 61131-3. Les blocs d'instruction incluent collectivement des blocs fonctionnels, des fonctions et des opérateurs. Vous pouvez connecter des entrées et des sorties de blocs d'instruction à des variables, contacts, bobines ou autres entrées et sorties de blocs d'instruction.

Conventions des blocs d'instruction

La norme CEI 61131-3 relative aux langages de programmation régit différents aspects des contrôleurs programmables, notamment l'exécution du système d'exploitation, les définitions de données, les langages de programmation et les jeux d'instructions. Cette norme offre un ensemble de fonctionnalités minimal qui peut être étendu pour les applications de l'utilisateur final.

Noms des blocs d'instruction

Les fonctions et les blocs de fonctions sont représentés par une boîte affichant le nom de l'instruction et la version courte des noms des paramètres. Pour les blocs de fonctions, le nom de l'instance est affiché au-dessus du nom du bloc de fonctions.

Paramètres de retour des blocs d'instruction

- Le paramètre de retour d'une fonction a le même nom que la fonction. Le paramètre de retour est la seule sortie ;

- Les paramètres de retour d'un bloc fonctionnel peuvent avoir n'importe quel nom. Des paramètres de retour multiples peuvent fournir des sorties multiples ;
- Vous pouvez définir les paramètres des programmes de plusieurs appareils en allant dans les onglets des appareils individuels affichés dans la vue **Paramètres**.

Utilisation de l'éditeur de programmation LD

Lors d'ajout d'éléments à une ligne de programme de schémas à relais (Ladder Diagram, LD), ces éléments sont ajoutés selon les critères suivants :

- Le premier élément d'une ligne est inséré à la position sélectionnée dans le langage à relais.
- Les éléments suivants sont insérés à droite de l'élément sélectionné sur la ligne de programme.
- Vous ne pouvez pas insérer d'élément à droite d'un saut ou d'un retour de bobine.

Méthodes différentes pour ajouter un élément à un programme LD :

- **Éditeur de schéma à relais LD**
  - Ajoutez des éléments, supprimez des éléments, copiez et collez des éléments.
  - Utilisez les raccourcis clavier LD sur la page 57 pour ajouter des éléments.
- **Éditeur de texte LD**
  - Ajoutez, modifiez et supprimez des éléments.
  - Copiez et collez des éléments depuis/vers RSLogix 500 et <RSLX5000>.
- **Éditeur multi langages** situé dans le menu **Outils**
  - Ajoutez des éléments.
  - Exportez une image du programme LD.
  - Activez ou désactivez l'ouverture automatique du **Sélecteur de variable** et du **Sélecteur de bloc d'instruction**.
- **Boîte à outils LD**
  - Ajoutez des éléments.
- **Barre d'outils Instructions**
  - Ajoutez des éléments.

Vous pouvez remplacer une variable affectée directement à partir de l'éditeur de programmation ou du **Sélecteur de variable**.

Pour modifier une variable à partir de l'éditeur de programmation

1. Dans l'éditeur de langue, cliquez sur la variable pour afficher une liste déroulante des variables globales et locales.

2. Effectuez l'une des opérations suivantes :
   - Saisissez un nouveau nom de variable et double-cliquez sur la variable pour ouvrir le **Sélecteur de variable**. Appuyez sur **Entrée** pour confirmer la nouvelle variable.
   - Sélectionnez un autre nom de variable dans la liste déroulante.

Pour modifier une variable à partir du sélecteur de variable

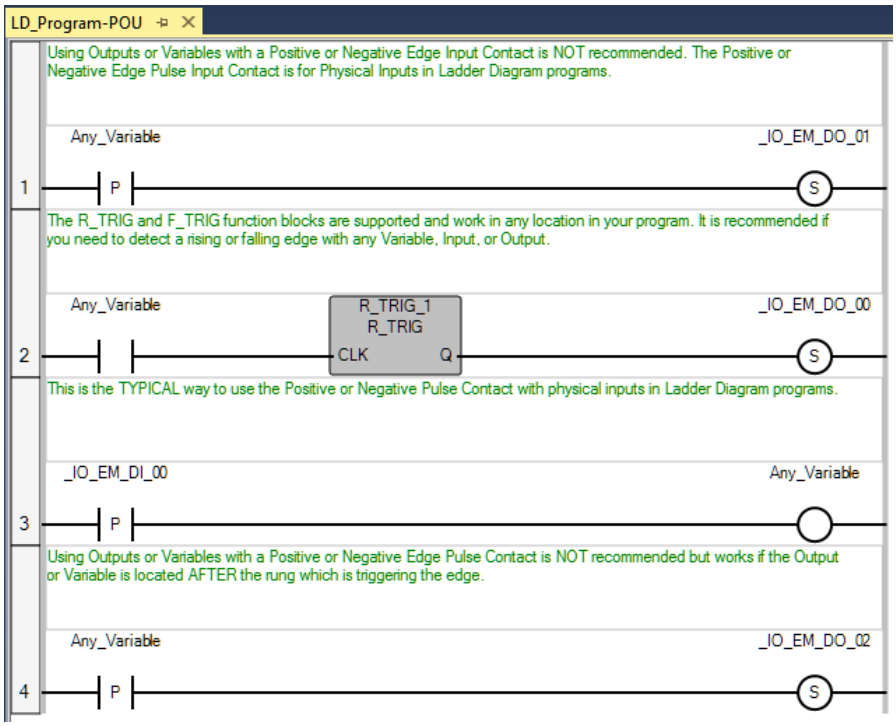
1. Dans l'éditeur de langue, double-cliquez sur la variable pour ouvrir le **sélecteur de variable**.
2. Cliquez sur le nom de la variable, puis sélectionnez une autre variable dans la liste déroulante des variables globales et locales.
3. Cliquez sur une variable existante, puis saisissez les constantes dans la zone de texte fournie.

Exemples de programme de schémas à relais (LD)

Voici quelques exemples de programmes de schémas à relais (LD) :

Exemple : bloc de fonctions R_TRIG

Le programme dans l'exemple suivant montre l'utilisation recommandée d'un bloc de fonctions R_TRIG servant à détecter un front tout en étant connecté au contrôleur.

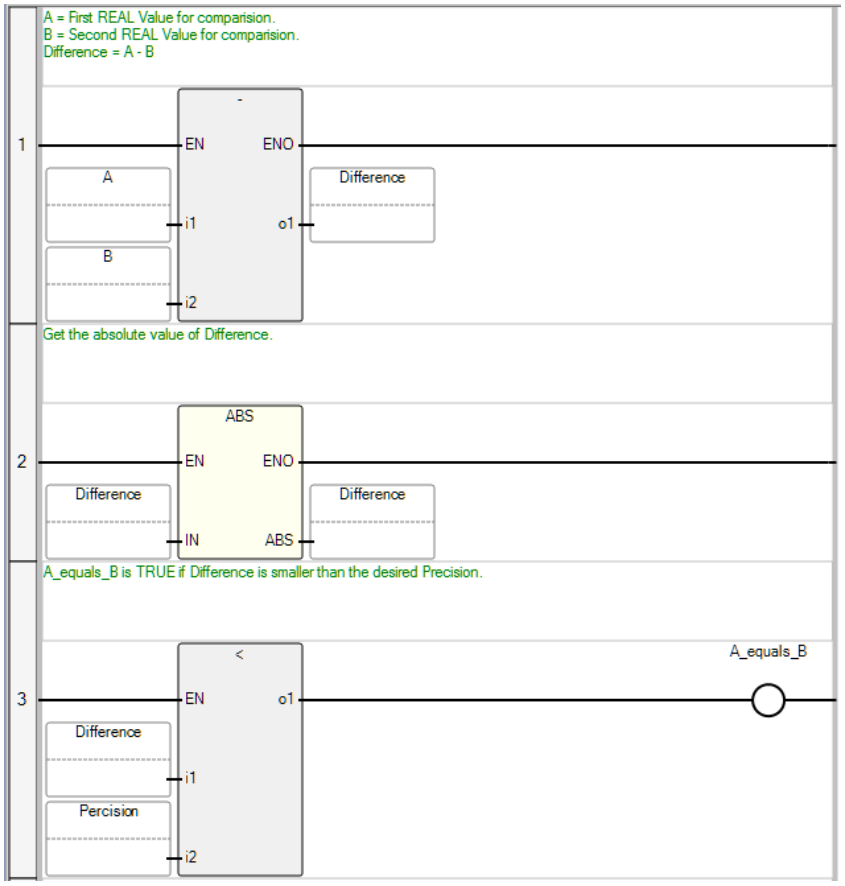


Exemple : comparaison de valeurs réelles avec les opérateurs de soustraction (-) ABS et d'infériorité (<)

Il est recommandé d'éviter d'utiliser le type de données REAL dans le cadre de la vérification de l'égalité de valeurs en raison des différences d'arrondi des chiffres. Deux valeurs de sortie peuvent être indiquées comme égales sur l'affichage de Connected Components Workbench, et générer un résultat négatif lors du test de comparaison.

Par exemple, les valeurs 23,500001 et 23,499999 s'afficheront toutes deux sous la forme 23,5 sur l'écran de saisie des variables, mais ne seront pas indiquées comme égales dans le contrôleur.

Pour vérifier l'égalité de deux valeurs REAL, vous pouvez utiliser une instruction Soustraction afin d'obtenir la différence entre ces valeurs, puis déterminer si cette différence est inférieure à une valeur de précision définie. L'exemple de programme LD ci-dessous illustre la comparaison de deux valeurs REAL.



Raccourcis clavier LD

Les raccourcis clavier suivants sont disponibles avec le langage LD sur la page 29.

Raccourci	Description
Ctrl+O	Insère une ligne après une ligne sélectionnée.[1]
Ctrl+Alt+O	Insère une ligne avant une ligne sélectionnée.[1]

Raccourci	Description
Ctrl+ 1	Insère une branche après un élément sélectionné.
Ctrl+Alt+ 1	Insère une branche avant un élément sélectionné.
Ctrl+2	Insère un bloc d'instruction après un élément sélectionné.[2]
Ctrl+Alt+2	Insère un bloc d'instruction avant un élément sélectionné.[2]
Ctrl+3	Insère un contact après un élément sélectionné.[2]
Ctrl+Alt+3	Insère un contact avant un élément sélectionné.[2]
Ctrl+4	Insère une bobine après un élément sélectionné.
Ctrl+5	Insère un saut après un élément sélectionné.
Ctrl+Alt+5	Insère un saut après un élément sélectionné.
Ctrl+6	Insère un retour après un élément sélectionné.
Ctrl+8	Insère une branche au-dessus de la branche sélectionnée.
Ctrl+Alt+8	Insère une branche en dessous de la branche sélectionnée.
Supprimer	Supprime une ligne ou un élément sélectionnés.
Entrée	Lorsqu'une ligne de programme est sélectionnée, appuyez sur la touche ENTRÉE pour sélectionner le premier élément de la ligne. Si aucun élément de ligne de programme n'est présent, rien ne se produit.
Barre d'espace	Lorsqu'une bobine ou un contact est sélectionné(e), appuyez sur la barre d'espace pour modifier le type de contact ou de bobine.
Maj+Entrée	Insère un saut de ligne.
Ctrl+Entrée	Ouvre une ligne au-dessus de la ligne actuelle.
Ctrl+Maj+Entrée	Ouvre une ligne au-dessous de la ligne actuelle.
Ctrl+Maj+L	Supprime la ligne actuelle.
Ctrl+Suppr	Supprime le prochain mot de la ligne actuelle.
Retour arrière	Supprime le caractère situé à gauche.
Ctrl+Retour arrière	Supprime le mot précédent de la ligne actuelle.
Ctrl+C	Copie le texte sélectionné dans le Presse-papiers.
Ctrl+Inser	Copie le texte sélectionné dans le Presse-papiers.
Ctrl+V	Copie le texte enregistré dans le Presse-papiers au niveau du point d'insertion.
Maj+Inser	Copie le texte enregistré dans le Presse-papiers au niveau du point d'insertion.
Ctrl+Z	Annule la commande précédente.
Ctrl+Y	Rétablit la commande précédente.
Ctrl+Maj+Z	Rétablit la commande précédente.
Ctrl+Flèche gauche	Accède au mot ou à la déclaration précédent(e).
Ctrl+Flèche droite	Accède au mot ou à la déclaration suivant(e).
Début	Se déplace vers le premier élément de la ligne de programme sélectionnée ; si aucun élément de ligne de programme n'est présent, rien ne se produit.
Fin	Se déplace vers le dernier élément de la ligne de programme sélectionnée ; si aucun élément de ligne de programme n'est présent, rien ne se produit.
Ctrl+Début	Se déplace vers le premier élément de la première ligne de programme : si aucun élément de ligne de programme n'est présent, la première ligne de programme est sélectionnée.
Ctrl+Fin	Se déplace vers le dernier élément de la dernière ligne de programme : si aucun élément de ligne de programme n'est présent, la dernière ligne de programme est sélectionnée.
Pg.Préc	Accède au début du code visible.
Pg.Suiv	Accède à la fin du code visible.

Raccourci	Description
Ctrl+J	Se déplace vers le crochet correspondant.
Ctrl+Flèche vers le bas	Fait défiler l'écran vers le bas.
Ctrl+Flèche vers le haut	Fait défiler l'écran vers le haut.
Maj+Flèche vers le bas	Effectue une sélection vers le bas.
Maj + clic gauche de souris	Sélectionne plusieurs lignes de programme. Cliquez sur chaque ligne individuellement.
Maj+Flèche vers le haut	Effectue une sélection vers le haut.
Maj+Flèche gauche	Effectue une sélection vers la gauche.
Maj+Flèche droite	Effectue une sélection vers la droite.
Ctrl+Maj+Flèche gauche	Sélectionne le mot ou la déclaration précédent(e).
Ctrl+Maj+Flèche droite	Sélectionne le mot ou la déclaration suivant(e).
Maj+Début	Effectue une sélection à partir du point d'insertion jusqu'au début de la ligne.
Maj+Fin	Effectue une sélection à partir du point d'insertion jusqu'à la fin de la ligne.
Ctrl+Maj+Début	Effectue une sélection à partir du point d'insertion jusqu'au début du document.
Ctrl+Maj+Fin	Effectue une sélection à partir du point d'insertion jusqu'à la fin du document.
Maj+Pg.Préc	Effectue une sélection à partir du point d'insertion jusqu'au début du code visible.
Maj+Pg.Suiv	Effectue une sélection à partir du point d'insertion jusqu'à la fin du code visible.
Ctrl+Maj+Pg.Préc	Effectue une sélection à partir du point d'insertion jusqu'au début du code visible.
Ctrl+Maj+Pg.Suiv	Effectue une sélection à partir du point d'insertion jusqu'à la fin du code visible.
Ctrl+A	Sélectionne l'ensemble du document.
Ctrl+D	Lorsqu'une ligne de programme ou un élément de la ligne est sélectionné, l'utilisateur peut modifier les commentaires de la ligne après avoir appuyé sur CTRL+D.
Ctrl+R	Active ou désactive l'Appel de sélecteur automatique. Par défaut, la boîte de dialogue Sélecteur de bloc d'instruction ou Sélecteur de variable s'ouvre lorsqu'un élément est ajouté à un programme de schémas à relais (LD).
Ctrl+Maj+W	Sélectionne le mot suivant.
Ctrl+Maj+J	Sélectionne jusqu'au crochet correspondant.
Maj+Alt+Flèche vers le bas	Sélectionne la ligne actuelle ainsi que les suivantes.
Maj+Alt+Flèche vers le haut	Sélectionne la ligne actuelle ainsi que les précédentes.
Maj+Alt+Flèche vers la gauche	Sélectionne vers la gauche sur la ligne actuelle.
Maj+Alt+Flèche vers la droite	Sélectionne vers la droite sur la ligne actuelle.
Ctrl+Maj+Alt+Flèche vers la gauche	Sélectionne les colonnes disponibles dans les lignes de code de gauche à droite.
Ctrl+Maj+Alt+Flèche vers la droite	Sélectionne les colonnes disponibles dans les lignes de code de droite à gauche.
Echap	Désélectionne le texte sélectionné.
Inser	Bascule entre le mode de saisie écraser/insérer.

[1] Lorsqu'aucune ligne n'est sélectionnée, une ligne est ajoutée à la fin de la liste des lignes.
[2] Lorsqu'une branche est sélectionnée, un élément est inséré à la fin de la branche.

Instruction d'alarme

Utilisez l'instruction d'alarme pour avertir lorsqu'une limite basse ou haute configurée est atteinte.

Bloc de fonctions	Description
LIM_ALRM sur la page 61	Hystérésis sur une valeur réelle pour des limites basse et haute.

LIM_ALRM (alarme limite)

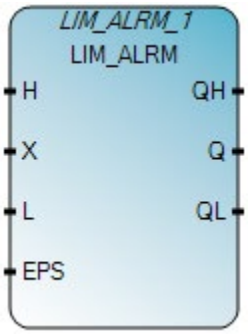
LIM_ALRM est une alarme avec hystérésis sur une valeur réelle pour des limites haute et basse.

Une hystérésis est appliquée sur les limites basse et haute. Le delta d'hystérésis utilisé pour la limite basse ou la limite haute correspond à la moitié du paramètre EPS.

Une alarme de traitement est une alarme qui se déclenche lorsque le contrôleur reçoit et traite un défaut. Les alarmes de niveau de traitement permettent d'avertir lorsque le module a franchi la limite basse ou haute configurée pour chaque canal.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

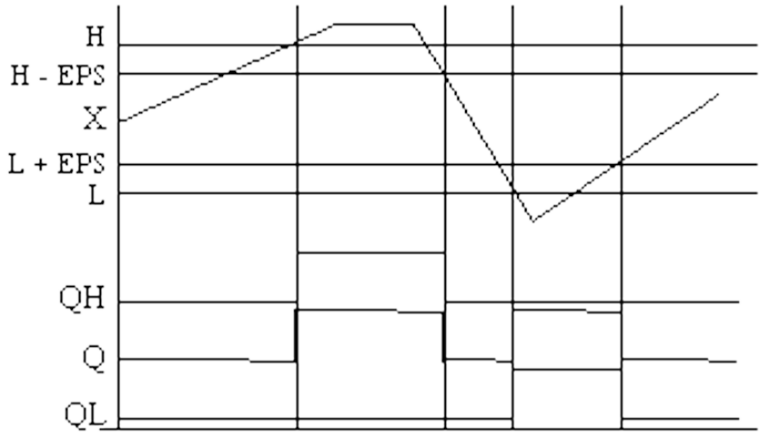


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

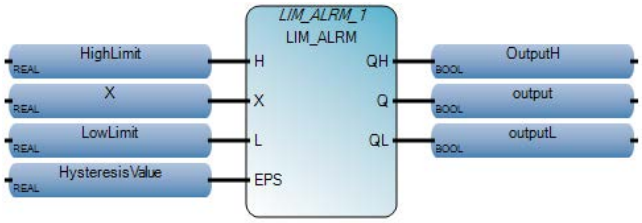
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Lorsque ce paramètre est VRAI, le bloc d'instruction est activé. VRAI : exécution du calcul LIM_ALRM actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
H	Entrée	REAL	Valeur de la limite haute.

Paramètre	Type de paramètre	Type de données	Description
X	Entrée	REAL	L'entrée est une valeur réelle, quelle qu'elle soit.
L	Entrée	REAL	Valeur de la limite basse.
EPS	Entrée	REAL	Valeur d'hystérésis (doit être supérieure à zéro).
QH	Sortie	BOOL	Alarme haute : VRAI si X est supérieur à la limite haute H.
Q	Sortie	BOOL	Sortie d'alarme : VRAI si X est hors limites.
QL	Sortie	BOOL	Alarme basse : VRAI si X est inférieur à la limite basse L.
ENO	Sortie	BOOL	Active les sorties. Ne s'applique qu'aux programmes de langage à relais.

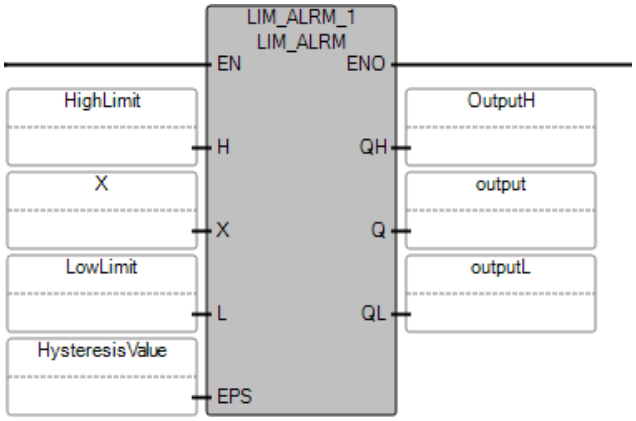
Exemple de diagramme de temporisation LIM_ALRM



Exemple de Diagramme de Bloc Fonction LIM_ALRM



Exemple de logique à relais LIM_ALRM



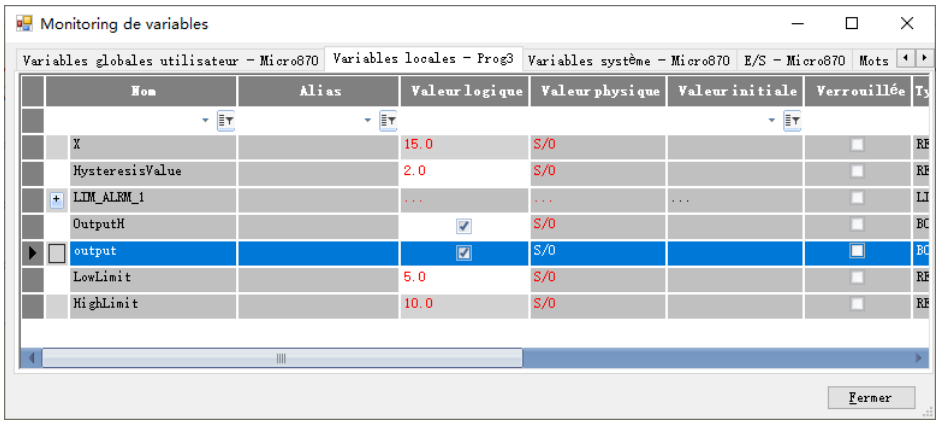
Exemple de texte structuré LIM_ALRM

```
LIM_ALRM_1(
```

void **LIM_ALRM_1**(REAL **H**, REAL **X**, REAL **L**, REAL **EPS**)
Type: LIM_ALRM, Alarme des limites hautes et basses avec hystérésis

```
1  HighLimit := 10.0;
2  X := 15.0;
3  LowLimit := 5.0;
4  HysteresisValue := 2.0;
5  LIM_ALRM_1(HighLimit, X, LowLimit, HysteresisValue);
6  OutputH := LIM_ALRM_1.QH;
7  OutputL := LIM_ALRM_1.QL;
8  output := LIM_ALRM_1.Q;
```

Résultats



Instructions arithmétiques

Utilisez les instructions arithmétiques pour effectuer des calculs mathématiques.

Fonction	Description
ABS sur la page 65	Renvoie la valeur absolue d'une valeur REAL.
ACOS sur la page 67	Calcule l'arc cosinus d'une valeur REAL.
ACOS_LREAL sur la page 68	Calcule l'arc cosinus d'une valeur REAL longue.
Addition sur la page 70	Ajoute plusieurs valeurs entières, réelles, de temps ou de chaîne.
ASIN sur la page 71	Calcule l'arc sinus d'une valeur REAL.
ASIN_LREAL sur la page 73	Calcule l'arc sinus d'une valeur REAL longue.
ATAN sur la page 74	Calcule l'arc tangente d'une valeur REAL.
ATAN_LREAL sur la page 75	Calcule l'arc tangente d'une valeur REAL longue.
COS sur la page 77	Calcule le cosinus d'une valeur REAL.
COS_LREAL sur la page 78	Calcule le cosinus d'une valeur REAL longue.
Division sur la page 80	Divise deux valeurs entières ou réelles.
EXPT sur la page 81	Calcule la valeur réelle d'un nombre de base à la puissance de l'exposant de type entier.
LOG sur la page 83	Calcule le logarithme (base 10) d'une valeur REAL.
MOD sur la page 84	Effectue un calcul du modulo sur les valeurs entières.
MOV sur la page 86	Copie une valeur d'entrée vers une sortie.
Multiplication sur la page 87	Multiplie plusieurs valeurs entières ou réelles.
Neg sur la page 89	Convertit une valeur en valeur inversée.
POW sur la page 89	Calcule la valeur d'un nombre réel à la puissance de l'exposant de type REAL.
RAND sur la page 91	Calcule des valeurs d'entiers aléatoires à partir d'une plage définie.
SIN sur la page 93	Calcule le sinus d'une valeur REAL.
SIN_LREAL sur la page 94	Calcule le sinus d'une valeur REAL longue.
SORT sur la page 96	Calcule la racine carrée d'une valeur REAL.
Soustraction sur la page 97	Soustrait une valeur entière, réelle ou de temps d'une autre valeur entière, réelle ou de temps.
TAN sur la page 98	Calcule la tangente d'une valeur REAL.
TAN_LREAL sur la page 100	Calcule la tangente d'une valeur REAL longue.
TRUNC sur la page 101	Tronque les valeurs réelles en ne gardant que l'entier.

ABS (valeur absolue)

Renvoie la valeur absolue (positive) d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

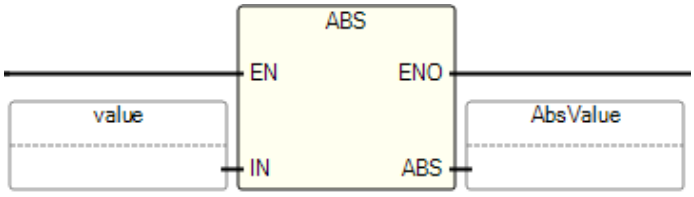
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

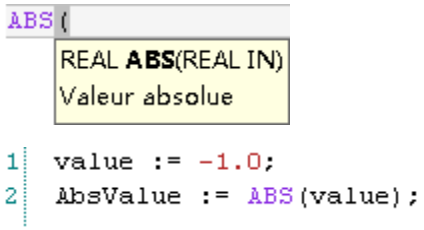
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la valeur absolue actuelle. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Valeur réelle signée, quelle qu'elle soit.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
ABS	Sortie	REAL	Valeur absolue (toujours positive).

Exemple de diagramme de bloc de fonctions ABS

Exemple de schéma à relais ABS



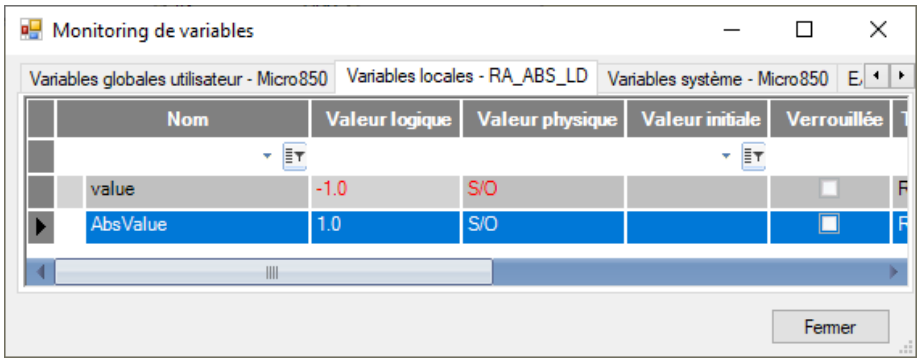
Exemple de diagramme en texte structuré ABS



(* Équivalence ST : *)

fonction := (ABS (delta) > plage) ;

Résultats



ACOS (arc cosinus de source)

Calcule l'arc cosinus d'une valeur REAL. Les valeurs d'entrée et de sortie sont exprimées en radians.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

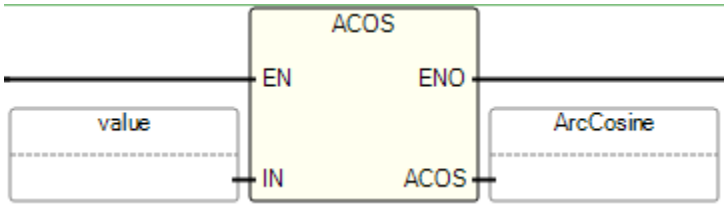
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de l'arc cosinus actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Doit être défini dans l'ensemble [-1,0 .. +1,0].
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
ACOS	Sortie	REAL	Arc cosinus de la valeur d'entrée (dans l'intervalle [-p1/2 ..+p1/2]). Renvoie 0 en cas d'entrée non valide.

Exemple de diagramme de bloc de fonctions ACOS

Exemple de schéma à relais ACOS



Exemple de texte structuré ACOS

```
1  value := 0.5;
2  ArcCosine := ACOS(value);
```

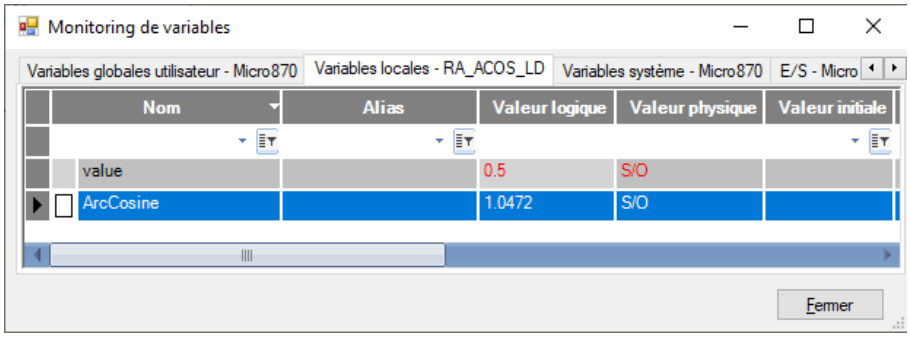
```
ACOS(
    REAL ACOS(REAL IN)
    Arc cosinus
```

(* Équivalence ST : *)

cosinus := COS (angle) ;

résultat := ACOS (cosinus) ; (* le résultat est égal à l'angle *)

Résultats

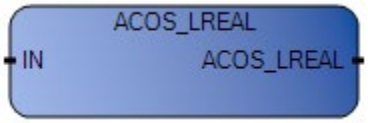


ACOS_LREAL (arc cosinus d'une valeur REAL longue)

Calcule l'arc cosinus d'une valeur REAL longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

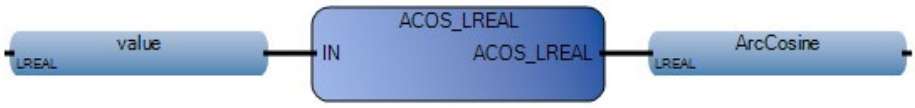
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



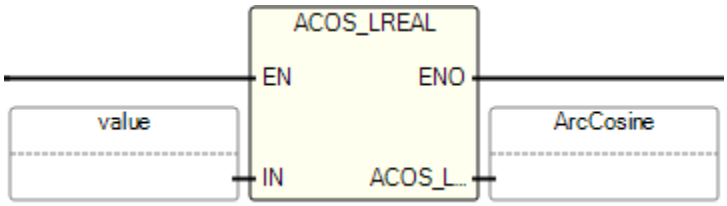
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	LREAL	Doit être défini dans l'ensemble [-1,0 .. +1,0].
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
ACOS_LREAL	Sortie	LREAL	Arc cosinus de la valeur d'entrée (dans l'ensemble [0,0 .. PI]) = 0,0 pour une entrée non valide.

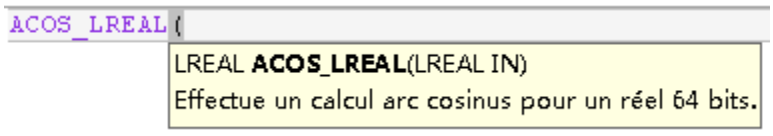
Exemple de Diagramme de Bloc Fonction ACOS_LREAL



Exemple de logique à relais ACOS_LREAL



Exemple de texte structuré ACOS_LREAL



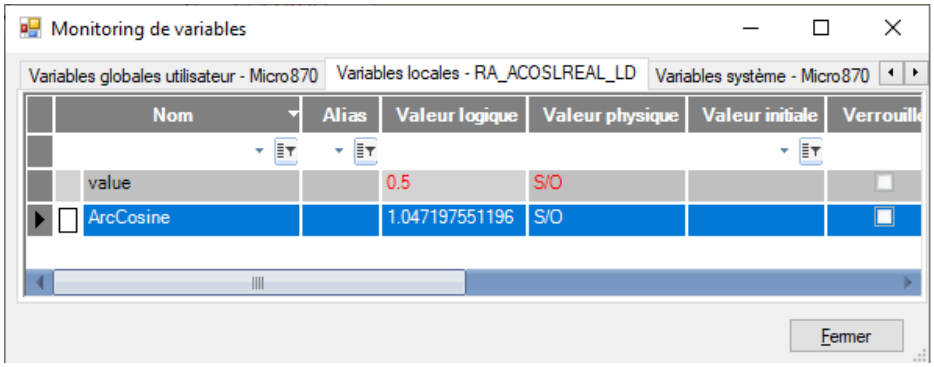

```
1  value := 0.5;
2  ArcCosine := ACOS_LREAL(value);
```

(* Équivalence ST : *)

cosinus := COS_LREAL (angle) ;

résultat := ACOS_LREAL (cosinus) ; (* le résultat est égal à l'angle *)

Résultats



Addition

Ajoute plusieurs valeurs entières, réelles, de temps ou de chaîne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul d'addition actuelle. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	Terme de type de données réel, de temps ou de chaîne. Toutes les entrées doivent être du même type de données.

Paramètre	Type de paramètre	Type de données	Description
i2	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	Terme de type de données réel, de temps ou de chaîne. Toutes les entrées doivent être du même type de données.
o1	Sortie	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	Somme des valeurs d'entrées au format réel, de temps ou de chaîne. L'entrée et la sortie doivent utiliser le même type de données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré d'addition

(* Équivalence ST : *)

```
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;
```

ASIN (arc sinus)

Calcule l'arc sinus d'une valeur REAL. Les valeurs d'entrée et de sortie sont exprimées en radians.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

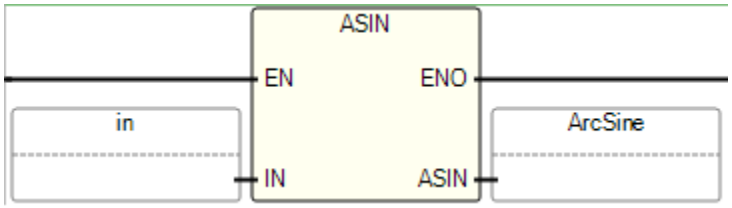
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul d'arc sinus actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Doit être défini dans l'ensemble [-1,0 .. +1,0].
ASIN	Sortie	REAL	Arc sinus de la valeur d'entrée (dans l'intervalle [-p1/2 ..+p1/2]). Renvoie 0 en cas d'entrée non valide.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ASIN

Exemple de schéma à relais ASIN



Exemple de texte structuré ASIN

```
1  in := 0.5;
2  ArcSine := ASIN(in);
```

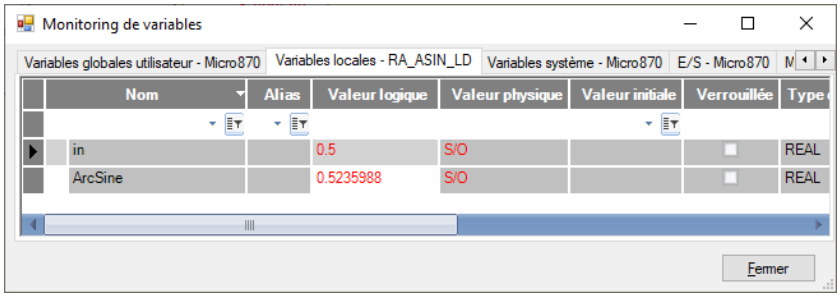
```
ASIN(
    REAL ASIN(REAL IN)
    Arc sinus
```

(* Équivalence ST : *)

sinus := SIN (angle) ;

résultat := ASIN (sinus); (* le résultat est égal à l'angle *)

Résultats

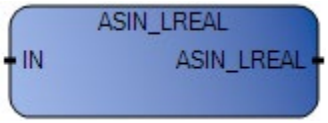


ASIN_LREAL (arc sinus d'une valeur REAL longue)

Calcule l'arc sinus d'une valeur REAL longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

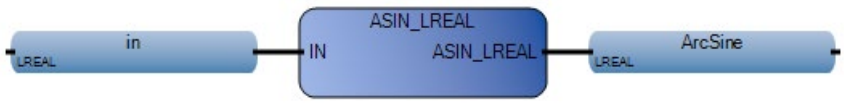
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



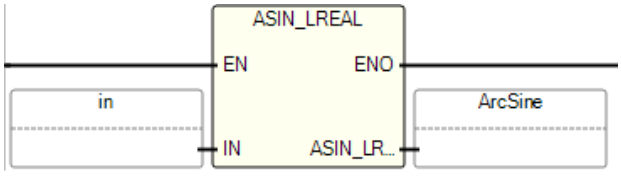
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	LREAL	Doit être défini dans l'ensemble [-1,0 .. +1,0].
ASIN_LREAL	Sortie	LREAL	Arc sinus de la valeur d'entrée (dans l'ensemble [-PI/2 .. +PI/2]) = 0,0 pour une entrée non valide.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ASIN_LREAL



Exemple de logique à relais ASIN_LREAL



Exemple de texte structuré ASIN_LREAL

```
1  in := 0.5;
2  ArcSine := ASIN_LREAL(in);
```

```
ASIN_LREAL(
```

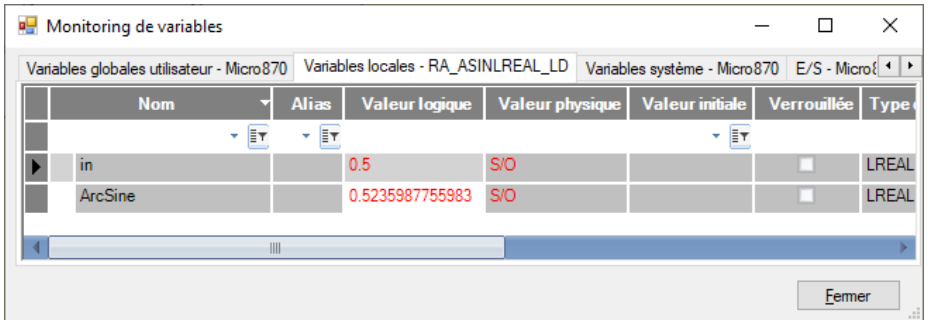
LREAL **ASIN_LREAL**(LREAL IN)
Effectue un calcul arc sinus pour un réel 64 bits.

(* Équivalence ST : *)

sinus := SIN_LREAL (angle);

résultat := ASIN_LREAL (sinus); (* le résultat est égal à l'angle *)

Résultats

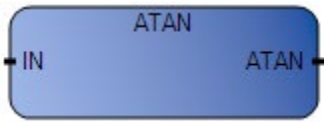


ATAN (arc tangente)

Calcule l'arc tangente d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

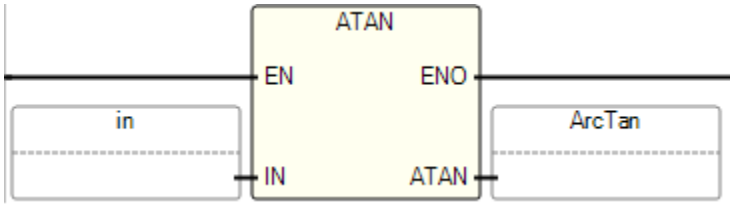


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul d'arc tangente actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Valeur réelle, quelle qu'elle soit.
ATAN	Sortie	REAL	Arc tangente de la valeur d'entrée (dans l'ensemble [-PI/2 .. +PI/2]) = 0,0 pour une entrée non valide.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ATAN

Exemple de schéma à relais ATAN



Exemple de texte structuré ATAN

```
1  in := 0.5;
2  ArcTan := ATAN(in);
```

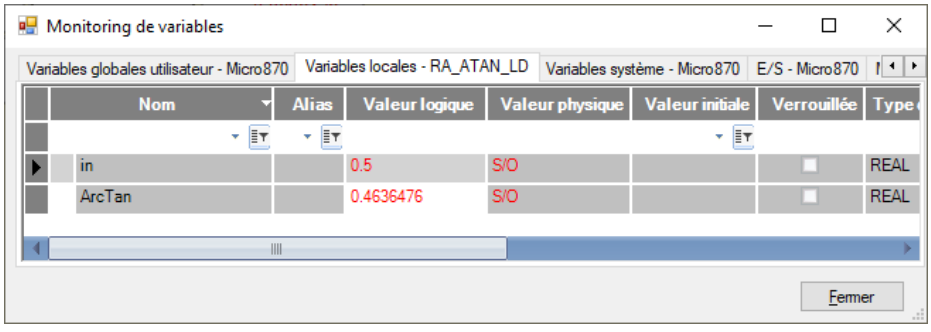
```
ATAN(
    REAL ATAN(REAL IN)
    Arc tangente
```

(* Équivalence ST : *)

tangente := TAN (angle) ;

résultat := ATAN (tangente) ; (* le résultat est égal à l'angle*)

Résultats

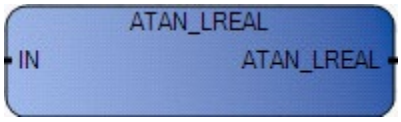


ATAN_LREAL (arc tangente d'une valeur REAL longue)

Calcule l'arc tangente d'une valeur REAL longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

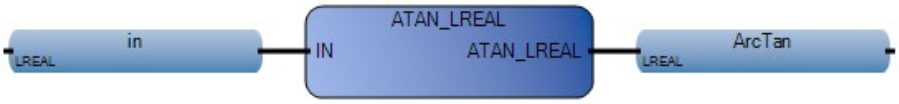
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



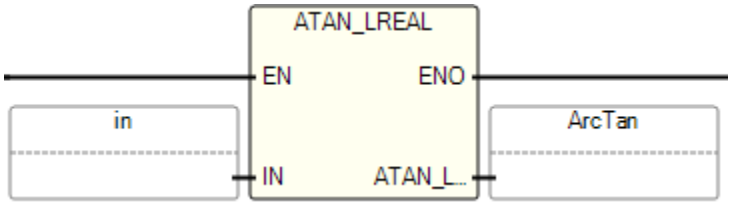
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	LREAL	Valeur réelle longue, quelle qu'elle soit.
ATAN_LREAL	Sortie	LREAL	Arc tangente de la valeur d'entrée (dans l'ensemble [-PI/2 .. +PI/2]) = 0,0 pour une entrée non valide.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ATAN_LREAL

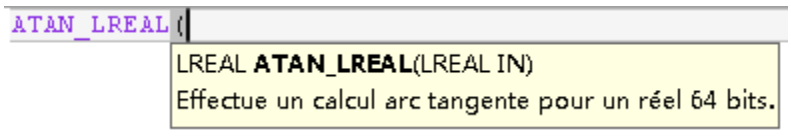


Exemple de logique à relais ATAN_LREAL



Exemple de texte structuré ATAN_LREAL

```
1  in := 0.5;
2  ArcTan := ATAN_LREAL(in);
```

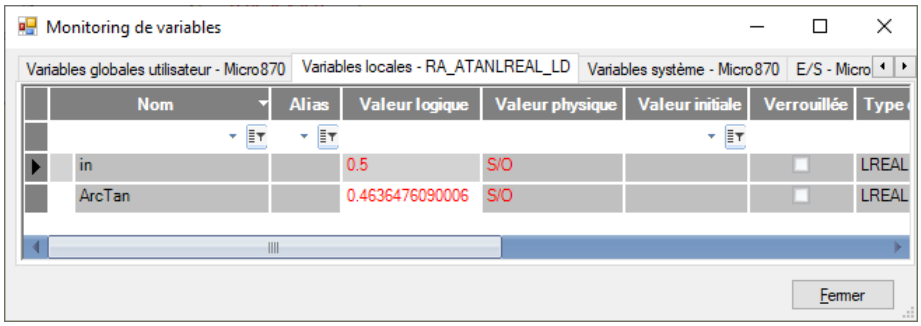


(* Équivalence ST : *)

tangente := TAN_LREAL (angle) ;

résultat := ATAN_LREAL (tangente) ; (* le résultat est égal à l'angle*)

Résultats



COS (cosinus)

Calcule le cosinus d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

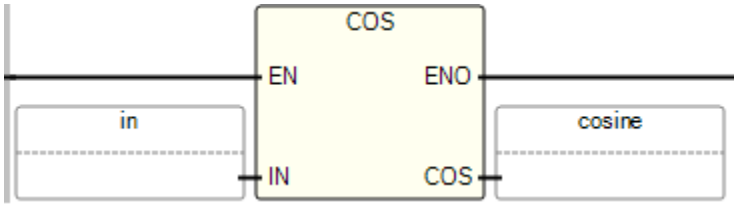
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

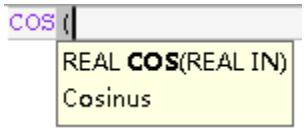
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du cosinus actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Valeur réelle, quelle qu'elle soit.
COS	Sortie	REAL	Cosinus de la valeur d'entrée (dans l'ensemble [-1,0 .. +1,0]).
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions COS

Exemple de schéma à relais COS



Exemple de texte structuré COS



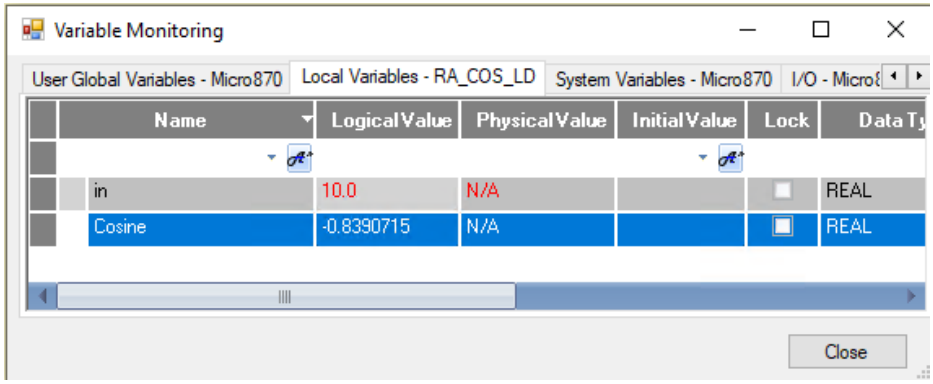

```
1  in := 10.0;
2  cosine := COS(in);
```

(* Équivalence ST : *)

cosinus := COS (angle) ;

résultat := ACOS (cosinus) ; (* le résultat est égal à l'angle *)

Résultats

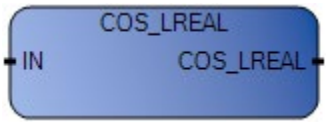


COS_LREAL (cosinus d'une valeur réelle longue)

Calcule le cosinus d'une valeur REAL longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

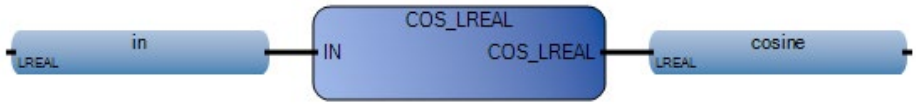
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



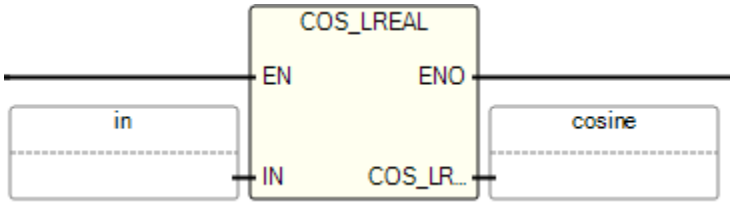
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du cosinus actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	LREAL	Valeur réelle longue, quelle qu'elle soit.
COS_LREAL	Sortie	LREAL	Cosinus de la valeur d'entrée (dans l'ensemble [-1,0 .. +1,0]).
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

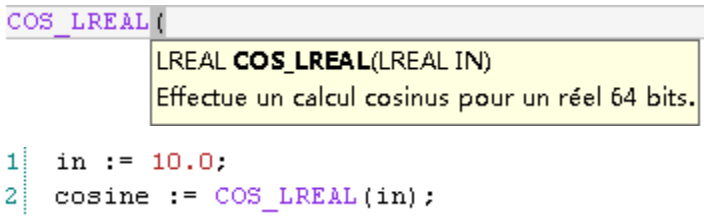
Exemple de Diagramme de Bloc Fonction COS_LREAL



Exemple de logique à relais COS_LREAL



Exemple de texte structuré COS_LREAL

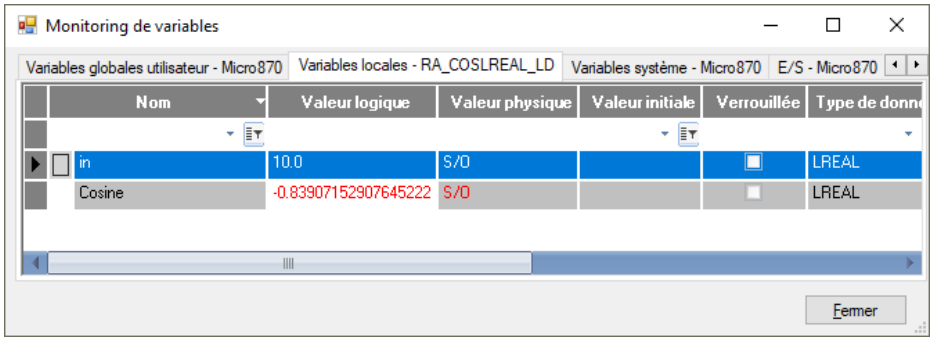


(* Équivalence ST : *)

cosinus := COS_LREAL (angle) ;

résultat := ACOS_LREAL (cosinus) ; (* le résultat est égal à l'angle *)

Résultats



Division

Divise la première valeur d'entrée entière ou réelle par la seconde valeur d'entrée entière ou réelle.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de division actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Dividende d'un type de données entier ou réel non nul. Toutes les entrées doivent être du même type de données.
i2	Entrée	SINT USINT BYTE	Diviseur d'un type de données entier ou réel non nul. Toutes les entrées doivent être du même type de données.

Paramètre	Type de paramètre	Type de données	Description
		INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	
o1	Sortie	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Quotient des entrées de type entier ou réel non nul. L'entrée et la sortie doivent utiliser le même type de données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré de division

(* Équivalence ST : *)

```
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;
```

EXPT (exposant)

Calcule la valeur IN (base) à la puissance EXP (exposant) et renvoie le résultat de type Real de l'opération.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

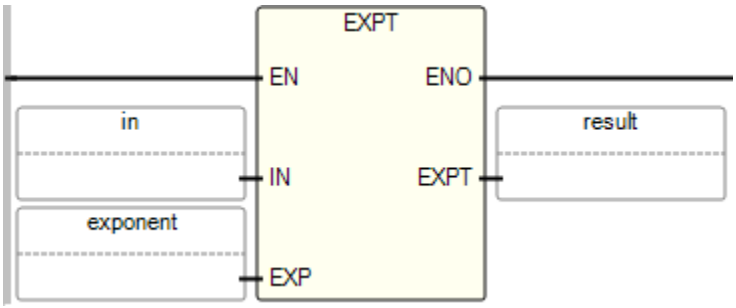
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de l'exposant actuel. FAUX : aucun calcul.
IN	Entrée	REAL	Valeur réelle signée, quelle qu'elle soit.
EXP	Entrée	DINT	Exposant de type entier.
EXPT	Sortie	REAL	Valeur réelle IN à la puissance EXP.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions EXPT

Exemple de schéma à relais EXPT



Exemple de texte structuré EXPT

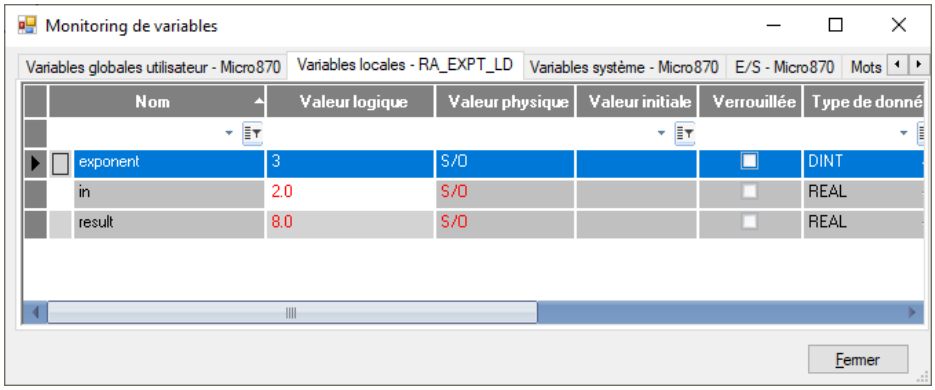
```
1  in := 2.0;
2  exponent := 3;
3  result := EXPT(in, exponent);
```

```
EXPT(
     REAL EXPT(REAL IN, DINT EXP)
     Exposant
```

(* Équivalence ST : *)

tb_size := ANY_TO_DINT (EXPT (2,0, plage)) ;

Résultats



LOG (logarithme de base 10)

Calcule le logarithme (base 10) d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

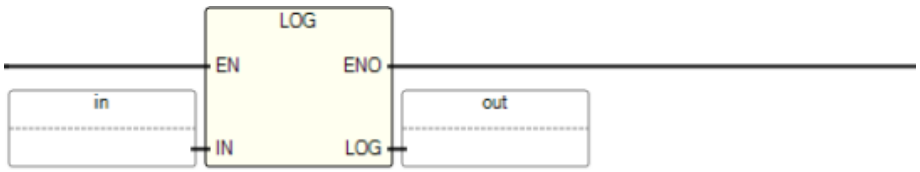
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du logarithme actuel. FAUX : aucun calcul.
IN	Entrée	REAL	Doit être supérieur à zéro.
LOG	Sortie	REAL	Logarithme (base 10) de la valeur d'entrée. Le résultat renvoyé est -3.4E+38 pour les valeurs IN nulles et négatives.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions LOG

Exemple de schéma à relais LOG



Exemple de texte structuré LOG

```
1  in := 10.0;
2  output := LOG(in);
```

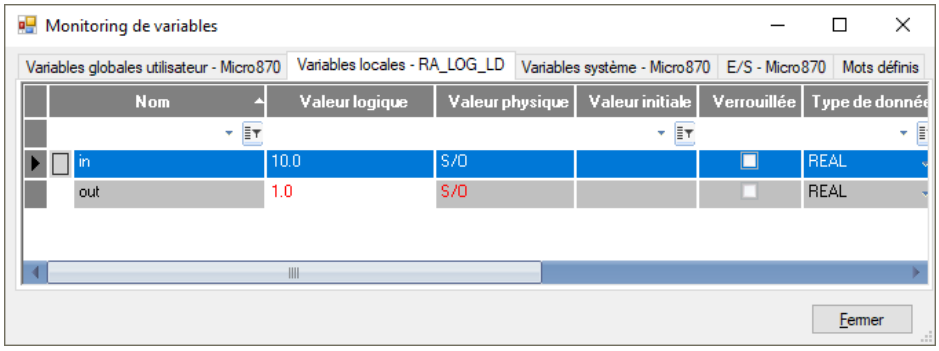
```
LOG(
    REAL LOG(REAL IN)
    Logarithme
```

(* Équivalence ST : *)

xpos := ABS (xval) ;

xlog := LOG (xpos) ;

Résultats



MOD (modulo)

Divise l'entrée IN par l'entrée Base et place le reste dans la sortie MOD.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

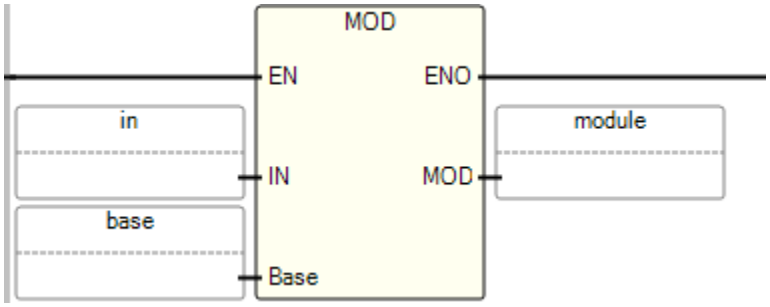
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

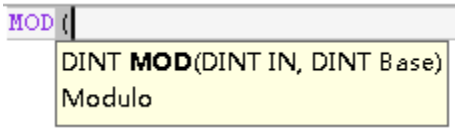
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de module. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Valeur entière signée, quelle qu'elle soit.
Base	Entrée	DINT	Doit être supérieur à zéro.
MOD	Sortie	DINT	Calcul du modulo (entrée MOD base) / retourne -1 si Base <= 0.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions MOD

Exemple de schéma à relais MOD



Exemple de texte structuré MOD



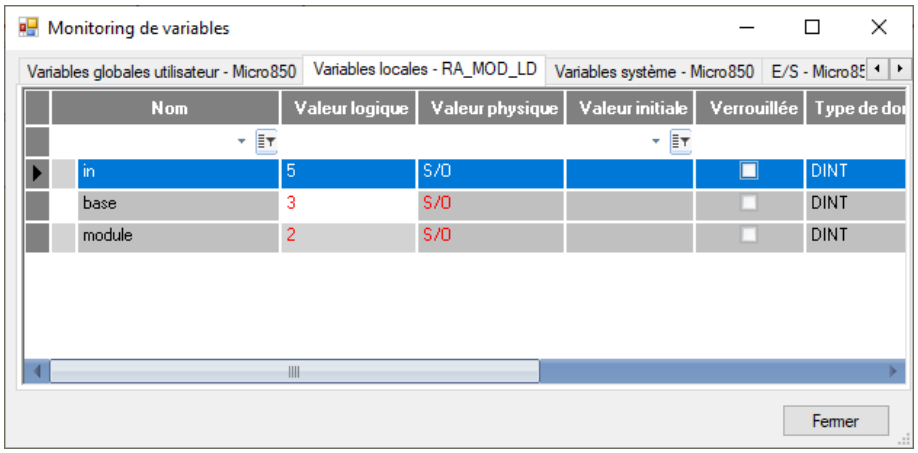

```
1  in := 5;
2  base := 3;
3  module := MOD(in, base);
```

(* Équivalence ST : *)

division_result := (valeur / diviseur) ; (* division d'un entier *)

```
rest_of_division := MOD (valeur, diviseur) ; (* reste de la division *)
```

Résultats



MOV (déplacer)

Affecte la valeur d'entrée (i1) à la sortie (o1).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Pour les programmes de texte structuré, utilisez l'opérateur d'égalité (=) à la place de MOV.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du lien direct vers un calcul de sortie. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL DINT REAL TIME STRING SINT USINT INT UINT UDINT	L'entrée et la sortie doivent utiliser le même type de données.

Paramètre	Type de paramètre	Type de données	Description
		LINT ULINT DATE LREAL BYTE WORD DWORD LWORD	
o1	Sortie	BOOL DINT REAL TIME STRING SINT USINT INT UINT UDINT LINT ULINT DATE LREAL BYTE WORD DWORD LWORD	L'entrée et la sortie doivent utiliser le même type de données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré

(* Équivalence ST : *)

```
ao23 := ai10;
```

Multiplication

Multiplie plusieurs valeurs entières ou réelles.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de multiplication actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Facteur d'un type de données entier ou réel. Toutes les entrées doivent être du même type de données.
i2	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT LINT LWORD REAL LREAL	Facteur d'un type de données entier ou réel. Toutes les entrées doivent être du même type de données.
o1	Sortie	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Produit des entrées de type entier ou réel. L'entrée et la sortie doivent utiliser le même type de données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré Multiplication

(* Equivalence ST *)

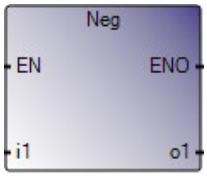
```
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;
```

Neg (négation)

Convertit une valeur en valeur inversée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de la conversion actuelle en calcul négatif. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT INT DINT LINT REAL LREAL	L'entrée et la sortie doivent utiliser le même type de données.
o1	Sortie	SINT INT DINT LINT REAL LREAL	L'entrée et la sortie doivent utiliser le même type de données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré Neg

(* Équivalence ST : *)

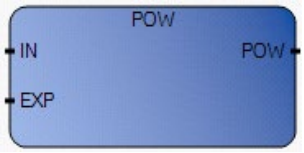
```
ao23 := - (ai10);
ro100 := - (ri1 + ri2);
```

POW (augmentation puissance)

Lorsque 'base' est le premier argument et 'exposant' le second, calcule le résultat de type REAL de (base exposant).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

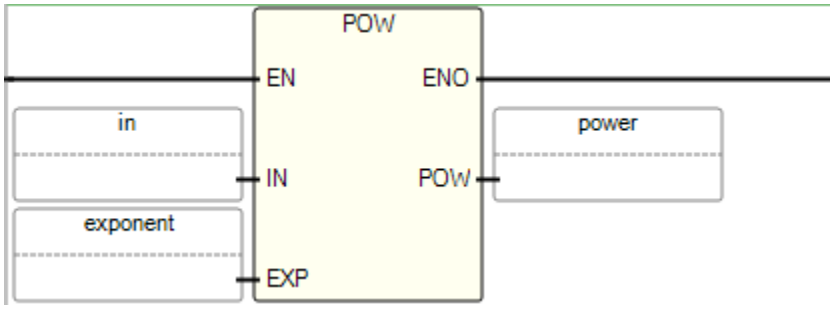


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de l'exposant actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Nombre réel auquel affecter l'exposant.
EXP	Entrée	REAL	Puissance (exposant).
POW	Sortie	REAL	(IN EXP) 1,0 si IN est différente de 0,0 et EXP égale à 0,0 0,0 si IN est égale à 0,0 et EXP est négative 0,0 si les valeurs IN et EXP sont toutes les deux égales à 0,0 0,0 si IN est négative et EXP ne correspond pas à un entier.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions POW

Exemple de schéma à relais POW



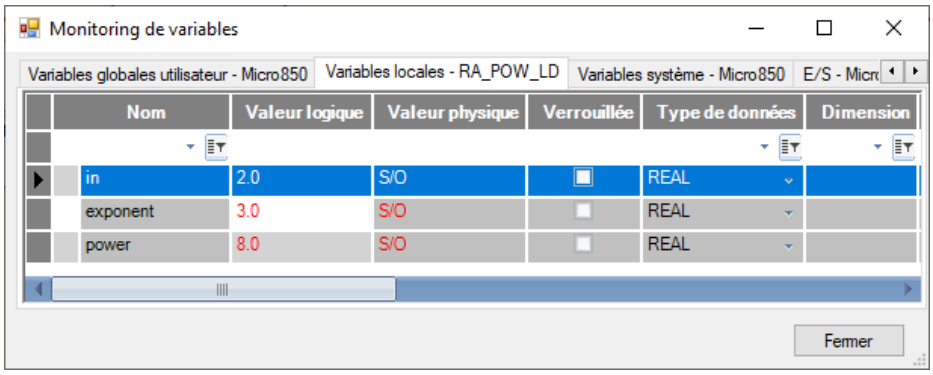
Exemple de texte structuré POW

```
POW(
    REAL POW(REAL IN, REAL EXP)
    Calcul de puissance

1  in := 2.0;
2  exponent := 3.0;
3  power := POW(in, exponent);
```

(* Équivalence ST : *)

résultat := POW (xval, puissance) ;

Résultats



RAND (valeur aléatoire)

Calcule des valeurs d'entiers aléatoires à partir d'une plage définie.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

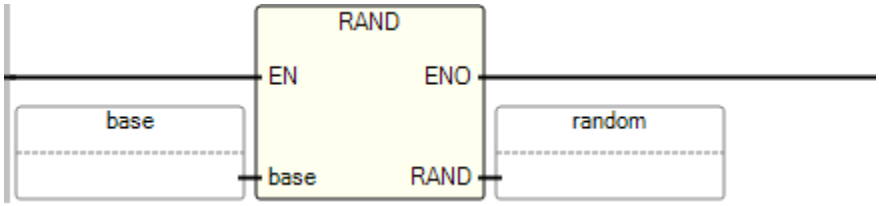
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la valeur entière aléatoire. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
base	Entrée	DINT	Définit l'ensemble de nombres pris en charge.
RAND	Sortie	DINT	Valeur aléatoire dans l'ensemble [0..base-1].
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions RAND

Exemple de schéma à relais RAND



Exemple de texte structuré RAND

```
RAND(
     DINT RAND(DINT base)
     Valeur au hasard
1  base := 10;
2  random := RAND(base);
```

(* Équivalence ST : *)

sélection := MUX4 (RAND (4), 1, 4, 8, 16) ;

(*

sélection aléatoire de l'une des 4 valeurs prédéfinies

la valeur émise par l'appel RAND est comprise dans l'ensemble [0..3] *)

donc la « sélection » effectuée à partir de MUX4 extraira « aléatoirement » la valeur

1 si 0 est émis par RAND,

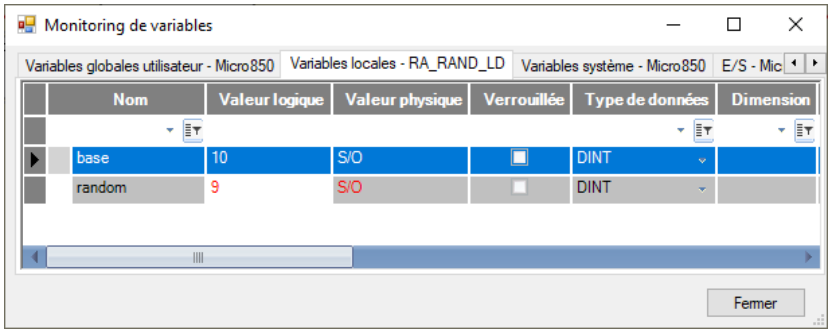
ou 4 si 1 est émis par RAND,

ou 8 si 2 est émis par RAND,

ou 16 si 3 est émis par RAND,

*)

Résultats



SIN (sinus)

Calcule le sinus d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

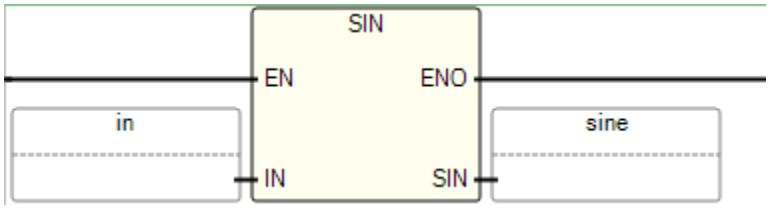
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

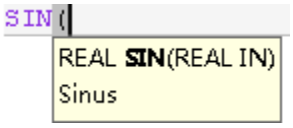
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du sinus actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Valeur réelle, quelle qu'elle soit.
SIN	Sortie	REAL	Sinus de la valeur d'entrée (dans l'ensemble [-1,0 .. +1,0]).
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions SIN

Exemple de schéma à relais SIN



Exemple de texte structuré SIN



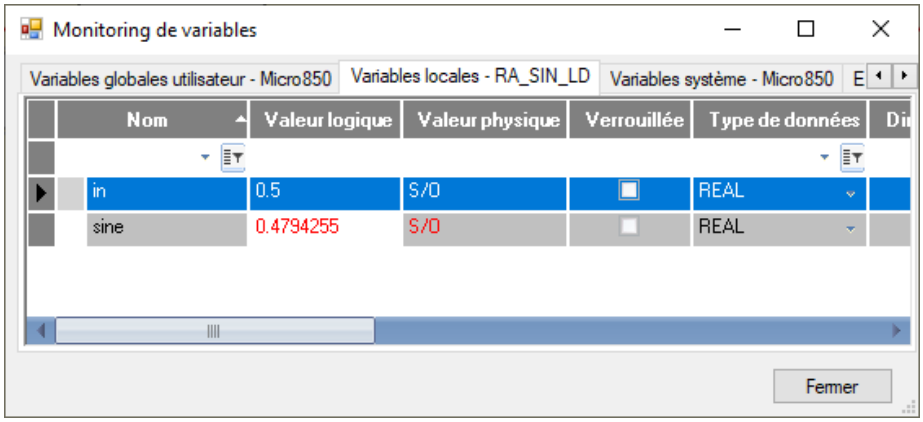

```
1  in := 0.5;
2  sine := SIN(in);
```

(* Équivalence ST : *)

sinus := SIN (angle) ;

résultat := ASIN (sinus); (* le résultat est égal à l'angle *)

Résultats

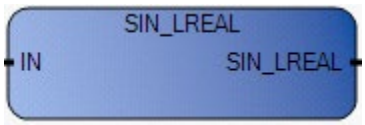


SIN_LREAL (sinus d'une valeur réelle longue)

Calcule le sinus d'une valeur REAL longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

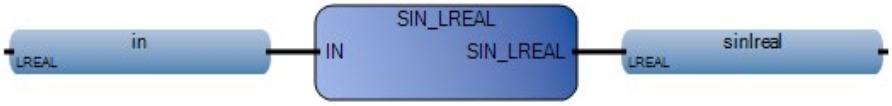


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

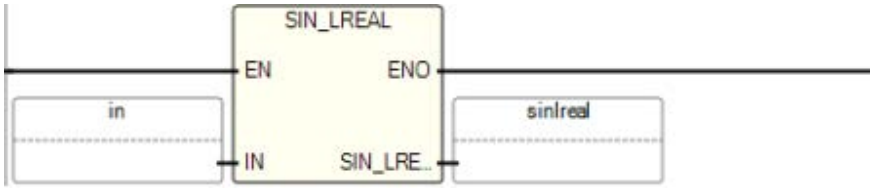
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	LREAL	Valeur réelle longue, quelle qu'elle soit.
SIN_LREAL	Sortie	LREAL	Sinus de la valeur d'entrée (dans l'ensemble [-1,0 .. +1,0]).

Paramètre	Type de paramètre	Type de données	Description
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

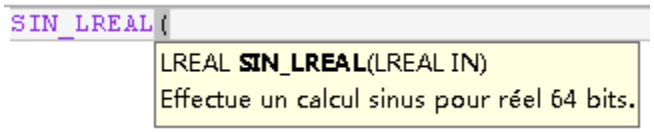
Exemple de diagramme de bloc de fonctions SIN_LREAL



Exemple de logique à relais SIN_LREAL



Exemple de texte structuré SIN_LREAL

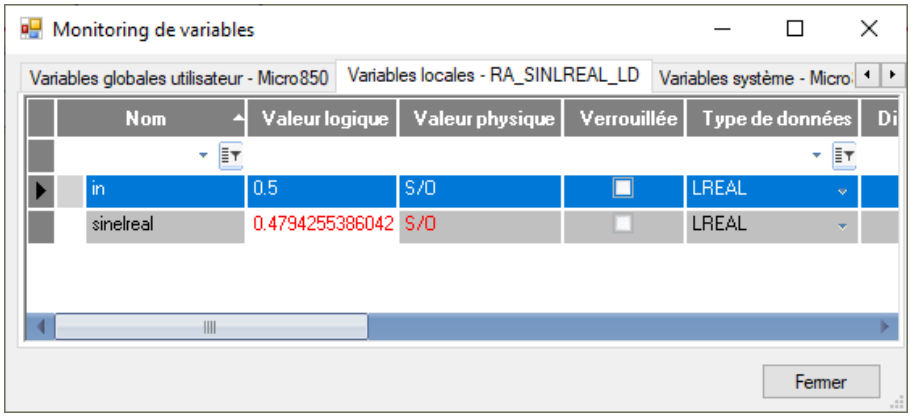



```
1  in := 0.5;
2  sinlreal := SIN_LREAL(in);
```

(* Équivalence ST : *)

TESTOUTPUT1 := SIN_LREAL(TESTINPUT1) ;

Résultats



SQRT (racine carrée)

Calcule la racine carrée d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

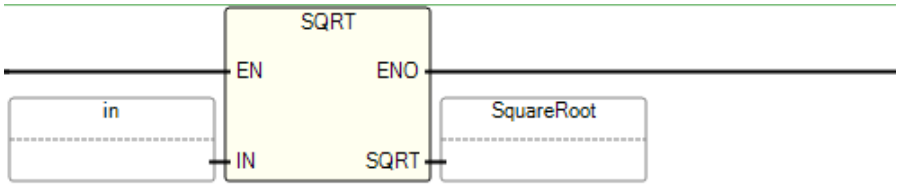
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la racine carrée actuelle. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Doit être supérieur ou égal à zéro.
SQRT	Sortie	REAL	Racine carrée de la valeur d'entrée. Le résultat renvoyé est 0 pour les valeurs IN négatives.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

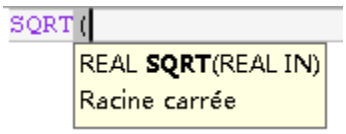
Exemple de diagramme de bloc de fonctions SQRT

Exemple de logique à relais SQRT



Exemple de texte structuré SQRT

```
1  in := 16.0;
2  SquareRoot := SQRT(in);
```

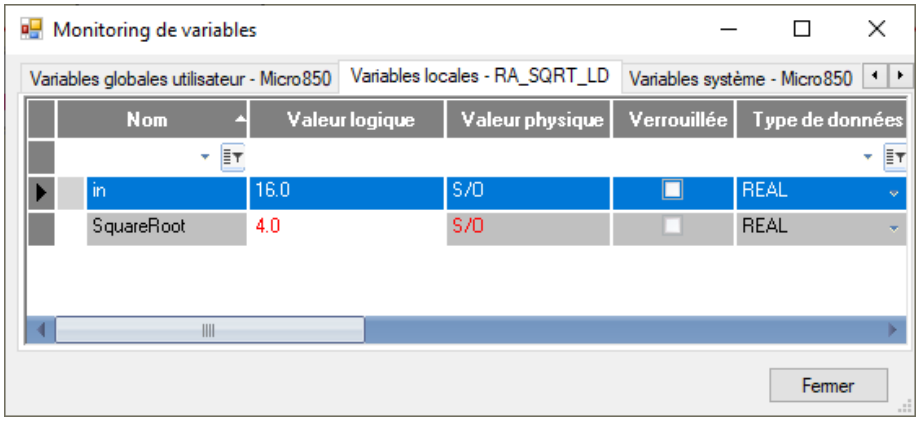


(* Équivalence ST : *)

xpos := ABS (xval) ;

xroot := SQRT (xpos).

Résultats

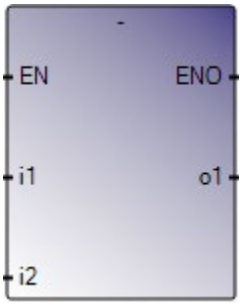


Soustraction

Soustrait une valeur entière, réelle ou de temps d'une autre valeur entière, réelle ou de temps.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul d'addition actuelle. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT	Diminuende d'un type de données entier, réel ou de temps. Toutes les entrées doivent être du même type de données.

Paramètre	Type de paramètre	Type de données	Description
		DWORD LINT ULINT LWORD REAL LREAL TIME	
i2	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME	Diminuteur d'un type de données entier, réel ou de temps. Toutes les entrées doivent être du même type de données.
o1	Sortie	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME	Différence entre le diminuende et le diminuteur dans un type de données entier, réel ou de temps. Les sorties doivent être du même type de données que les entrées.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré de soustraction

(* Équivalence ST : *)

```
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;
```

TAN (tangente)

Calcule la tangente d'une valeur REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

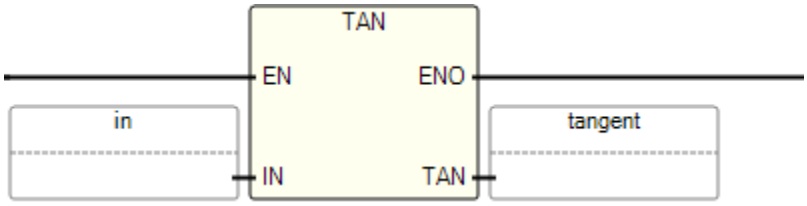
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la tangente actuelle. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Ne peut pas être égal à modulo PI/2 ni à modulo PI.
TAN	Sortie	REAL	Tangente de la valeur d'entrée = 1E+38 pour les entrées non valides.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions TAN

Exemple de schéma à relais TAN



Exemple de texte structuré TAN

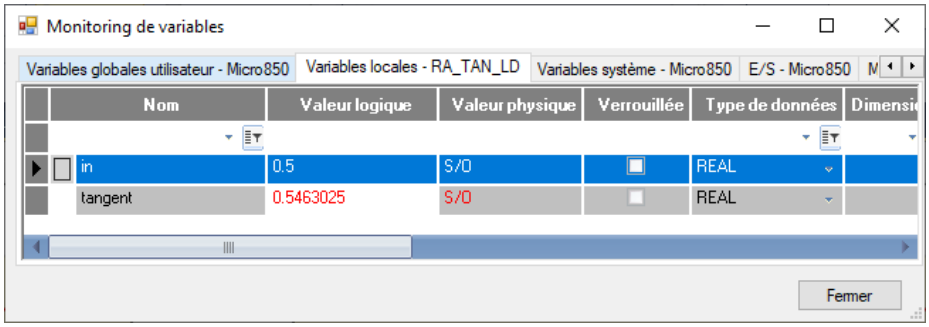
```
1  in := 0.5;
2  tangent := TAN(in);
```

(* Équivalence ST : *)

tangente := TAN (angle) ;

résultat := ATAN (tangente) ; (* le résultat est égal à l'angle*)

Résultats

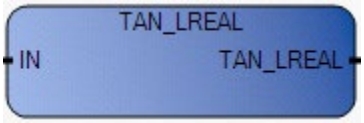


TAN_LREAL (tangente d'une valeur réelle longue)

Calcule la tangente d'une valeur REAL longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

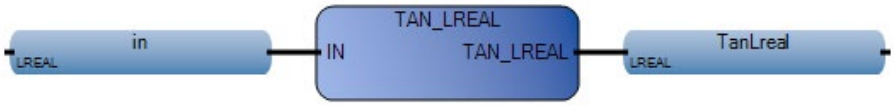
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



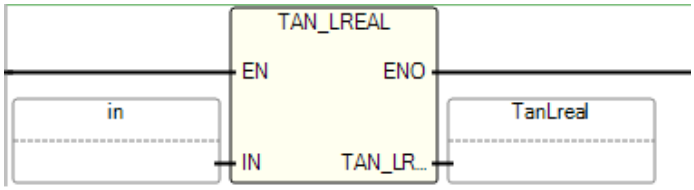
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	LREAL	Ne peut pas être égal à modulo PI/2 ni à modulo PI.
TAN_LREAL	Sortie	LREAL	Tangente de la valeur d'entrée = 1E+38 pour les entrées non valides.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction TAN_LREAL



Exemple de logique à relais TAN_LREAL



Exemple de texte structuré TAN_LREAL

```
TAN_LREAL(
```

LREAL **TAN_LREAL**(LREAL IN)
Effectue un calcul tangente pour réel 64 bits.

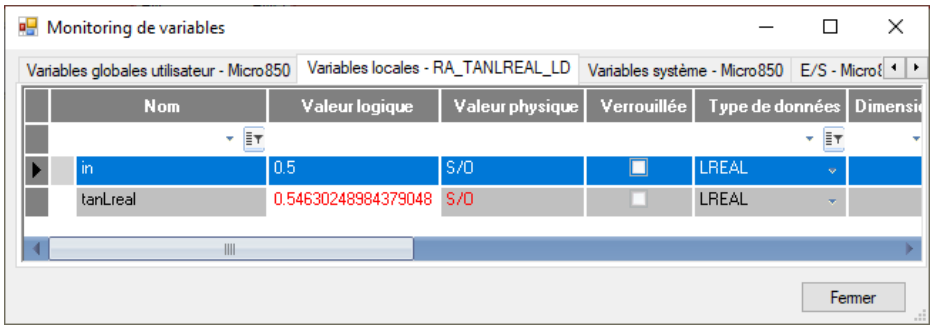
```
1  in := 0.5;
2  TanLreal := TAN_LREAL(in);
```

(* Équivalence ST : *)

tangente := TAN_LREAL (angle) ;

résultat := ATAN_LREAL (tangente) ; (* le résultat est égal à l'angle*)

Résultats



TRUNC (troncation)

Tronque les valeurs réelles en ne gardant que l'entier.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

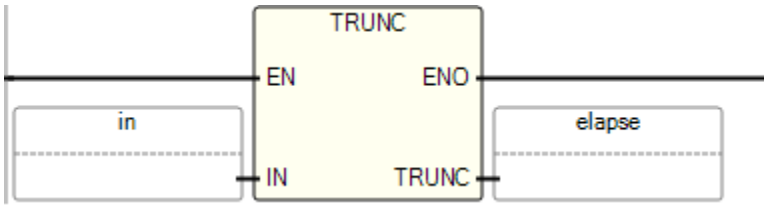
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la troncation de la valeur REAL. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	REAL	Valeur réelle, quelle qu'elle soit.
TRUNC	Sortie	REAL	Si IN>0, entier le plus grand inférieur ou égal à l'entrée. Si IN<0, entier le plus petit supérieur ou égal à l'entrée.

Paramètre	Type de paramètre	Type de données	Description
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions TRUNC

Exemple de schéma à relais TRUNC



Exemple de texte structuré TRUNC

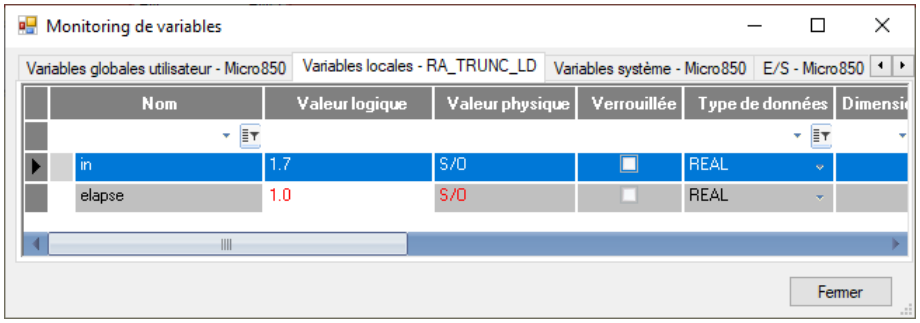
```
1  in := 1.7;
2  truncation := TRUNC(in);
```

(* Équivalence ST : *)

résultat := TRUNC (+2,67) + TRUNC (-2,0891) ;

(* signification : résultat := 2,0 + (-2,0) := 0,0; *)

Résultats



Instructions de port série ASCII

Utilisez les instructions de port série ASCII afin d'utiliser ou de modifier le canal de communication pour la réception ou la transmission de données.

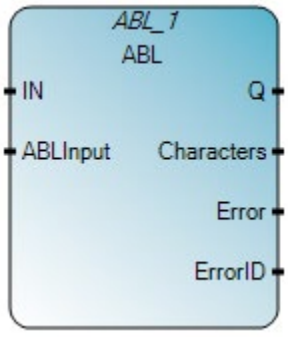
Bloc de fonctions	Description
ABL sur la page 103	Compte le nombre de caractères dans la mémoire tampon, jusqu'au caractère de fin de ligne inclus.
ACB sur la page 109	Compte le nombre total de caractères dans la mémoire tampon.
ACL sur la page 105	Vide les mémoires tampons de réception et de transmission.
AHL sur la page 107	Définit ou réinitialise les lignes de contrôle de flux avec le modem.
ARD sur la page 111	Lit les caractères de la mémoire tampon d'entrée et les place dans une chaîne.
ARL sur la page 114	Lit une ligne de caractères à partir de la mémoire tampon d'entrée et insère ces caractères dans une chaîne.
AWA sur la page 116	Écrit une chaîne avec deux caractères (configurés par l'utilisateur) ajoutés à un appareil externe.
AWT sur la page 118	Écrit les caractères provenant d'une chaîne source vers un appareil externe.

ABL (test ASCII pour la ligne de la mémoire tampon)

Compte le nombre de caractères ASCII dans la mémoire tampon d'entrée, jusqu'aux caractères de fin de ligne inclus.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc de fonctions à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.

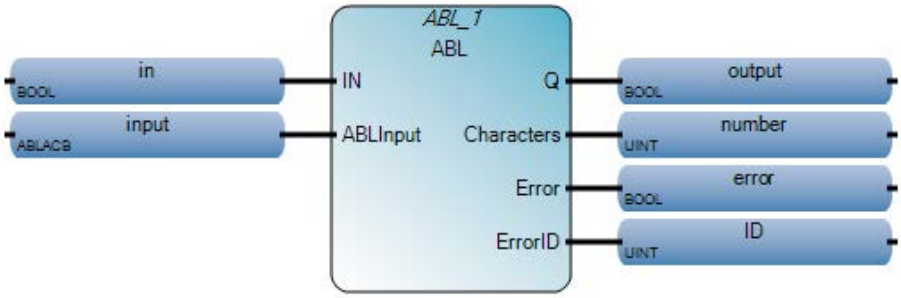
Paramètre	Type de paramètre	Type de données	Description
ABLInput	Entrée	ABLACB	Canal à exploiter. Utilisez le type de données ABLACB sur la page 120 pour définir les paramètres Canal, Type de déclenchement et Abandon pour ABLInput.
Q	Sortie	BOOL	Indique si le compte de caractères est en cours ou terminé. La sortie est mise à jour de manière asynchrone à partir de l'analyse du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
Characters	Sortie	UINT	Nombre de caractères dans la mémoire tampon. La limite de la mémoire tampon est de 82 caractères.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

Codes d'erreur ABL

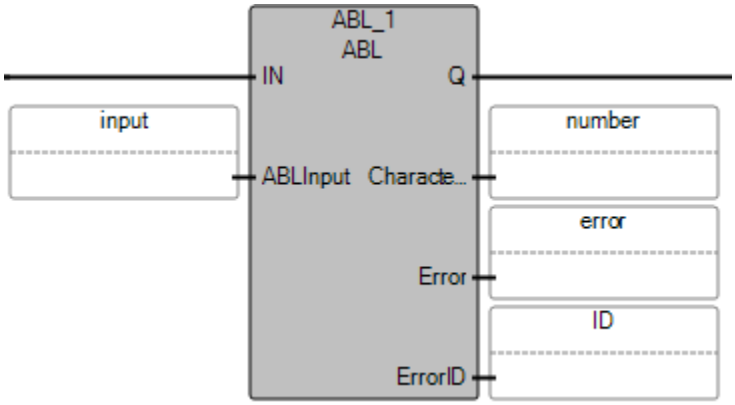
Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions ABL



Exemple de schéma à relais ABL



Exemple de texte structuré ABL

```
ABL_1(
```

void **ABL_1**(BOOL **IN**, ABLACB **ABLInput**)
Type: ABL, Spécifie le nombre de caractères dans le tampon (incluant la fin de ligne).

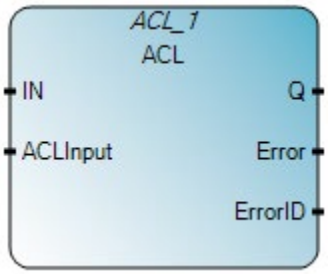
```
1  ABL_1(in, input);
2  output := ABL_1.Q;
3  number := ABL_1.Characters;
4  error := ABL_1.Error;
5  ID := ABL_1.ErrorID;
```

ACL (vidage de la mémoire tampon ASCII)

Vide la ou les mémoires tampon de réception et de transmission et supprime les instructions dans la file d'attente ASCII.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc de fonctions à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.

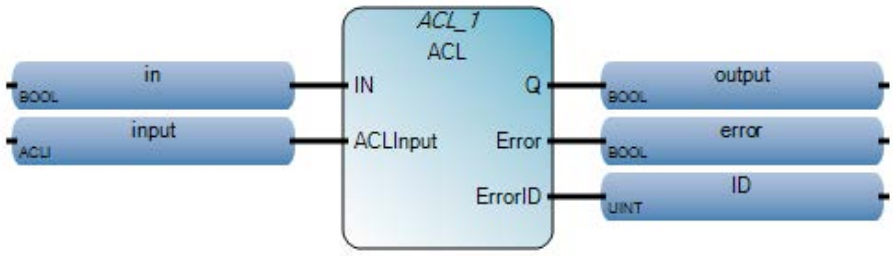
Paramètre	Type de paramètre	Type de données	Description
ACLInput	Entrée	ACLI	Le canal à utiliser, et l'état de la transmission et des mémoires tampon de réception. Pour RXBuffer, vide la mémoire tampon de réception et supprime les blocs de fonctions ASCII de réception (ARL et ARD) de la file d'attente ASCII. Pour TXBuffer, vide la mémoire tampon de transmission et supprime les blocs de fonctions ASCII de transmission (AWA et AWT) de la file d'attente ASCII. Utilisez le type de données ACLI sur la page 121 pour définir les paramètres Canal, RXBuffer et TXBuffer pour ACLInput.
Q	Sortie	BOOL	Indique si le processus de suppression de la file d'attente ASCII est en cours ou terminé. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

Codes d'erreur ABL

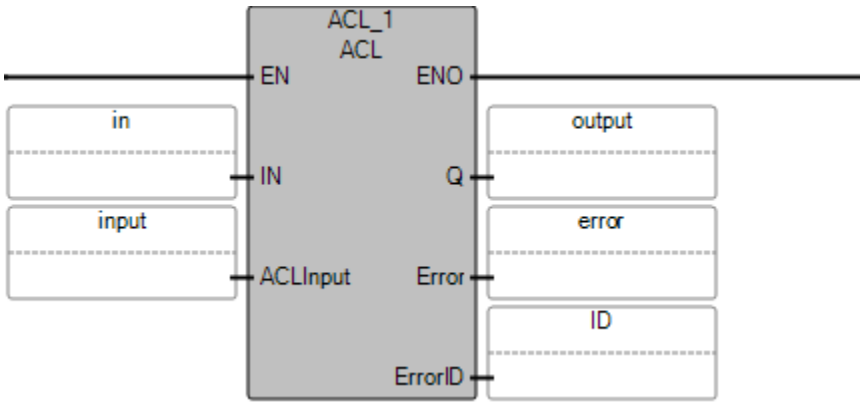
Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions ACL



Exemple de schéma à relais ACL



Exemple de texte structuré ACL

```
ACL_1(
```

void **ACL_1**(BOOL **IN**, ACLI **ACLInput**)
Type: ACL, Efface le contenu des tampons de réception et transmission.

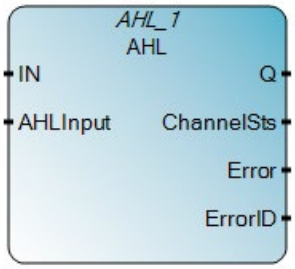
```
1  ACL_1(in, input);
2  output := ACL_1.Q;
3  error := ACL_1.Error;
4  ID := ACL_1.ErrorID;
```

AHL (lignes de contrôle de flux ASCII)

Définit ou réinitialise les lignes de contrôle de flux RS-232 RTS (Request to Send) pour un modem.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.

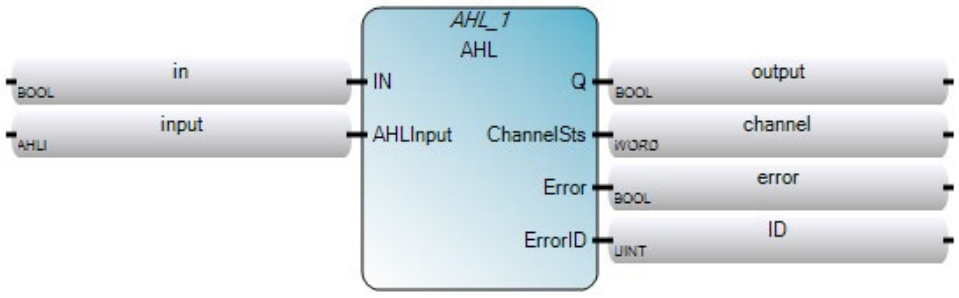
Paramètre	Type de paramètre	Type de données	Description
AHLInput	Entrée	AHLI	Le canal à utiliser, et la définition ou réinitialisation de la ligne de contrôle RTS pour le modem. Utilisez le type de données AHLI sur la page 121 pour définir les paramètres Canal, SetRts, ClrRts et Abandon pour AHLInput.
Q	Sortie	BOOL	Indique la fin de la définition ou de la réinitialisation. Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
ChannelSts	Sortie	WORD	Affiche le statut actuel (de 0000 à 001F) des lignes d'établissement de liaison pour le canal spécifié.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

Codes d'erreur ABL

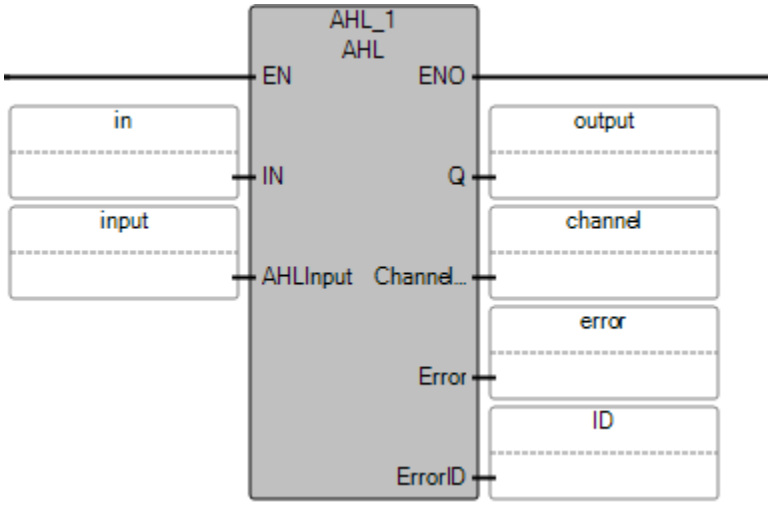
Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions AHL



Exemple de schéma à relais AHL



Exemple de texte structuré AHL

```
AHL_1(
```

void **AHL_1**(BOOL **IN**, AHLI **AHLInput**)
Type: AHL, Établi ou rétabli l'établissement de liaison pour le modem.

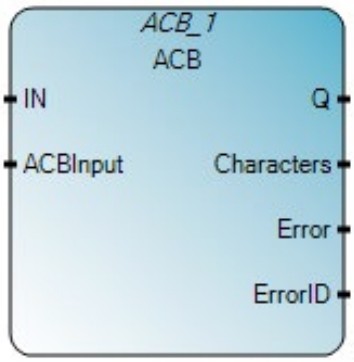
```
1  AHL_1(in, input);
2  output := AHL_1.Q;
3  channel := AHL_1.ChannelSts;
4  error := AHL_1.Error;
5  ID := AHL_1.ErrorID;
```

ACB (caractères ASCII dans la mémoire tampon)

Compte le nombre total de caractères ASCII dans la mémoire tampon, y compris la fin de ligne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

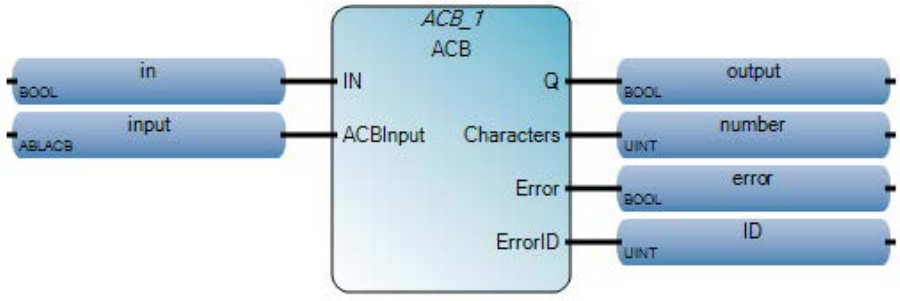
Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.
ACBInput	Entrée	ABLACB sur la page 120	Canal à exploiter. Utilisez le type de données ABLACB pour définir les paramètres Canal, Type de déclenchement et Abandon pour ACBInput.
Q	Sortie	BOOL	Indique si le compte de caractères est en cours ou terminé. Les sorties de ce bloc de fonctions sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le comptage est terminé. FAUX : le comptage est en cours.
Characters	Sortie	UINT	Nombre de caractères dans la mémoire tampon.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. FAUX : aucune erreur. VRAI : erreur détectée.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

Codes d'erreur ACB

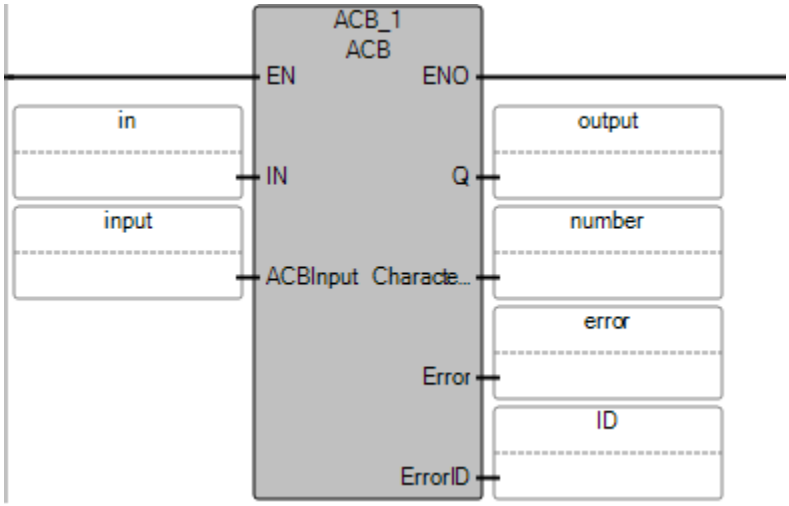
Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions ACB



Exemple de schéma à relais ACB



Exemple de texte structuré ACB

```
ACB_1(
```

void **ACB_1**(BOOL **IN**, ABLACB **ACBInput**)
Type: ACB, Définie le nombre total de caractères pour le tampon.

```
1  ACB_1(in, input);
2  output := ACB_1.Q;
3  number := ACB_1.Characters;
4  error := ACB_1.Error;
5  ID := ACB_1.ErrorID;
```

ARD (lecture ASCII)

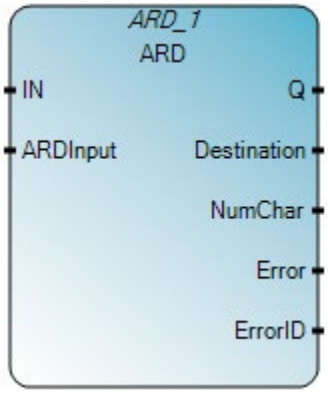
Lit les caractères de la mémoire tampon d'entrée et les stocke dans une chaîne.

Détails de l'opération :

- L'instruction ARD s'exécute jusqu'à ce que tous les caractères dans la mémoire tampon ASCII soient reçus. Si une autre instruction ASCII sur la page 103 est exécutée, celle-ci est mise en attente jusqu'à ce que l'exécution de l'instruction ARD se termine.
- Pour annuler l'instruction ARD, exécutez une instruction ACL.
- Pour éviter que l'instruction ARD ne retarde la file d'attente ASCII pendant qu'elle attend le nombre de caractères requis, utilisez les résultats de l'instruction ACB pour déclencher l'instruction ARD.
- L'état de l'instruction peut être extrait à partir du bit de contrôle de l'instance d'instruction (par exemple, ARD_1.controlbit). Cela indique si l'instruction est en train de bloquer la file d'attente d'instructions ASCII en attendant davantage de caractères :
  - 7e bit = L'instruction est activée.
  - 6e bit = L'instruction est en file d'attente.
  - 5e bit = L'instruction est terminée.
  - 3e bit = L'instruction comporte une erreur.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.
ARDInput	Entrée	ARDARL sur la page 122	Lit les caractères dans la mémoire tampon. Le maximum est de 82 caractères. Utilisez le type de données ARDARL pour définir les paramètres Canal, Longueur et Abandon pour ARDInput.
Q	Sortie	BOOL	Indique si la lecture de la mémoire tampon est en cours ou terminée. Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
Destination	Sortie	ASCIILOCADDR	L'élément de chaîne dans lequel les caractères sont stockés.
NumChar	Sortie	UINT	Nombre de caractères.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

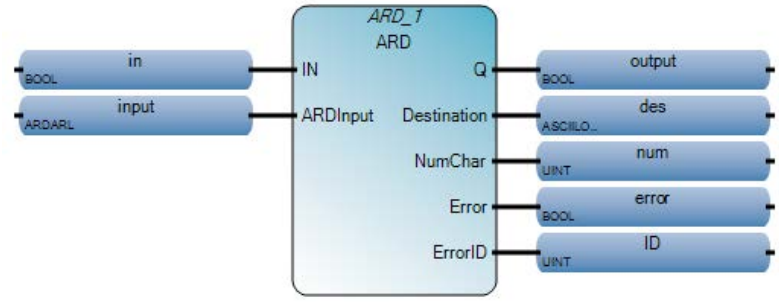
Codes d'erreur ABL

Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

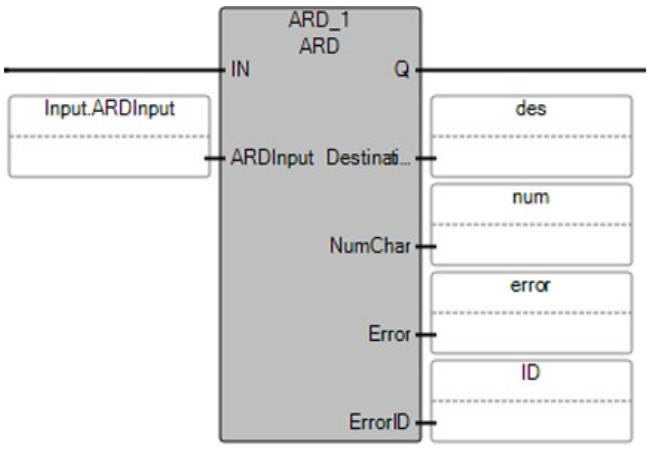
Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.

Code d'erreur	Description de l'erreur
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions ARD



Exemple de schéma à relais ARD



Exemple de texte structuré ARD

```
ARD_1(
```

void **ARD_1**(BOOL **IN**, ARDARL **ARDInput**)
Type: ARD, Lecture des caractères du tampon d'entrée à placer dans une chaîne.

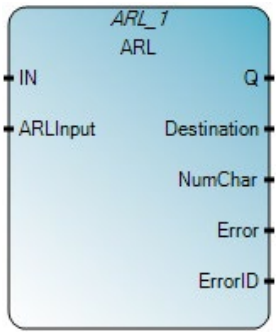
```
1  ARD_1(in, input);
2  output := ARD_1.Q;
3  des := ARD_1.Destination;
4  num := ARD_1.NumChar;
5  error := ARD_1.Error;
6  ID := ARD_1.ErrorID;
```

ARL (ligne de lecture ASCII)

Lit une ligne de caractères ASCII sur la page 103 dans la mémoire tampon, jusqu'aux caractères de fin inclus, et les stocke dans une chaîne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.
ARLInput	Entrée	ARDARL sur la page 122	Lit une ligne de caractères ASCII dans la mémoire tampon. Le maximum est de 82 caractères. Utilisez le type de données ARDARL pour définir les paramètres Canal, Longueur et Abandon pour ARDLInput.
Q	Sortie	BOOL	Indique si la lecture de la ligne dans la mémoire tampon est en cours ou terminée. Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
Destination	Sortie	ASCIILOCADDR	L'élément de chaîne dans lequel les caractères sont stockés.
NumChar	Sortie	UINT	Le nombre de caractères dans la ligne, y compris les caractères de fin.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

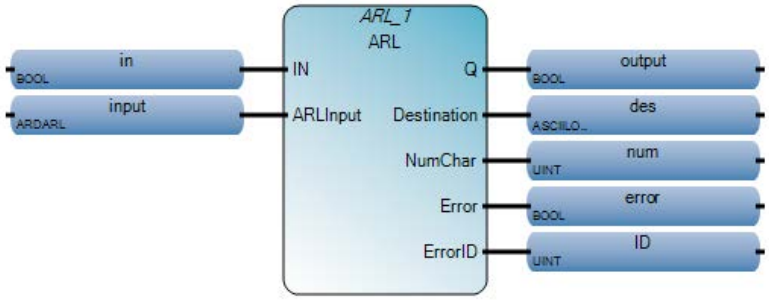
Codes d'erreur ABL

Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

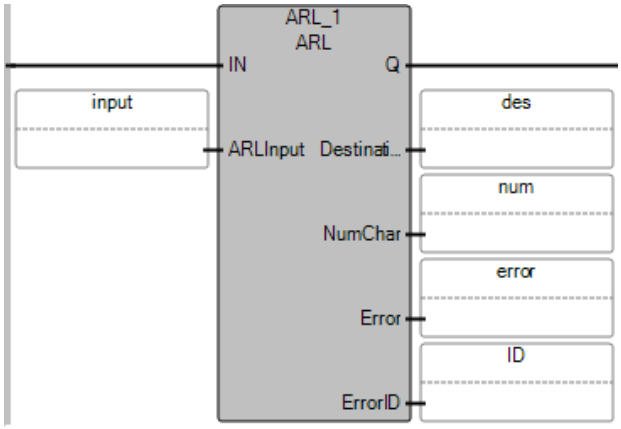
Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.

Code d'erreur	Description de l'erreur
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions ARL



Exemple de langage à relais ARL



Exemple de texte structuré ARL

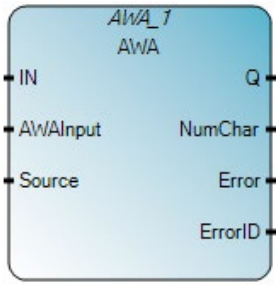
```
1  ARL_1(in, input);
2  output := ARL_1.Q;
3  des := ARL_1.Destination;
4  num := ARL_1.NumChar;
5  error := ARL_1.Error;
6  ID := ARL_1.ErrorID;
```

AWA (ajout en écriture ASCII)

Écrit une chaîne avec deux caractères (configurés par l'utilisateur) ajoutés à un appareil externe.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.
AWAInput	Entrée	AWAAWT	Le canal et le nombre (la longueur) des caractères à écrire dans la mémoire tampon. Le maximum est 82. Utilisez le type de données AWAAWT sur la page 122 pour définir les paramètres Canal, Longueur et Abandon pour AWAInput.
Source	Entrée	ASCIILOCADDR	La chaîne source sortie en tant que tableau de caractères par instruction ARD ou ARL.
Q	Sortie	BOOL	Indique si l'écriture est en cours ou terminée. Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
NumChar	Sortie	UINT	Nombre de caractères. Il est possible que le paramètre NumChar soit inférieur à la longueur à transférer si la chaîne source est plus courte que la longueur demandée. Met à jour les informations lorsque la transmission est terminée et que la valeur Q est VRAI.

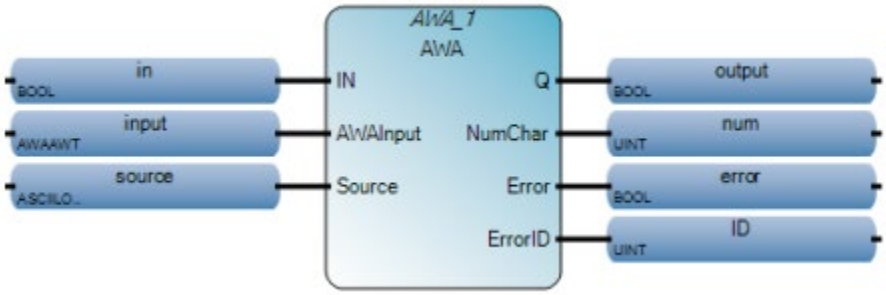
Paramètre	Type de paramètre	Type de données	Description
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

Codes d'erreur ABL

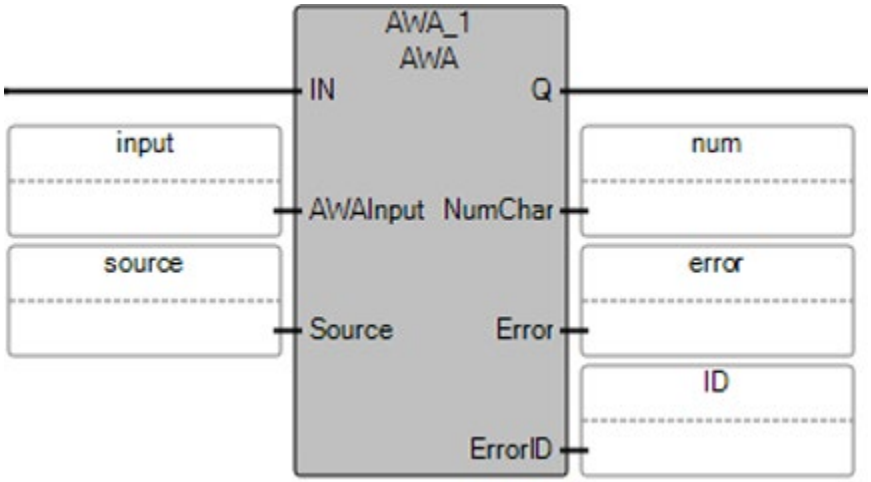
Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions AWA



Exemple de schéma à relais AWA



Exemple de texte structuré AWA

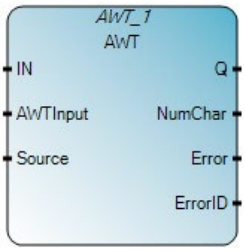
```
AWA_1(
    void AWA_1(BOOL IN, AWAAWT AWAInput, ASCIILOCADDR Source)
    Type: AWA, Écriture d'une chaîne avec caractères ajoutés jusqu'à un équipement externe.

1  AWA_1(in, input, source);
2  output := AWA_1.Q;
3  num := AWA_1.NumChar;
4  error := AWA_1.Error;
5  ID := AWA_1.ErrorID;
```

AWT (écriture ASCII)

Écrit les caractères ASCII provenant d'une chaîne source vers un appareil externe.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : le bloc d'instruction est en attente.

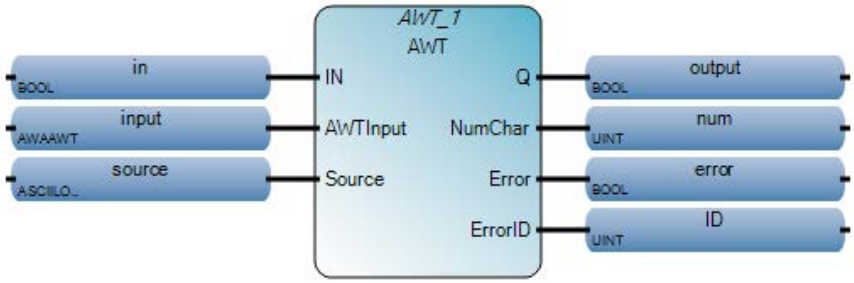
Paramètre	Type de paramètre	Type de données	Description
AWTInput	Entrée	AWAAWT	Le canal et le nombre (la longueur) des caractères à écrire dans la mémoire tampon. Le maximum est 82. Utilisez le type de données AWAAWT sur la page 122 pour définir les paramètres Canal, Longueur et Abandon pour AWTInput.
Source	Entrée	ASCIILOCADDR	La chaîne source sortie en tant que tableau de caractères par instruction ARD ou ARL.
Q	Sortie	BOOL	Indique si l'écriture est en cours ou terminée. Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : le bloc de fonctions est terminé. FAUX : le bloc de fonctions n'est pas terminé.
NumChar	Sortie	UINT	Nombre de caractères. Il est possible que le paramètre NumChar soit inférieur à la longueur à transférer si la chaîne source est plus courte que la longueur demandée. Met à jour les informations lorsque la transmission est terminée et que la valeur Q est VRAI.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur ABL.

Codes d'erreur ABL

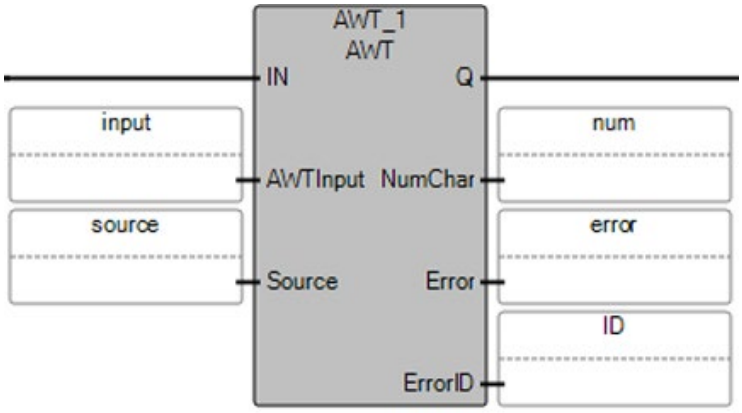
Utilisez ce tableau pour déterminer les codes d'erreur ABL et les descriptions.

Code d'erreur	Description de l'erreur
03	La transmission ne peut pas être terminée car le signal CTS a été perdu.
06	Paramètre illégal détecté.
07	Impossible de terminer l'envoi ou la réception ASCII, car le canal de configuration a été arrêté à l'aide de la boîte de dialogue de configuration des canaux.
08	Impossible de terminer l'écriture ASCII en raison d'une transmission ASCII en cours.
09	La requête de communication ASCII n'est pas prise en charge par la configuration des canaux actuelle.
10	L'annulation a été définie, ce qui a provoqué l'arrêt de l'exécution de l'instruction. Aucune action n'est nécessaire.
11	La longueur requise pour la chaîne est soit non valide, soit négative ou supérieure à 82, soit égale à 0. S'applique aux blocs de fonctions ARD et ARL.
13	La valeur (.LEN) requise dans le bloc de contrôle est un nombre négatif ou supérieur à la taille de la chaîne stockée avec la chaîne source. S'applique aux blocs de fonctions AWA et AWT.
14	Le bloc de fonctions ACL a été annulé.
16	Le port série ne prend pas en charge les lignes de contrôle RTS ou CTS.

Exemple de diagramme de bloc de fonctions AWT



Exemple de schéma à relais AWT



Exemple de texte structuré AWT

```
AWT_1(
```

void **AWT_1**(BOOL **IN**, AWAAWT **AWTInput**, ASCIILOCADDR **Source**)
Type: AWT, Écriture des caractères provenant d'une chaîne source vers un équipement externe.

```
1  AWT_1(in, input, source);
2  output := AWT_1.Q;
3  num := AWT_1.NumChar;
4  error := AWT_1.Error;
5  ID := AWT_1.ErrorID;
```

Détail des paramètres ASCII

Vous trouverez dans les rubriques suivantes des informations complémentaires sur les paramètres ASCII et les types de données structurés.

- Type de données ABLACB
- Type de données ACL
- Type de données AHL ChannelSts
- Type de données AHLI
- Type de données ARDARL
- Type de données AWAAWT

Type de données ABLACB

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données ABLACB.

Paramètre	Type de données	Description
Channel	UINT	Numéro du port série : • 2 pour le port série intégré ou • 5 à 9 pour les ports série des modules enfichables installés sur les emplacements 1 à 5 : • 5 pour l'emplacement 1 ; • 6 pour l'emplacement 2 ; • 7 pour l'emplacement 3 ; • 8 pour l'emplacement 4 ; • 9 pour l'emplacement 5 ;
TriggerType	USINT	Représente l'une des valeurs suivantes : • 0: envoi de message déclenché une fois (lorsque IN passe de Faux à Vrai) • 1: envoi de message déclenché de façon continue lorsque IN est défini sur Vrai • Autre valeur : Réservé

Paramètre	Type de données	Description
Cancel	BOOL	Lorsque cette entrée est définie sur VRAI, ce bloc fonctionnel n'est pas exécuté.

Type de données ACL

Cette table permet de déterminer les valeurs de paramètre pour le type de données ABL.

Paramètre	Type de données	Description
Channel	UINT	Numéro du port série : • 2 pour le port série intégré ou • 5 à 9 pour les ports série des modules enfichables installés sur les emplacements 1 à 5 : • 5 pour l'emplacement 1 ; • 6 pour l'emplacement 2 ; • 7 pour l'emplacement 3 ; • 8 pour l'emplacement 4 ; • 9 pour l'emplacement 5 ;
RXBuffer	BOOL	Vide la mémoire tampon de réception lorsque ce paramètre est défini sur TRUE et supprime les blocs de fonctions ASCII de réception (ARL et ARD) de la file d'attente ASCII.
TXBuffer	BOOL	Vide la mémoire tampon de transmission lorsque ce paramètre est défini sur TRUE et supprime les blocs de fonctions ASCII de transmission (AWA et AWT) de la file d'attente ASCII.

Type de données AHL ChannelSts

Cette table permet de déterminer les valeurs de paramètre pour le type de données AHL ChannelSts.

Paramètre	Type de données	Description
DTRstatus	UINT	Utilisé pour le signal DTR (réservé)
DCDstatus	UINT	Utilisé pour le signal DCD (bit 3 du mot) 1 indique un état actif
DSRstatus	UINT	Utilisé pour le signal DSR (réservé)
RTSstatus	UINT	Utilisé pour le signal RTS (bit 1 du mot) 1 indique un état actif
CTSstatus	UINT	Utilisé pour le signal CTS (bit 0 du mot) 1 indique un état actif

Type de données AHLI

Cette table permet de déterminer les valeurs de paramètre pour le type de données AHL.

Paramètre	Type de données	Description
Channel	UINT	Numéro du port série : • 2 pour le port série intégré ou • 5 à 9 pour les ports série des modules enfichables installés sur les emplacements 1 à 5 : • 5 pour l'emplacement 1 ; • 6 pour l'emplacement 2 ; • 7 pour l'emplacement 3 ; • 8 pour l'emplacement 4 ; • 9 pour l'emplacement 5 ;
ClrRts	BOOL	Utilisé pour réinitialiser la ligne de contrôle RTS.

Paramètre	Type de données	Description
SetRts	BOOL	Utilisé pour définir la ligne de contrôle RTS.
Cancel	BOOL	Lorsque cette entrée est définie sur VRAI, ce bloc fonctionnel n'est pas exécuté.

Type de données ARDARL

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données ARDARL.

Paramètre	Type de données	Description
Channel	UINT	Numéro du port série : • 2 pour le port série intégré ou • 5 à 9 pour les ports série des modules enfichables installés sur les emplacements 1 à 5 : • 5 pour l'emplacement 1 ; • 6 pour l'emplacement 2 ; • 7 pour l'emplacement 3 ; • 8 pour l'emplacement 4 ; • 9 pour l'emplacement 5 ;
Length	UINT	Nombre de caractères que vous voulez lire à partir de la mémoire tampon (82 au maximum).
Cancel	BOOL	Lorsque cette entrée est définie sur VRAI, ce bloc fonctionnel n'est pas exécuté. Si l'exécution est déjà en cours, l'opération s'arrête.

Type de données AWAAWT

Utilisez ce tableau pour déterminer les valeurs de paramètre pour le type de données AWAAWT.

Paramètre	Type de données	Description
Channel	UINT	Numéro du port série : • 2 pour le port série intégré ou • 5 à 9 pour les ports série des modules enfichables installés sur les emplacements 1 à 5 : • 5 pour l'emplacement 1 ; • 6 pour l'emplacement 2 ; • 7 pour l'emplacement 3 ; • 8 pour l'emplacement 4 ; • 9 pour l'emplacement 5 ;
Length	UINT	Définit le nombre de caractères à écrire dans la mémoire tampon (82 au maximum). Lorsque la longueur est définie sur 0, AWA envoie 0 octet de données utilisateur et 2 octets de caractères ajoutés à la mémoire tampon.
Cancel	BOOL	Lorsque ce paramètre est « VRAI », ce bloc de fonctions n'est pas exécuté. Si l'exécution est déjà en cours, l'opération s'arrête.

Instructions binaires

Utilisez les instructions binaires pour effectuer des opérations mathématiques.

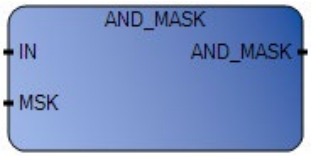
Opérateur	Description
AND_MASK sur la page 123	Effectue un AND bit à bit entre deux valeurs entières.
NOT_MASK sur la page 131	Masque de négation bit à bit entier, inverse une valeur de paramètre.
BSL sur la page 125	Déplace un bit dans un élément du tableau vers la gauche.
BSR sur la page 128	Déplace un bit dans un élément du tableau vers la droite.
OR_MASK sur la page 132	Masque avec opérateur OR bit à bit entier, active les bits.
ROL sur la page 134	Pour des entiers 32 bits, fait pivoter les bits d'entiers vers la gauche.
ROR sur la page 135	Pour des entiers 32 bits, fait pivoter les bits d'entiers vers la gauche.
SHL sur la page 137	Pour des entiers 32 bits, déplace les entiers vers la gauche et place 0 dans le bit de poids faible.
SHR sur la page 139	Pour des entiers 32 bits, déplace les entiers vers la droite et place 0 dans le bit de poids fort.
XOR_MASK sur la page 140	Masque avec opérateur OR exclusif bit à bit entier, renvoie les valeurs de bits inversées.

AND_MASK (masque avec opérateur AND)

Effectue un AND bit à bit entre deux valeurs entières.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

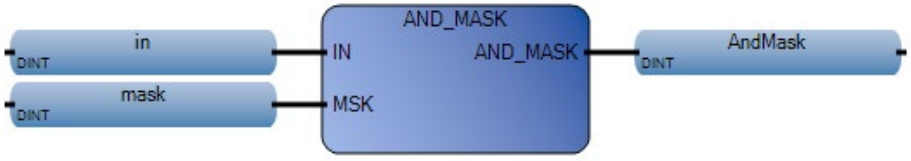
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



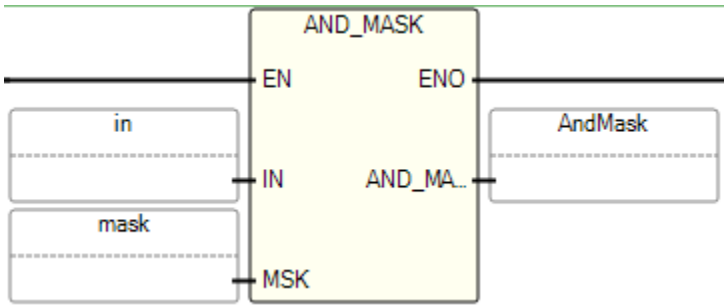
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du masque bit à bit avec opérateur AND entier. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Doit être au format entier.
MSK	Entrée	DINT	Doit être au format entier.
AND_MASK	Sortie	DINT	Opérateur AND logique bit à bit entre IN et MSK.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions AND_MASK



Exemple de logique à relais AND_MASK



Exemple de texte structuré AND_MASK

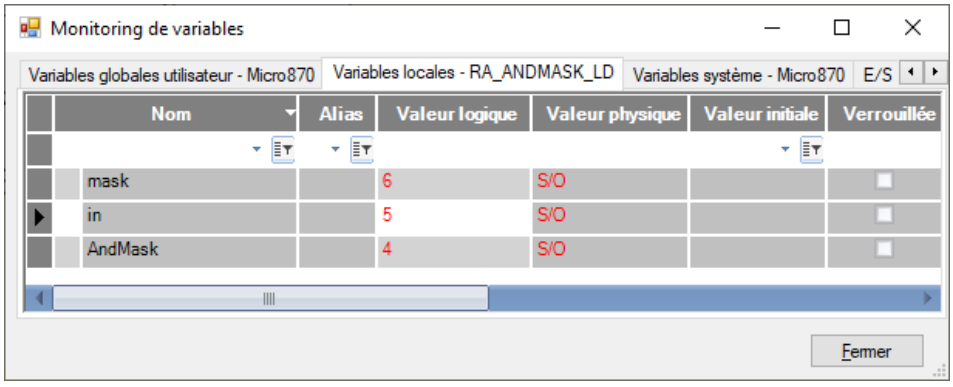
```
1  in := 5;
2  mask := 6;
3  AndMask := AND_MASK(in, mask);
```

(* Équivalence ST : *)

parité := AND_MASK (xvalue, 1) ; (* 1 si xvalue est impaire *)

résultat := AND_MASK (16#abc, 16#fof) ; (* est égal à 16#aoc *)

Résultats



BSL (déplacement à gauche de bits)

Déplace un bit dans un élément du tableau vers la gauche.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Détails de l'opération :

L'instruction BSL est un processus immédiat sur la transition d'une ligne de FAUX à VRAI et met à jour la sortie de manière synchrone. Lorsque Execute est défini sur VRAI, le bit le plus à gauche (Src + SrcOffset et Length) est copié dans le bit Unload et tous les bits du tableau (ou non) sont déplacés d'un bit vers la gauche. La longueur et la limite de 16 bits sont prises en compte, sauf pour les types de données BOOL. Le bit externe est ensuite déplacé vers le bit 0 (Src + SrcOffset) du premier élément.

Pour les opérations de renvoi à la ligne, définissez la position de BitAddr sur le dernier bit ou sur le bit Unload. Une utilisation possible de l'instruction BSL est le suivi de bouteilles dans une ligne de mise en bouteille où chaque bit représente une bouteille.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

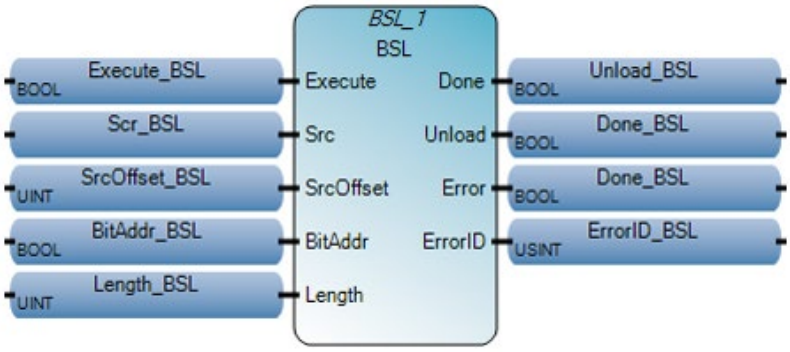
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation de l'instruction. VRAI : front montant détecté, déplace un bit vers la gauche d'une position. • Vérifiez d'abord les conditions de défaut. • Si Length = 0, le bit externe est déplacé dans le bit Unload. Aucun changement de bit n'est effectué dans Scr. Les bits Error et ErrorID sont réinitialisés. Le bit Done est défini. • Si Length > 0 et Length ≤ 2048, les bits Error et ErrorID sont réinitialisés. Lorsque le changement de bit est terminé, le bit Done est défini. • Si Length > 0 et Length ≤ 2048, le bit le plus à gauche (adressés par Src + SrcOffset et Length) est copié dans le bit Unload et tous les bits faisant partie du tableau (ou non) sont déplacés d'un bit vers la gauche (jusqu'au bit Length et à la limite de 16 bits, sauf pour BOOL). Le bit externe est déplacé vers le bit 0 (Src + SrcOffset) du premier élément. FAUX : front montant non détecté, ne pas activer l'opération BSL.

Paramètre	Type de paramètre	Type de données	Description
Scr	Entrée	ANY_ELEMENTARY	L'adresse de Src (bit) doit être changée. Types de données pris en charge : BOOL, DWORD, INT, UINT, WORD, DINT et UDINT. • Tableaux : définissez Scr sur une adresse basée sur une variable, par exemple : Source1, Source1[0] ou Source1[1]. • Éléments non-tableaux : définissez Scr sur une adresse basée sur une variable, par exemple Source1.
SrcOffset	Entrée	UINT	Si SrcOffset est défini sur 0, commence par le premier élément. • Tableaux : définissez SrcOffset sur 0. Si la valeur est définie sur Source1[0] ou Source1[1], une erreur se produit : « Source offset exceeds the size of the array. » (La sortie de la source dépasse la taille du tableau.) • Éléments non-tableaux : définissez SrcOffset sur 0 ou une erreur survient : « Source offset exceeds the size of the array. » (La sortie de la source dépasse la taille du tableau.)
BitAddr	Entrée	BOOL	Emplacement du bit décalé dans Src.
Length	Entrée	UINT	Length contient le nombre de bits dans le Src à déplacer. Prend en charge le déplacement d'éléments du tableau. • Pour le type de données BOOL, le nombre de valeurs booléennes dans le tableau doit être modifié. • Pour les types de données 16 et 32 bits, les bits sont déplacés dans des multiples de 16 (par exemple, 16, 32 et 64). Si Length n'est pas un multiple pair de 16, les bits déplacés sont envoyés à la limite de 16 bits suivante. • Length est basé sur la taille du type de données. Si Length dépasse la plage, une erreur se produit : « Source offset exceeds the size of the array. » (La sortie de la source dépasse la taille du tableau.) Valeurs de Length : • BOOL : 1 • Mot de 16 bits : 1-16 • Mot de 32 bits : 1-32 • Mot de 64 bits : 1-64
Done	Sortie	BOOL	Lorsque la valeur est « TRUE », l'opération s'est déroulée correctement. Lorsqu'elle est « FALSE », l'opération a rencontré une erreur.
Unload	Sortie	BOOL	Bit décalé hors de l'adresse de Src.
Error	Sortie	BOOL	Lorsqu'un défaut se produit, Error est défini sur VRAI.
ErrorID	Sortie	USINT	Lorsqu'un défaut se produit, ErrorID contient le code d'erreur.

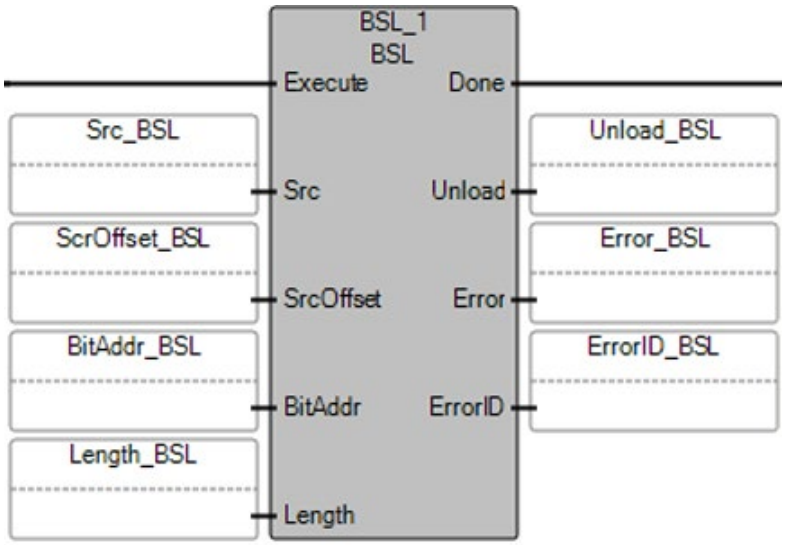
Codes d'erreur BSL

Code d'erreur	Description de l'erreur
01	Dimension non prise en charge.
02	Type de donnée non pris en charge.
03	La longueur des bits dépasse 2 048.
04	Le décalage source dépasse la taille du tableau.
05	La longueur des bits dépasse la taille du tableau.
07	Paramètres non valides.

Exemple de diagramme de bloc de fonctions BSL



Exemple de schéma à relais BSL



Exemple de texte structuré BSL

```
BSL_1(
```

void **BSL_1**(BOOL **Execute**, ANY_ELEMENTARY[1..1] **Src**, UINT **SrcOffset**, BOOL **BitAddr**, UINT **Length**)
Type: BSL, Effectue une opération de déplacement de bits vers la gauche.

```
1  BSL_1(Execute,Src,SrcOffset,BitAddr,Length);
2  Done_BSL := BSL_1.Done;
3  Unload_BSL := BSL_1.Unload;
4  Error_BSL := BSL_1.Error;
5  ErrorID_BSL := BSL_1.ErrorID;
```

Résultats

	Nom	Alias	Valeur logique	Valeur physique	Valeur initiale	Verrouillée	Type de données	Dimension
▶ +	BSL_1		...	...	...	☐	BSL	
	Execute		☐	S/O		☐	BOOL	
+	Src		...	...	...	☐	BOOL	[1..4]
	SrcOffset		0	S/O		☐	UINT	
	BitAddr		☐	S/O		☐	BOOL	
	Length		0	S/O		☐	UINT	
	Done_BSL		☐	S/O		☐	BOOL	
	Upload_BSL		☐	S/O		☐	BOOL	
	Error_BSL		☐	S/O		☐	BOOL	
	ErrorID_BSL		0	S/O		☐	USINT	

BSR (déplacement à droite à droite de bits)

Déplace un bit dans un élément du tableau vers la droite.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Détails de l'opération :

L'instruction BSR est un processus immédiat sur la transition d'une ligne de FAUX à VRAI et met à jour la sortie de manière synchrone. Lorsque Execute est défini sur VRAI, le bit le plus à droite (bit 0 de l'élément traité par Src + SrcOffset) est copié dans le bit Unload, et tous les bits du tableau (ou non) sont déplacés d'un bit vers la droite. La longueur et la limite de 16 bits sont prises en compte, sauf pour les types de données BOOL. Le bit externe est ensuite déplacé vers le bit 0 (Src + SrcOffset) du premier élément.

Pour les opérations de renvoi à la ligne, définissez la position de BitAddr sur le dernier bit ou sur le bit Unload. Une utilisation possible de l'instruction BSL est le suivi de bouteilles dans une ligne de mise en bouteille où chaque bit représente une bouteille.

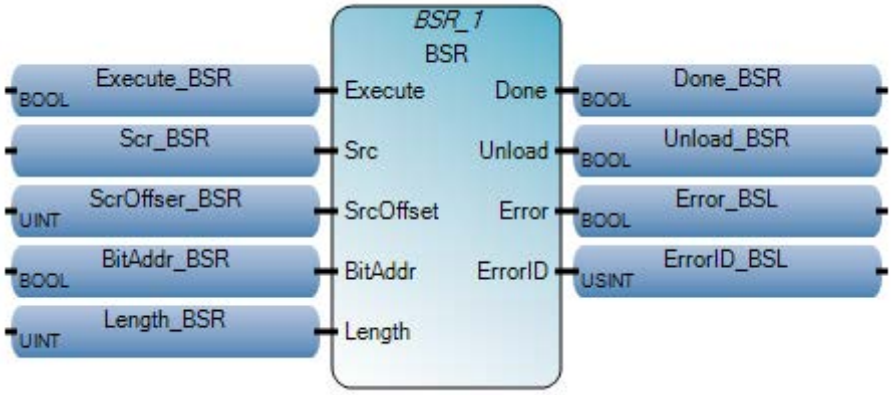
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation de l'instruction. VRAI : front montant détecté, déplace un bit vers la droite d'une position. FAUX : front montant non détecté, ne pas activer l'opération BSR.
Scr	Entrée	ANY_ELEMENTARY	L'adresse de Src (bit) doit être changée. Types de données pris en charge : BOOL, DWORD, INT, UINT, WORD, DINT et UDINT. • Tableaux : définissez Scr sur une adresse basée sur une variable, par exemple : Source1, Source1[0] ou Source1[1]. • Éléments non-tableaux : définissez Scr sur une adresse basée sur une variable, par exemple Source1.
SrcOffset	Entrée	UINT	Si SrcOffset est défini sur 0, commence par le premier élément. • Tableaux : définissez SrcOffset sur 0. Si la valeur est définie sur Source1[0] ou Source1[1], une erreur se produit : « Source offset exceeds the size of the array. » (La sortie de la source dépasse la taille du tableau.) • Éléments non-tableaux : définissez SrcOffset sur 0 ou une erreur survient : « Source offset exceeds the size of the array. » (La sortie de la source dépasse la taille du tableau.)
BitAddr	Entrée	BOOL	Emplacement du bit décalé dans Src.
Length	Entrée	UINT	Length contient le nombre de bits dans le Src à déplacer. Prend en charge le déplacement d'éléments du tableau. • Pour le type de données BOOL, le nombre de valeurs booléennes dans le tableau doit être modifié. • Pour les types de données 16 et 32 bits, les bits sont déplacés dans des multiples de 16 (par exemple, 16, 32 et 64). Si Length n'est pas un multiple pair de 16, les bits déplacés sont envoyés à la limite de 16 bits suivante. • Length est basé sur la taille du type de données. Si Length dépasse la plage, une erreur se produit : « Source offset exceeds the size of the array. » (La sortie de la source dépasse la taille du tableau.) Valeurs de Length : • BOOL : 1 • Mot de 16 bits : 1-16 • Mot de 32 bits : 1-32 • Mot de 64 bits : 1-64
Done	Sortie	BOOL	Lorsque la valeur est « TRUE », l'opération s'est déroulée correctement. Lorsqu'elle est « FALSE », l'opération a rencontré une erreur.
Unload	Sortie	BOOL	Bit décalé hors de l'adresse de Src.
Error	Sortie	BOOL	Lorsqu'un défaut se produit, Error est défini sur VRAI.
ErrorID	Sortie	USINT	Lorsqu'un défaut se produit, ErrorID contient le code d'erreur.

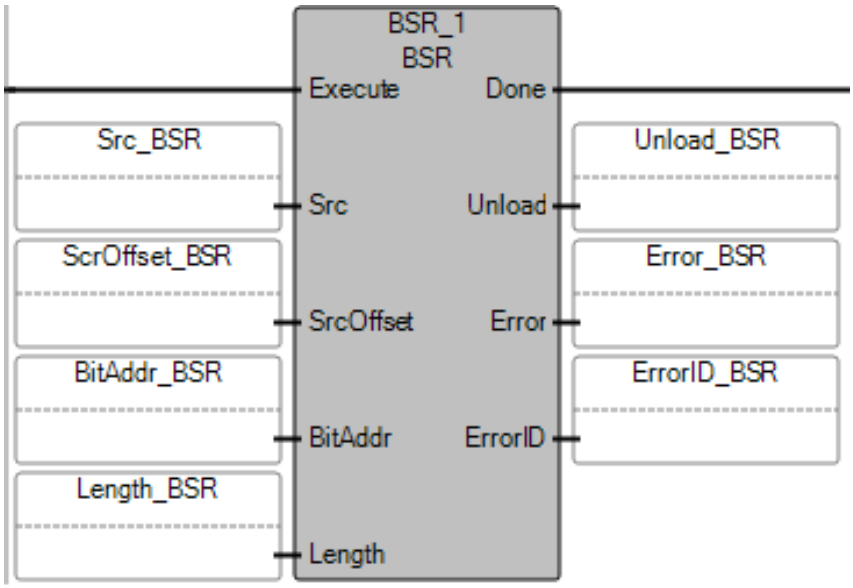
Codes d'erreur BSR

Code d'erreur	Description de l'erreur
01	Dimension non prise en charge.
02	Type de donnée non pris en charge.
03	La longueur des bits dépasse 2 048.
04	Le décalage source dépasse la taille du tableau.
05	La longueur des bits dépasse la taille du tableau.
07	Paramètres non valides.

Exemple de diagramme de bloc de fonctions BSR



Exemple de schéma à relais BSR



Exemple de texte structuré BSR

```
BSR_1(
```

void **BSR_1**(BOOL **Execute**, ANY_ELEMENTARY[1..1] **Src**, UINT **SrcOffset**, BOOL **BitAddr**, UINT **Length**)
Type: BSR, Effectue une opération de déplacement de bits vers la droite.

```
1  BSR_1(Execute,Scr,SrcOffset,BitAddr,Length);
2  Done_BSR := BSR_1.Done;
3  Unload_BSR := BSR_1.Unload;
4  Error_BSR := BSR_1.Error;
5  ErrorID_BSR := BSR_1.ErrorID;
```

Résultats

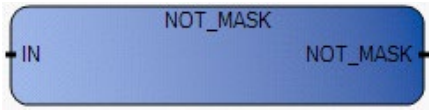
Nom	Alias	Valeur logique	Valeur physique	Verrouillée	Type de données	Dimension
BSR_1		...	...		BSR	
Execute			S/O		BOOL	
Scr		...	...		DINT	[1..4]
SrcOffset		0	S/O		UINT	
BitAddr			S/O		BOOL	
Length		0	S/O		UINT	
Done_BSR			S/O		BOOL	
Unload_BSR			S/O		BOOL	
Error_BSR			S/O		BOOL	
ErrorID_BSR		0	S/O		USINT	

NOT_MASK (masque NOT de bit à bit)

Masque de négation bit à bit entier, inverse une valeur de paramètre.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

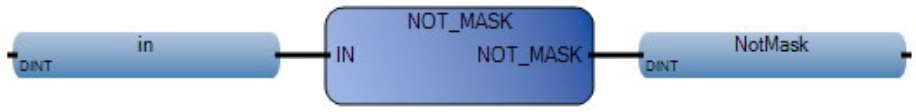
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



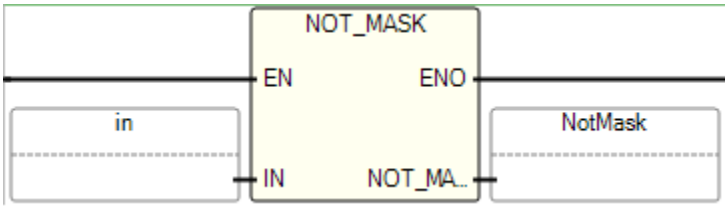
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du masque de négation bit à bit. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Doit être au format entier.
NOT_MASK	Sortie	DINT	Négation bit à bit sur 32 bits de la chaîne IN.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction NOT_MASK



Exemple de logique à relais NOT_MASK



Exemple de texte structuré NOT_MASK

```
NOT_MASK(
```

DINT **NOT_MASK**(DINT IN)
Négation bit à bit

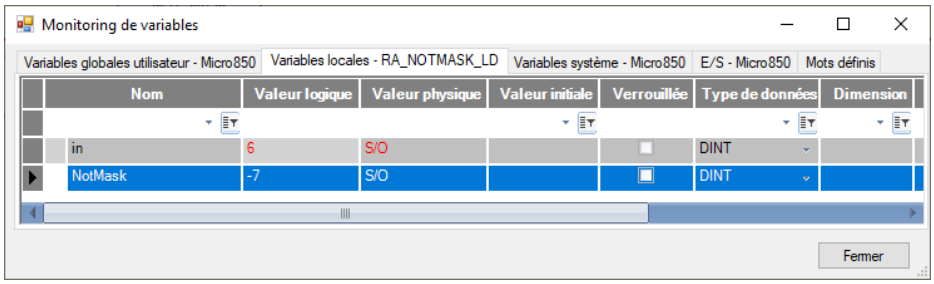
```
1  in := 6;
2  NotMask := NOT_MASK(in);
```

(* Équivalence ST : *)

résultat := NOT_MASK (16#1234) ;

(* le résultat est égal à 16#FFFF_EDCB *)

Résultats



OR_MASK (masque OR bit à bit)

Masque avec opérateur OR bit à bit entier, active les bits.

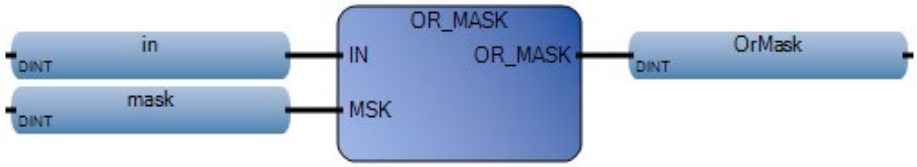
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

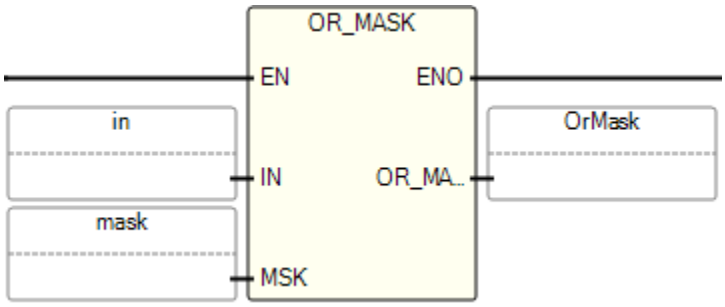
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du masque bit à bit avec opérateur OR entier. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Doit être au format entier.
MSK	Entrée	DINT	Doit être au format entier.
OR_MASK	Sortie	DINT	Opérateur OR logique bit à bit entre IN et MSK.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction OR_MASK



Exemple de logique à relais OR_MASK



Exemple de texte structuré OR_MASK

```
OR_MASK(
```

DINT **OR_MASK**(DINT IN, DINT MSK)
Analogue bit à bit OR mask

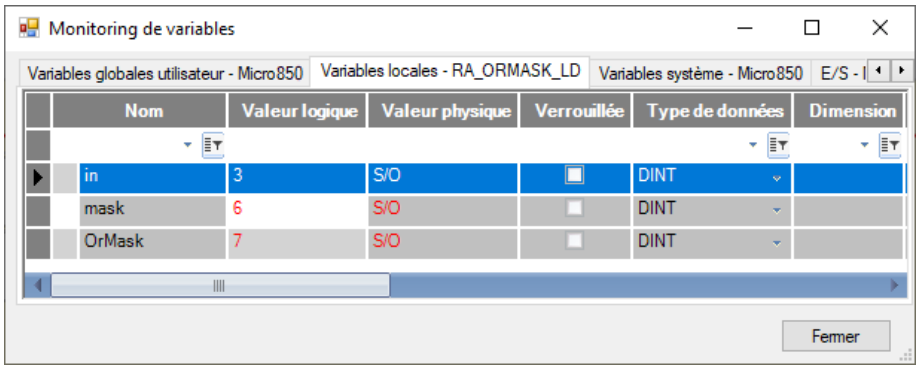
```
1  in := 3;
2  mask := 6;
3  OrMask := OR_MASK(in, mask);
```

(* Équivalence ST : *)

parité := OR_MASK (xvalue, 1) ; (* convertit toujours les valeurs en valeurs impaires *)

résultat := OR_MASK (16#abc, 16#fof) ; (* est égal à 16#fbf *)

Résultats

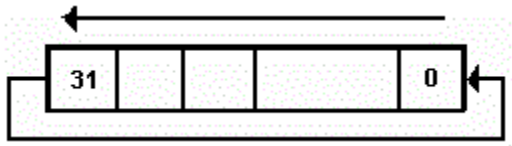


ROL (pivoter à gauche)

Fait pivoter l'entrée de type DINT par des bits NbR vers la gauche sous forme circulaire et remplit les bits à droite avec les bits qui ont pivotés.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

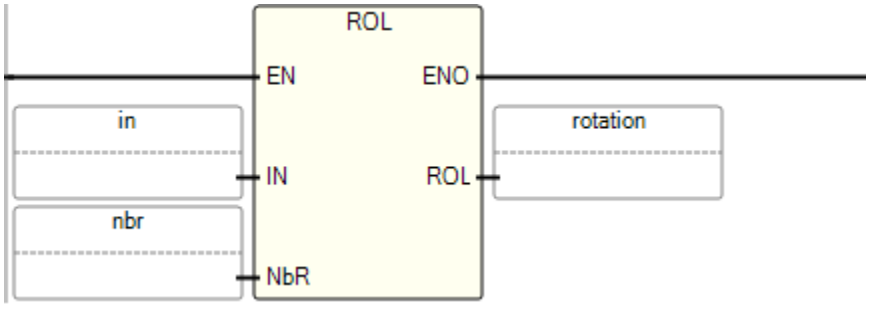


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la valeur entière pour la rotation des bits vers la gauche. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Valeur entière.
NbR	Entrée	DINT	Nombre de rotations 1 bit (dans l'ensemble [1..31]).
ROL	Sortie	DINT	Valeur de rotation vers la gauche. Lorsque NbR <= 0, aucune modification n'est apportée.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ROL

Exemple de schéma à relais ROL



Exemple de texte structuré ROL

```
1  in := 123;
2  nbr := 2;
3  rotation := ROL(in, nbr);
```

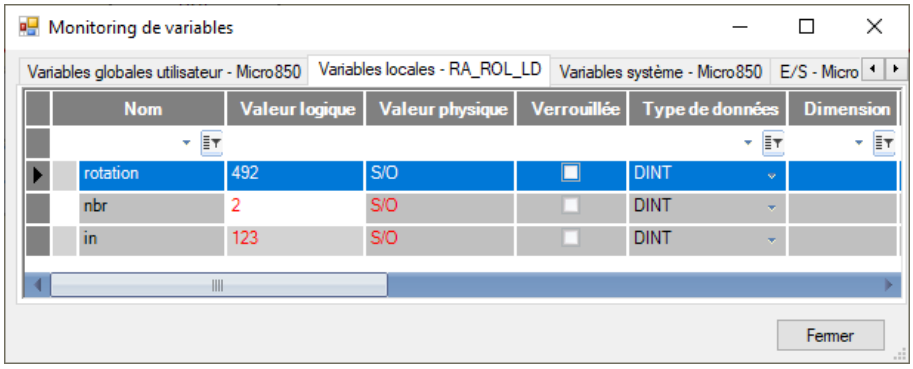
(* Équivalence ST : *)

résultat := ROL (registre, 1) ;

(* registre = 2#0100_1101_0011_0101*)

(* résultat = 2#1001_1010_0110_1010*)

Résultats

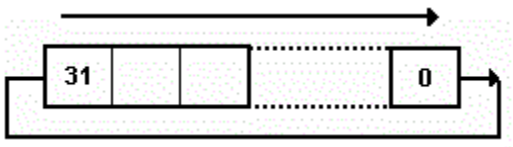


ROR (pivoter à droite)

Fait pivoter l'entrée de type DINT par des bits NbR vers la droite sous forme circulaire et remplit les bits de gauche avec les bits qui ont pivotés.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

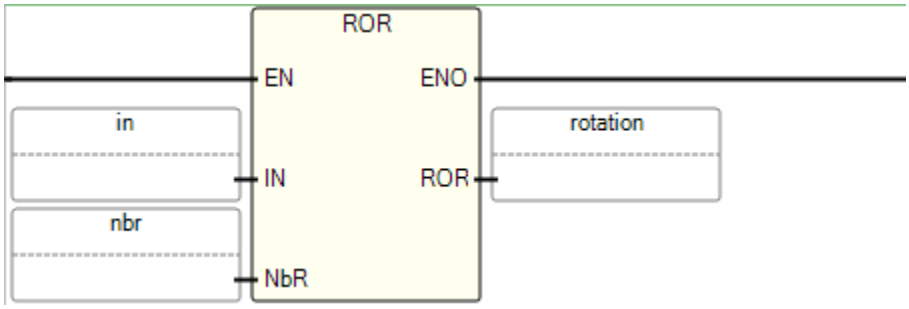


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la valeur entière pour la rotation des bits vers la droite. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Valeur entière, quelle qu'elle soit.
NbR	Entrée	DINT	Nombre de rotations 1 bit (dans l'ensemble [1..31]).
ROR	Sortie	DINT	Valeur de rotation vers la droite. La fonction n'a aucun effet si NbR <= 0.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ROR

Exemple de schéma à relais ROR



Exemple de texte structuré ROR

```
1  in := 123;
2  nbr := 2;
3  rotation := ROR(in, nbr);
```

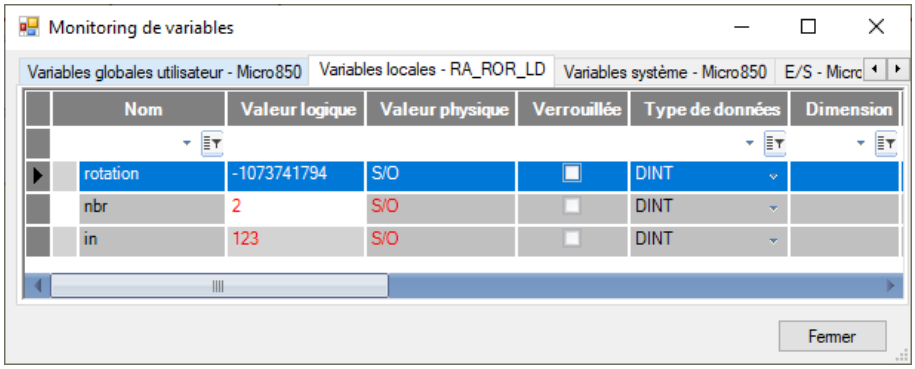
(* Équivalence ST : *)

résultat := ROR (registre, 1) ;

(* registre = 2#0100_1101_0011_0101 *)

(* résultat = 2#1010_0110_1001_1010 *)

Résultats

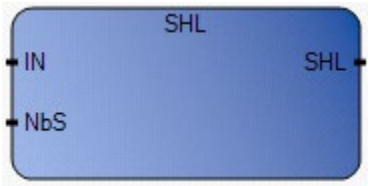


SHL (décalage à gauche)

Pour des entiers 32 bits, déplace les entiers vers la gauche et place 0 dans le bit de poids faible.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

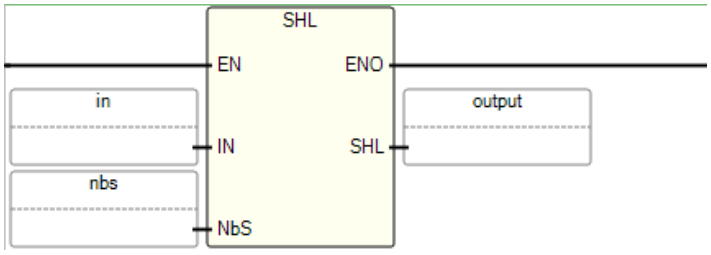


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : déplacement des entiers vers la gauche. FAUX : aucun mouvement d'entiers. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Valeur entière, quelle qu'elle soit.
NbS	Entrée	DINT	Nombre de décalages 1 bit (dans l'ensemble [1..31]).
SHL	Sortie	DINT	Valeur de décalage vers la gauche. La fonction n'a aucun effet si NbS <= 0. Si la valeur est égale à 0, la fonction remplace le bit de poids faible.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions SHL

Exemple de schéma à relais SHL



Exemple de texte structuré SHL

```
1  in := 123;
2  nbs := 2;
3  output := SHL(in, nbs);
```

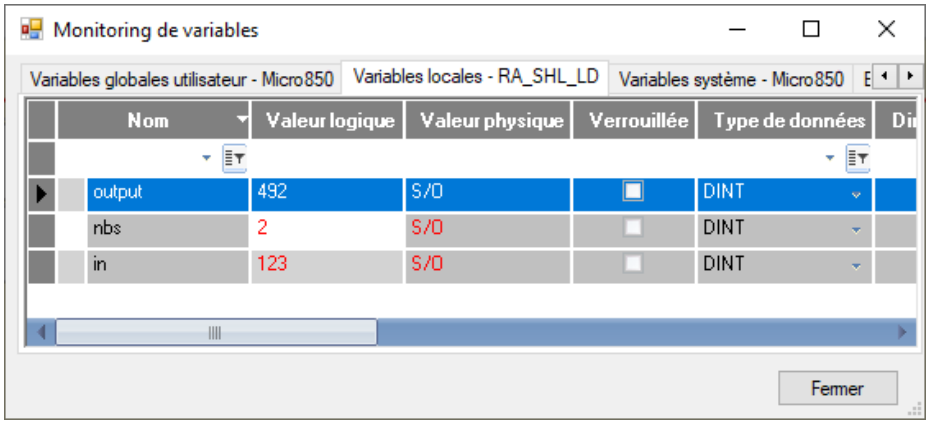
(* Équivalence ST : *)

résultat := SHL (registre,1) ;

(* registre = 2#0100_1101_0011_0101 *)

(* résultat = 2#1001_1010_0110_1010 *)

Résultats

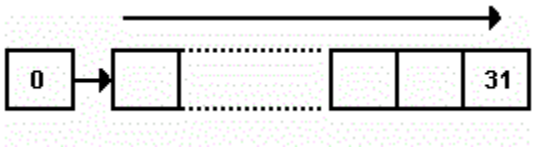


SHR (décalage à droite)

Décale les 32 bits d'un entier vers la droite et réplique le bit le plus à gauche (bit significatif) pour remplir les bits vacants.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

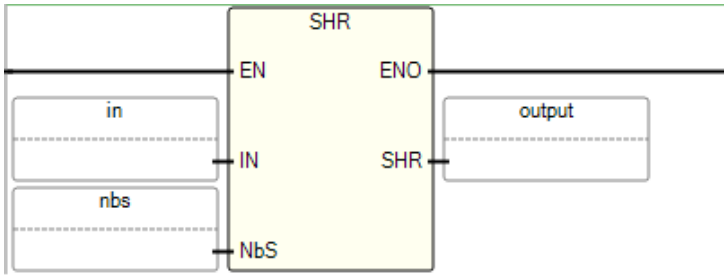


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : déplacement des entiers vers la droite. FAUX : aucun mouvement d'entiers. S'applique aux programmes de schémas à relais.
IN	Entrée	DINT	Valeur entière, quelle qu'elle soit.
NbS	Entrée	DINT	Nombre de décalages 1 bit (dans l'ensemble [1..31]).
SHR	Sortie	DINT	Valeur de décalage vers la droite. La fonction n'a aucun effet si NbS <= 0. Si la valeur est égale à 0, la fonction remplace le bit de poids fort.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions SHR

Exemple de schéma à relais SHR



Exemple de texte structuré SHR

```
1  in := 123;
2  nbs := 2;
3  output := SHR(in, nbs);
```

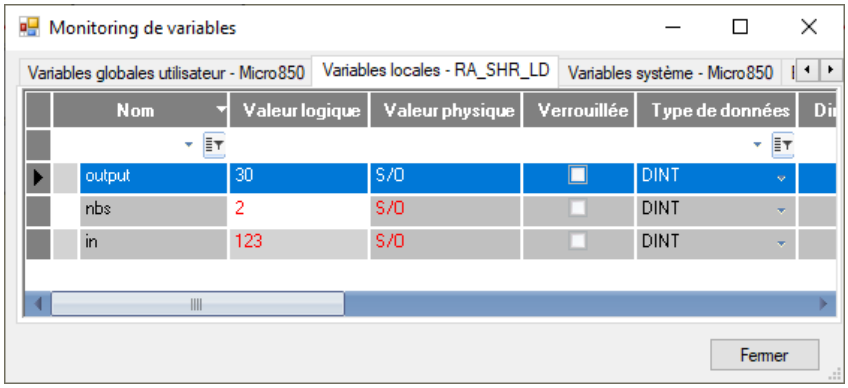
(* Équivalence ST : *)

résultat := SHR (registre,1) ;

(* registre = 2#1100_1101_0011_0101 *)

(* résultat = 2#0110_0110_1001_1010 *)

Résultats



XOR_MASK (masque OR exclusif)

Masque avec opérateur OR exclusif bit à bit entier, renvoie les valeurs de bits inversées.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

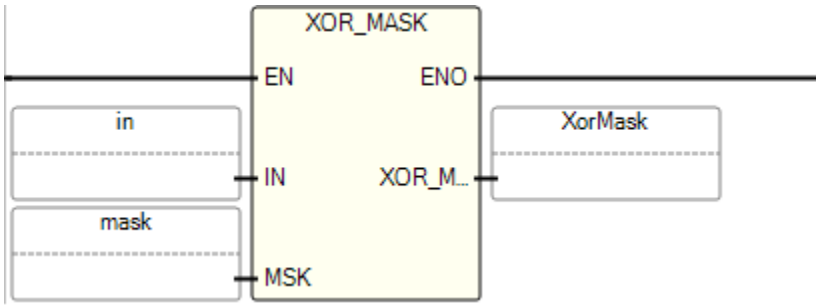
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul du masque bit à bit avec opérateur OR exclusif. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.

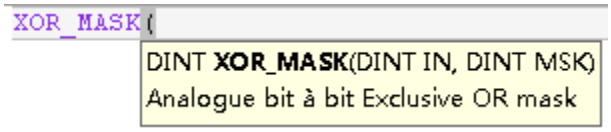
Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	DINT	Doit être au format entier.
MSK	Entrée	DINT	Doit être au format entier.
XOR_MASK	Sortie	DINT	Opérateur OR exclusif logique bit à bit entre IN et MSK.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction XOR_MASK

Exemple de logique à relais XOR_MASK



Exemple de texte structuré XOR_MASK



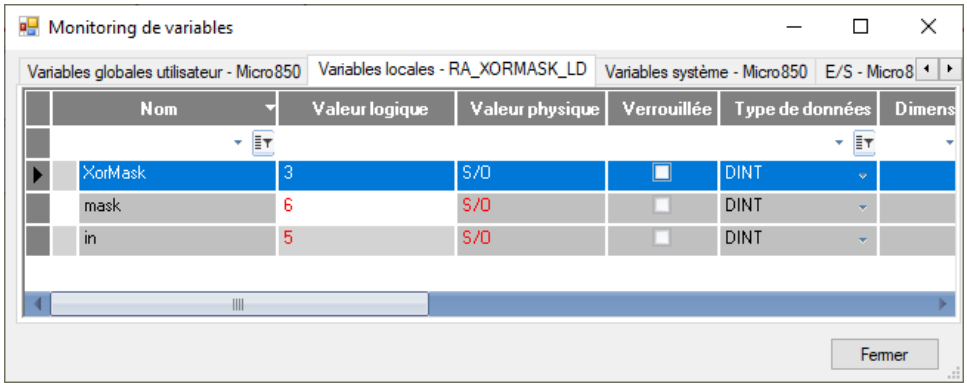

```
1  in := 5;
2  mask := 6;
3  XorMask := XOR_MASK(in, mask);
```

(* Équivalence ST : *)

crc32 := XOR_MASK (prevcrc, nextc) ;

résultat := XOR_MASK (16#012, 16#011) ; (* est égal à 16#003 *)

Résultats


Monitoring de variables
Variables globales utilisateur - Micro850
Variables locales - RA_XORMASK_LD
Variables système - Micro850
E/S - Micro8
Nom
Valeur logique
Valeur physique
Verrouillée
Type de données
Dimens
XorMask
3
S/O
DINT
mask
6
S/O
DINT
in
5
S/O
DINT
Fermer

Instructions booléennes

Utilisez des instructions booléennes pour déterminer une valeur de sortie en fonction d'un calcul logique à partir des entrées. Les sorties de module peuvent être directement contrôlées à partir du programme ou de manière indépendante par le module, à l'aide des instructions booléennes.

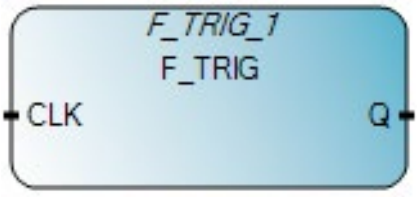
Fonction	Description
MUX4B sur la page 158	Multiplexeur entre quatre entrées BOOL, une valeur BOOL de sortie.
MUX8B sur la page 154	Multiplexeur entre huit entrées BOOL, une valeur BOOL de sortie.
TTABLE sur la page 152	Renvoie la valeur de la sortie déterminée en fonction de la combinaison des entrées.
Bloc de fonctions	**Description**
F_TRIG sur la page 143	Détecte un front descendant d'une variable booléenne.
RS sur la page 146	Remise à zéro du bistable dominant (priorité la plus élevée lors de la détermination du comportement de l'instruction).
R_TRIG sur la page 145	Détecte un front montant d'une variable booléenne.
SR sur la page 150	Définit le bistable dominant.
Opérateur	**Description**
AND sur la page 148	Effectue une opération booléenne AND (ET) entre deux ou plusieurs valeurs.
NOT sur la page 150	Convertit des valeurs booléennes en valeurs inversées.
XOR sur la page 149	Opérateur OR (OU) exclusif booléen entre deux valeurs.
OR sur la page 148	Opérateur booléen OR (OU) entre deux valeurs ou plus.

F_TRIG (détection d'un front descendant)

Détecte un front descendant d'une variable booléenne. Le bloc F_TRIG définit la sortie Q pour un cycle lorsque l'entrée CLK bascule de définie à effacée (c.-à-d. qu'un front descendant est détecté à l'entrée CLK).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

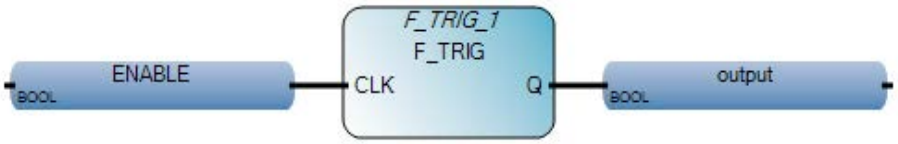
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
CLK	Entrée	BOOL	Permet de vérifier l'entrée d'un front descendant. Variable booléenne, quelle qu'elle soit. VRAI = aucun front descendant détecté. FAUX = front descendant détecté sur l'entrée CLK, définit la sortie Q sur VRAI.
Q	Sortie	BOOL	Indique l'état pour la sortie Q. VRAI = front descendant détecté, définit la sortie Q pour un cycle supplémentaire. FAUX = aucune modification de la sortie Q.

Exemple de Diagramme de Bloc Fonction F_TRIG



Exemple de logique à relais F_TRIG

Exemple de texte structuré F_TRIG

```
F_TRIG_1(
```

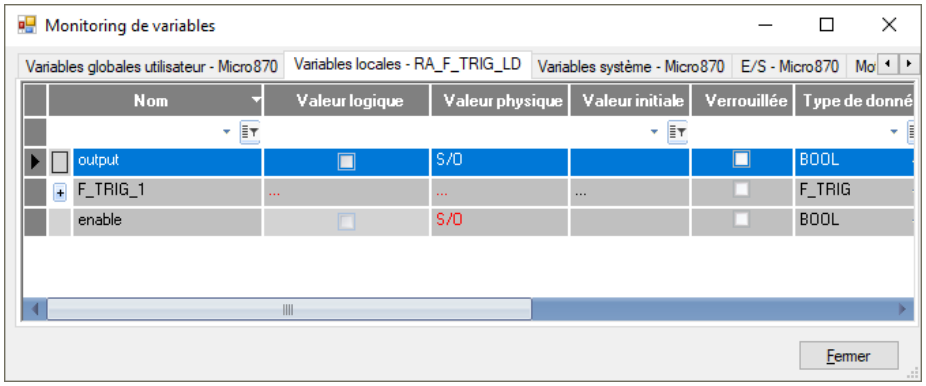
```
void F_TRIG_1(BOOL CLK)
Type: F_TRIG, Détection de front descendant
```

```
1  F_TRIG_1(ENABLE);
2  IF F_TRIG_1.Q THEN
3     output := TRUE;
4  END_IF;
```

(* Équivalence ST : F_TRIG1 est une instance d'un bloc F_TRIG *)

```
F_TRIG1(cmd) ;
nb_edge := ANY_TO_DINT(F_TRIG1.Q) + nb_edge ;
```

Résultats

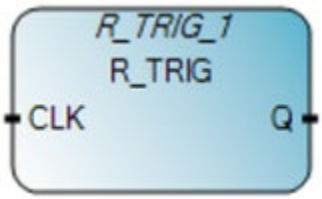


R_TRIG (détecteur de front montant)

Détecte un front montant d'une variable booléenne. Le bloc R_TRIG définit la sortie Q pour un cycle lorsque l'entrée CLK bascule d'effacée à définie (c.-à-d. qu'un front montant est détecté à l'entrée CLK).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

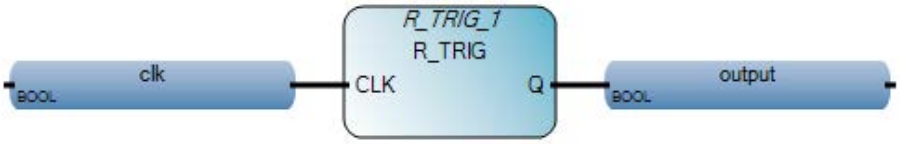
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

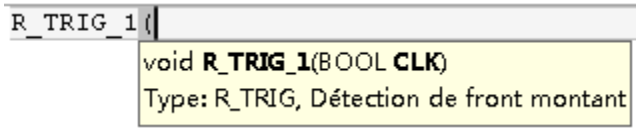
Paramètre	Type de paramètre	Type de données	Description
CLK	Entrée	BOOL	Variable booléenne, quelle qu'elle soit. VRAI : front montant détecté, définir Q sur VRAI. FAUX : aucun front montant détecté, définir Q sur FAUX.
Q	Sortie	BOOL	VRAI : lorsque CLK est défini sur VRAI. FAUX : dans tous les autres cas.

Exemple de diagramme de bloc de fonctions R_TRIG



Exemple de schéma à relais R_TRIG

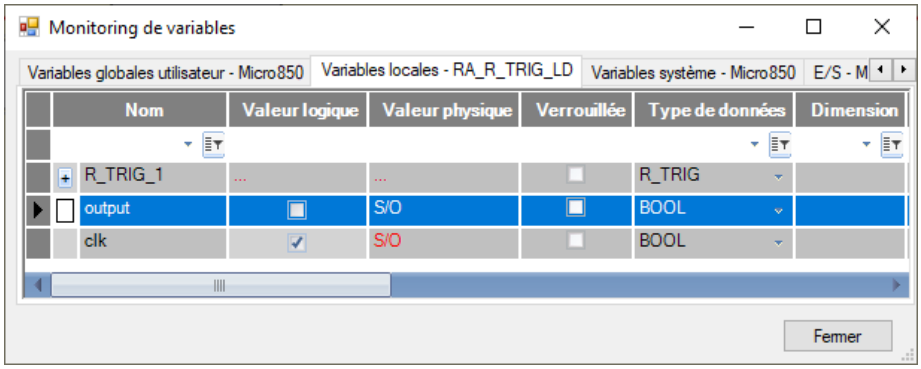
Exemple de texte structuré R_TRIG



(* Équivalence ST : R_TRIG1 est une instance d'un bloc R_TRIG *)

```
R_TRIG1(cmd) ;
nb_edge := ANY_TO_DINT(R_TRIG1.Q) + nb_edge ;
```

Résultats



RS (réinitialisation/définition)

Réinitialise ou définit le bistable dominant (priorité la plus élevée lors de la détermination du comportement de l'instruction).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

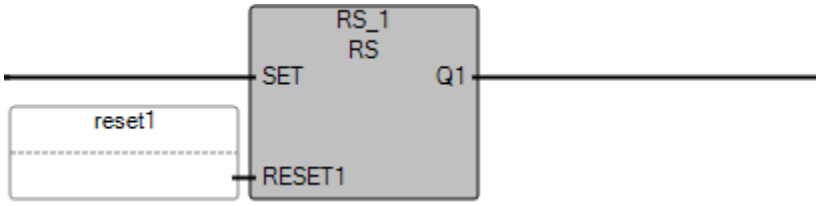
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
SET	Entrée	BOOL	VRAI : définit Q1 sur VRAI.
RESET1	Entrée	BOOL	VRAI : définit Q1 sur FAUX (prioritaire).
Q1	Sortie	BOOL	Etat de la mémoire booléenne.

Exemple de diagramme de bloc de fonctions RS

Exemple de schéma à relais RS



Exemple de texte structuré RS

```
RS_1(
```

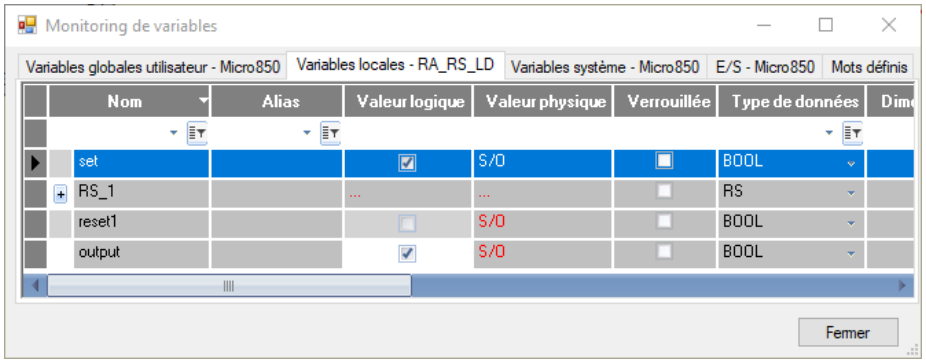
void **RS_1**(BOOL **SET**, BOOL **RESET1**)
Type: RS, Réapplique le bistable dominant

```
1  set := TRUE;
2  reset1 := FALSE;
3  RS_1(set, reset1);
4  output := RS_1.Q1;
```

(* Équivalence ST : RS1 est une instance d'un bloc RS *)

```
RS1(start_cmd, (stop_cmd OR alarme)) ;
commande := RS1.Q1 ;
```

Résultats

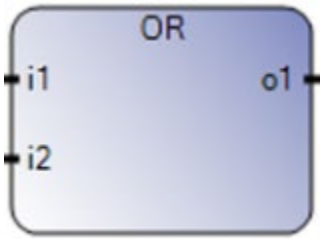


OR

Effectue une opération OR logique de deux valeurs booléennes ou plus et renvoie la valeur booléenne VRAI si l'entrée est « vraie » ; dans le cas contraire, renvoie une valeur booléenne FAUX.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	BOOL	
i2	Entrée	BOOL	
o1	Sortie	BOOL	Booléen OR des termes d'entrée. VRAI : lorsqu'une ou plusieurs entrées sont VRAIES. FAUX : lorsque les entrées sont FAUSSES.

Exemple de texte structuré OR

(* Équivalence ST : *)

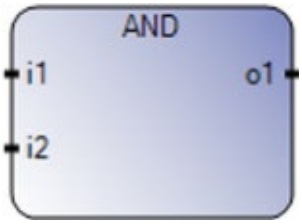
```
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;
```

AND

Effectue une opération booléenne AND (ET) entre deux ou plusieurs valeurs.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	BOOL	Valeur de type de données booléen.
i2	Entrée	BOOL	Valeur de type de données booléen.
o1	Sortie	BOOL	Résultat de l'opération booléenne AND (ET) sur les valeurs d'entrée.

Exemple de texte structuré AND

(* Équivalence ST : *)

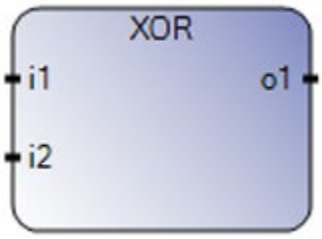
```
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

XOR (OR exclusif)

Effectue une opération OR exclusive de deux valeurs booléennes.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	BOOL	
i2	Entrée	BOOL	
o1	Sortie	BOOL	OR exclusif booléen de deux termes d'entrée. TRUE : lorsque l'une des entrées est TRUE. FALSE : lorsque les deux entrées sont FALSE ou que les deux entrées sont TRUE.

Exemple de texte structuré XOR

(* Équivalence ST : *)

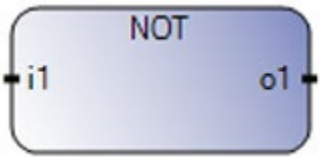
```
bo10 := bi101 XOR NOT (bi102) ;
bo5 := (bi51 XOR bi52) XOR bi53;
```

NOT

Convertit des valeurs booléennes en valeurs inversées.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	BOOL	Toute valeur booléenne ou expression complexe.
o1	Sortie	BOOL	TRUE lorsque IN est FALSE. FALSE lorsque IN est TRUE.

Exemple de texte structuré NOT

(* Équivalence ST : *)

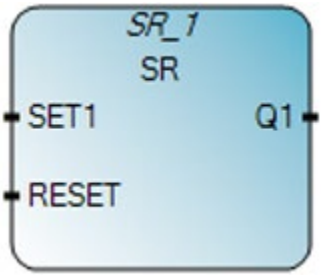
```
bo10 := NOT (bi101);
```

SR (définition/réinitialisation)

Définit le bistable à Reset prioritaire.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description

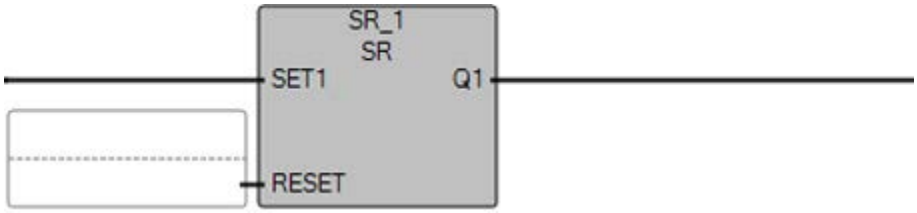
Paramètre	Type de paramètre	Type de données	Description
SET1	Entrée	BOOL	TRUE : définit Q1 sur TRUE (dominant : priorité la plus élevée lors de la détermination du comportement de l'instruction).
RESET	Entrée	BOOL	VRAI : définit Q1 sur FAUX.
Q1	Sortie	BOOL	Etat de la mémoire booléenne. VRAI : lorsque SET1 est défini sur VRAI. FAUX : lorsque RESET est défini sur VRAI.

Exemple de bistable dominant

Set1	Réinitialiser	Q1	Résultat Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Exemple de diagramme de bloc de fonctions SR

Exemple de schéma à relais SR



Exemple de texte structuré SR

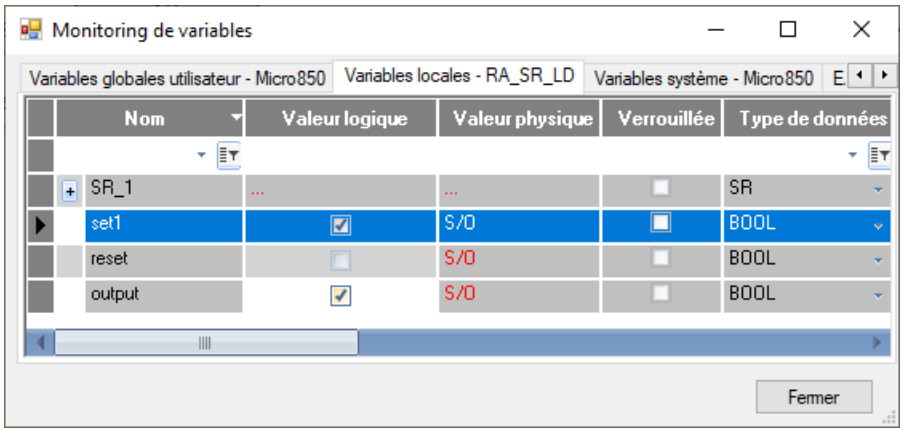
```
SR_1(
```

void **SR_1**(BOOL **SET1**, BOOL **RESET**)
Type: SR, Applique le bistable dominant

```
1  set1 := TRUE;
2  reset := FALSE;
3  SR_1(set1, reset);
4  output := SR_1.Q1;
```

```
(* Équivalence ST : SR1 est une instance d'un bloc SR *)
SR1((auto_mode & start_cmd), stop_cmd) ;
commande := SR1.Q1 ;
```

Résultats



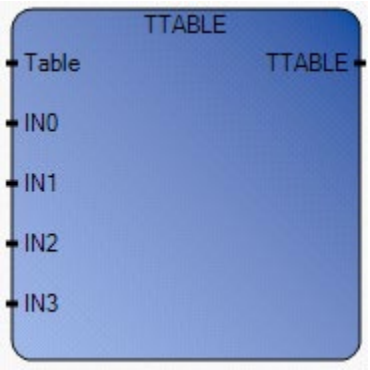
TTABLE (tableau de vérité)

Renvoie la valeur de la sortie déterminée en fonction de la combinaison des entrées.

Si la valeur est égale à 0xABCD et que In3 à In0 correspondent au nombre 7, la fonction TTABLE renvoie la valeur du bit 7 du tableau (1 en l'occurrence). Le bit de poids faible dans ce tableau correspond au bit 0.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Table	Entrée	UINT	Tableau de vérités pour fonction booléenne.
IN0	Entrée	BOOL	Toute valeur d'entrée booléenne
IN1	Entrée	BOOL	Toute valeur d'entrée booléenne
IN2	Entrée	BOOL	Toute valeur d'entrée booléenne

Paramètre	Type de paramètre	Type de données	Description
IN3	Entrée	BOOL	Toute valeur d'entrée booléenne
TTABLE	Sortie	BOOL	Valeur de la sortie déterminée en fonction de la combinaison des entrées.

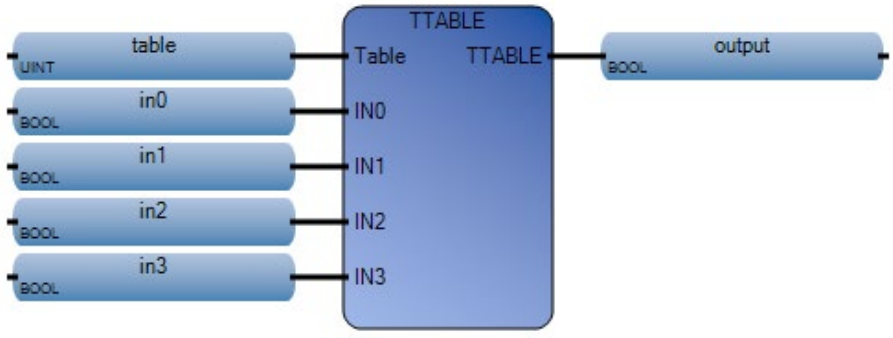
Combinaisons des entrées TTABLE

L'instruction TTABLE comporte quatre entrées et peut donc générer 16 combinaisons. Ces combinaisons sont répertoriées dans un tableau de vérité ; pour chacune d'entre elles, la valeur de sortie peut être ajustée. Le nombre de combinaisons configurables dépend du nombre d'entrées connectées à la fonction.

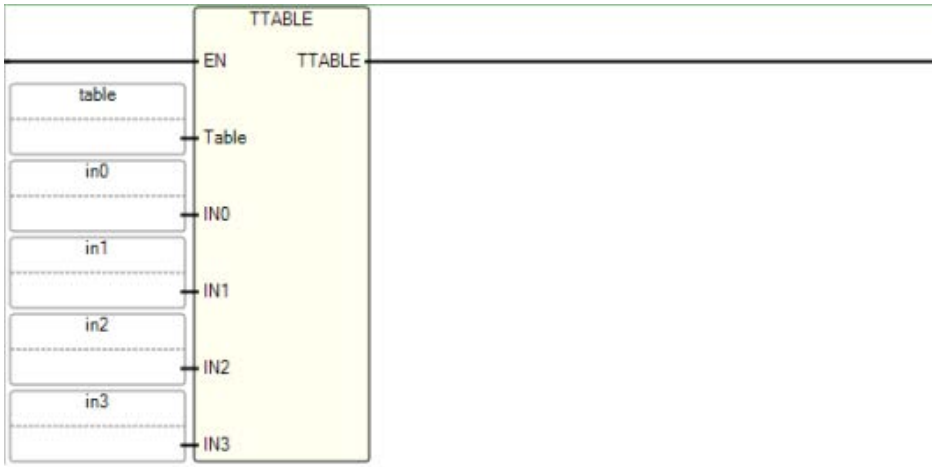
Exemple de combinaison de tableau de vérité.

Nombre	In3	In2	In1	In0
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	1	1	1
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

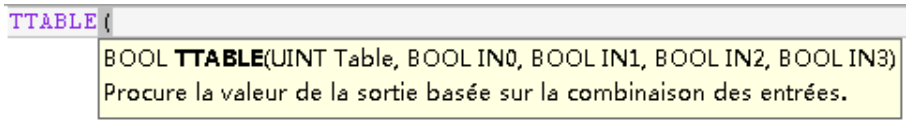
Exemple de Diagramme de Bloc Fonction TTABLE



Exemple de schéma à relais TTABLE

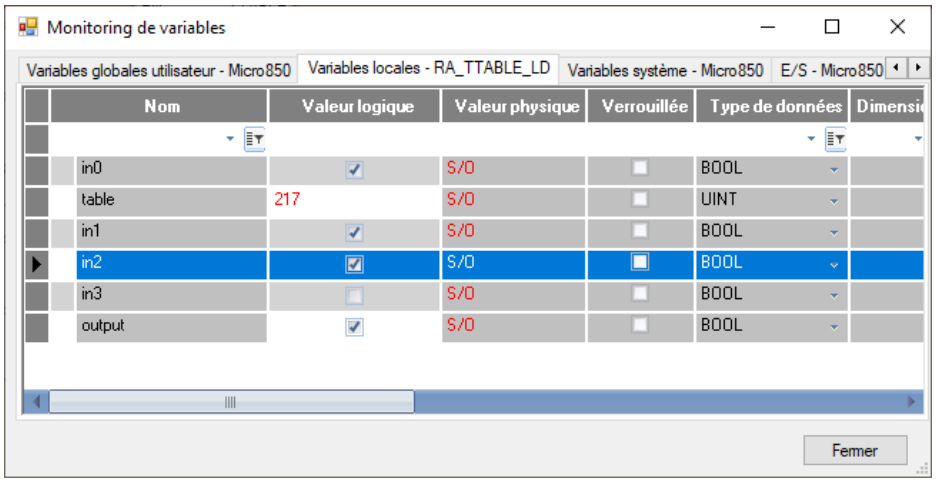


Exemple de texte structuré TTABLE




```
1  table := 217;
2  in0 := TRUE;
3  in1 := TRUE;
4  in2 := TRUE;
5  in3 := FALSE;
6  output := TTABLE(table, in0, in1, in2, in3);
```

Résultats



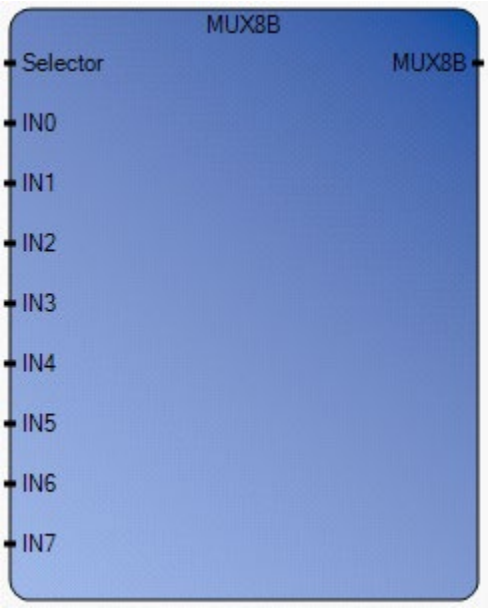
MUX8B (multiplexeur de 8 entrées BOOL)

Multiplexeur entre huit entrées BOOL, une valeur BOOL de sortie.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de

schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

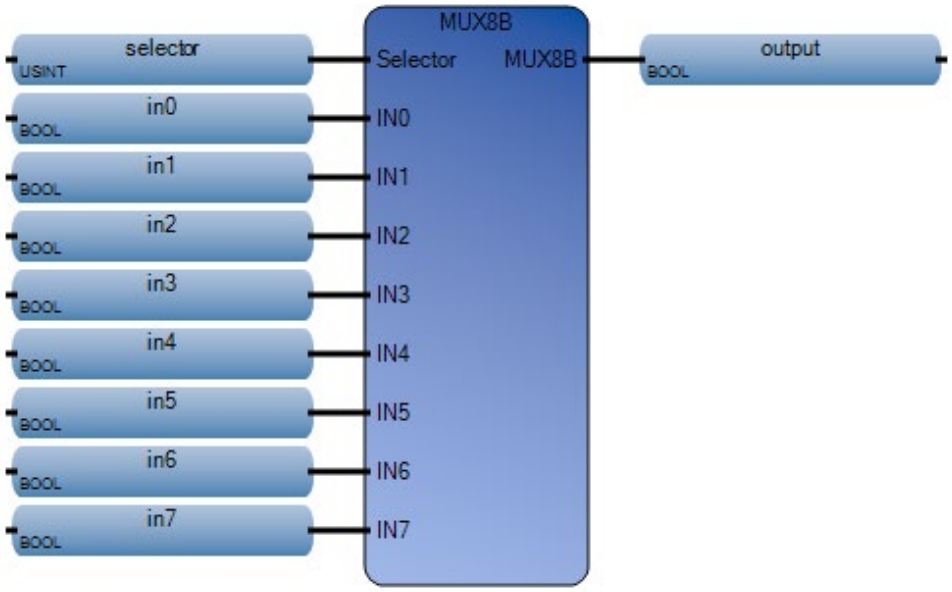


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

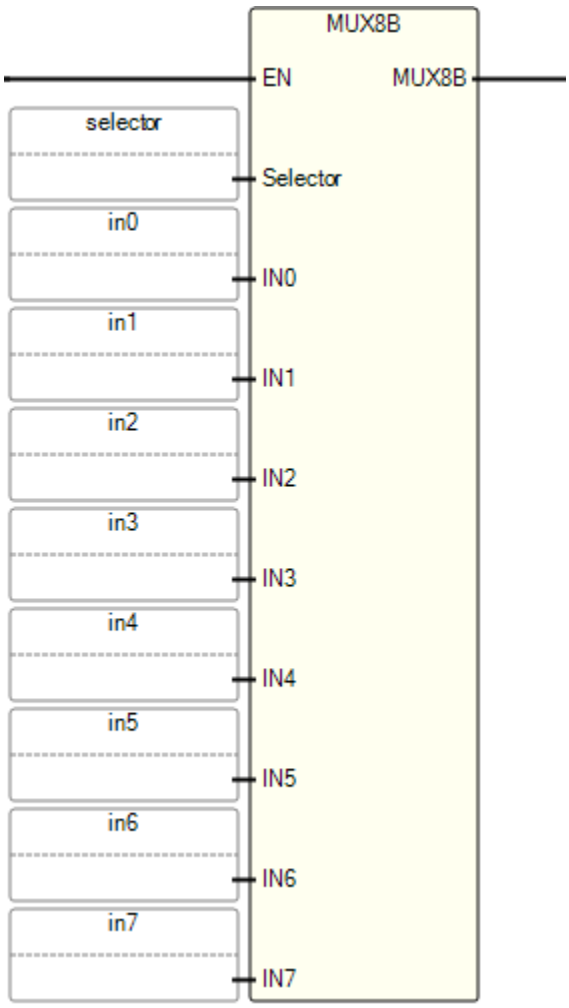
Paramètre	Type de paramètre	Type de données	Description
Selector	Entrée	USINT	Valeur entière de sélection (doit être dans l'intervalle [0..7])
IN0	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 0. FAUX : lorsque le sélecteur n'est pas 0.
IN1	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 1. FAUX : lorsque le sélecteur n'est pas 1.
IN2	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 2. FAUX : lorsque le sélecteur n'est pas 2.
IN3	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 3. FAUX : lorsque le sélecteur n'est pas 3.
IN4	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 4. FAUX : lorsque le sélecteur n'est pas 4.
IN5	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 5. FAUX : lorsque le sélecteur n'est pas 5.
IN6	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 6. FAUX : lorsque le sélecteur n'est pas 6.
IN7	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 7. FAUX : lorsque le sélecteur n'est pas 7.

Paramètre	Type de paramètre	Type de données	Description
MUX8B	Sortie	BOOL	VRAI : lorsque • In0 si sélecteur = 0 ; • In1 si sélecteur = 1 ; • In2 si sélecteur = 2 ; • In3 si sélecteur = 3 ; • In4 si sélecteur = 4 ; • In5 si sélecteur = 5 ; • In6 si sélecteur = 6 ; • In7 si sélecteur = 7 ; FAUX : pour toutes les autres valeurs du sélecteur.

Exemple de Diagramme de Bloc Fonction MUX8B



Exemple de logique à relais MUX8B



Exemple de texte structuré MUX8B

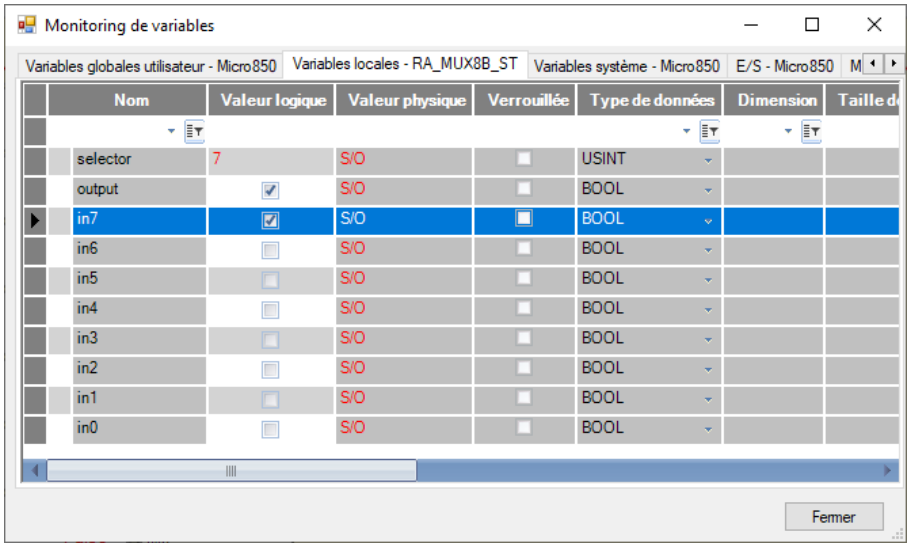
```
MUX8B(
    BOOL MUX8B(USINT Selector, BOOL IN0, BOOL IN1, BOOL IN2, BOOL IN3, BOOL IN4, BOOL IN5, BOOL IN6, BOOL IN7)
    Multiplexeur (8 entrées) - accepte les valeurs d'entrées et de sorties BOOL.

 1  selector := 7;
 2  in0 := FALSE;
 3  in1 := FALSE;
 4  in2 := FALSE;
 5  in3 := FALSE;
 6  in4 := FALSE;
 7  in5 := FALSE;
 8  in6 := FALSE;
 9  in7 := TRUE;
10  output := MUX8B(selector, in0, in1, in2, in3, in4, in5, in6, in7);
```

(* Équivalence ST : *)

plage = MUX8 (choix, 1, 5, 10, 50, 100, 500, 1000, 5000) ;

(* choisir parmi 8 plages prédéfinies ; si vous sélectionnez 3, par exemple, la plage équivaut à 50 *)

Résultats

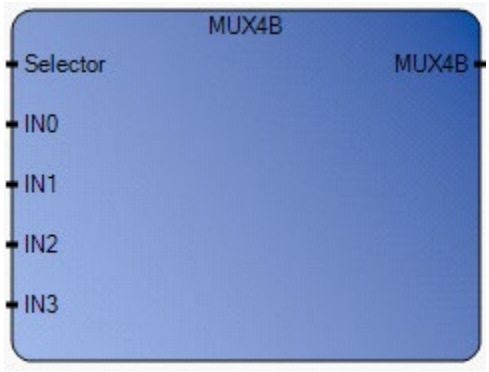


MUX4B (multiplexeur de 4 entrées BOOL)

Multiplexeur entre quatre entrées BOOL, une valeur BOOL de sortie.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

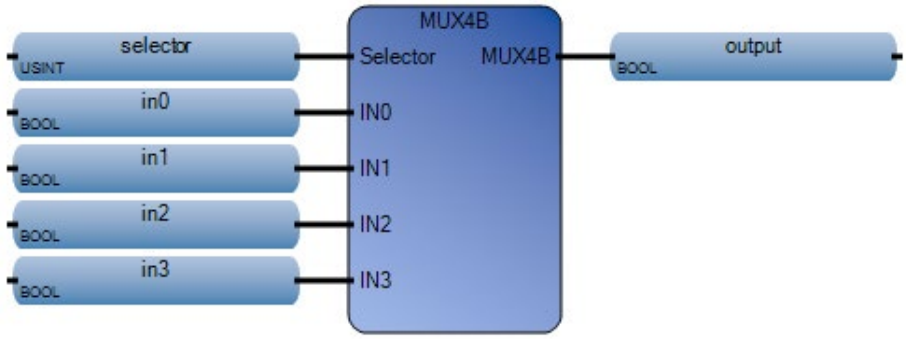


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

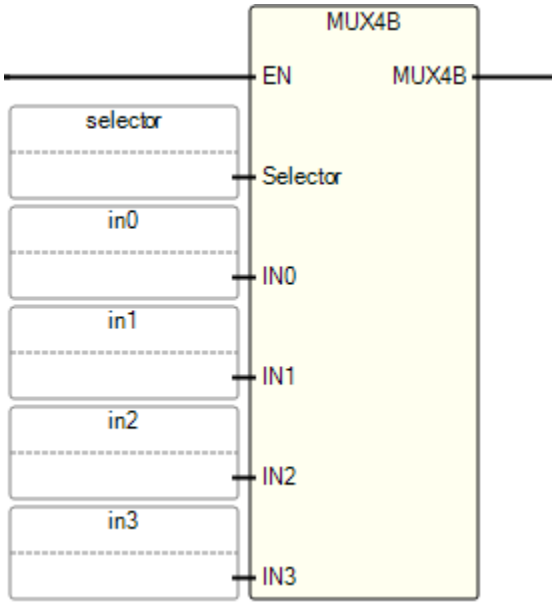
Paramètre	Type de paramètre	Type de données	Description
Selector	Entrée	USINT	Valeur entière de sélection (doit être dans l'intervalle [0..3])
IN0	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 0. FAUX : lorsque le sélecteur n'est pas 0.
IN1	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 1. FAUX : lorsque le sélecteur n'est pas 1.
IN2	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 2. FAUX : lorsque le sélecteur n'est pas 2.

Paramètre	Type de paramètre	Type de données	Description
IN3	Entrée	BOOL	Toute valeur d'entrée booléenne VRAI : lorsque le sélecteur est 3. FAUX : lorsque le sélecteur n'est pas 3.
MUX4B	Sortie	BOOL	VRAI : lorsque • In0 si sélecteur = 0 ; • In1 si sélecteur = 1 ; • In2 si sélecteur = 2 ; • In3 si sélecteur = 3 ; FAUX : pour toutes les autres valeurs du sélecteur.

Exemple de Diagramme de Bloc Fonction MUX4B



Exemple de logique à relais MUX4B



Exemple de texte structuré MUX4B

```
MUX4B(
```

BOOL **MUX4B**(USINT Selector, BOOL IN0, BOOL IN1, BOOL IN2, BOOL IN3)
Multiplexeur (4 entrées) - accepte les valeurs d'entrées et de sorties BOOL.

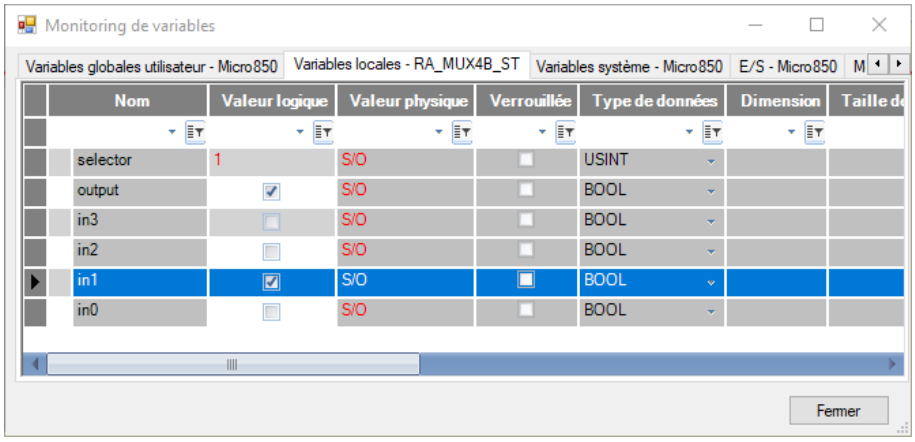
```
1  selector := 1;
2  in0 := FALSE;
3  in1 := TRUE;
4  in2 := FALSE;
5  in3 := FALSE;
6  output := MUX4B(selector, in0, in1, in2, in3);
```

(* Équivalence ST : *)

plage = MUX4 (choix, 1, 10, 100, 1000) ;

(* choisir parmi 4 plages prédéfinies ; si vous sélectionnez 1, par exemple, la plage équivaut à 10 *)

Résultats



Instructions de communication

Utilisez les instructions de communication pour lire, écrire, comparer et convertir les chaînes de communication.

Bloc de fonctions	Description
COM_IO_WDOG sur la page 161	Surveille les communications vers le contrôleur.
MSG_CIPGENERIC sur la page 163	Envoie un message explicite générique CIP.
MSG_CIPSYMBOLIC sur la page 171	Envoie un message explicite symbolique CIP.
MSG_MODBUS sur la page 175	Envoie un message Modbus.
MSG_MODBUS2 sur la page 181	Envoie un message MODBUS/TCP via un canal Ethernet.
MSG_PCCC sur la page 187	Envoie un message PCCC via un canal Ethernet ou un port série.

COM_IO_WDOG

Surveille les messages externes vers les entrées et sorties du contrôleur. Par exemple, si la commande d'écriture CIP pour la variable _IO_EM_DO_00 n'est pas reçue via EtherNet/IP dans le temps configuré, le temporisateur du chien de garde expirera et toutes les sorties du contrôleur seront réinitialisées.

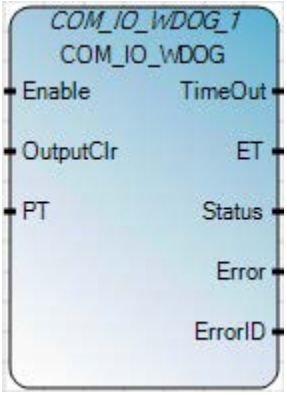
Les protocoles EtherNet/IP, Modbus TCP et RTU Modbus sont pris en charge.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator. Il n'est pris en charge qu'avec la version 12.00 du micrologiciel et les versions ultérieures.

Remarque :

- bien que plusieurs instances de cette instruction puissent exister, une seule instance peut être activée. Sinon, une erreur se produira avec ErrorID défini sur 2.
- Seules les E/S intégrées au contrôleur, enfichables et d'extension numérique sont prises en charge.



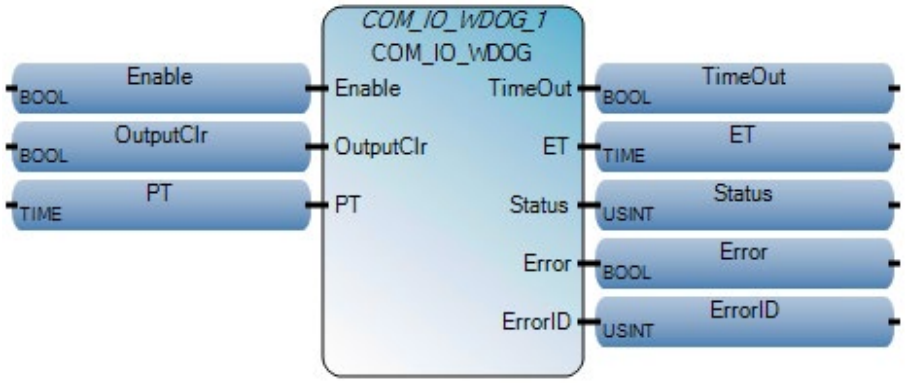
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	VRAI : voici les trois cas de la condition VRAI. 1. Sur le front montant de VRAI, l'exécution du bloc de fonctions démarre. Le bit Status est défini sur 0 et le reste des bits est effacé. Vérifiez ensuite les erreurs ci-dessous. Si aucune erreur ne se produit, passez à l'étape suivante. • Si PresetValue est inférieur à une seconde, Error est défini sur VRAI et ErrorID, sur 1, le bit Status 4 est défini, TimeOut et ElapsedTime sont effacés. • Si le temporisateur est déjà acquis par l'autre instance, Error est défini sur VRAI et ErrorID, sur 2, le bit Status 4 est défini, TimeOut et ElapsedTime sont effacés. 2. PresetTime > ElapsedTime. Lorsque Enable est défini sur VRAI et que l'exécution du bloc de fonctions démarre. Vérifiez si des commandes sont reçues. Si des commandes sont reçues, réinitialiser le temporisateur immédiatement. Définir ElapsedTime sur 0 et le bit Status 2. Le reste des bits est effacé. Si aucune commande n'est reçue, définir le bit Status 1, et le reste des bits est effacé. 3. PresetTime = ElapsedTime. Lorsque Enable est défini sur VRAI et que l'exécution du bloc de fonctions démarre. Vérifiez si des commandes sont reçues. Si c'est le cas, réinitialiser immédiatement le temporisateur. Définir ElapsedTime sur 0 et définir le bit Status 2. Le reste des bits est effacé. Sinon, définir le bit TimeOut, définir le bit Status 3, et le reste des bits est effacé. Définir la variable interne pour effacer les sorties numériques à la fin de la scrutation si l'action configurée consiste à effacer toutes les sorties numériques à la fin de la scrutation. FAUX : l'instruction ne s'exécute pas, et les sorties de bloc de fonctions sont effacées.
OutputClr	Entrée	BOOL	0 : ne rien faire si le temps est dépassé. 1 : effacer toutes les sorties numériques à la fin de la scrutation (Embedded, EXIO et UPM) si le temps est dépassé.
TI	Entrée	TIME	Temps à attendre avant que le temps ne soit dépassé. La valeur de temps dépassé ne peut pas être inférieure à une seconde, ou une erreur se produit. La valeur maximale de PresetTime peut être la valeur maximale dans l'intervalle du type de données TIME.
TimeOut	Sortie	BOOL	VRAI : ElapsedTime est égal à PresetTime. FAUX : Enable est défini sur FAUX ; le temporisateur n'est pas écoulé ou une erreur s'est produite.
ET	Sortie	TIME	Le temps actuellement écoulé. Les valeurs possibles sont comprises entre 0 ms et 1 193 h 2 m 47 s 294 ms.
Status	Sortie	USINT	État du bloc de fonctions. Bit 0 : activer Bit 1 : le temporisateur s'exécute, aucune sortie ou entrée n'a été reçue. Bit 2 : la commande de sortie ou d'entrée a été reçue. Bit 3 : le temps est dépassé. Aucune commande de sortie ou d'entrée n'a été reçue. Bit 4 : une erreur s'est produite. Les autres bits sont réservés.
Error	Sortie	BOOL	Indique une erreur.
ErrorID	Sortie	USINT	Lorsqu'une erreur se produit, ErrorID contient le code d'erreur.

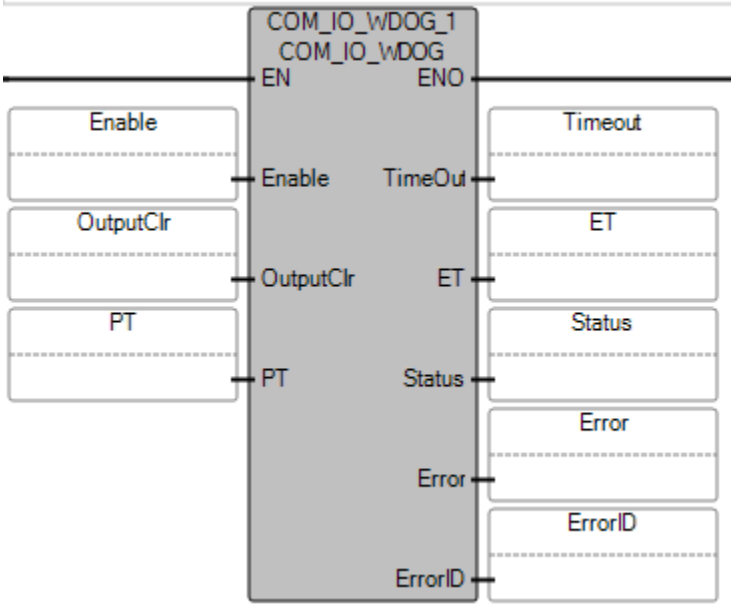
Code d'erreur COM_IO_WDOG

Code ErrorID	Description de l'erreur
1	PresetTime est inférieur à une seconde.
2	Une autre instance de bloc de fonctions COM_IO_WDOG est déjà en cours d'exécution.

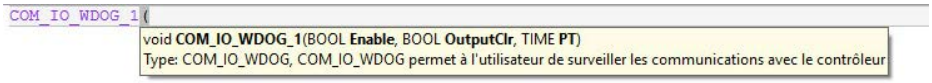
Exemple de diagramme de bloc fonction COM_IO_WDOG



Exemple de langage à relais COM_IO_WDOG



Exemple de texte structuré COM_IO_WDOG




```
1  COM_IO_WDOG_1(Enable, OutputCrl, PT);
2   Timeout :=COM_IO_WDOG_1.Timeout;
3   ET :=COM_IO_WDOG_1.ET;
4   Status :=COM_IO_WDOG_1.Status;
5   Error :=COM_IO_WDOG_1.Error;
6   ErrorID :=COM_IO_WDOG_1.ErrorID;
```

MSG_CIPGENERIC (message générique du protocole CIP)

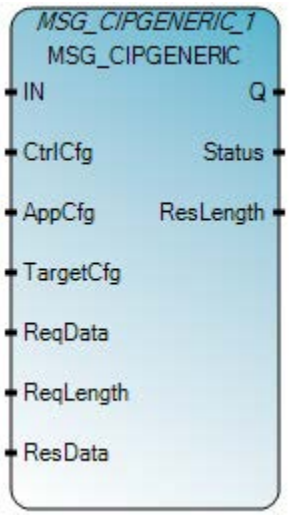
Envoie un message explicite CIP (common industrial protocol) via un canal Ethernet ou un port série.

Au maximum quatre demandes de message par canal peuvent être traitées au

cours de chaque scrutation. Pour les programmes de schémas à relais, les demandes de message sont exécutées à la fin de la scrutation à relais.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

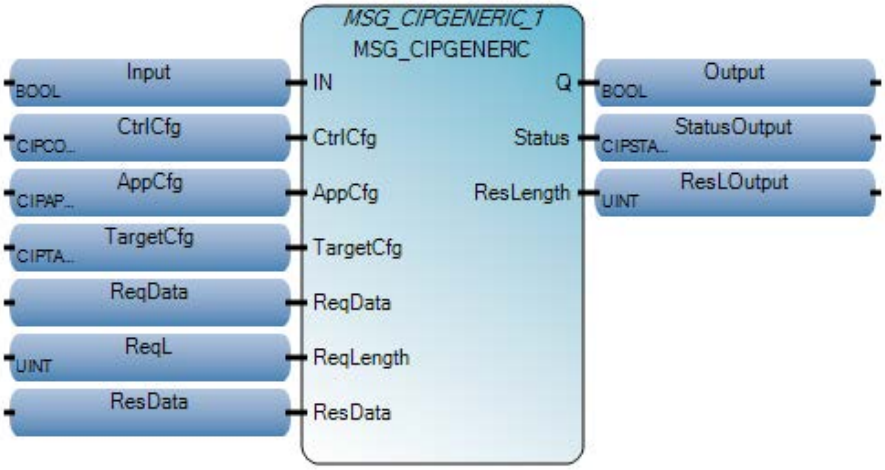


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

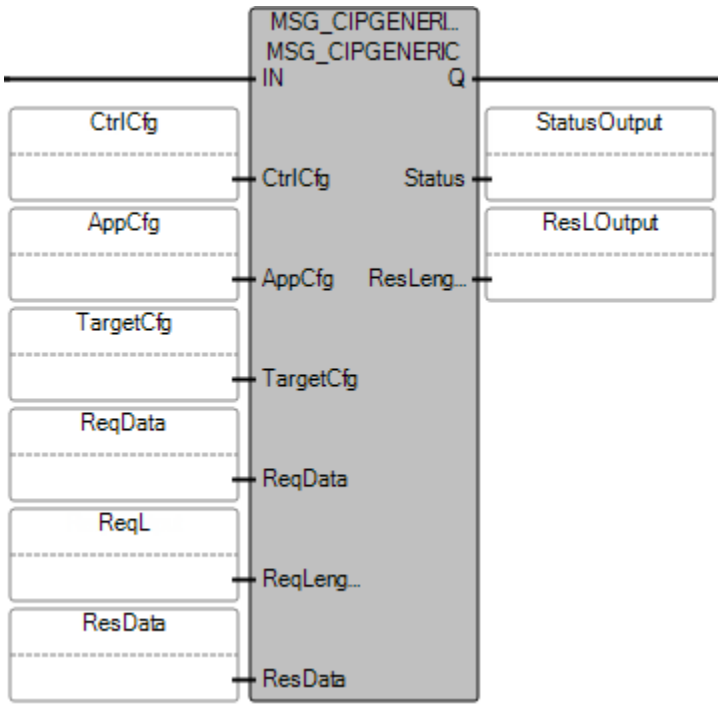
Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : front montant non détecté, en attente.
CtrlCfg	Entrée	CIPCONTROLCFG	Configuration du contrôle d'exécution du bloc d'instruction Utilisez les paramètres de type de données CIPCONTROLCFG sur la page 166 pour définir CtrlCfg.
AppCfg	Entrée	CIPAPPCFG	Configuration du service CIP et du chemin de l'application (EPATH) Utilisez les paramètres de type de données CIPAPPCFG sur la page 166 pour définir AppCfg.
TargetCfg	Entrée	CIPTARGETCFG	Configuration de l'appareil cible. Utilisez les paramètres de type de données CIPTARGETCFG sur la page 169 pour définir TargetCfg.
ReqData	Entrée	USINT[1..1]	Données de la demande de message CIP. La taille du tableau doit être supérieure à la valeur ReqLength.
ReqLength	Entrée	UINT	Longueur des données de la demande de message CIP : • 0 - 490
ResData	Entrée	USINT[1..1]	Données de la réponse au message CIP. La taille du tableau doit être supérieure à la valeur ReqLength. Lorsqu'une instruction MSG est déclenchée ou redéclenchée, les données du tableau ResData sont effacées.
Q	Sortie	BOOL	Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : l'instruction MSG s'est terminée correctement. FAUX : l'instruction MSG n'est pas terminée.

Paramètre	Type de paramètre	Type de données	Description
Status	Sortie	CIPSTATUS	L'état du bloc d'instruction. Lorsqu'une instruction MSG est déclenchée ou redéclenchée, tous les éléments d'état sont réinitialisés. La sortie d'état est définie dans le type de données CIPSTATUS sur la page 167.
ResLength	Sortie	UINT	Longueur des données de la réponse au message CIP : • 0 - 490 Lorsqu'une instruction MSG est déclenchée ou redéclenchée, la valeur ResLength est remise à zéro.

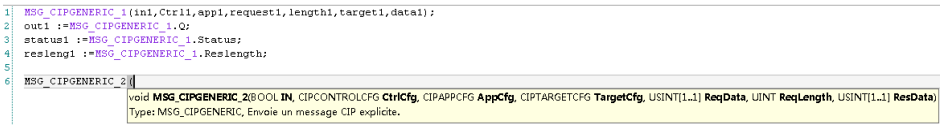
Exemple de diagramme de bloc de fonctions MSG_CIPGENERIC



Exemple de langage à relais MSG_CIPGENERIC



Exemple de texte structuré MSG_CIPGENERIC



Type de données CIPAPPCFG

Ce tableau permet de définir les paramètres pour le type de données CIPAPPCFG.

Paramètre	Type de données	Description
Service	USINT	Code de service : 1 – 127
Class	UINT	Valeur d'ID de classe du segment logique : 1 – 65535
Instance	UDINT	Valeur d'ID d'instance du segment logique : 0 – 4294967295
Attribute	UINT	Valeur d'ID d'attribut du segment logique : 1 - 65535, 0 - Aucun ID d'attribut n'est utilisé
MemberCnt	USINT	Nombre d'ID de membre. Valeurs maximum d'ID de membre utilisées : 1 - 3, 0 - Aucun ID de membre n'est utilisé
MemberId	UINT[3]	Valeurs d'ID de membre : 0 - 65535

Type de données CIPCONTROLCFG

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données CIPCONTROLCFG.

Paramètre	Type de données	Description
Cancel	BOOL	VRAI - Annule l'exécution du bloc fonctionnel. Le bit est effacé lorsque le message est activé. Si le paramètre Cancel est configuré et si le message est activé (bit EN défini) et non terminé (bit DN non défini), l'exécution du message est annulée et le bit ER est défini.
TriggerType	USINT	Représente l'une des valeurs suivantes : • 0: envoi de message déclenché une fois (lorsque IN passe de Faux à Vrai) • 1 à 65 535 : Valeur de déclenchement cyclique en millisecondes. Message envoyé périodiquement lorsque IN est défini sur Vrai. Définissez la valeur sur 1 pour déclencher l'instruction MSG le plus rapidement possible.
StrMode	USINT	Réservé pour une utilisation future.

Déclenchement de message CIP

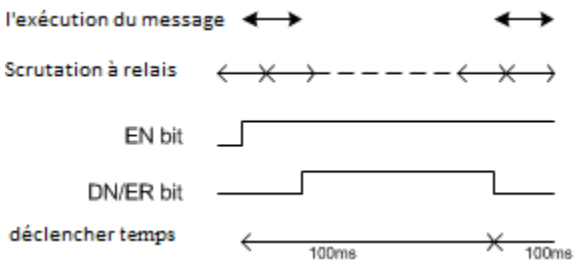
Un message CIP peut être déclenché périodiquement en définissant le paramètre TriggerType sur une valeur différente de zéro.

Ce tableau permet de définir les actions pour le paramètre Type de Déclenchement.

Action	Résultats
Le message est activé.	Le temporisateur de déclenchement démarre.
Le temporisateur de déclenchement arrive à expiration avant que le message soit terminé.	Le message est immédiatement déclenché lors du cycle de scrutation de relais suivant.
Le message se termine avant l'expiration du délai de déclenchement	Le message est déclenché avant l'expiration du délai de déclenchement

Exemple : déclenchement de message

Dans l'exemple suivant, la valeur TriggerType est définie sur 100.



Type de données CIPSTATUS

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données CIPSTATUS.

Paramètre	Type de données	Description
Error	BOOL	Ce bit est défini sur TRUE lorsque l'exécution du bloc fonctionnel rencontre une erreur.
ErrorID	UINT	Valeur du code d'erreur. Les éléments ErrorID sont définis dans les codes d'erreur CIPSTATUS sur la page 168.
SubErrorID	UINT	Sous-valeur du code d'erreur. Les éléments SubErrorID sont définis dans les codes d'erreur CIPSTATUS.
ExtErrorID	UINT	Valeur du code d'erreur du statut prolongé CIP.
StatusBits	UINT	Ce paramètre peut être utilisé pour vérifier les bits de contrôle : • Bit 0 : EN – Enable • Bit 1 : EW – Enable Wait • Bit 2 : ST – Start • Bit 3 : ER – Error • Bit 4 : DN – Done • Bit 5 : CIPCONN : coupure de connexion CIP • Bit 6 : EIPSESS : fermeture de session EIP • Les autres bits sont réservés. Les éléments StatusBits sont définis pour les bits d'état CIPSTATUS sur la page 167.

Bits de statut CIPSTATUS

Les bits de statut CIPSTATUS sont définis sur la base du statut d'exécution du message, des mémoires tampon de communication et des conditions de ligne.

-	-	-	-	-	-	-	-	-	-	-	4	3	2	1	0

Bit	Nom	Description	Comportement
0	EN	Enable	Est à 1 lorsque la ligne passe à Vrai et reste à 1 jusqu'à ce que le bit DN ou ER passe à 1 et que la ligne passe à Faux.
1	EW	Enable Waiting	Est à 1 lorsque la mémoire tampon de communication est affectée à la demande de message. Est à 0 lorsque le bit ST est à 1.
2	ST	Start	Est à 1 lorsque le message a été transmis et est en attente de réponse. Est à 0 lorsque le bit DN est à 1.
3	ER	Error	Est à 1 lorsque la transmission du message a échoué. Un code d'erreur est écrit sur ErrorID. Le bit ER et les valeurs de code d'erreur sont effacés lorsque la ligne passe à nouveau de Faux à Vrai.

Bit	Nom	Description	Comportement
4	DN	Done	Est à 1 lorsque l'envoi du message a réussi. Le bit DN passe à 0 lorsque la ligne passe à nouveau de Faux à Vrai. Lorsque le bit Done est à 1, tous les autres bits passent à 0 (effacés) pour indiquer le l'instruction MSG a été exécutée avec succès. Lorsqu'une erreur est détectée et que le bit Error est à 1, les autres bits de statut (EN/EW/ST) ne passent pas à 0.
5	CIPCONN	Done	Défini lorsque la connexion CIP pour la communication est coupée. Le bit CIPCONN s'applique lorsque ConnClose est défini sur True ; pour les autres cas, le bit CIPCONN est défini sur False. Le bit CIPCONN est également utilisé pour les connexions Ethernet, série et USB.
6	EIPSESS	Done	Défini lorsque la session CIP d'encapsulation pour la communication est fermée. Le bit EIPSESS s'applique lorsque ConnClose est défini sur True ; pour les autres cas, le bit EIPSESS est défini sur False. Ce bit est uniquement utilisé pour Ethernet.

Codes d'erreur CIPSTATUS

Ce tableau permet de déterminer les valeurs de paramètre pour les champs ErrorID et SubErrorID du paramètre CIPSTATUS lorsque le bit ER est défini.

Code ErrorID	SubErrorID	Description du code d'erreur
33	Erreurs relatives à la configuration du paramètre	
	32	Numéro de canal incorrect.
	36	Type de connexion CIP non pris en charge.
	40	Type de données symbolique CIP non pris en charge.
	41	Nom de symbole CIP non valide.
	43	Valeur de classe CIP ou nombre de MemberID non pris en charge.
	48	Taille de tableau de données d'entrée du bloc d'instructions insuffisante.
	49	Chemin d'accès à la cible non valide.
	50	Code de service incorrect.
	51	Taille de tableau de données de transmission du bloc d'instructions trop élevée pour la communication CIP. la longueur maximale des données utilisateur à transmettre varie en fonction de la configuration du message. Si la charge utile totale du message CIP (y compris la surcharge du message CIP et des données utilisateur) est supérieure à 504 octets, une erreur 0x21 (subError 0x33) est signalée.
	52	Valeur de type de segment incorrect.
	53	Valeur de temps dépassé UCCM incorrecte. Si la valeur de temps dépassé d'encapsulation est inférieure au temps dépassé UCCM ou que la différence entre le temps dépassé d'encapsulation et le temps dépassé UCCM est inférieure ou égale à une seconde, une erreur 0x21 (subError 0x35) est signalée.
	54	Valeur de temps dépassé de connexion incorrecte. Si la valeur de temps dépassé d'encapsulation est inférieure au temps dépassé du message CONNECTED ou que la différence entre le temps dépassé d'encapsulation et le temps dépassé du message CONNECTED est inférieure ou égale à une seconde, une erreur 0x21 (subError 0x36) est signalée.
55	Erreurs de temporisation	
	112	Le délai du message a expiré pendant que celui-ci se trouvait dans la file d'attente.
	113	Le délai du message a expiré pendant que celui-ci attendait l'établissement de la connexion à la couche liaison.
	114	Le délai du message a expiré pendant que celui-ci attendait la transmission vers la couche liaison.
	115	Le délai du message a expiré pendant que celui-ci attendait une réponse de la couche liaison.
69	Codes d'erreur relatifs au format de réponse du serveur	
	65	La réponse au message ne correspond pas à la demande.
	68	Le type de données de la réponse au message n'est pas valide/pris en charge (MSG_CIPSYMBOLIC).
208	Aucune adresse IP n'a été configurée pour le réseau.	

Code ErrorID	SubErrorID	Description du code d'erreur
209	Le nombre maximal de connexions autorisé a été atteint ; aucune connexion n'est disponible.	
210	Adresse Internet ou adresse de station non valide.	
217	L'exécution du message a été annulée par l'utilisateur (le paramètre Cancel a été défini sur TRUE).	
218	Aucun espace disponible dans la mémoire tampon du réseau.	
222	Réservé.	
223	L'adresse du lien n'est pas disponible. Un changement de configuration TCP/IP ou Ethernet est en cours.	
224	Code d'erreur de réponse CIP. SubErrorID spécifie le statut CIP et ExtErrorID spécifie la valeur du statut prolongé CIP. Consultez la spécification CIP pour connaître les valeurs de code d'erreur possibles.	
255	Le canal est fermé ou une reconfiguration est en cours. Le code d'erreur est émis immédiatement après la mise sous tension et jusqu'à ce que la connexion soit établie ; il s'agit d'un comportement normal. Il peut aussi se produire dans l'une des situations suivantes : • un câble Ethernet est déconnecté • une adresse IP ne peut pas être détectée • Un module enfichable de port série est présent, mais n'est pas configuré	

Type de données CIPTARGETCFG

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données CIPTARGETCFG.

Paramètre	Type de données	Description
Path	STRING[80]	Chemin d'accès de la cible. Vous pouvez spécifier au maximum deux tronçons. La syntaxe du chemin d'accès est la suivante : • {"<port>,<node/slot address>"}2
CipConnMode	USINT	Type de connexion CIP. • 0 - Non connecté (par défaut) • 1 - Connexion Class3
UcmmTimeout	UDINT	Délai de temporisation du message non connecté (en millisecondes). Temps d'attente pour obtenir une réponse aux messages non connectés, y compris le délai d'établissement de la connexion pour les messages connectés. • Valeurs autorisées : de 250 à 10 000 ms • Définissez sur 0 pour utiliser la valeur par défaut de 3 000 ms (3 secondes). • Toute valeur spécifiée inférieure à 250 ms sera ramenée automatiquement à 250 ms (valeur minimum). • Toute valeur spécifiée supérieure à 10 000 ms sera ramenée automatiquement à 10 000 ms (valeur maximum).
ConnMsgTimeout	UDINT	Délai de temporisation de connexion Class3 (en millisecondes). Temps d'attente pour obtenir une réponse aux messages connectés. La connexion est coupée lorsque le délai de temporisation expire. • Valeurs autorisées : de 800 à 10 000 ms • Définissez sur 0 pour utiliser la valeur par défaut de 10 000 ms (10 secondes). • Toute valeur spécifiée inférieure à 800 ms sera ramenée automatiquement à 800 ms (valeur minimum). • Toute valeur spécifiée supérieure à 10 000 ms sera ramenée automatiquement à 10 000 ms (valeur maximum).
ConnClose	BOOL	Comportement de coupure de connexion : • TRUE - Coupe la connexion lorsque le message est terminé. • FALSE - Ne coupe pas la connexion lorsque le message est terminé (par défaut).

Chemin d'accès cible pour la messagerie CIP

Le chemin d'accès cible pour la messagerie CIP contient des paramètres qui déterminent le chemin et la destination des messages CIP.

Le paramètre de chaîne du chemin d'accès cible utilise la syntaxe suivante :

- "<port local>, <adresse de la 1re cible>, [<port local de la 1re cible>, <adresse de la 2e cible>]"

Le 1er tronçon est indispensable, le 2e facultatif.

Élément de chaîne	Description
Port local	Port local utilisé pour envoyer le message. Il doit s'agir d'un port série CIP ou EtherNet/IP actif. Les ports USB ne sont pas pris en charge.
Adresse de la 1re cible	Adresse cible du 1er tronçon. • Dans le cas d'EIP, spécifiez l'adresse IP de la cible. Il doit s'agir d'une adresse de monodiffusion et non d'une adresse de multidiffusion, d'une adresse de diffusion, d'une adresse locale ou d'une adresse de boucle (127.x.x.x) ; en outre, l'adresse IP ne peut être égale à 0. • Dans le cas d'un port série CIP, spécifiez l'adresse de nœud de la cible. La valeur prise en charge est 1.
Port local de la 1re cible	Port local utilisé pour envoyer le message.
Adresse de la 2e cible	Adresse cible du 2e tronçon.

Exemple de chemin d'accès de la cible

Le tableau suivant répertorie des exemples de valeurs utilisées dans une chaîne de chemin d'accès de cible et décrit le résultat de chacune de ces chaînes.

Élément de chaîne	Description
"0,0"	L'appareil cible est l'appareil local.
"6,1"	Cible le Nœud 1 via le Port 6 (port série UPM Micro830).
"4,192.168.1.100"	Cible le Nœud 192.168.1.100 via le Port 4 (port Ethernet intégré Micro850).
"4,192.168.1.100,1,0"	Cible le Nœud 192.168.0.100 (module ENET Logix) via le Port 4 (port Ethernet intégré Micro850). Cible le contrôleur Logix au niveau de l'emplacement 0 depuis le module ENET, via le port Fond de panier (Port 1).

Connexions de message CIP/EIP

Un maximum de 16 connexions CIP (classe 3) et 16 connexions EIP est pris en charge pour l'exécution du message client. Le tableau suivant décrit le comportement de connexion CIP/EIP.

Scénario	Résultats
La demande de message est activée et CipConnMode=1.	S'il n'existe pas de connexion à la cible, une connexion CIP est établie. S'il existe déjà une connexion à la cible, la connexion CIP existante est utilisée.
La demande de message est activée, CipConnMode=1 et le port local du message est de type Ethernet.	S'il n'existe aucune connexion EIP à la cible, une connexion EIP est établie avant l'établissement d'une connexion CIP.
La demande de message est activée, CipConnMode=0 et le port local du message est de type Ethernet.	S'il n'existe aucune connexion EIP à la cible, une connexion EIP est établie.

Scénario	Résultats
L'exécution du message est terminée et ConnClose est défini sur Vrai.	S'il existe une seule connexion à la cible, la connexion est fermée. S'il existe plusieurs connexions pour la cible, la connexion est coupée lorsque l'exécution du dernier message se termine. Lorsqu'une connexion CIP est coupée, toute connexion EIP associée est également coupée. Si plusieurs connexions CIP utilisent la même connexion EIP, celle-ci sera coupée une fois toutes les connexions CIP associées coupées.
Lorsque ConnClose est défini sur True, la connexion de message est coupée et la session EIP est fermée à la fin de l'exécution du message.	Si plusieurs messages partagent la même connexion, la connexion est coupée à la fin de l'exécution du dernier message.
Une connexion CIP ou EIP qui n'est pas associée à un message actif est coupée si elle est en attente pendant x secondes. Où x correspond à une valeur de délai d'inactivité d'encapsulation configurable qui peut être réinitialisée à l'aide du service CIP défini.	Reportez-vous aux objets TCP/IP du Volume II de la spécification CIP pour en savoir plus sur le service CIP défini.
L'exécution du message est terminée et ConnClose est défini sur Faux.	La connexion n'est pas fermée.
La connexion n'est pas associée à un message actif et reste en attente pendant le délai spécifié dans le paramètre ConnTimeOut.	La connexion est coupée.
Le contrôleur passe d'un mode d'exécution (Exécution, Exécution à distance, Scrutation unique pour test à distance et Ligne unique distante) à un mode sans exécution.	La coupure est forcée pour l'ensemble des connexions.

Temporisateurs de message CIP

Le tableau suivant décrit la manière dont les temporisateurs des paramètres de temporisation CIPTARGETCFG (UcmmTimeout et ConnMsgTimeout) se comportent, suivant le statut et la demande de message.

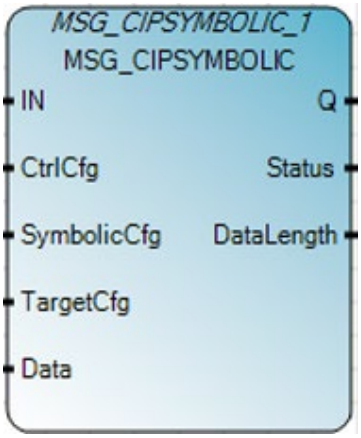
Action	Résultats
Le message est activé.	Le temporisateur UcmmTimeout est activé.
La connexion est demandée.	Le temporisateur ConnMsgTimeout est activé.
Le temporisateur ConnMsgTimeout est activé.	Le temporisateur UcmmTimeout est activé.
La demande de connexion est finalisée.	Le temporisateur UcmmTimeout est réactivé.

MSG_CIPSYMBOLIC (message symbolique du protocole CIP)

Envoie un message symbolique CIP (common industrial protocol) via un canal Ethernet ou un port série.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



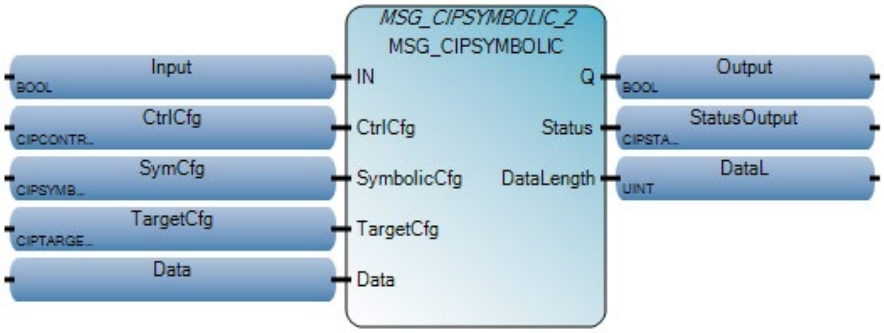
Exécution de MSG_CIPSYMBOLIC

Lorsque le bloc fonctionnel est activé, les mémoires tampon de réception pour les opérations de lecture sont effacées sur le front montant d'Enable.

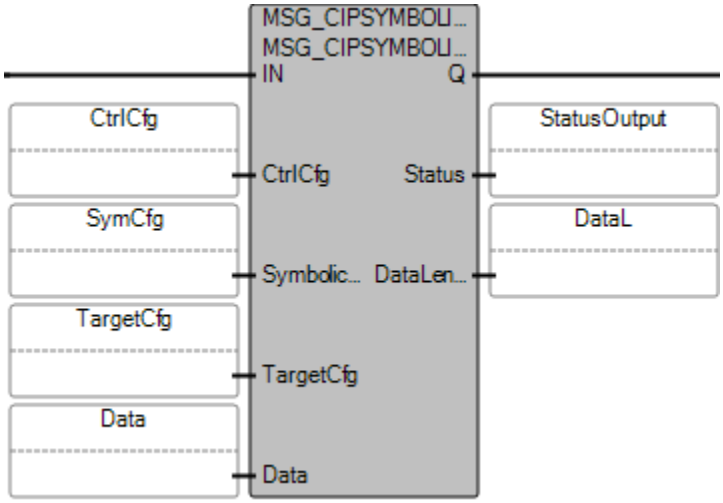
Arguments

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : front montant non détecté, en attente.
CtrlCfg	Entrée	CIPCONTROLCFG	Configuration du contrôle d'exécution du bloc d'instruction Utilisez les paramètres de type de données CIPCONTROLCFG sur la page 166 pour définir CtrlCfg.
SymbolicCfg	Entrée	CIPSYMBOLICCFG sur la page 173	Informations sur le symbole de lecture/écriture.
TargetCfg	Entrée	CIPTARGETCFG	Configuration de l'appareil cible. Utilisez les paramètres de type de données CIPTARGETCFG sur la page 169 pour définir TargetCfg.
Data	Entrée	USINT[490]	La commande de lecture stocke les données renvoyées par le serveur. La commande d'écriture met en mémoire tampon les données à envoyer au serveur. Lorsque l'envoi d'un MSG est déclenché ou redéclenché, les données sont effacées pour la commande de lecture MSG.
Q	Sortie	BOOL	Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : l'instruction MSG s'est terminée correctement. FAUX : l'instruction MSG n'est pas terminée.
Status	Sortie	CIPSTATUS	État d'exécution du bloc de fonctions Lorsqu'une instruction MSG est déclenchée ou redéclenchée, tous les éléments d'état sont réinitialisés. La sortie d'état est définie dans le type de données CIPSTATUS sur la page 167.
DataLength	Sortie	UDINT	Nombre d'octets de données pour le service de lecture. Pour le service d'écriture : 0. Lorsqu'une instruction MSG est déclenchée ou redéclenchée, la valeur DataLength est remise à zéro pour la commande de lecture MSG.

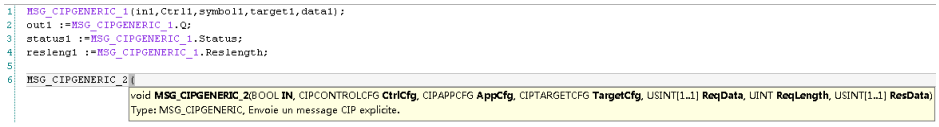
Exemple de diagramme de bloc de fonctions MSG_CIPSYMBOLIC



Exemple de langage à relais MSG_CIPSYMBOLIC



Exemple de texte structuré MSG_CIPSYMBOLIC



Type de données CIPSYMBOLICCFG

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données CIPSYMBOLICCFG.

Paramètre	Type de données	Description
Service	USINT	Code de service : • 0 - Lecture (par défaut) • 1 - Écriture
Symbol	STRING	Nom de la variable à lire/écrire. • Le nom ne doit pas dépasser 80 caractères. • Le champ ne peut être vide. Syntaxe des symboles définie dans la syntaxe de lecture/d'écriture symbolique sur la page 174.
Count	UINT	Nombre d'éléments de variable à lire/écrire : • Valeurs autorisées : 1 - 490 • 1 est utilisé si la valeur est définie sur 0.

Paramètre	Type de données	Description
Type	Défini par l'utilisateur	Type de données défini par l'utilisateur pour la variable ciblée. Type défini dans la prise en charge des types de données symboliques.
Offset	USINT	Réservé pour une utilisation future. Décalage d'octet pour la variable de lecture/d'écriture utilisée pour lire/écrire une variable de grande taille, ne pouvant être traitée dans un seul message. • 0 – 0xFF
Réservé pour une utilisation future.		

Prise en charge des types de données symboliques

Utilisez ce tableau pour déterminer les types de données pris en charge par MSG_CIPSYMBOLIC.

Type de données	Valeur de type de données (hexadécimale)	Description
BOOL	193 (0xC1)	Expression booléenne logique avec les valeurs TRUE (1) ou FALSE (0)
SINT	194 (0xC2)	Valeur d'entier signé 8 bits
INT	195 (0xC3)	Valeur d'entier signé 16 bits
DINT	196 (0xC4)	Valeur d'entier signé 32 bits
LINT	197 (0xC5)	Valeur d'entier signé 64 bits
USINT	198 (0xC6)	Valeur d'entier non signé 8 bits
UINT	199 (0xC7)	Valeur d'entier non signé 16 bits
UDINT	200 (0xC8)	Valeur d'entier non signé 32 bits
ULINT	201 (0xC9)	Valeur d'entier non signé 64 bits
REAL	202 (0xCA)	Valeur à virgule flottante 32 bits
LREAL	203 (0xCB)	Valeur à virgule flottante 64 bits
STRING	218 (0xDA)	Chaîne de caractères

Syntaxe de l'écriture/lecture symbolique

La syntaxe définit les combinaisons de symboles d'un bloc d'instruction en lecture/écriture valide.

Noms de symboles valides

Pour être valide, chaque nom de symbole doit remplir les conditions suivantes :

- Commencer par une lettre ou un trait de soulignement suivi d'une lettre, d'un chiffre ou d'un seul trait de soulignement.
- Être composé de 40 caractères au maximum.
- Ne pas contenir deux traits de soulignement consécutifs.
- Utiliser des caractères spéciaux [ ] . , comme séparateurs.

Syntaxe des symboles

Ce tableau permet de définir la syntaxe valide pour les symboles. seules les variables globales sont prises en charge.

Symbole	Syntaxe	Exemple
Variable	PROGRAM:<nom du programme>,<nom du symbole>	PROGRAM:POU1.MyTag
Tableau	<nom du symbole>[dim3, dim2, dim1] La dimension maximale prise en charge est 3.	MyTag1[0] MyTag2[3,6] MyTag3[1,0,4]
Structure	<nom du symbole>.<nom du symbole du champ struct>	MyTag4.time.year MyTag5.local.time[1].year

Taille des paquets de données prise en charge pour la fonction Série CIP

Pour les contrôleurs Micro830, Micro850 et Micro870, le port série intégré et les ports série enfichables peuvent prendre en charge la communication CIP en série. Un paquet de données de communication série CIP est constitué de données utilisateur et d'un en-tête.

Lors d'un fonctionnement en tant que client série CIP, les ports série d'un Micro830/Micro850 peuvent prendre en charge au maximum 490 octets de données utilisateur en lecture/écriture. Ce maximum est possible pour les paquets de données série CIP dont l'en-tête est réduit au minimum. Si la taille de l'en-tête d'un paquet dépasse ce minimum, la taille maximale de données utilisateur que le client CIP peut prendre en charge est inférieure à 490 octets. Si la taille du paquet de données est supérieure à la taille de données maximale prise en charge par le client CIP, le bloc de fonctions signale une erreur (0x21) et une sous-erreur (0x33).

Lors d'un fonctionnement en tant que serveur CIP en série, les ports série Micro830, Micro850 et Micro870 peuvent prendre en charge au moins 255 octets de données utilisateur en lecture/écriture. Ce minimum s'applique aux paquets de données série CIP avec un en-tête de taille maximale. Si la taille de l'en-tête du paquet CIP est inférieure à ce maximum, le client CIP peut prendre en charge des paquets de données de plus de 255 octets. Cependant, si la taille des données utilisateur est supérieure à la taille de données maximale prise en charge par la fonction serveur CIP, le paquet de données CIP peut être abandonné et le délai de communication du client arriver à expiration.

IMPORTANT Pour la fonction de serveur série CIP, il est recommandé de ne pas lire/écrire plus de 255 octets de données utilisateur dans chaque message CIP.

MSG_MODBUS (message Modbus)

Envoie un message Modbus via un port série.

Détails de l'opération :

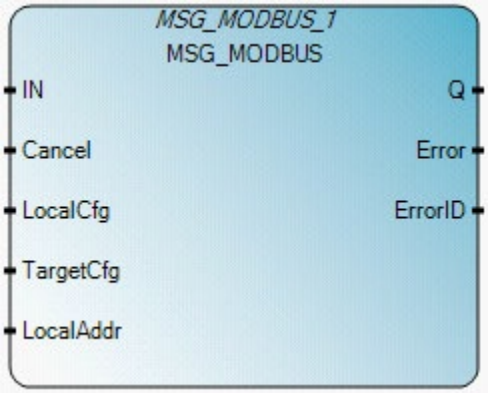
- Au maximum quatre demandes de message par canal peuvent être

traitées au cours de chaque scrutation. Pour les programmes de schémas à relais, les demandes de message sont exécutées à la fin de la scrutation à relais.

- si le déclenchement est réglé sur « en continu », les codes d'erreur sont également effacés en continu. Pour afficher des codes d'erreur, ajoutez une ligne avant l'instruction MSG_MODBUS.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : front montant non détecté, pas démarré.
Cancel	Entrée	BOOL	VRAI : annulation de l'exécution du bloc d'instruction. FAUX : lorsque IN est VRAI. L'annulation de la saisie est dominante.
LocalCfg	Entrée	MODBUSLOCPARA	Définit l'entrée de la structure (appareil local). Définit la structure d'entrée de l'appareil local à l'aide du type de données MODBUSLOCPARA sur la page 178.
TargetCfg	Entrée	MODBUSTARPARA	Définit l'entrée de la structure (appareil cible). Définit la structure d'entrée de l'appareil cible à l'aide du type de données MODBUSTARPARA sur la page 181.
LocalAddr	Entrée	MODBUSLOCADDR	MODBUSLOCADDR correspond à un tableau de 125 mots utilisé d'une part par les commandes de lecture pour stocker les données (1 à 125 mots) renvoyées par l'esclave Modbus, et d'autre part par les commandes d'écriture pour mettre en mémoire tampon les données (1 à 125 mots) à transmettre à l'esclave Modbus.
Q	Sortie	BOOL	Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : l'instruction MSG s'est terminée correctement. FAUX : l'instruction MSG n'est pas terminée.

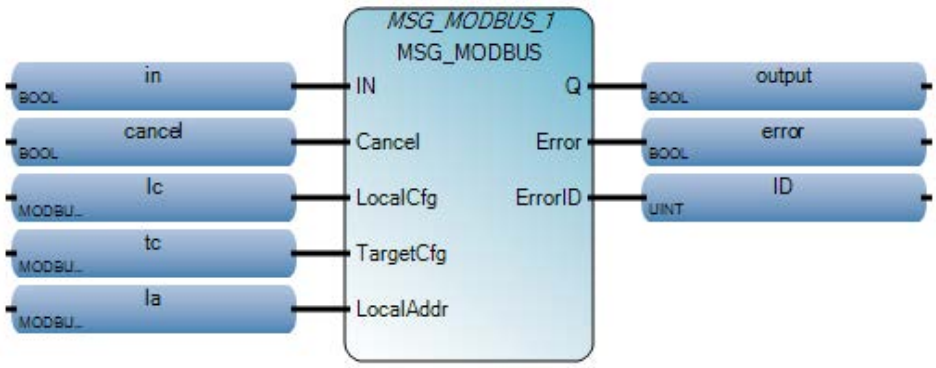
Paramètre	Type de paramètre	Type de données	Description
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur MSG_MODBUS.

Codes d'erreur MSG_MODBUS

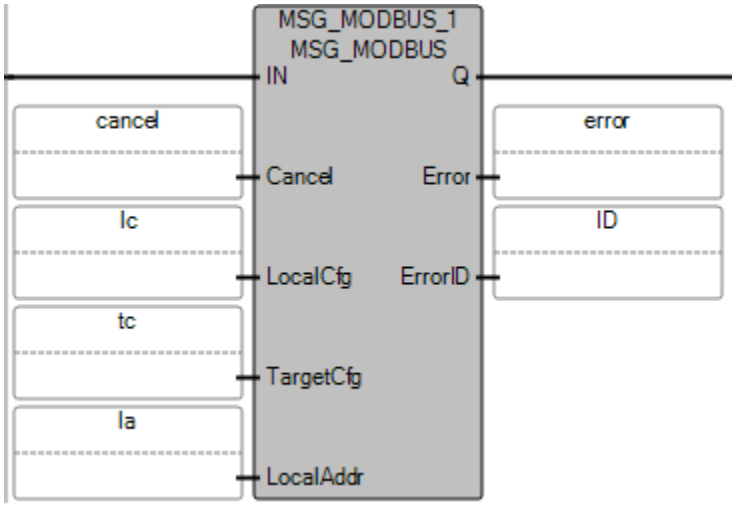
Ce tableau décrit les codes d'erreur MSG_MODBUS.

Code d'erreur	Description de l'erreur
3	La valeur de TriggerType est passée de 2 à 255.
20	Le pilote de communication locale est incompatible avec l'instruction MSG.
21	Il existe une erreur au niveau du paramètre de configuration du canal local.
22	L'adresse cible ou de pont local est supérieure à l'adresse de nœud maximale.
33	Un paramètre de fichier MSG est incorrect.
54	Un modem a été perdu.
55	Le délai du message a expiré dans le processeur local. Le délai pour une couche de lien a expiré.
217	L'utilisateur a annulé le message.
129	Fonction illégale.
130	Adresse de données illégale.
131	Valeur de données illégale.
132	Échec de l'appareil esclave.
133	Accusé de réception.
134	L'appareil esclave est occupé.
135	Accusé de réception négatif.
136	Erreur de parité dans la mémoire.
137	Réponse non standard.
255	Le canal a été fermé.

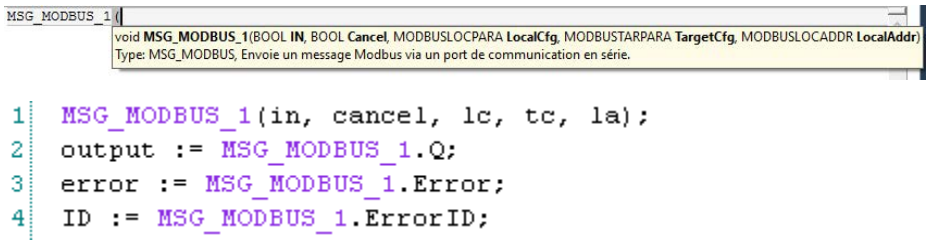
Exemple de diagramme de bloc de fonctions MSG_MODBUS



Exemple de langage à relais MSG_MODBUS



Exemple de texte structuré MSG_MODBUS



Type de données MODBUSLOCPARA

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données MODBUSLOCPARA.

Paramètre	Type de données	Description
Channel	UINT	Numéro du port série Micro800 PLC : • 2 pour le port série intégré ou • 5-9 pour les ports série des modules enfichables installés sur les emplacements 1 à • 5 pour l'emplacement 1 ; • 6 pour l'emplacement 2 ; • 7 pour l'emplacement 3 ; • 8 pour l'emplacement 4 ; • 9 pour l'emplacement 5 ;
TriggerType	USINT	Représente l'une des valeurs suivantes : • 0: envoi de message déclenché une fois (lorsque IN passe de Faux à Vrai) • 1: envoi de message déclenché de façon continue lorsque IN est défini sur Vrai • Autre valeur : Réservé

Paramètre	Type de données	Description
Cmd	USINT	Représente l'une des valeurs suivantes : • 01: Lecture du statut de la bobine (0xxxx) • 02: Lecture du statut de l'entrée (1xxxx) • 03: Lecture du registre de conservation (4xxxx) • 04: Lecture des registres d'entrée (3xxxx) • 05: Écriture d'une seule bobine (0xxxx) • 06: Écriture d'un seul registre (4xxxx) • 15: Écriture de plusieurs bobines (0xxxx) • 16: Écriture de plusieurs registres (4xxxx) • Autres : Prise en charge de la commande personnalisée. Prise en charge de la commande personnalisée MODBUSLOCPARA : Les commandes personnalisées comprises dans la plage 0-255 dont les valeurs ne sont pas encore attribuées à une commande Modbus sont également prises en charge. Si une commande personnalisée est utilisée, la commande LocalCfg:ElementCnt contient le nombre d'octets reçus. La réponse est reçue dans au niveau des données relatives à l'adresse locale et écrase les données ayant fait l'objet de la demande. • Exemple pour CMD=0x2B • Données d'adresse locale 1:0x0E, READ_DEVICE_ID_MEI • Données d'adresse locale 2:0x01, READ_DEV_ID_BASIC • Données d'adresse locale 3:0x00, lecture de l'objet fournisseur
ElementCnt	UINT	Limites • Pour la lecture de la bobine et des entrées discrètes : 2000 bits • Pour la lecture des registres : 125 mots • Pour l'écriture de la bobine : 1968 bits • Pour l'écriture du registre : 123 mots

Déclenchement de message MSG_MODBUS

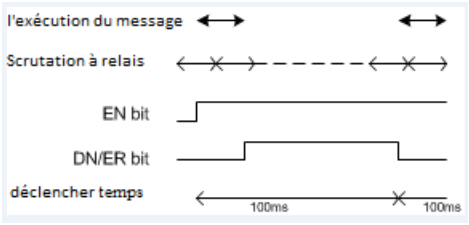
Un message Modbus peut être déclenché périodiquement en définissant le paramètre TriggerType sur une valeur différente de zéro.

Ce tableau décrit le comportement du paramètre TriggerType lorsqu'il est utilisé avec le bloc de fonctions MSG_MODBUS sur la page 175.

Action	Résultats
Le message est activé.	Le temporisateur de déclenchement démarre.
Le temporisateur de déclenchement arrive à expiration avant que le message soit terminé.	Le message est immédiatement déclenché lors du cycle de scrutation de relais suivant.
Le message se termine avant l'expiration du délai de déclenchement	Le message est déclenché avant l'expiration du délai de déclenchement

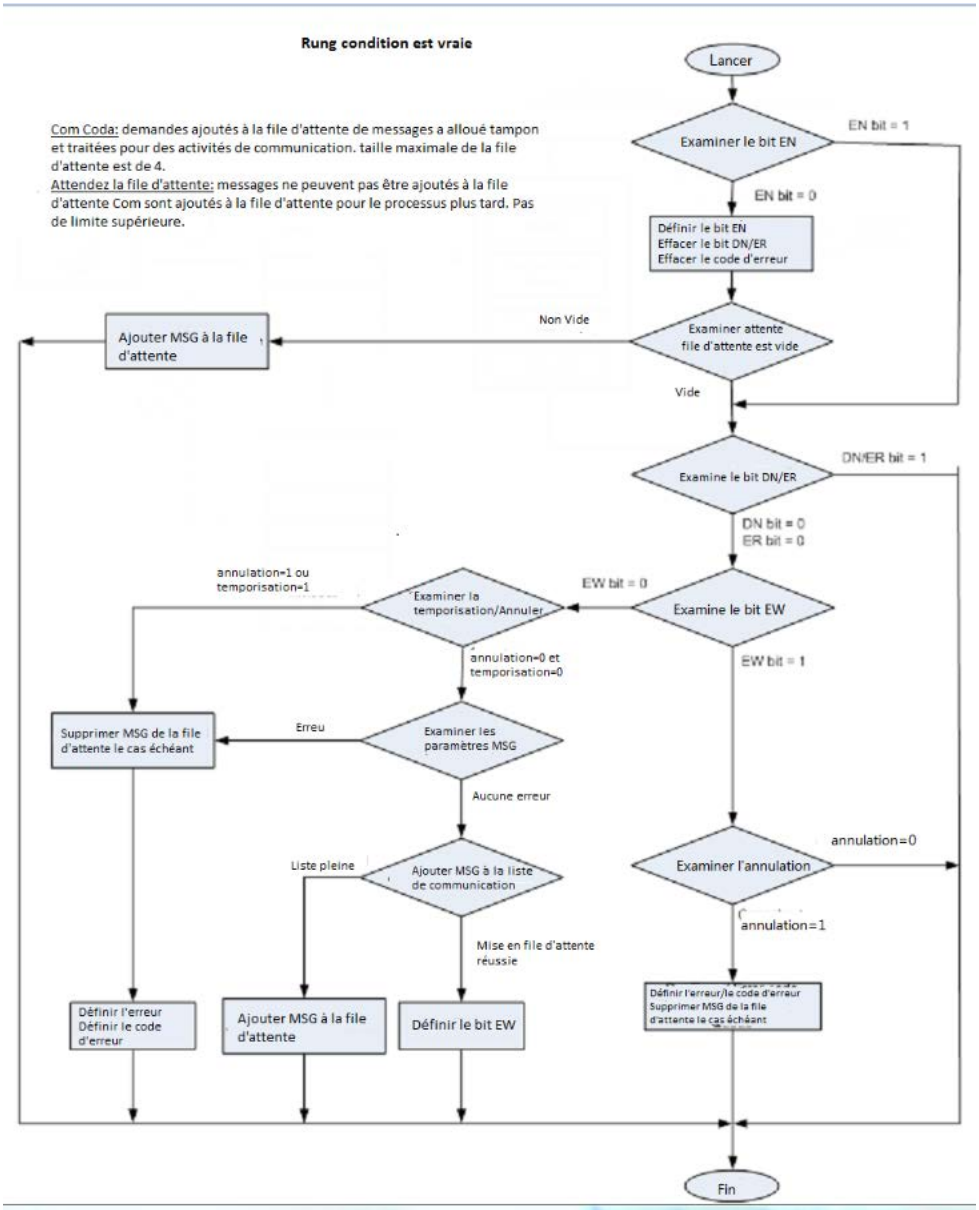
Exemple : déclenchement de message

Dans l'exemple suivant, la valeur TriggerType est définie sur 100.



Processus d'exécution des messages (Ligne = TRUE)

Le diagramme de processus suivant décrit les événements d'instruction de message qui surviennent lorsque la condition de ligne est définie sur Vrai.



File d'attente de communication : Les demandes de message ajoutées à la file d'attente de communication (Com) disposent d'une mémoire tampon dédiée et sont traitées par la tâche de communication. Le nombre d'éléments maximum dans cette file d'attente est de 4.

File d'attente : les messages qui ne peuvent pas être ajoutés à la liste de communication sont placés dans la file d'attendre pour être traités ultérieurement. Le nombre de messages contenus dans la file d'attente n'est pas limité.

Type de données MODBUSTARPARA

Le tableau suivant décrit le type de données MODBUSTARPARA.

Paramètre	Type de données	Description
Addr	UDINT	Adresse des données cible (1 - 65 536). Diminue de un à chaque envoi.
Node	USINT	L'adresse du nœud esclave par défaut est 1. La plage se situe entre 1 et 247. Zéro correspond à l'adresse de diffusion Modbus et n'est valide que pour les commandes d'écriture Modbus (par exemple, 5, 6, 15 et 16).

MSG_MODBUS2 (message MODBUS/TCP)

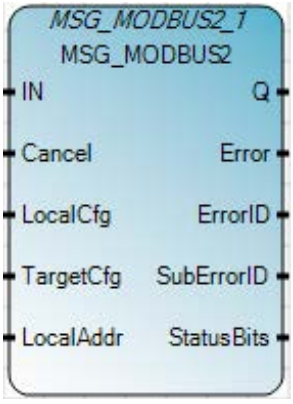
Envoie un message MODBUS/TCP via un canal Ethernet.

Détails de l'opération :

- Au maximum quatre demandes de message par canal peuvent être traitées au cours de chaque scrutation. Pour les programmes de schémas à relais, les demandes de message sont exécutées à la fin de la scrutation à relais.
- Lorsque MSG_MODBUS2 est activé, les mémoires tampon de réception pour les opérations de lecture sont effacées sur le front montant d'Enable.
- L'annulation de l'exécution de l'instruction MSG_MODBUS2 ne garantit pas que la demande de message sortante est abandonnée, mais elle garantit que la réponse n'est pas traitée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : front montant non détecté, en attente.
Cancel	Entrée	BOOL	VRAI : annulation de l'exécution du bloc d'instruction. L'annulation de l'exécution de l'instruction MSG_MODBUS2 ne garantit pas que la demande de message sortante est abandonnée, mais elle garantit que la réponse n'est pas traitée. FAUX : lorsque IN est VRAI. L'annulation de la saisie est dominante.
LocalCfg	Entrée	MODBUS2LOCPARA	Définit l'entrée de la structure (appareil local). Définit la structure d'entrée de l'appareil local à l'aide du type de données MODBUS2LOCPARA sur la page 184.
TargetCfg	Entrée	MODBUS2TARPARA	Définit l'entrée de la structure (appareil cible). Définit la structure d'entrée de l'appareil cible à l'aide du type de données MODBUS2TARPARA sur la page 185.
LocalAddr	Entrée	MODBUSLOCADDR	Le type de données MODBUSLOCADDR est un tableau de 125 mots. Utilisation de LocalAddr : • Pour les commandes de lecture, stockez les données (1 à 125 mots) renvoyées par l'esclave Modbus. • Pour les commandes d'écriture, mettez en mémoire tampon les données (1 à 125 mots) à envoyer à l'esclave Modbus.
Q	Sortie	BOOL	Les sorties de cette instruction sont mises à jour de façon asynchrone par la scrutation du programme. La sortie Q ne peut pas être utilisée pour déclencher de nouveau l'instruction, car IN est déclenchée par un front. VRAI : l'instruction MSG s'est terminée correctement. FAUX : l'instruction MSG n'est pas terminée.
Error	Sortie	BOOL	Indique qu'une erreur a été détectée. VRAI : une erreur s'est produite. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs pour cette instruction sont définies dans les codes d'erreur Modbus2.
SuberrorID	Sortie	UINT	Est utilisé pour vérifier les bits d'état : • Bit 0 : EN – Enable • Bit 1 : EW – Enable Wait • Bit 2 : ST – Start • Bit 3 : ER – Error • Bit 4 : DN – Done Les autres bits sont réservés.
StatusBits	Sortie	UINT	Valeur de code SubError lorsque Error est défini sur TRUE. Lorsqu'une instruction MSG est déclenchée ou redéclenchée, le jeu SubErrorID précédent est effacé.

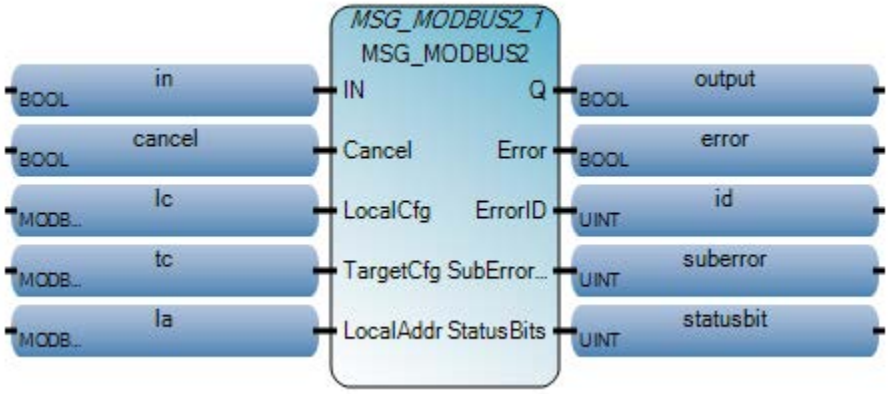
Codes d'erreur et sous-erreur MSG_MODBUS2

Lorsque le bit ER est défini, les champs ErrorID et SubErrorID affichent les codes d'erreur suivants.

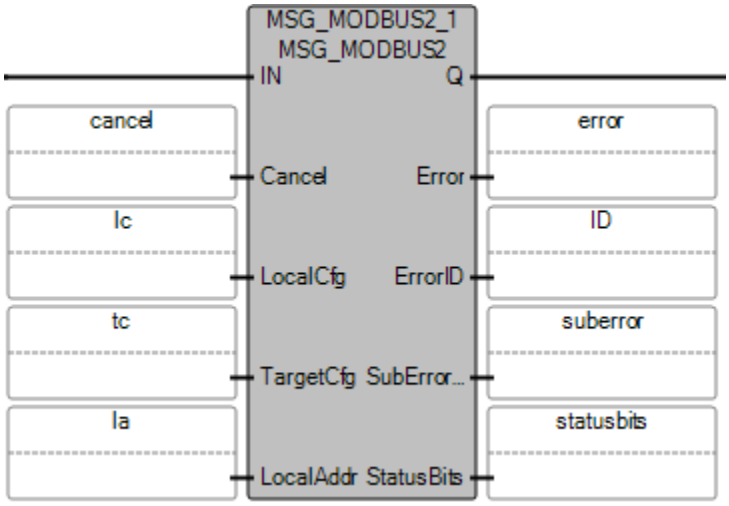
ID d'erreur	SubErrorID	Description
33	Erreurs relatives à la configuration du paramètre	

ID d'erreur	SubErrorID	Description
	32	Numéro de canal incorrect.
	37	Nombre d'éléments incorrect.
	38	Adresse de données incorrecte.
55	**Erreurs de temporisation**	
	112	Le délai du message a expiré pendant que celui-ci se trouvait dans la file d'attente.
	113	Le délai du message a expiré pendant que celui-ci attendait l'établissement de la connexion à la couche liaison.
	114	Le délai du message a expiré pendant que celui-ci attendait la transmission vers la couche liaison.
	115	Le délai du message a expiré pendant que celui-ci attendait une réponse de la couche liaison.
69	**Codes d'erreur relatifs au format de réponse du serveur.**	
208	Aucune adresse IP n'a été configurée pour le réseau.	
209	Le nombre maximal de connexions autorisé a été atteint ; aucune connexion n'est disponible.	
210	Adresse Internet ou adresse de station non valide.	
217	L'exécution du message a été annulée par l'utilisateur (le paramètre Cancel a été défini sur TRUE).	
222	Impossible d'établir la connexion réseau avant la période de temps dépassé.	
255	Le canal est fermé ou une reconfiguration est en cours. Le code d'erreur est émis immédiatement après la mise sous tension et jusqu'à ce que la connexion soit établie ; il s'agit d'un comportement normal. Cela peut se produire lorsqu'un câble Ethernet est débranché ou qu'aucune adresse IP n'a été détectée.	
	Code d'erreur de réponse d'esclave.	
129	Code de fonction incorrect	
130	Adresse de données incorrecte	
131	Valeur de données incorrecte	
132	Panne de serveur	
133	Accusé de réception	
134	Accusé de réception négatif	
136	Erreur de parité dans la mémoire	
137	Code d'erreur de réponse non standard Vous trouverez le code d'erreur actuel dans le SubErrorID.	

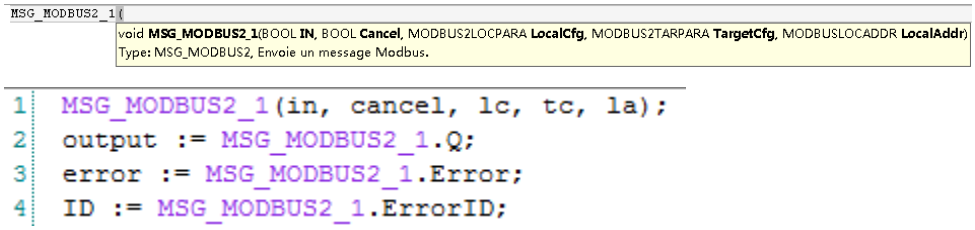
Exemple de diagramme de bloc de fonctions MSG_MODBUS2



Exemple de langage à relais MS_MODBUS2



Exemple de texte structuré MSG_MODBUS2



Type de données MODBUS2LOCPARA

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données MODBUS2LOCPARA.

Paramètre	Type de données	Description
Channel	UINT	Numéro de port Ethernet local : • 4 pour le port Ethernet intégré du Micro850 et du Micro820
TriggerType	UDINT	Type de déclenchement de message : • 0: envoi de message déclenché une fois (lorsque IN passe de Faux à Vrai) • 1 à 65535 - Valeur de déclenchement cyclique en millisecondes. Message déclenché périodiquement lorsque IN est défini sur Vrai et que l'exécution du message précédent est terminée. • Définissez la valeur sur 1 pour déclencher les messages le plus rapidement possible. Voir ci-dessous le déclenchement de message MSG_MODBUS2.
Cmd	USINT	Commande Modbus : • 01: Lecture du statut de la bobine (0xxxx) • 02: Lecture du statut de l'entrée (1xxxx) • 03: Lecture du registre de conservation (4xxxx) • 04: Lecture des registres d'entrée (3xxxx) • 05: Écriture d'une seule bobine (0xxxx) • 06: Écriture d'un seul registre (4xxxx) • 15: Écriture de plusieurs bobines (0xxxx) • 16: Écriture de plusieurs registres (4xxxx) • Autres : Prise en charge de la commande personnalisée Prise en charge de la commande personnalisée MODBUS2LOCPARA :

Paramètre	Type de données	Description
		Les commandes personnalisées comprises dans la plage 0-255 dont les valeurs ne sont pas encore attribuées à une commande Modbus sont également prises en charge. Si une commande personnalisée est utilisée, la commande LocalCfg:ElementCnt contient le nombre d'octets reçus. La réponse est reçue au niveau des données relatives à l'adresse locale et écrase les données ayant fait l'objet de la demande. Exemple pour CMD=0x2B : • Données d'adresse locale 1:0x0E, READ_DEVICE_ID_MEI • Données d'adresse locale 2:0x01, READ_DEV_ID_BASIC • Données d'adresse locale 3:0x00, lecture de l'objet fournisseur
ElementCnt	UINT	Limites • Pour la lecture de la bobine et des entrées discrètes : 2000 bits • Pour la lecture des registres : 125 mots • Pour l'écriture de la bobine : 1968 bits • Pour l'écriture du registre : 123 mots

Déclenchement de message MSG_MODBUS2

Un message Modbus peut être déclenché périodiquement en définissant le paramètre TriggerType sur une valeur différente de zéro.

Ce tableau décrit ce qui se produit lors de l'utilisation du paramètre TriggerType avec le bloc de fonctions MSG_MODBUS2 sur la page 181.

Action	Résultats
Le message est activé.	Le temporisateur de déclenchement démarre.
Le temporisateur de déclenchement arrive à expiration avant que le message soit terminé.	Le message est immédiatement déclenché lors du cycle de scrutation de relais suivant.
Le message se termine avant l'expiration du délai de déclenchement.	Le message est déclenché avant l'expiration du délai de déclenchement

Type de données MODBUS2TARPARA

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données MODBUS2TARPARA.

Paramètre	Type de données	Description
Addr	UDINT	Adresse de données Modbus de l'appareil cible : • 1 - 65536 • Diminue de un à chaque envoi. • Le firmware utilise un mot de poids faible pour l'adresse si la valeur de celle-ci est supérieure à 65 536.
NodeAddress[4]	USINT	Adresse IP de l'appareil cible : Il doit s'agir d'une adresse de monodiffusion valide et non d'une adresse de multidiffusion, d'une adresse de diffusion, d'une adresse locale ou d'une adresse de boucle (127.x.x.x) ; en outre, l'adresse IP ne peut être égale à 0. Par exemple, pour spécifier 192.168.2.100 : • NodeAddress[0]=192 • NodeAddress[1]=168 • NodeAddress[2]=2 • NodeAddress[3]=100

Paramètre	Type de données	Description
Port	UINT	Numéro de port TCP cible. Le port Modbus/TCP standard est 502. 1 - 65535 Définir sur 0 pour utiliser la valeur par défaut, 502.
UnitId	USINT	Identifiant de l'unité. Utilisé pour communiquer avec les appareils esclave via un pont Modbus. Consultez la spécification Modbus pour plus de détails. Notez que le Micro800 ne doit pas tenter de valider cette valeur. 0 - 255 Définir sur 255 si l'appareil cible n'est pas un pont.
MsgTimeOut	UDINT	Délai de temporisation du message (en millisecondes). Temps d'attente pour obtenir une réponse dans le cas d'une commande initiée. • 250-10 000 • Définir sur 0 pour utiliser la valeur par défaut, 3 000. • Toute valeur inférieure à 250 (minimum) est définie sur 250. • Toute valeur supérieure à 10 000 (minimum) est définie sur 10 000. Voir Temporisateurs de message Modbus/TCP.
ConnTimeOut	UDINT	Délai de temporisation de l'établissement de la connexion TCP (en millisecondes). Temps d'attente pour établir avec succès une connexion TCP avec l'appareil cible. • 250-10 000 • Définir sur 0 pour utiliser la valeur par défaut, 5 000. • Toute valeur inférieure à 250 (minimum) est définie sur 250. • Toute valeur supérieure à 10 000 (minimum) est définie sur 10 000. Voir Temporisateurs de message Modbus/TCP.
ConnClose	BOOL	Comportement de coupure de connexion TCP. • VRAI : coupe la connexion TCP une fois le message terminé. • FAUX : ne coupe pas la connexion TCP une fois le message terminé [par défaut]. Voir Connexions de message Modbus/TCP.

Temporisateurs de message Modbus/TCP

Ce tableau décrit le comportement de MsgTimeOut et ConnTimeOut en fonction de l'état et des demandes de message.

Action	Résultats
Le message est activé.	Active le temporisateur MsgTimeOut.
La connexion TCP est demandée.	Active le temporisateur ConnTimeOut.
Le temporisateur ConnTimeOut est actif.	Désactive le temporisateur MsgTimeOut.
La demande de connexion est terminée.	Réactive le temporisateur MsgTimeOut.

Connexions de message Modbus/TCP

Le client Modbus/TCP prend en charge un maximum de 16 connexions. Ce tableau décrit le comportement de connexion Modbus/TCP.

Scénario	Résultats
La demande de message est activée et il n'existe aucune connexion avec la cible.	S'il n'existe aucune connexion à la cible, une nouvelle connexion est établie. S'il existe déjà une connexion à la cible, la connexion existante est utilisée.

Scénario	Résultats
L'exécution du message est terminée et ConnClose est défini sur Vrai.	S'il existe une seule connexion à la cible, la connexion est fermée. S'il existe plusieurs connexions pour la cible, la connexion est coupée lorsque l'exécution du dernier message se termine.
L'exécution du message est terminée et ConnClose est défini sur Faux.	La connexion n'est pas fermée.
La connexion n'est pas associée à un message actif et reste inactive pendant le délai spécifié dans le paramètre ConnTimeOut.	La connexion est coupée.
Le contrôleur passe d'un mode d'exécution (Exécution, Exécution à distance, Scrutation unique pour test à distance et Ligne unique distante) à un mode sans exécution.	La coupure est forcée pour l'ensemble des connexions.

MSG_PCCC (message de commandes de communication de contrôleur programmable)

Envoie un message PCCC via un canal Ethernet ou un port série.

Commandes prises en charge : Lecture de type SLC et écriture de type SLC.

Détails de l'opération :

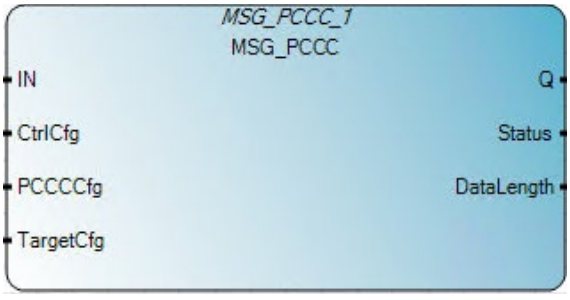
- Lorsque vous envoyez le message PCCC via un canal Ethernet et que plus de quatre demandes de blocs de fonctions sont dans la file d'attente pour CIP_GENERIC_FB, CIP_SYMBOLIC ou PCCC_MSG_FB, jusqu'à quatre messages par canal peuvent être traités au cours d'une seule scrutation.

  Pour les programmes de schémas à relais, les demandes de message sont exécutées à la fin de la scrutation à relais.
- Lorsque vous envoyez le message PCCC sur un port série et que plus de huit demandes de blocs de fonctions sont dans la file d'attente pour les communications série, huit messages au total sur tous les canaux peuvent être traités au cours d'une seule scrutation.

  Pour les programmes de schémas à relais, les demandes de message sont exécutées à la fin de la scrutation à relais.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro870 L70E.

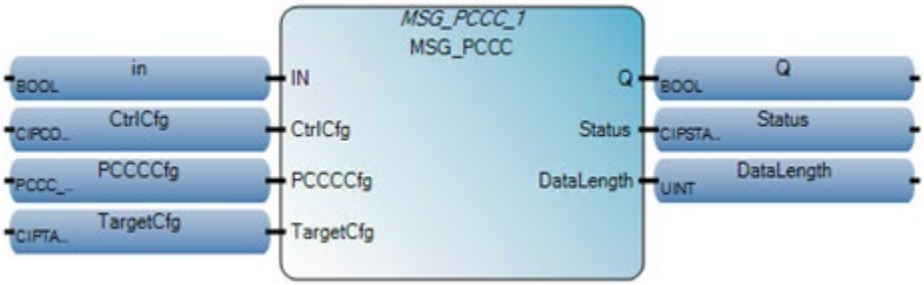


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

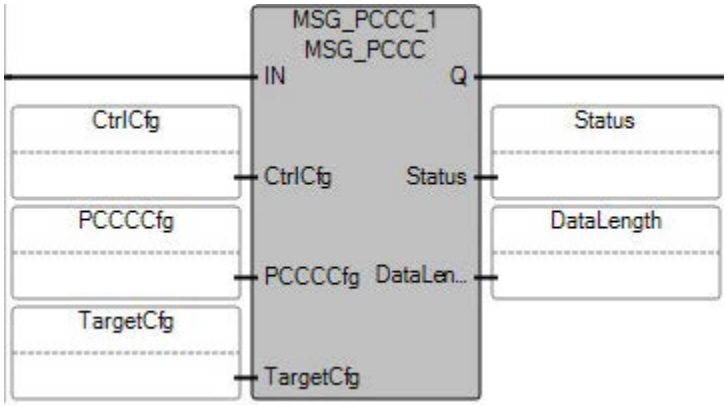
Paramètre	Type de paramètre	Type de données	Description

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	État de l'entrée de l'échelon. VRAI : lorsque le front montant est détecté, démarrez le bloc de fonctions à condition que la dernière opération soit terminée. FAUX : front montant non détecté, pas démarré.
CtrlCfg	Entrée	CIPCONTROLCFG	Configuration du contrôle d'exécution du bloc de fonctions. Utilisez le type de données CIPCONTROLCFG sur la page 189 pour définir CtrlCfg.
PCCCCfg	Entrée	PCCC_CFG	Commandes et adresse PCCC. Utilisez le type de données PCCCCFG sur la page 190 pour définir PCCCCfg.
TargetCfg	Entrée	CIPTARGETCFG	Configuration de l'appareil cible. Utilisez le type de données TARGETCFG sur la page 191 pour définir TargetCfg.
Q	Sortie	BOOL	Vrai : l'instruction MSG est terminée. Faux : l'instruction MSG n'est pas terminée.
Status	Sortie	CIPSTATUS	État d'exécution du bloc de fonctions. Lorsqu'une instruction MSG est déclenchée ou redéclenchée, tous les éléments d'état sont réinitialisés. La sortie d'état est définie dans le type de données CIPSTATUS sur la page 193.
DataLength	Sortie	UINT	Nombre d'octets de données pour la réponse au message CIPPCCC. Lorsqu'une instruction MSG est déclenchée ou redéclenchée, la valeur DataLength est réinitialisée sur 0. La valeur DataLength est prise en charge lorsqu'un message PCCC est envoyé sur un canal Ethernet.

Exemple de diagramme de bloc de fonctions MSG_PCCC



Exemple de langage à relais MSG_PCCC



Exemple de texte structuré MSG_PCCC

```
MSG_PCCC_1(
          void MSG_PCCC_1(BOOL IN, CIPCONTROLCFG CtrlCfg, PCCC_CFG PCCCCfg, CIPTARGETCFG TargetCfg)
          Type: MSG_PCCC, Envoie un message PCCC
2  Q=MSG_PCCC_1.Q;
3  Status=MSG_PCCC_1.Status;
4  DataLength=MSG_PCCC_1.DataLength
```

Type de données CIPCONTROLCFG

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données CIPCONTROLCFG.

Paramètre	Type de données	Description
Cancel	BOOL	VRAI - Annule l'exécution du bloc fonctionnel. Le bit est effacé lorsque le message est activé.
TriggerType	UDINT	Représente l'une des valeurs suivantes : • 0: envoi de message déclenché une fois (lorsque IN passe de Faux à Vrai) • 1 à 65 535 : Valeur de déclenchement cyclique en millisecondes. Message envoyé périodiquement lorsque IN est défini sur Vrai. Définissez la valeur sur 1 pour déclencher l'instruction MSG le plus rapidement possible.
StrMode	USINT	Réservé pour une utilisation future.

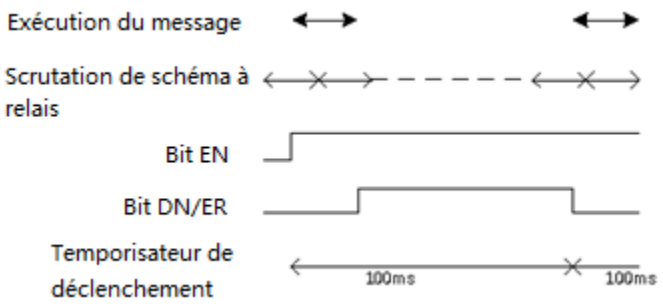
Déclenchement de message PCCC

Le message est déclenché périodiquement en définissant le paramètre TriggerType sur une valeur différente de zéro.

Action	Résultats
Le message est activé	Le temporisateur de déclenchement démarre.
Le temporisateur de déclenchement arrive à expiration avant que le message soit terminé	Le message est immédiatement déclenché lors du cycle de scrutation de relais suivant.
Le message se termine avant l'expiration du délai de déclenchement	Le message est déclenché avant l'expiration du délai de déclenchement.

Exemple : déclenchement de message

Dans l'exemple suivant, la valeur TriggerType est définie sur 100.



Type de données PCCCCFG

Ce tableau permet de définir les paramètres pour le type de données PCCCCFG.

Paramètre	Type de données	Description
Service	USINT	Code de service : • 0 - Lecture (par défaut) • 1 - Écriture
SrcAdr	STRING	Adresse source : • Nom du fichier source si la valeur de Source est 0 Syntaxe : *<Type de fichier><Numéro du fichier>:<Numéro d'élément>* Exemple : N7:1 Les types de fichiers pris en charge sont : • N (Entier) • F (Flottant) • B (Bool) • L (Long) • ST (Chaîne) • Nom de la variable de point si la valeur de Service est 1 • Exemple d'un tableau 1-D : testarrayVariable1*[x]* où x est la position du tableau dans la plage de tableaux. Les tableaux monodimensionnels sont pris en charge. • Exemple de variable normale ou de tableau 1-D dont l'index commence à partir de 0 : VariableName1.
NumOfElements	UINT	Nombre d'éléments : • Lorsque Service est égal à 0, le nombre maximal d'éléments (valeur d'élément * octets occupés par un seul élément) ne peut pas dépasser 244. • Lorsque Service est égal à 1, le nombre maximal d'éléments (valeur d'élément * octets occupés par un seul élément) ne peut pas dépasser 232.
DstAdr	STRING	Adresse de destination : • Nom du fichier source si la valeur de Service est 1 Syntaxe : *<Type de fichier><Numéro du fichier>:<Numéro d'élément>* Exemple : N7:1 Les types de fichiers pris en charge sont : • N (Entier)

Paramètre	Type de données	Description
		• F (Flottant) • B (Bool) • L (Long) • ST (Chaîne) • Nom de la variable de point si la valeur de Service est 0 • Exemple d'un tableau 1-D : testarrayVariable2*[x]* où x est la position du tableau dans la plage de tableaux. Les tableaux monodimensionnels sont pris en charge. • Exemple de variable normale ou de tableau 1-D dont l'index commence à partir de 0 : VariableName2.

Type de données TARGETCFG

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données TARGETCFG.

Paramètre	Type de données	Description
Chemin	STRING[80]	Chemin de la cible défini à l'aide de la syntaxe *{"<port>,<adresse de station/d'emplacement>"}* Définition du numéro de port : • 2 pour le port série intégré • 4 pour le port Ethernet intégré • 5 à 7 pour le module enfichable 2080-SERIALISOL installé dans les emplacements 1 à 3
CipConnMode	USINT	Type de connexion CIP. • 1 - Connexion Class3 Le paramètre doit être défini sur 1 pour les messages PCCC Ethernet et série.
UcmmTimeout	UDINT	Délai de temporisation du message non connecté (en millisecondes). Temps d'attente pour obtenir une réponse aux messages non connectés, y compris le délai d'établissement de la connexion pour les messages connectés. • Valeurs autorisées : de 250 à 10 000 ms. • Définissez sur 0 pour utiliser la valeur par défaut de 3 000 ms (3 secondes). • Toute valeur spécifiée inférieure à 250 ms sera ramenée automatiquement à 250 ms (valeur minimum). • Toute valeur spécifiée supérieure à 10 000 ms sera ramenée automatiquement à 10 000 ms (valeur maximum). Le paramètre s'applique à EtherNet/IP.
ConnMsgTimeout	UDINT	Délai de temporisation de connexion Class3 (en millisecondes). Temps d'attente pour obtenir une réponse aux messages connectés. La connexion est coupée lorsque le délai de temporisation expire. • Valeurs autorisées : de 800 à 10 000 ms. • Définissez sur 0 pour utiliser la valeur par défaut de 10 000 ms (10 secondes).

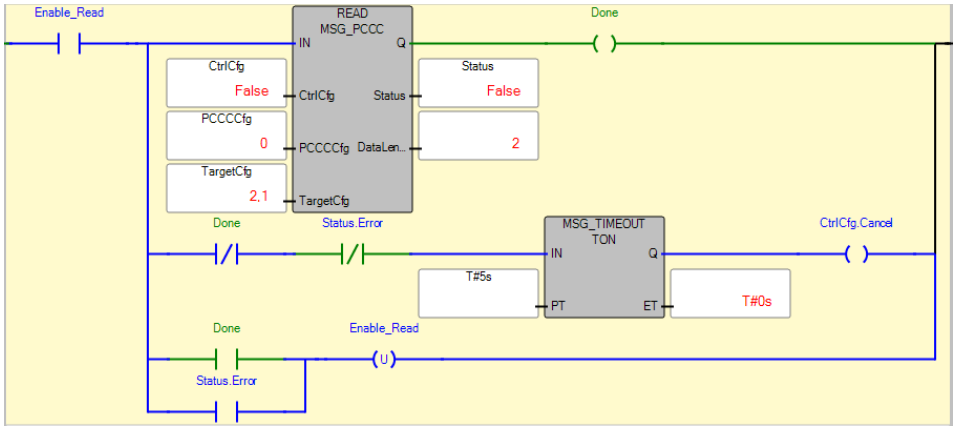
Paramètre	Type de données	Description
		• Toute valeur spécifiée inférieure à 800 ms sera ramenée automatiquement à 800 ms (valeur minimum). • Toute valeur spécifiée supérieure à 10 000 ms sera ramenée automatiquement à 10 000 ms (valeur maximum). Le paramètre s'applique à EtherNet/IP et série.
ConnClose	BOOL	Comportement de coupure de connexion : • TRUE - Coupe la connexion lorsque le message est terminé. • FALSE - Ne coupe pas la connexion lorsque le message est terminé (par défaut). Le paramètre s'applique à EtherNet/IP.

Exemple de chemin d'accès de la cible

Le tableau suivant répertorie des exemples de valeurs utilisées dans une chaîne de chemin d'accès de cible et décrit le résultat de chacune de ces chaînes.

Élément de chaîne	Description
"6,1"	Par le module enfichable de port série monté dans l'emplacement 2 jusqu'au nœud à l'adresse 1. Notez que les emplacements de module enfichable 1 à 3 correspondent aux ports 5 à 7.
"4,192.168.1.100"	Via le Port 4 (port Ethernet intégré Micro870), atteint le Nœud 192.168.1.100.
Port local	Port local utilisé pour envoyer le message. Il doit s'agir d'un port série CIP ou EtherNet/IP actif. Les ports USB ne sont pas pris en charge.
Adresse cible	• Dans le cas d'EIP, l'adresse cible spécifie l'adresse IP de la cible. Il doit s'agir d'une adresse IP unicast et non de 0 ou d'une adresse multicast, broadcast, locale ou de boucle (127.x.x.x). • Dans le cas d'un port série CIP, l'adresse cible spécifie l'adresse de station de la cible. La plage de valeurs autorisée est comprise entre 0 et 254.

IMPORTANT	Lors du déclenchement d'une instruction MSG_PCCC à partir d'un port série, l'utilisateur doit inclure un temporisateur de délai de temporisation de message pour provoquer une erreur dans l'instruction de message si une réponse sans erreur n'est pas reçue dans un délai raisonnable, en particulier lors de l'utilisation de modems radio. L'heure prédéfinie du temporisateur doit être définie par l'utilisateur en fonction de son application. Un exemple est présenté ci-dessous.



Type de données CIPSTATUS

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données CIPSTATUS.

Paramètre	Type de données	Description
Error	BOOL	Ce bit est défini sur TRUE lorsque l'exécution du bloc fonctionnel rencontre une erreur.
ErrorID	UINT	Valeur du code d'erreur. Pour plus d'informations, reportez-vous au tableau des codes d'erreur sur la page 195
SubErrorID	UINT	Sous-valeur du code d'erreur.
ExtErrorID	UINT	Valeur du code d'erreur du statut prolongé CIP.
StatusBits	UINT	Ce paramètre peut être utilisé pour vérifier les bits de contrôle : • Bit 0 : EN – Enable • Bit 1 : EW – Enable Wait • Bit 2 : ST – Start • Bit 3 : ER – Error • Bit 4 : DN – Done • Bit 5 : CIPCONN : coupure de connexion CIP • Bit 6 : EIPSESS : fermeture de session EIP Le bit 5 et le bit 6 sont pour Ethernet/IP et les autres bits sont réservés. Les éléments StatusBits sont définis pour les bits d'état CIPSTATUS sur la page 193.

Bits de statut CIPSTATUS

Les bits de statut CIPSTATUS sont définis sur la base du statut d'exécution du message, des mémoires tampon de communication et des conditions de ligne.

Nom du bit	Description	Comportement
EN	Activer	Est à 1 lorsque la ligne de programme passe à Vrai et reste à 1 jusqu'à ce que le bit DN ou ER passe à 1 et que la ligne passe à Faux.
EW	Activer l'attente	Est à 1 lorsque la mémoire tampon de communication est affectée à la demande de message. Est à 0 lorsque le bit ST est à 1.

Nom du bit	Description	Comportement
ST	Démarrer	Est à 1 lorsque le message a été transmis et est en attente de réponse. Est à 0 lorsque le bit DN est à 1.
ER	Erreur	Est à 1 lorsque la transmission du message a échoué. Un code d'erreur est écrit sur ErrorID. Le bit ER et les valeurs de code d'erreur sont effacés lorsque la ligne passe à nouveau de Faux à Vrai.
DN	Terminé	Est à 1 lorsque l'envoi du message a réussi. Le bit DN passe à 0 lorsque la ligne passe à nouveau de Faux à Vrai.
CIPCONN	Terminé	Défini lorsque la connexion CIP pour la communication est coupée. Le bit CIPCONN s'applique lorsque ConnClose est défini sur True ; pour les autres cas, le bit CIPCONN est défini sur False. Le bit est uniquement utilisé pour Ethernet.
EIPSESS	Terminé	Défini lorsque la session CIP d'encapsulation pour la communication est fermée. Le bit EIPSESS s'applique lorsque ConnClose est défini sur True ; pour les autres cas, le bit EIPSESS est défini sur False. Le bit est uniquement utilisé pour Ethernet.

Conseil : Lorsque le message est activé (bit EN mis à un) et n'est pas terminé (le bit DN n'est pas mis à un), si le paramètre Abandon est défini, l'exécution du message se termine et le bit ER est mis à un.

Connexions de message CIP/EIP

Un maximum de 16 connexions CIP (classe 3) et 16 connexions EIP est pris en charge pour l'exécution du message client. Le tableau suivant décrit le comportement de connexion CIP/EIP. Tous les scénarios suivants s'appliquent à Ethernet.

Scénario	Résultats
La demande de message est activée et CipConnMode=1.	S'il n'existe pas de connexion à la cible, une connexion CIP est établie. Si le port local du message est Ethernet et qu'il n'existe aucune connexion EIP à la cible, une connexion EIP est établie avant l'établissement d'une connexion CIP.
La demande de message est activée, CipConnMode=0 et le port local du message est de type Ethernet.	S'il n'existe aucune connexion EIP à la cible, une connexion EIP est établie. Si plusieurs messages sont désignés pour la même cible, les messages doivent utiliser la même connexion.
ConnClose est défini sur True, la connexion de message est coupée et la session EIP est fermée à la fin de l'exécution du message.	Si plusieurs messages partagent la même connexion, la connexion est coupée à la fin de l'exécution du dernier message.
L'exécution du message est terminée et ConnClose est défini sur Faux.	La connexion n'est pas fermée une fois l'exécution du message terminée.

Scénario	Résultats
Une connexion CIP ou EIP qui n'est pas associée à un message actif est coupée si elle est en attente pendant x secondes, où x correspond à une valeur de temporisation d'inactivité d'encapsulation configurable qui peut être réinitialisée à l'aide du service CIP défini.	Reportez-vous aux objets TCP/IP du Volume II de la spécification CIP pour en savoir plus sur le service CIP défini.
Une connexion CIP est coupée et toute connexion EIP est également coupée.	Si plusieurs connexions CIP utilisent la même connexion EIP, celle-ci est coupée lorsque toutes les connexions CIP associées sont coupées.
Le contrôleur passe d'un mode d'exécution (Exécution, Exécution à distance, Scrutation unique pour test à distance et Ligne unique distante) à un mode sans exécution.	La coupure est forcée pour l'ensemble des connexions.

Temps du message dépassé

Le tableau suivant décrit la manière dont les temporisateurs des paramètres inUcmmTimeout et ConnMsgTimeout se comportent, suivant le statut et la demande de message.

Action	Résultats
Le message est activé	Le temporisateur UcmmTimeout est activé
La connexion CIP est demandée	Le temporisateur ConnMsgTimeout est activé
Le temporisateur ConnMsgTimeout est activé	Le temporisateur UcmmTimeout est désactivé Applicable pour Ethernet/IP
La demande de connexion est finalisée	Le temporisateur UcmmTimeout est réactivé Applicable pour Ethernet/IP

Codes d'erreur MSG_PCCC

Ce tableau permet de déterminer les valeurs de paramètre pour les champs ErrorID et SubErrorID lorsque le bit ER est mis à un.

Code ErrorID	Code SubErrorID	Description du code d'erreur
30		SubErrorID représente l'erreur à distance de l'état (STS). Pour plus d'informations, reportez-vous au chapitre 8 du DF1 Protocol and Command Set Reference Manual.
31		SubErrorID représente l'erreur à distance de l'état étendu (STS EXT). Pour plus d'informations, reportez-vous au chapitre 8 du DF1 Protocol and Command Set Reference Manual.
32	Erreur de configuration PCCC	
	1	Code de service non pris en charge Les codes de service pris en charge sont : • 0 - Lecture (par défaut) • 1 - Écriture
	2	Adresse source non valide Pour connaître les adresses source correctes, reportez-vous à SrcAdr dans le tableau des types de données PCCCCFG sur la page 190. Pour le type de données Chaîne, la taille de chaîne fixe prise en charge est 82.
	3	Adresse de destination non valide Pour connaître les adresses source correctes, reportez-vous à DstAdr dans le tableau des types de données PCCCCFG sur la page 190. Pour le type de données Chaîne, la taille de chaîne fixe prise en charge est 82.

Code ErrorID	Code SubErrorID	Description du code d'erreur
	4	Type de données source non valide Les types de fichiers pris en charge sont : • N (Entier) • F (Flottant) • B (Bool) • L (Long) • ST (Chaîne)
	5	Type de données de destination non valide Les types de fichiers pris en charge sont : • N (Entier) • F (Flottant) • B (Bool) • L (Long) • ST (Chaîne)
	6	Le type de fichier et le type de données du symbole ne sont pas compatibles. Le type de données flottant n'envoie des messages que vers ou depuis des types de données flottants.
	41	Nom de symbole CIP non valide
	48	Taille du tableau de données d'entrée du bloc de fonctions insuffisante. Vérifiez que la taille de la mémoire tampon des données dépasse la longueur des données demandée.
33	**Erreurs relatives à la configuration du paramètre**	
	32	Numéro de canal incorrect.
	36	Type de connexion CIP non pris en charge. Assurez-vous que le paramètre CipConnMode est défini sur 1 pour les messages PCCC.
	48	Nombre d'éléments non valide La plage du nombre d'éléments est comprise entre 1 et le maximum. Pour plus d'informations, reportez-vous au tableau du nombre d'éléments.
	49	Chemin d'accès à la cible non valide.
	51	La charge utile est incorrecte ou la taille du tableau de données à transmettre du bloc de fonctions est excessive pour la communication CIP.
	52	Valeur de type de segment incorrect
	53	Valeur de temps dépassé UCCM incorrecte Si la valeur de temps dépassé d'encapsulation est inférieure au temps dépassé UCCM ou que la différence entre le temps dépassé d'encapsulation et le temps dépassé UCCM est inférieure ou égale à une seconde, l'erreur est signalée. Ce code s'applique à Ethernet/IP.
	54	Valeur de temps dépassé de connexion incorrecte Si la valeur de temps dépassé d'encapsulation est inférieure au temps dépassé du message CONNECTED ou que la différence entre le temps dépassé d'encapsulation et le temps dépassé du message CONNECTED est inférieure ou égale à une seconde, l'erreur est signalée. Ce code s'applique à EtherNet/IP.
	Erreurs de temporisation	
55	112	Le délai du message a expiré pendant que celui-ci se trouvait dans la file d'attente. Pour éviter ce problème, vérifiez que le nombre de blocs de fonctions en cours d'exécution est égal ou inférieur à quatre pour chaque port.
	113	Le délai du message a expiré pendant que celui-ci attendait l'établissement de la connexion à la couche liaison. Pour éviter ce problème, vous devez vous assurer que : • Le câble fonctionne et la cible existe. • Le contrôleur prend en charge le traitement de huit messages client CIP au maximum.

Code ErrorID	Code SubErrorID	Description du code d'erreur
	114	Le délai du message a expiré pendant que celui-ci attendait la transmission vers la couche liaison. Pour éviter ce problème, vérifiez que le contrôleur prend en charge le traitement de huit messages client CIP au maximum.
	115	Le délai du message a expiré pendant que celui-ci attendait une réponse de la couche liaison. Pour éviter ce problème, vérifiez que la cible existe et que les câbles fonctionnent.
	Codes d'erreur relatifs au format de réponse du serveur	
69	65	La réponse au message ne correspond pas à la demande.
	68	Le type de données de la réponse au message n'est pas valide ou pris en charge.
208		Aucune adresse IP n'a été configurée pour le réseau. Pour résoudre ce problème, configurez une adresse IP valide. Ce code s'applique à EtherNet/IP.
209		Le nombre maximal de connexions autorisé a été atteint ; aucune connexion n'est disponible. Pour éviter ce problème, réduisez le nombre de connexions utilisées. Ce code s'applique à EtherNet/IP.
210		Adresse Internet ou adresse de station non valide Pour éviter ce problème, n'utilisez pas 0 ou une adresse broadcast, multicast ou de boucle pour l'adresse IP cible. Ce code s'applique à EtherNet/IP.
217		L'exécution du message a été annulée par l'utilisateur (le paramètre Abandon a été défini sur VRAI). Pour éviter ce problème, ne définissez pas le bit Abandon ou n'exécutez pas l'opération RMD.
218		Aucun espace de mémoire tampon réseau disponible Pour éviter ce problème, vérifiez qu'il n'y a pas une quantité excessive d'activité de communication en cours. Ce code s'applique à EtherNet/IP.
222		Réservé.
223		L'adresse de liaison n'est pas disponible. Un changement de configuration TCP/IP ou Ethernet est en cours. Ce code s'applique à EtherNet/IP.
224		Code d'erreur de réponse CIP SubErrorID spécifie le statut CIP et ExtErrorID spécifie la valeur du statut prolongé CIP. Consultez la spécification CIP pour connaître les valeurs de code d'erreur possibles. Ce code s'applique à EtherNet/IP.
255		Le canal est fermé ou une reconfiguration est en cours. Vous pouvez rencontrer cette erreur lorsque la reconfiguration est en cours ou que le port n'est pas disponible lorsque vous exécutez les instructions MSG_CIPPCCC. Pour éviter ce problème, attendez que la configuration se termine avant d'exécuter les instructions. Ce code s'applique à EtherNet/IP.

Tableau du nombre d'éléments

Type de données source/destination	Nombre maximal d'éléments pour le code de service 0 - service de lecture	Nombre maximal d'éléments pour le code de service 1 - service d'écriture
BIT	122	116
INT	122	116
FLOAT	61	58
STRING	2	2
DOUBLE	61	58

Pour plus d'informations sur la création d'un programme MSG_PCCC, consultezl'exemple suivant : Création d'un programme de messagerie MSG_PCCC pour y écrire une valeur et lire une valeur à partir d'une adresse dans MicroLogix sur la page 198.

Exemple : Créez un programme de messagerie MSG_PCCC pour y écrire une valeur et lire une valeur à partir d'une adresse dans MicroLogix

Cet exemple montre comment créer un programme de messagerie pour écrire une valeur sur une adresse héritée du contrôleur A à partir d'une variable globale du contrôleur B et lire une valeur à partir de l'adresse héritée du contrôleur A et l'enregistrer dans la variable globale du contrôleur B via Ethernet.

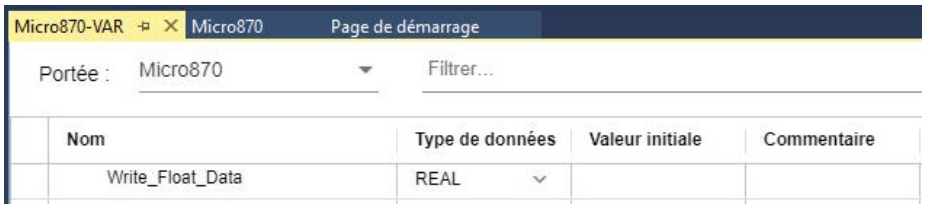
Exemple d'écriture PCCC : Ajout d'un bloc de fonctions MSG_ PCCC, de variables et d'un contact

Pour créer un programme de messagerie MSG_PCCC utilisé pour écrire une valeur sur une adresse héritée dans MicroLogix 1400 à partir d'un contrôleur Micro870E, effectuez les tâches suivantes.

Pour créer une variable globale pour la variable source, procédez comme suit :

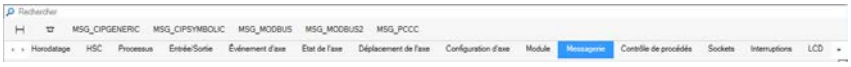
1. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables globales** pour afficher le sélecteur de **variables globales**.
2. Entrez **Write_Float_Data** dans la colonne **Nom**.
3. Vérifiez que le type de données est REAL.

Le sélecteur de **variables globales** doit être similaire à celui illustré dans l'image suivante.

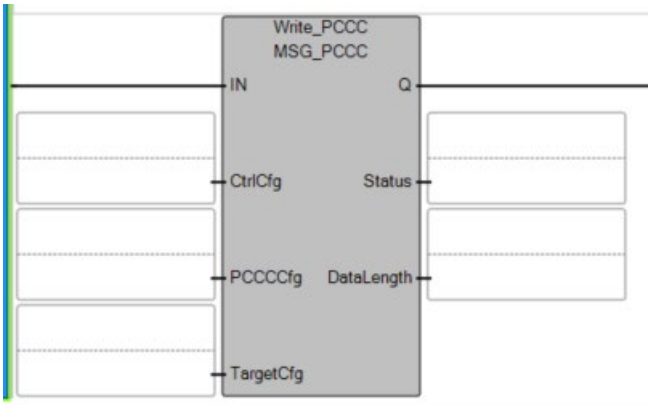


Pour ajouter un bloc de fonctions et des variables, procédez comme suit :

1. Dans la **Barre d'outils Instructions - Messagerie**, sélectionnez **MSG_PCCC** et faites-le glisser sur la ligne de programme de schéma à relais.

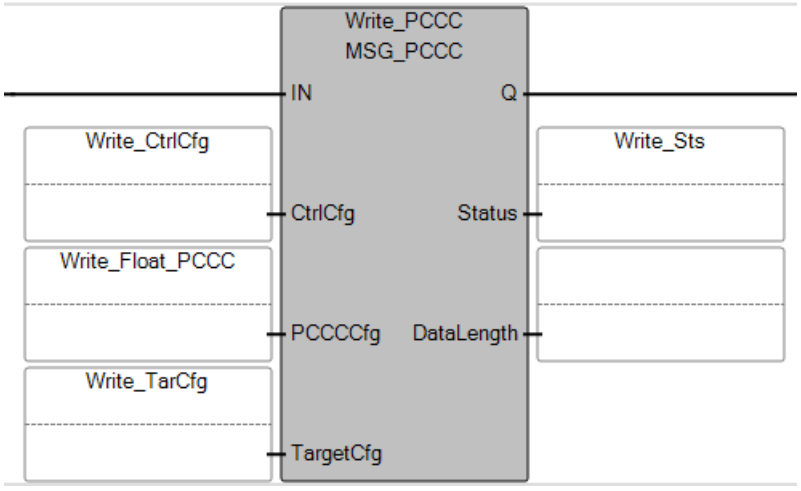


2. (facultatif) Double-cliquez sur **MSG_PCCC** pour faire passer l'instance du bloc de fonctions à Write_PCCC.



3. Affectez des noms de variable.
   a. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables locales** pour afficher la page **Variables locales**.
   b. Dans le sélecteur de **variables**, ajoutez les variables et les types de données répertoriés dans le tableau.
   c. Affectez les variables aux paramètres d'entrée et de sortie de l'instruction.

Paramètre	Nom de variable	Type de données
CtrlCfg	Write_CtrlCfg	CIPCONTROLCFG
PCCCCfg	Write_PCCC	PCCC_CFG
TargetCfg	Write_TarCfg	CIPTARGETCFG
Status	Write_Sts	CIPSTATUS



4. Configurez les valeurs initiales pour les variables locales et ajoutez les valeurs initiales CtrlCfg.
   a. Dans la page **Variables locales**, développez le paramètre CtrlCfg pour afficher des paramètres complémentaires.
   b. Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale
Write_CtrlCfg.Cancel	Laisser vide
Write_CtrlCfg.TriggerType	50
Write_CtrlCfg.StrMode	Laisser vide

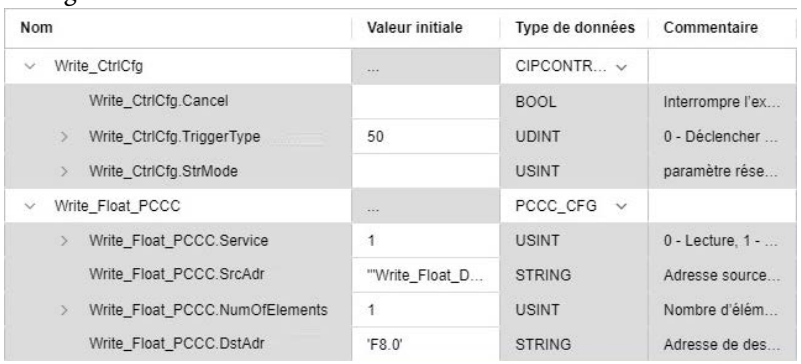
Conseil : La valeur **50** est entrée dans cet exemple pour définir un intervalle de déclenchement cyclique de 50 ms.

5. Ajoutez les valeurs initiales PCCCCfg.
   a. Dans la page **Variables locales**, développez le paramètre PCCCCfg pour afficher des paramètres complémentaires.
   b. Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale
Write_Float_PCCC.Service	1
Write_Float_PCCC.SrcAdr	'Write_Float_Data'
Write_Float_PCCC.NumOfElements	1
Write_Float_PCCC.DstAdr	'F8:0'

Conseil : **F8:0** est entré comme adresse de destination dans le contrôleur MicroLogix.

Le sélecteur Variables locales doit être similaire à celui illustré dans l'image suivante.



Nom	Valeur initiale	Type de données	Commentaire
Write_CtrlCfg	...	CIPCONTR...	
Write_CtrlCfg.Cancel		BOOL	Interrompre l'ex...
Write_CtrlCfg.TriggerType	50	UDINT	0 - Déclencher ...
Write_CtrlCfg.StrMode		USINT	paramètre rése...
Write_Float_PCCC	...	PCCC_CFG	
Write_Float_PCCC.Service	1	USINT	0 - Lecture, 1 - ...
Write_Float_PCCC.SrcAdr	'"Write_Float_D...	STRING	Adresse source...
Write_Float_PCCC.NumOfElements	1	USINT	Nombre d'élém...
Write_Float_PCCC.DstAdr	'F8.0'	STRING	Adresse de des...

6. Configurez les valeurs initiales pour TarCfg.
   a. Dans la page **Variables locales**, développez le paramètre TarCfg pour afficher des paramètres complémentaires.
   b. Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

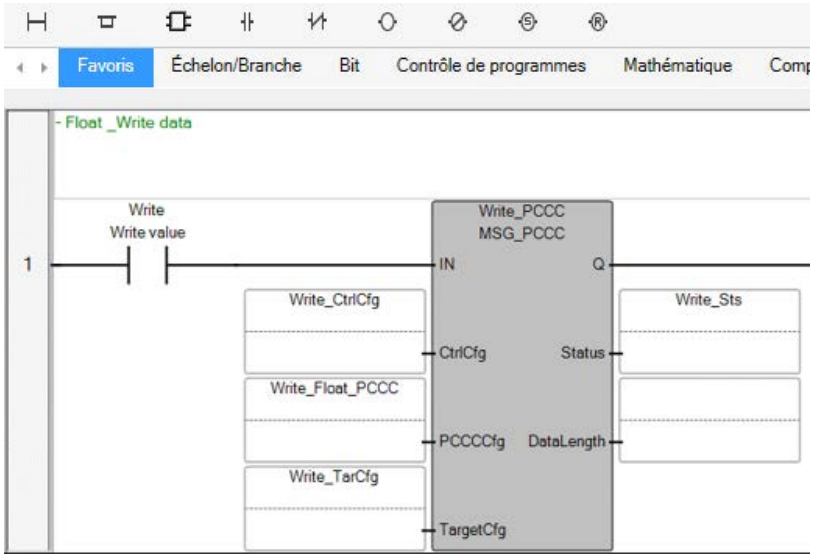
Paramètre	Valeur initiale
Write_TarCfg.Path	'4,192.168.3.33'
Write_TarCfg.CipConnMode	1
Write_TarCfg.UcmmTimeout	0
Write_TarCfg.ConnMsgTimeout	0
Write_TarCfg.ConnClose	FALSE

Le sélecteur Variables locales doit être similaire à celui illustré dans l'image suivante.

Nom	Valeur initiale	Type de données	Commentaire
Write_TarCfg	...	CIPTARGE...	
Write_TarCfg.Path	'4,192.168.3.33'	STRING	Chemin de des...
Write_TarCfg.CipConnMode	1	USINT	0 - Non connec...
Write_TarCfg.UcmmTimeout	0	UDINT	Expiration du m...
Write_TarCfg.ConnMsgTimeout	0	UDINT	Expiration du m...
Write_TarCfg.ConnClose	FALSE	BOOL	TRUE : Fermer...

7. Ajoutez un contact.
   a. Dans la **Barre d'outils Instructions - Favoris**, sélectionnez l'élément **Contact normalement ouvert** et faites-le glisser sur la gauche de l'entrée du bloc de fonctions MSG_PCCC sur la première ligne de programme de schéma à relais.
   b. Cliquez sur le contact et affectez la variable **Écriture** au contact.

   La ligne de votre programme de schémas à relais pour la messagerie MSG_PCCC doit être similaire à celle illustrée dans l'image suivante.



8. Chargez le programme.

   Une fois le chargement terminé, entrez la valeur à écrire dans MicroLogix dans la variable **Write_Float_Data** du contrôleur Micro800 et déclenchez la variable **d'écriture** pour qu'il exécute une valeur d'écriture depuis les Write_Float_Data à l'adresse du contrôleur MicroLogix F8:0. Observez que l'adresse MicroLogix est mise à jour en fonction de la valeur.

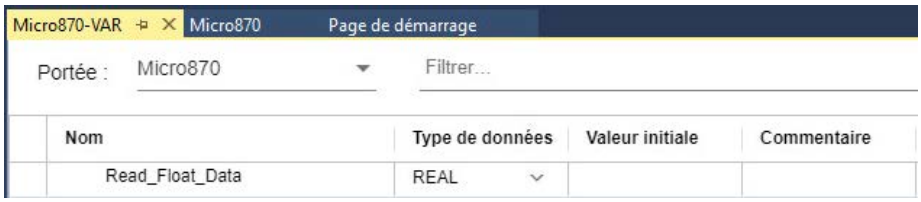
Exemple de lecture PCCC : Ajout d'un bloc de fonctions PCCC_MSG, de variables et d'un contact

Pour créer un programme de messagerie MSG_PCCC utilisé pour lire une valeur d'une adresse héritée de MicroLogix 1400 vers un contrôleur Micro870E, effectuez les tâches suivantes.

Pour créer une variable globale permettant de stocker les résultats, procédez comme suit :

1. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables globales** pour afficher le sélecteur de **variables globales**.
2. Entrez **Read_Float_Data** dans la colonne **Nom**.
3. Vérifiez que le type de données est REAL.

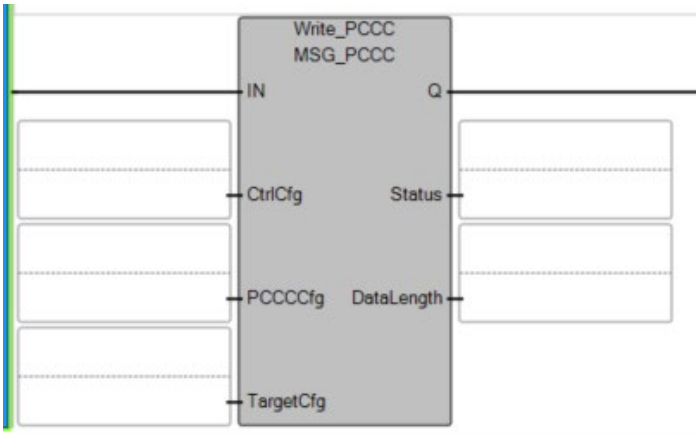
Le sélecteur de **variables globales** doit être similaire à celui illustré dans l'image suivante.



Pour ajouter un bloc de fonctions et des variables, procédez comme suit :

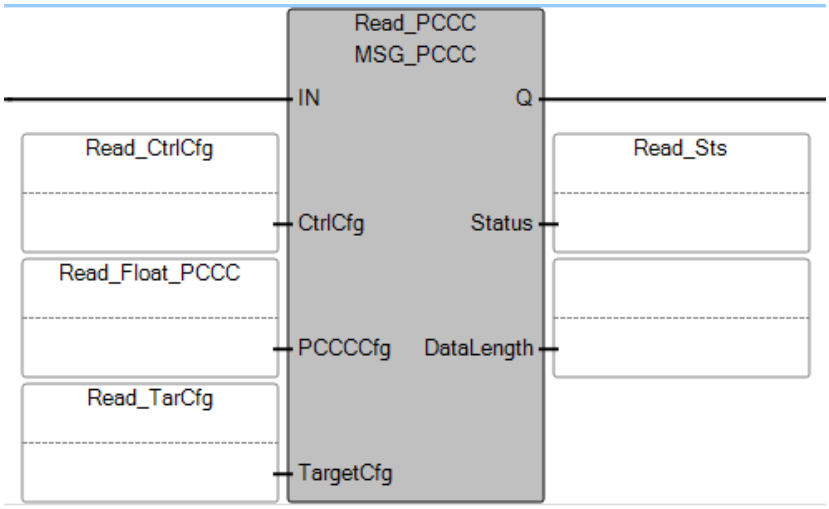
1. Dans la **Barre d'outils Instructions - Messagerie**, sélectionnez **MSG_PCCC** et faites-le glisser sur la ligne de programme de schéma à relais.

2. (facultatif) Double-cliquez sur **MSG_PCCC** pour faire passer l'instance du bloc de fonctions à Read_PCCC.



3. Affectez des noms de variable.
   a. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables locales** pour afficher la page **Variables locales**.
   b. Dans le sélecteur de **variables**, ajoutez les variables et les types de données répertoriés dans le tableau.
   c. Affectez les variables aux paramètres d'entrée et de sortie de l'instruction.

Paramètre	Nom de variable	Type de données
CtrlCfg	Read_CtrlCfg	CIPCONTROLCFG
PCCCCfg	Read_PCCC	PCCC_CFG
TargetCfg	Read_TarCfg	CIPTARGETCFG
Status	Read_Sts	CIPSTATUS



4. Configurez les valeurs initiales pour les variables locales et ajoutez les valeurs initiales CtrlCfg.
   a. Dans la page **Variables locales**, développez le paramètre CtrlCfg pour afficher des paramètres complémentaires.
   b. Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale
Read_CtrlCfg.cancel	Laisser vide
Read_CtrlCfg.TriggerType	50
Read_CtrlCfg.StrMode	Laisser vide

Conseil : La valeur **50** est entrée dans cet exemple pour définir un intervalle de déclenchement cyclique de 50 ms.

5. Ajoutez les valeurs initiales PCCCCfg.
   a. Dans la page **Variables locales**, développez le paramètre PCCCCfg pour afficher des paramètres complémentaires.
   b. Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale
Read_Float_PCCC.Service	1
Read_Float_PCCC.SrcAdr	'F8:0'
Read_Float_PCCC.NumOfElements	1
Read_Float_PCCC.DstAdr	'Read_Float_Data'

Conseil : **F8:0** est entré comme adresse source dans le contrôleur MicroLogix.

Le sélecteur Variables locales doit être similaire à celui illustré dans l'image suivante.

Nom	Valeur initiale	Type de données	Commentaire
Read_Ctrlcfg	...	CIPCONTR...	
Read_Ctrlcfg.Cancel		BOOL	Interrompre l'ex...
Read_Ctrlcfg.TriggerType	50	UDINT	0 - Déclencher ...
Read_Ctrlcfg.StrMode		USINT	paramètre rése...
Read_Float_PCCC	...	PCCC_CFG	
Read_Float_PCCC.Service	1	USINT	0 - Lecture, 1 - ...
Read_Float_PCCC.SrcAdr	'F8.0'	STRING	Adresse source...
Read_Float_PCCC.NumOfElements	1	USINT	Nombre d'élém...
Read_Float_PCCC.DstAdr	'Read_Float_D...	STRING	Adresse de des...

6. Configurez les valeurs initiales pour TarCfg.
   a. Dans la page **Variables locales**, développez le paramètre TarCfg pour afficher des paramètres complémentaires.
   b. Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

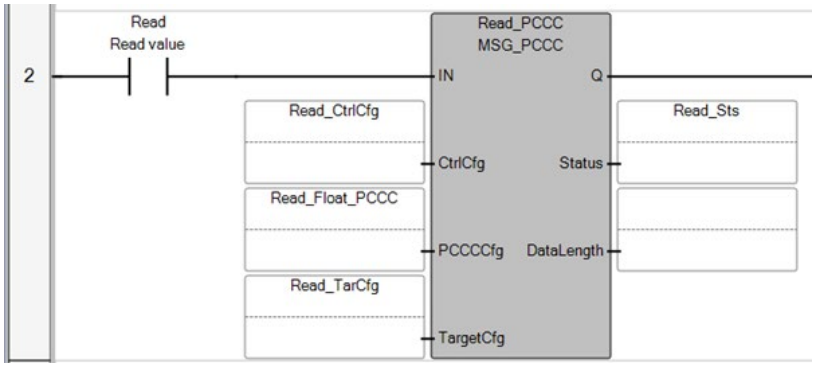
Paramètre	Valeur initiale
Read_TarCfg.Path	'4,192.168.3.33'
Read_TarCfg.CipConnMode	1
Read_TarCfg.UcmmTimeout	0
Read_TarCfg.ConnMsgTimeout	0
Read_TarCfg.ConnClose	FALSE

Le sélecteur Variables locales doit être similaire à celui illustré dans l'image suivante.

Nom	Valeur initiale	Type de données	Commentaire
Read_TarCfg	...	CIPTARGE...	
Read_TarCfg.Path	'4,192.168.3.33'	STRING	Chemin de des...
Read_TarCfg.CipConnMode	1	USINT	0 - Non connec...
Read_TarCfg.UcmmTimeout	0	UDINT	Expiration du m...
Read_TarCfg.ConnMsgTimeout	0	UDINT	Expiration du m...
Read_TarCfg.ConnClose	FALSE	BOOL	TRUE : Fermer...

7. Ajoutez un contact.
   a. Dans la **Barre d'outils Instructions - Favoris**, sélectionnez l'élément **Contact normalement ouvert** et faites-le glisser et déposez-le sur la gauche de l'entrée du bloc de fonctions MSG_PCCC sur la deuxième ligne de programme de schéma à relais.
   b. Cliquez sur le contact et affectez la variable **Lecture** au contact.

La ligne de votre programme de schémas à relais pour la messagerie MSG_PCCC doit être similaire à celle illustrée dans l'image suivante.



8. Chargez le programme.

   Une fois le chargement terminé, entrez la valeur dans l'adresse du contrôleur MicroLogix F8:0 et déclenchez la variable de lecture dans le contrôleur Micro800 pour effectuer une valeur de lecture à partir de l'adresse MicroLogix F8:0 et observer la valeur écrite dans les Read_Float_Data du contrôleur Micro800.

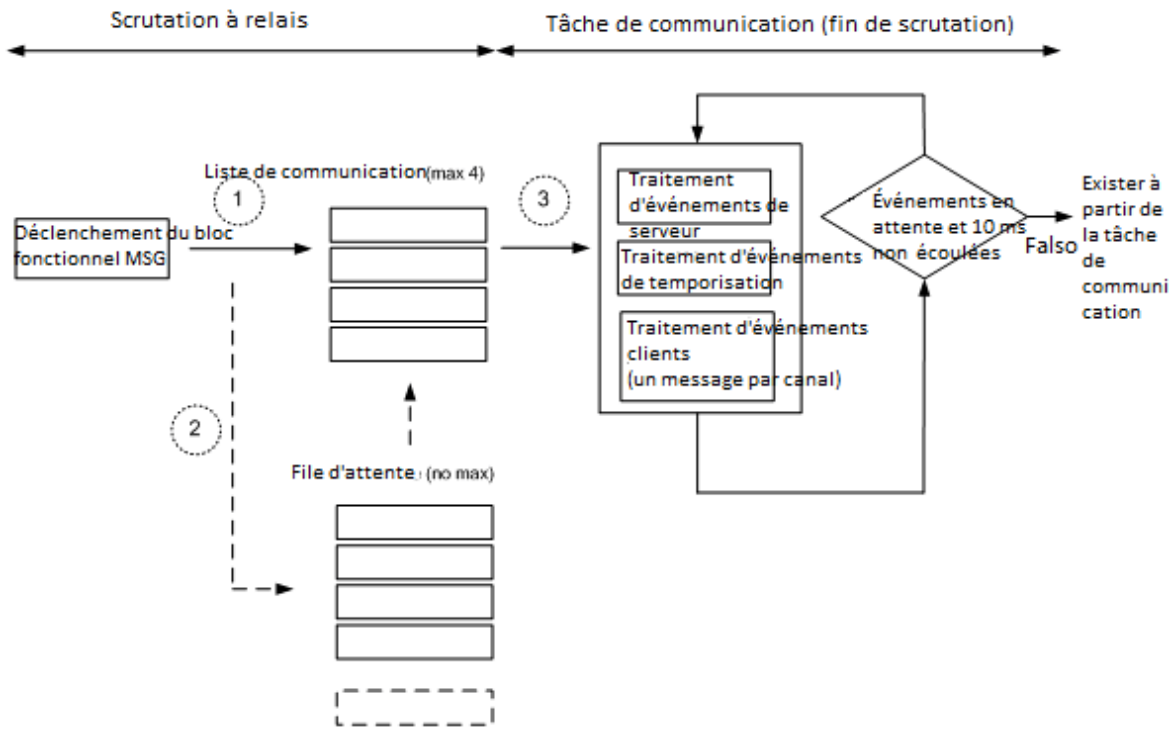
Processus d'exécution des messages et diagrammes de temporisation

Les rubriques suivantes décrivent de quelle manière et à quel moment les instructions de message MSG_CIPGENERIC sur la page 163, MSG_CIPSYMBOLIC sur la page 171 et MSG_MODBUS2 sur la page 181 et MSG_PCCC sur la page 187 s'exécutent selon les conditions de leur bit et de leur ligne de programme.

- Processus d'exécution de message (généralités)
- Processus d'exécution des messages (Ligne = TRUE)
- Diagramme de temporisation d'exécution des messages (Ligne = Vrai)
- Processus d'exécution des messages (Ligne = FALSE)
- Diagramme de temporisation d'exécution des messages (ligne = FALSE)
- Processus d'exécution des messages (erreur)
- Diagramme de temporisation d'exécution des messages (erreur de configuration PCCC)
- Diagramme de temporisation d'exécution de message (message abandonné)

Processus d'exécution de message (généralités)

Le diagramme suivant illustre comment et quand les messages sont exécutés, en fonction de leur statut dans la file d'attente de communication.

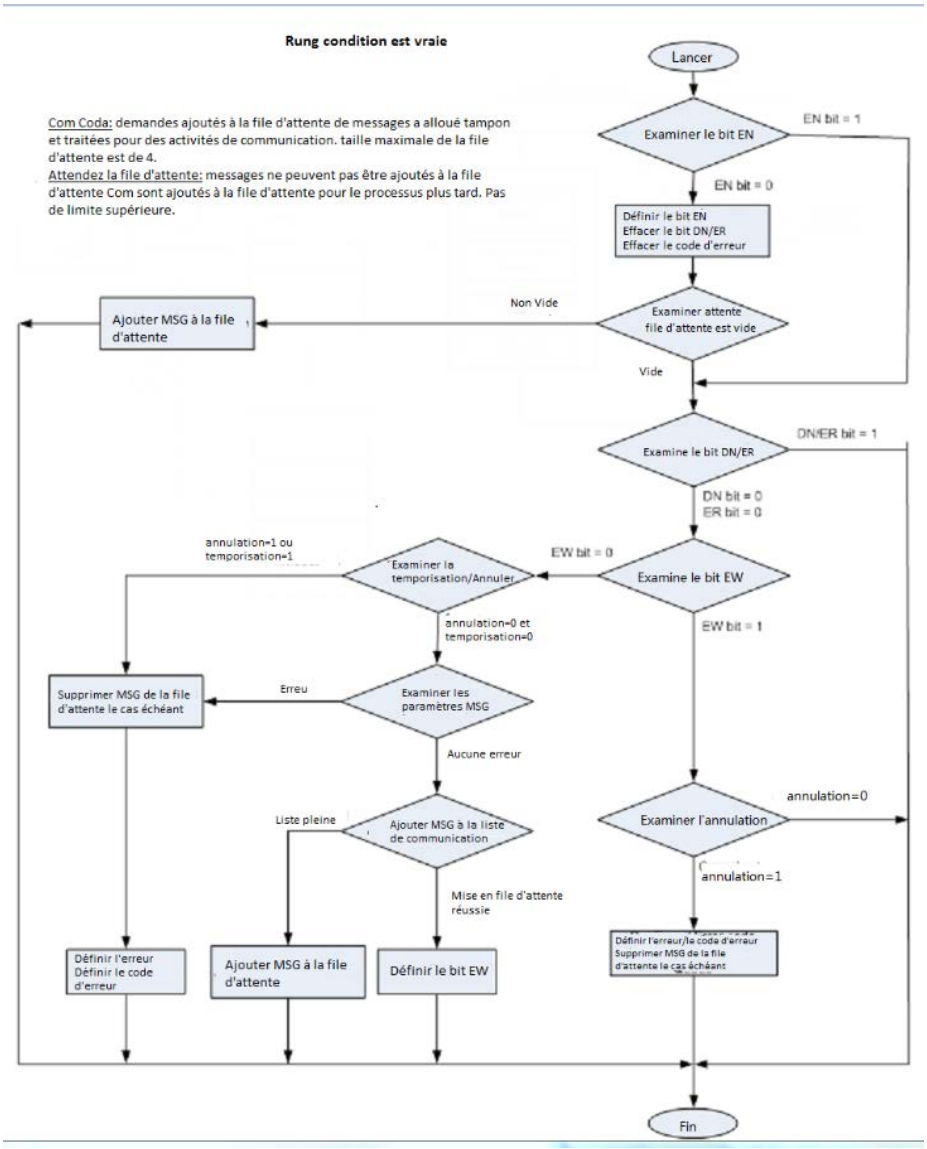


Le tableau suivant décrit la séquence d'événements identifiée dans le précédent diagramme.

N°	Description des événements
1	Le message est activé. Si la liste de communication est vide, la mémoire tampon est attribuée au message et le message est ajouté à la liste de communication pour être transmis. La taille de la file d'attente Com est de 4 et chaque canal possède sa propre file d'attente.
2	Si la liste de communication est pleine, le message est ajouté à la file d'attente. Lorsque la file d'attente Com s'est vidée, le message contenu dans la file d'attente standard est ajouté à la file d'attente Com. La file d'attente standard n'a pas de limite de taille et chaque canal possède sa propre file d'attente.
3	La tâche de communication exécute les messages de la liste de communication à chaque fin de scrutation pour qu'ils soient transmis. Les files d'attente de canal sont traitées une par une, en mode tourniquet (Round Robin). Un message de chaque canal est exécuté et le traitement se poursuit jusqu'à ce que tous les messages aient été exécutés ou que le planning de communication (10 ms) arrive à expiration. Le canal suivant celui ayant été traité en dernier est planifié pour apparaître en premier dans la fin de scrutation suivante

Processus d'exécution des messages (Ligne = TRUE)

Le diagramme de processus suivant décrit les événements d'instruction de message qui surviennent lorsque la condition de ligne est définie sur Vrai.



File d'attente de communication : Les demandes de message ajoutées à la file d'attente de communication (Com) disposent d'une mémoire tampon dédiée et sont traitées par la tâche de communication. Le nombre d'éléments maximum dans cette file d'attente est de 4.

File d'attente : les messages qui ne peuvent pas être ajoutés à la liste de communication sont placés dans la file d'attendre pour être traités ultérieurement. Le nombre de messages contenus dans la file d'attente n'est pas limité.

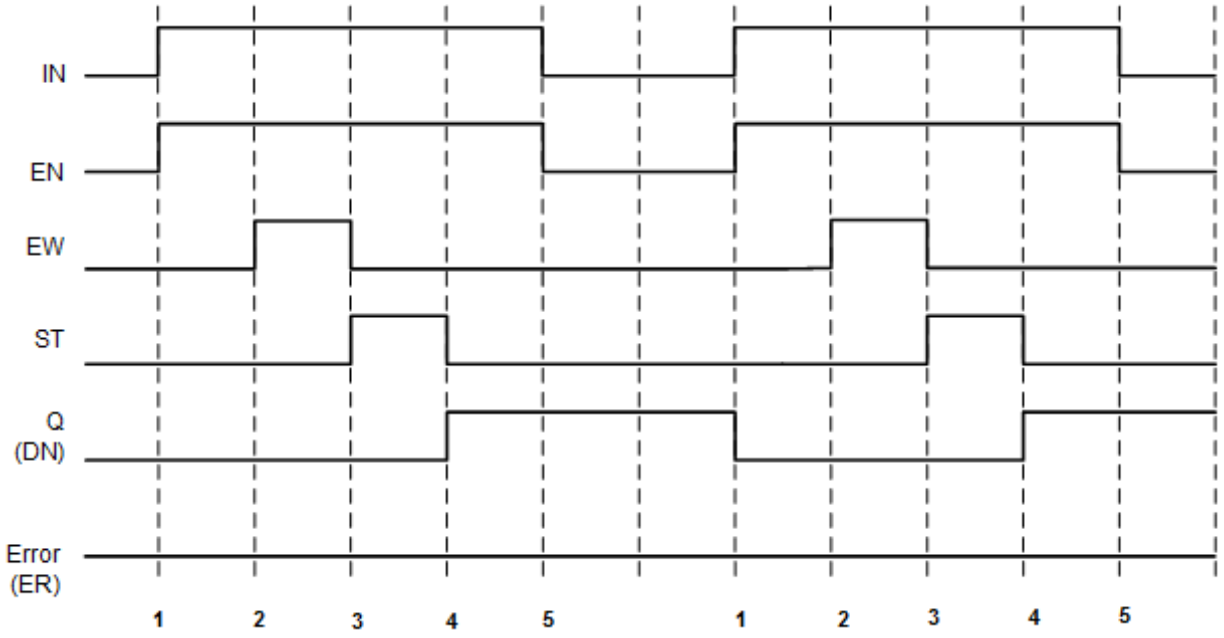
Diagramme de temporisation d'exécution des messages (Ligne = VRAI)

Le tableau suivant décrit les conditions de message et le statut de bit pour chaque étape d'exécution identifiée dans le diagramme de temporisation lorsque la condition de ligne reste définie sur Vrai.

Étape	Description du message	Statut du bit
1	La condition de ligne passe sur TRUE. L'exécution du message est activée.	Le bit EN bit est mis à 1. Tous les autres bits sont effacés.

Étape	Description du message	Statut du bit
2	La mémoire tampon du contrôle de message est acquise. À ce stade, les données d'entrée (c'est-à-dire le paramètre de « données » pour écrire les messages) sont copiées, pour transmission. Les modifications appliquées ultérieurement aux données d'entrée n'apparaîtront pas dans le message transmis.	Le bit EW est défini sur 1.
3	La transmission de message débute.	Le bit EW est effacé (0). Le bit ST est défini sur 1.
4	La réponse au message est reçue.	Le bit ST est effacé (0). Le bit DN est défini sur 1.
5	La condition de ligne passe sur FALSE.	Le bit EN est effacé.

Diagramme de temporisation pour (Ligne = TRUE)



Processus d'exécution des messages (Ligne = FALSE)

Le diagramme de processus suivant décrit les événements d'instruction de message qui surviennent lorsque la condition de ligne de programme est définie sur False.

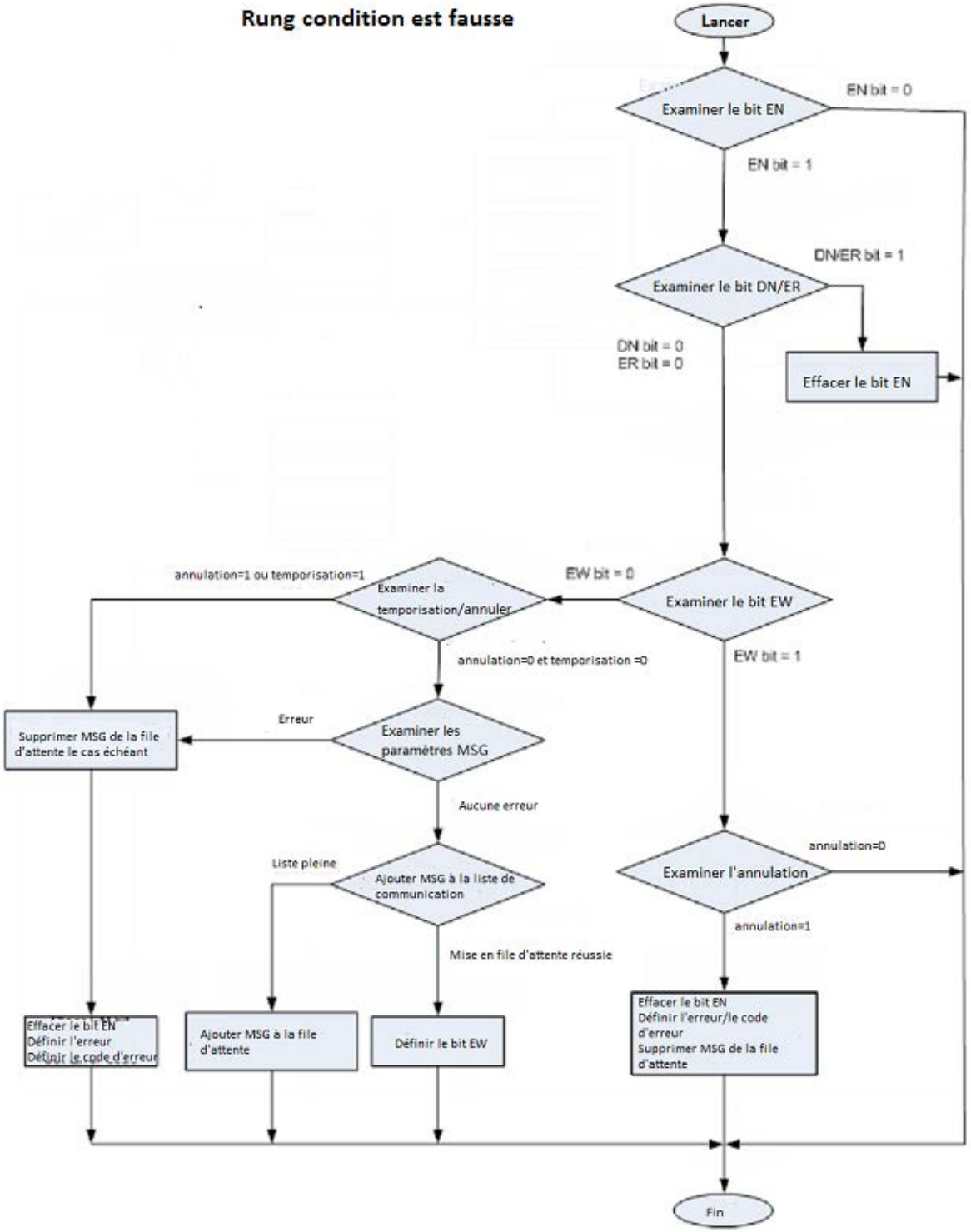


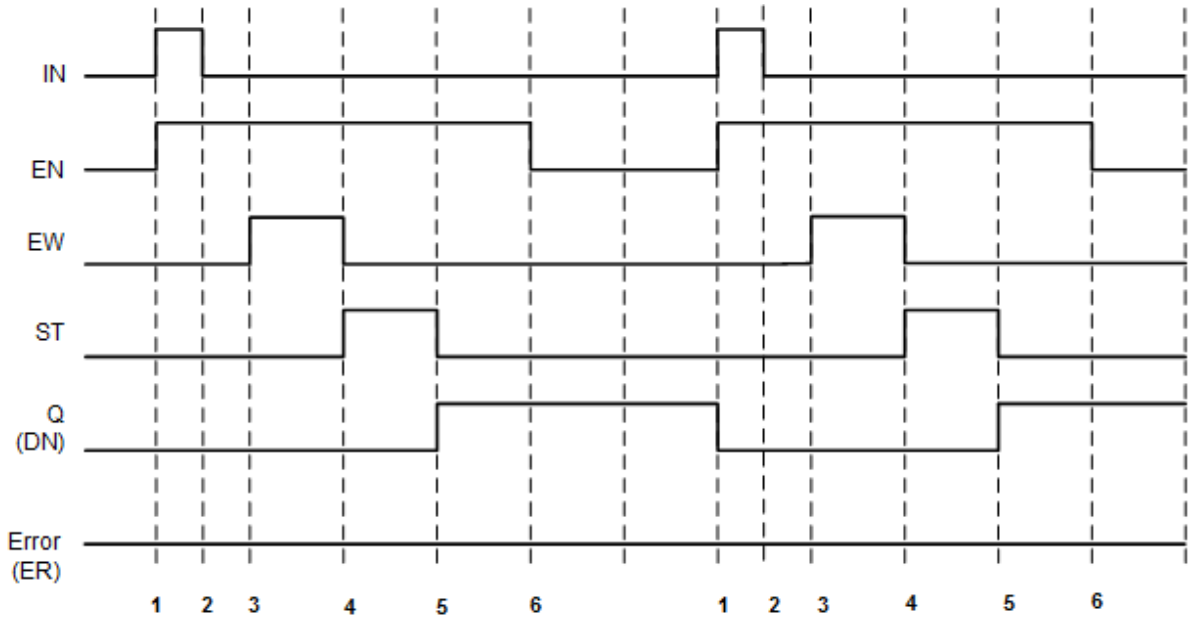
Diagramme de temporisation d'exécution des messages (ligne = FALSE)

Le tableau suivant décrit les conditions de message et le statut de bit pour chaque étape d'exécution identifiée dans le diagramme de temporisation lorsque la condition de ligne passe sur FALSE pendant l'exécution.

Étape	Description du message	Statut du bit
1	La condition de ligne passe sur TRUE. L'exécution du message est activée.	Le bit EN bit est mis à 1. Tous les autres bits sont effacés.

Étape	Description du message	Statut du bit
2	La condition de ligne passe sur FALSE. L'exécution du message se poursuit.	
3	La mémoire tampon du contrôle de message est acquise. À ce moment, les données d'entrée (c'est-à-dire le paramètre « données » pour l'écriture de messages) sont copiées pour transmission. Les modifications appliquées ultérieurement aux données d'entrée n'apparaîtront pas dans le message transmis.	Le bit EW est défini sur 1.
4	La transmission de message débute.	Le bit EW est effacé (0). Le bit ST est défini sur 1.
5	La réponse au message est reçue.	Le bit ST est effacé (0). Le bit DN est défini sur 1.
6	Le message est de nouveau scruté après l'étape 5.	Le bit EN est effacé.

Diagramme de temporisation pour (Ligne = FALSE)



Processus d'exécution des messages (erreur)

Le tableau suivant décrit les conditions de message et le statut de bit de chaque étape d'exécution identifiée dans le diagramme de temporisation, lorsqu'une erreur se produit durant l'exécution.

Étape	Description du message	Statut du bit
1	La condition de ligne passe sur TRUE. L'exécution du message est activée.	Le bit EN bit est mis à 1. Tous les autres bits sont effacés.
2	La mémoire tampon du message est acquise.	Le bit EW est défini sur 1.
3	La transmission de message débute.	Le bit EW est effacé (0). Le bit ST est défini sur 1.
4	Le délai de transmission du message a expiré.	Les bits EW et ST ne changent pas.
4-6	La condition de ligne passe sur FALSE.	Le bit EN est effacé. Le bit ER est mis à 1.

Diagramme de temporisation d'exécution des messages (erreur)

Le diagramme de temporisation suivant montre ce qui arrive généralement lorsqu'une erreur se produit pendant l'exécution.

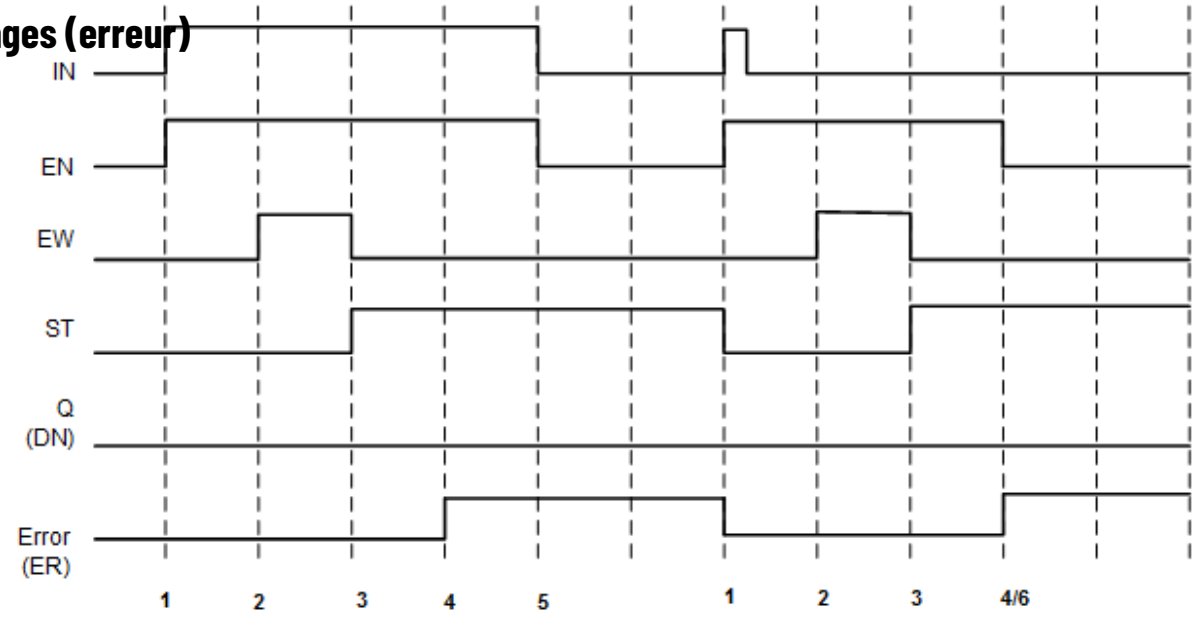


Diagramme de temporisation d'exécution des messages (erreur de configuration PCCC)

Le tableau suivant décrit les conditions de message et le statut de bit de chaque étape d'exécution identifiée dans le diagramme de temporisation, lorsqu'une erreur de configuration PCCC se produit durant l'exécution.

Étape	Description du message	Statut du bit
1	La condition de ligne passe sur TRUE. L'exécution du message est activée.	Le bit EN est défini sur 1.
2	La mémoire tampon du message est acquise.	Le bit EW est défini sur 1.
3	Des éléments non valides existent dans PCCCCfg.	Le bit ER est défini sur 1 avant que le bit ST ne soit défini sur 1. Les bits EW et ST ne changent pas.
4	La condition de ligne passe sur FALSE.	Les bits EN, EW et ER sont mis à zéro.

Le diagramme suivant montre la condition d'erreur typique lors de la modification dans PCCCCfg du nombre d'éléments, passé à 17, ce qui n'est pas valide.

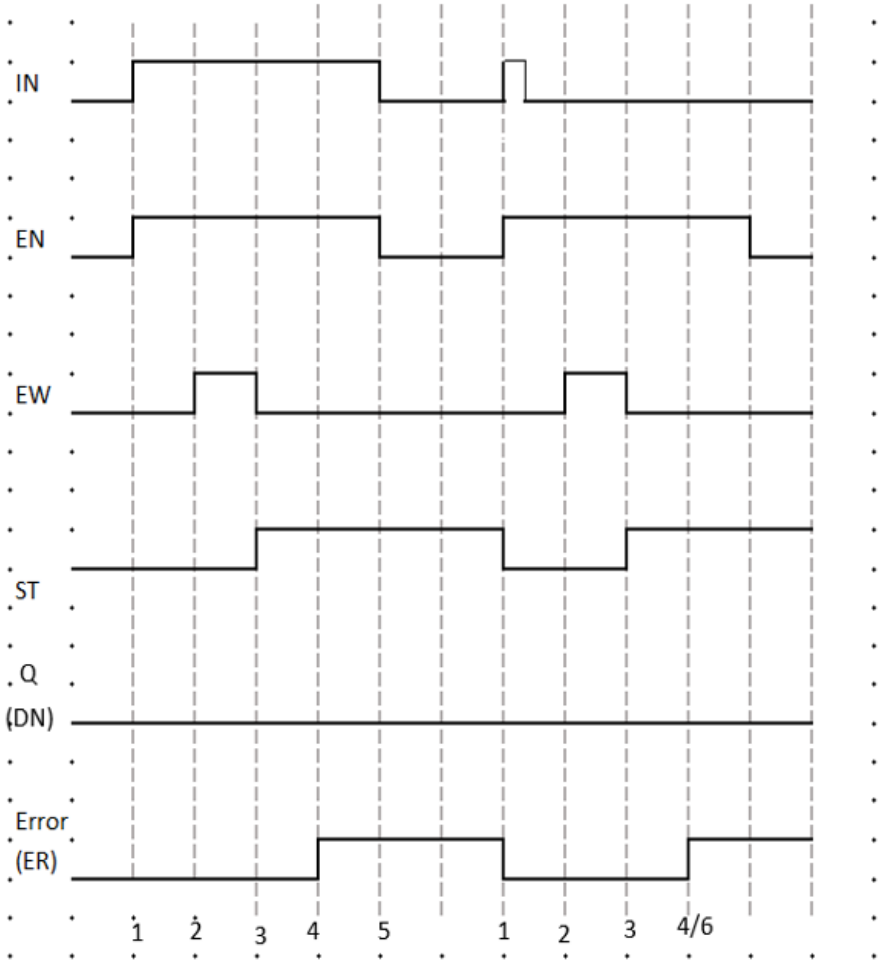
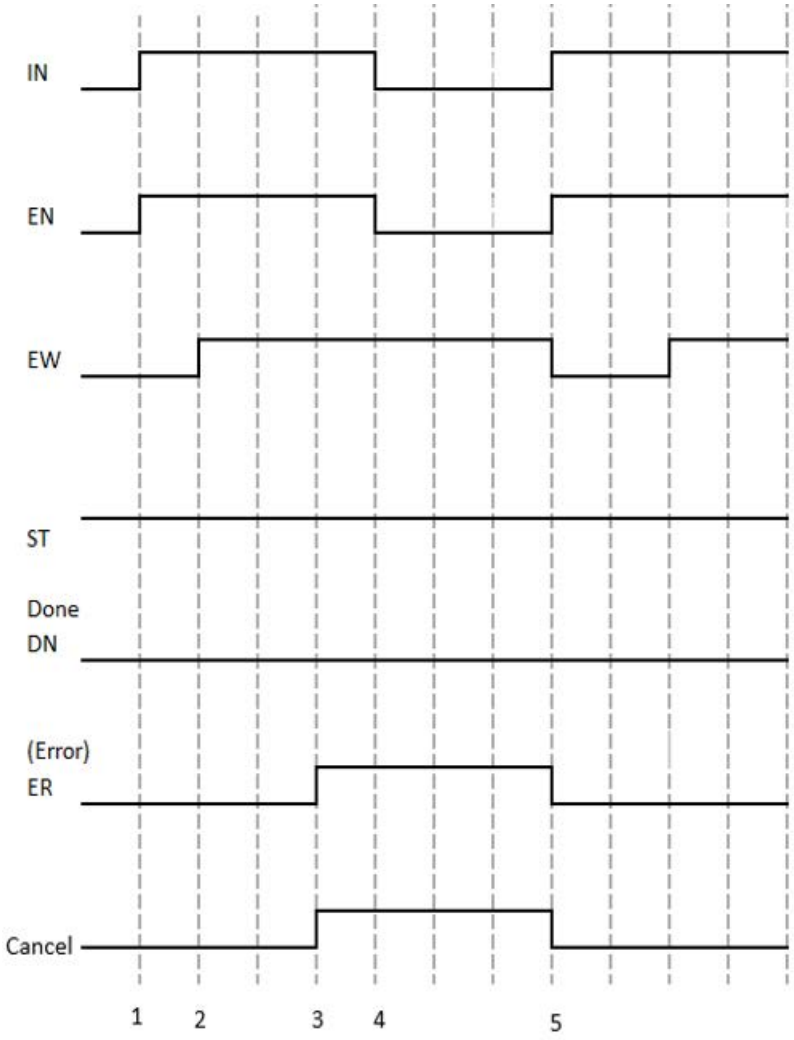


Diagramme de temporisation d'exécution de message (message abandonné)

Le tableau suivant décrit les conditions de message et le statut de bit de chaque étape d'exécution identifiée dans le diagramme de temporisation, lorsqu'un message est abandonné.

Étape	Description du message	Statut du bit
1	La condition de ligne passe sur TRUE. L'exécution du message est activée.	Le bit EN est défini sur 1. Tous les autres bits sont mis à zéro.
2	La mémoire tampon du message est acquise.	Le bit EW est défini sur 1.
3	L'exécution du message est abandonnée.	Le bit ER est défini sur 1. Les bits EW et ST ne changent pas.
4	La condition de ligne passe sur FALSE.	Le bit EN est effacé.
5	La condition de ligne passe sur TRUE.	L'entrée Abandonner est mise à zéro. Le bloc de fonctions commence à s'exécuter.

Diagramme de temporisation lors de l'abandon d'un message



Utilisation des blocs de fonction de communication (message)

Cette section fournit des détails précis et des exemples sur l'utilisation des instructions de communication sur la page 161 dans les programmes logiques. Consultez les rubriques suivantes pour obtenir des détails et des exemples sur l'utilisation des blocs fonctionnels MSG_CIPGENERIC sur la page 215 et MSG_CIPSYMBOLIC dans la création de programmes sur la page 223.

Configuration des valeurs de données d'objet pour les messages explicites (MSG_CIPGENERIC)

Pour utiliser le bloc de fonctions MSG_CIPGENERIC sur la page 163 pour les messages explicites, configurez le paramètre AppCfg avec les valeurs correctes.

Pour plus d'informations sur la communication des messages

Il y a plusieurs sources d'informations qui couvrent l'implémentation et l'utilisation de la communication des messages, y compris l'aide de Connected Components Workbench, les manuels d'utilisation et la bibliothèque de documents de Rockwell Automation.

Le tableau suivant répertorie les sources d'informations complémentaires disponibles relatives à la communication des messages.

Source d'informations	Description	Comment trouver les informations
Manuel d'utilisation de votre appareil de communication	Contient des informations importantes sur la messagerie et des informations précises concernant la configuration des blocs fonctionnels de message.	Menu Aide de l'application Connected Components Workbench
Adaptateur EtherNet/IP 22-COMM-E FRN 1.xxx, Annexe C	Fournit des informations sur les objets EtherNet/IP accessibles à l'aide des messages explicites.	Menu Aide de l'application Connected Components Workbench
Spécification EtherNet/IP	Indique les objets à inclure dans chaque appareil CIP : Objet identité, objet routeur de messages et objet réseau.	Site Web ODVA (http://www.odva.org)
Contrôleurs programmables Micro800 : Mise en route de la messagerie client CIP	Contient des instructions de démarrage rapide pour l'utilisation de la messagerie CIP GENERIC et CIP Symbolic sur les contrôleurs logiques programmables (PLC) Micro830 et Micro850.	Bibliothèque de documents de Rockwell Automation

Pour accéder aux manuels d'utilisation et aux guides de démarrage rapide, procédez comme suit :

1. Pour accéder aux guides de démarrage rapide, cliquez sur **Afficher l'aide** dans le menu **Aide**.
2. Double-cliquez sur **Connected Components Workbench**.
3. Double-cliquez sur **Mise en route de Connected Components Workbench**.
4. Pour accéder aux manuels des variateurs, dans le menu **Aide**, cliquez sur **Manuels d'utilisation** pour afficher la boîte de dialogue Manuels.
5. Cliquez sur le signe plus (+) à côté de Variateurs pour développer la catégorie, puis développez la classe jusqu'à localiser le manuel souhaité.
6. Double-cliquez sur le titre du manuel pour ouvrir le fichier .pdf correspondant.
7. Pour accéder aux manuels EtherNet/IP, dans le menu **Aide**, cliquez sur **Manuels d'utilisation** pour afficher la boîte de dialogue Manuels.
8. Cliquez sur le signe plus (+) situé à côté de **Variateurs** pour développer la catégorie, puis développez la classe Périphériques PowerFlex Classe 4.
9. Double-cliquez sur le Manuel d'utilisation de l'adaptateur EtherNet/IP 22-COMM-E pour ouvrir le fichier .pdf correspondant.

Pour accéder aux manuels de la bibliothèque de documents de Rockwell Automation, procédez comme suit :

1. Accédez à http://literature.rockwellautomation.com.
2. Pour accéder aux manuels d'utilisation dans des langues autres que l'anglais, sélectionnez la langue dans la liste déroulante Langue de publication (à droite).
3. Saisissez le numéro complet ou partiel du catalogue d'appareils dans la zone **Recherche**. Par exemple, saisissez 2080-LC30 pour consulter les manuels d'utilisation de Micro830.

4. Dans la zone **Recherche**, tapez le numéro de catalogue complet ou partiel de l'appareil. Par exemple, saisissez 2080-LC30 pour consulter les manuels d'utilisation de Micro830.

Données de l'objet de registre CIP

Les blocs fonctionnels MSG_CIPGENERIC utilisent les données d'objet de registre CIP dans le paramètre AppCfg. Les données d'objet incluent les éléments suivants :

- Code de classe
- Instance
- Attribut d'instance
- Service

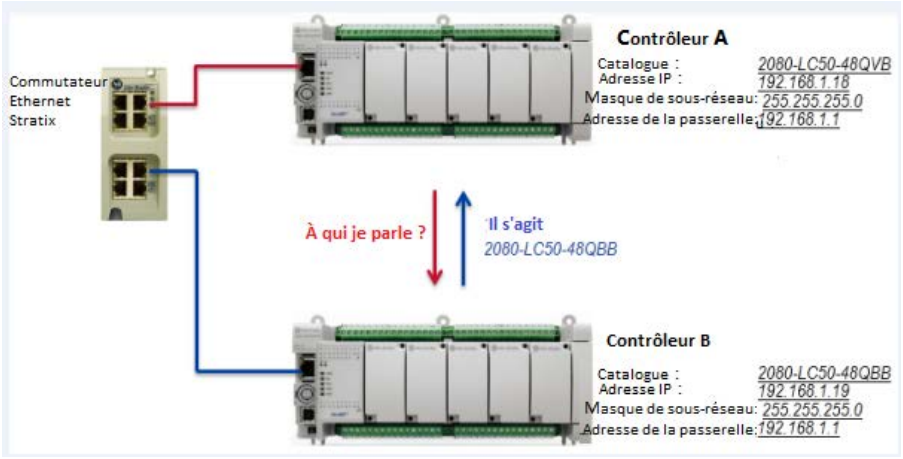
Valeurs du paramètre MSG_CIPGENERIC AppCfg

Utilisez les valeurs de l'objet de registre CIP dans vos variables d'entrée pour configurer les paramètres du bloc fonctionnel MSG_CIPGENERIC. L'image suivante illustre les valeurs de données d'objet de registre CIP utilisées dans les paramètres du bloc de fonctions MSG_CIPGENERIC.

Nom	Alias	Type de donnée	Dimens	Valeur Proje	Valeur initiale
MSG_ReadDrive		MSG_CIPGENE		...	...
MyAppCfg		CIPAPPCFG		...	...
MyAppCfg.Service		USINT			14
MyAppCfg.Class		UINT			7
MyAppCfg.Instance		UDINT			4
MyAppCfg.Attribute		UINT			4
MyAppCfg.MemberCnt		USINT			

Exemple : création d'un programme de messagerie MSG_CIPGENERIC pour lire des données depuis un contrôleur

Cet exemple explique comment créer un programme de message qui récupère les informations de catalogue du contrôleur B, auprès du contrôleur A, en utilisant un bloc de fonctions MSG_CIPGENERIC et un bloc de fonctions COP.



Réalisez les tâches suivantes pour créer un programme de messagerie MSG_CIPGENERIC qui lit des informations sur un contrôleur différent.

Non	Tâche
1	Identification des valeurs initiales pour les variables d'entrée (MSG_CIPGENERIC) sur la page 216
2	Ajout de variables et d'un bloc fonctionnel MSG_CIPGENERIC sur la page 217
3	Configuration des valeurs initiales pour les variables sur la page 218
4	Ajout d'un contact et d'une bobine sur la page 220
5	Ajout d'un bloc fonctionnel COP, de variables et d'un contact (MSG_CIPGENERIC) sur la page 221
6	Vérification de la configuration IP sur le contrôleur B sur la page 222

Identification des valeurs initiales pour les variables d'entrée (MSG_CIPGENERIC)

Procédez comme suit pour ajouter des variables d'entrée et leurs valeurs initiales, puis obtenez les valeurs d'objet d'identité pour configurer les valeurs initiales du paramètre AppCfg.

Pour ajouter des variables d'entrée et des valeurs initiales :

1. Dans le menu **Aide**, cliquez sur **Manuels d'utilisation**.
2. Développez la section Variateurs et recherchez le manuel d'utilisation de l'adaptateur de communication que vous utilisez (Manuel d'utilisateur de l'adaptateur EtherNet/IP).
3. Double-cliquez sur le titre du manuel pour l'ouvrir.
4. Consultez les titres de l'Annexe afin de localiser la section contenant des informations sur les objets EtherNet/IP accessibles à l'aide de messages explicites (Annexe C).
5. Rendez-vous à la section d'Annexe correspondante et identifiez le type d'objet associé à votre message explicite (objet identité).
6. Déterminez les valeurs initiales requises pour les paramètres AppCfg, en fonction du type d'informations que vous souhaitez récupérer.

Exemples de paramètre AppCfg et de données d'objet Ethernet/IP

Le tableau suivant indique les données d'objet spécifiques Ethernet/IP utilisées pour lire les informations de catalogue à partir d'un contrôleur.

Exemple de variable d'entrée	Paramètre AppCfg	Option de données d'objet Ethernet/IP	Description	Valeur initiale
MyAppCfg.Service	Service	Code de service	Mettre en œuvre pour la classe = Oui Mettre en œuvre pour l'instance = Oui Obtenir un attribut simple	14 (0x0E en hexadécimal)
MyAppCfg.Class	Classe	Code de classe	Classe d'objet EtherNet/IP = Objet d'identité	01
MyAppCfg.Instance	Instance	Instances	22-COMM-E	01
MyAppCfg.Attribute	Attribut	Attribut d'instance	Obtenir le nom du produit et l'évaluation dans une CHAÎNE COURTE	07

Ajout de variables et d'un bloc fonctionnel MSG_CIPGENERIC

Pour ajouter un bloc de fonctions MSG_CIPGENERIC pour un programme de langage à relais et ajouter des variables d'entrée pour le bloc de fonctions, procédez comme suit.

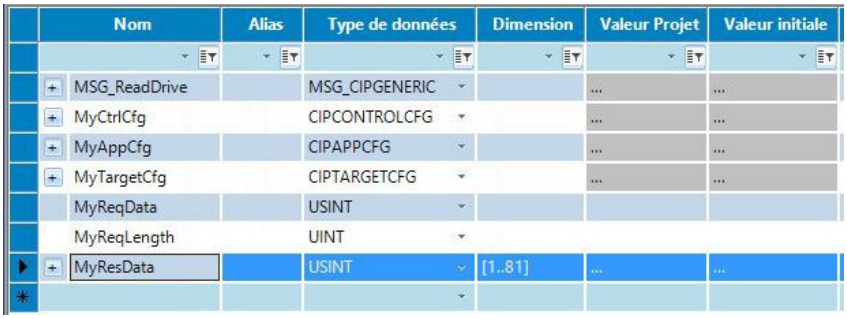
Pour ajouter un bloc de fonctions MSG_CIPGENERIC, procédez comme suit :

1. Ajoutez un contrôleur :
   - Développez le dossier **Contrôleurs** et le dossier Micro850 pour afficher tous les contrôleurs Micro850.
   - Double-cliquez sur un contrôleur (2080-LC50-48QVB) pour l'ajouter au **Gestionnaire de projet**.
2. Ajout d'un programme de logique à relais
   - Dans le **Gestionnaire de projet**, cliquez avec le bouton droit de la souris sur **Programmes**, cliquez sur **Ajouter**, puis cliquez sur **Nouvelle LD : Logique à relais**.
   - Cliquez avec le bouton droit de la souris sur l'icône du langage à relais dans le **Gestionnaire de projet**, cliquez sur **Renommer**, puis saisissez CIPExplicitMessage.
   - Double-cliquez sur le programme de langage à relais dans le **Gestionnaire de projet**, afin d'afficher le POU LD dans l'éditeur de langage.
3. Ajoutez le bloc fonctionnel MSG_CIPGENERIC :
   - Dans la **Boîte à outils**, sélectionnez l'élément **Bloc d'instructions** et faites-le glisser sur la ligne de logique à relais, afin d'afficher le **Sélecteur de bloc d'instruction**.
   - Dans le champ Rechercher, saisissez **MSG** pour afficher les blocs fonctionnels de message.
   - Saisissez **MSG_ReadDrive** dans le champ **Instance**.
   - Double-cliquez sur **MSG_CIPGENERIC** pour ajouter une instance du bloc fonctionnel à la logique à relais.
4. Ajoutez des variables d'entrée locales MSG_CIPGENERIC :
   - Dans le **Gestionnaire de projet**, double-cliquez sur **Variables locales** pour afficher la page **Variables locales**.
   - Dans la page **Variables**, ajoutez les variables et les types de données répertoriés dans le tableau.

Paramètre	Nom de variable	Type de données
CtrlCfg	MyCtrlCfg	CIPCONTROLCFG
AppCfg	MyAppCfg	CIPAPPCFG
TargetCfg	MyTargetCfg	CIPTARGETCFG
ReqData	MyReqData	USINT
ReqLength	MyReqLength	UINT
ResData	MyResData	USINT (tableau)

5. Pour la variable MyResData, double-cliquez sur **Dimension** et modifiez la taille du tableau [1..81].

   La page **Variables** doit être similaire à celle illustrée dans l'image suivante.



	Nom	Alias	Type de données	Dimension	Valeur Projet	Valeur initiale
	MSG_ReadDrive		MSG_CIPGENERIC		...	...
	MyCtrlCfg		CIPCONTROLCFG		...	...
	MyAppCfg		CIPAPPCFG		...	...
	MyTargetCfg		CIPTARGETCFG		...	...
	MyReqData		USINT			
	MyReqLength		UINT			
▶	MyResData		USINT	[1..81]	...	...

Configuration des valeurs initiales pour les variables

Observez la procédure suivante pour ajouter des valeurs initiales aux variables d'entrée précédemment créées, puis affecter les variables au paramètre d'entrée approprié du bloc fonctionnel MSG_CIPGENERIC.

Pour affecter des variables à MSG_CIPGENERIC, procédez comme suit :

1. Pour configurer des valeurs initiales pour la variable d'entrée MyCtrlCfg, procédez comme suit :
   - Dans la page **Variables locales**, développez MyCtrlCfg pour afficher les paramètres correspondants.
   - Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale	Commentaires
MyCtrlCfg.Cancel	Laisser vide	Sans objet.
MyCtrlCfg.TriggerType	0	Il n'est nécessaire de récupérer le numéro de catalogue qu'une seule fois.
MyCtrlcfg.StrMode	Laisser vide	Sans objet.

2. Configuration des valeurs initiales pour la variable d'entrée MyAppCfg
   - Dans la page **Variables locales**, développez MyAppCfg pour afficher les paramètres correspondants.
   - Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale
MyAppCfg.Service	14
MyAppCfg.Class	01
MyAppCfg.Instance	01
MyAppCfg.Attribute	07

3. Configuration des valeurs initiales pour la variable d'entrée MyTargetCfg

- Dans la page **Variables locales**, développez MyTargetCfg pour afficher les paramètres correspondants.
- Saisissez les valeurs suivantes dans la colonne **Valeur initiale** pour chaque paramètre.

Paramètre	Valeur initiale	Commentaires
MyTargetCfg.Path	'4,192.168.100.4'	Le premier « 4 » indique que le message est envoyé au port Ethernet intégré. 192.168.100.4 est l'adresse IP de l'interface Ethernet du variateur.
MyTargetCfg.CipConnMode	0	La valeur Non connecté est préférée pour les messages CIP.
MyTargetCfg.UcmmTimeout	Vide	Les messages non connectés ont un délai de temporisation par défaut de 3 000 millisecondes si leurs Valeurs initiales restent vides.
MyTargetCfg.ConnMsgTimeout	Vide	Les messages connectés ont un délai de temporisation par défaut de 3 000 millisecondes si leurs Valeurs initiales restent vides.
MyTargetCfg.ConnClose	FALSE	Pour les messages connectés, la connexion CIP peut être coupée immédiatement après la fin de l'instruction de message, en définissant la valeur initiale sur TRUE.

Les paramètres de la page **Variables** doivent être similaires à ceux illustrés dans l'image suivante.

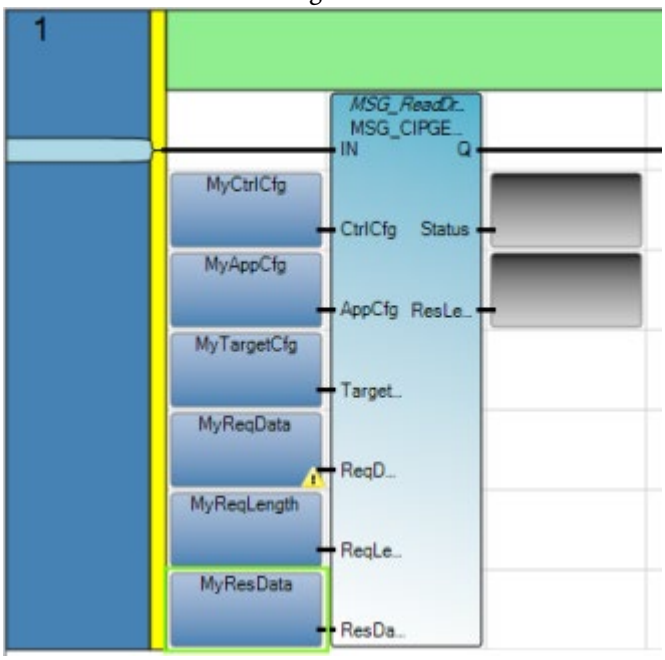
Nom	Alias	Type de données	Dimension	Valeur Projet	Valeur initiale
MSG_ReadDrive		MSG_CIPGENERIC		...	...
MyCtrlCfg		CIPCONTROLCFG		...	...
MyCtrlCfg.Cancel		BOOL			
MyCtrlCfg.TriggerType		UDINT			0
MyCtrlCfg.StrMode		USINT			
MyAppCfg		CIPAPPCFG		...	...
MyAppCfg.Service		USINT			14
MyAppCfg.Class		UINT			01
MyAppCfg.Instance		UDINT			01
MyAppCfg.Attribute		UINT			07
MyAppCfg.MemberCnt		USINT			
MyAppCfg.MemberId		CIPMEMBERID		...	...
MyTargetCfg		CIPTARGETCFG		...	...
MyTargetCfg.Path		STRING			'4,192.168.100.4
MyTargetCfg.CipConnMode		USINT			0
MyTargetCfg.UcmmTimeout		UDINT			
MyTargetCfg.ConnMsgTimeout		UDINT			
MyTargetCfg.ConnClose		BOOL			FALSE
MyReqData		USINT			
MyReqLength		UINT			
MyResData		USINT	[1..81]	...	...
MyResData[1]		USINT			

4. Affectation de variables aux paramètres
   - Dans le POU de logique à relais, cliquez sur la section supérieure du bloc d'entrée des variables afin d'afficher la liste déroulante des variables.
   - Dans cette liste, associez chaque paramètre d'entrée à la variable d'entrée appropriée, indiquée dans le tableau suivant.

Paramètre	Variable d'entrée	Commentaires
CtrlCfg	MyCtrlCfg	Le numéro de catalogue ne doit être récupéré qu'une seule fois ; la valeur initiale de MyCtrlCfg.TriggerType est donc définie sur 0.
AppCfg	MyAppCfg	Les valeurs initiales sont déterminées en consultant les valeurs de données d'objet pour Service, Classe, Instance et Attribut.
Target	MyTargetCfg	Les valeurs initiales servent à la configuration de l'appareil cible.
ReqData	MyReqData	Ce message étant de type Lecture, il n'existe aucune donnée de demande. Les paramètres ReqData ne sont pas utilisés.

Paramètre	Variable d'entrée	Commentaires
ReqLength	MyReqLength	Ce message étant de type Lecture, il n'existe aucune donnée de demande. Les paramètres MyReqLength ne sont pas utilisés.
ResData	MyResData	La chaîne du numéro de catalogue est stockée dans le tableau, au format de chaîne courte ODVA. Le premier élément de tableau définit la longueur de chaîne et le reste stocke la valeur hexadécimale du caractère de chaîne. Le nombre maximal de caractères autorisé est 80, plus l'élément de longueur ; MyResData est donc défini comme tableau 1 dimension avec 81 éléments.

L'instance du bloc de fonctions MSG_CIPGENERIC doit être similaire à celle illustrée dans l'image suivante.



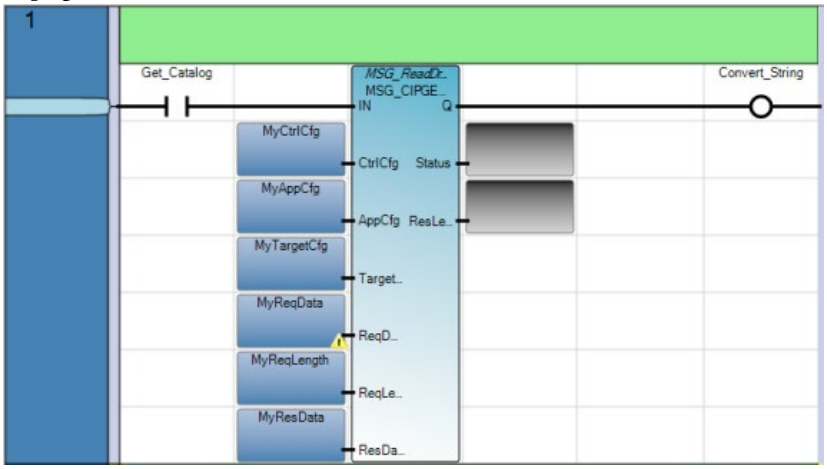
Ajout d'un contact et d'une bobine

Suivez les étapes suivantes pour ajouter une bobine et un contact à l'instruction MSG_CIPGENERIC, qui convertit les informations de catalogue dans un format de chaîne compréhensible.

Pour ajouter une bobine à MSG_CIPGENERIC :

1. Dans la **Boîte à outils**, sélectionnez **Contact normalement ouvert** et faites-le glisser sur la gauche de l'entrée du bloc de fonctions MSG_CIPGENERIC, sur la première ligne de programme de schéma à relais.
2. Dans le **Sélecteur de variable**, saisissez **Get_Catalog** dans le champ Nom du contact.
3. Dans la **Boîte à outils**, sélectionnez **Bobine directe** et faites-le glisser sur la droite de la sortie du bloc de fonctions MSG_CIPGENERIC, sur la première ligne de programme de schéma à relais.
4. Dans le **Sélecteur de variable**, saisissez **Convert_String** dans le champ Nom de la bobine.

La première ligne de votre programme de logique à relais pour la messagerie MSG_CIPGENERIC doit être similaire à celle illustrée dans la page suivante.



Ajout d'un bloc fonctionnel COP, de variables et d'un contact (MSG_CIPGENERIC)

Suivez les étapes suivantes pour ajouter un bloc de fonctions COP, des variables et un contact. L'instruction COP est utilisée pour convertir des données du type de données de la source (par exemple, DINT ou REAL) dans le type de données de la destination. Dans cet exemple, les informations de catalogue seront converties dans un format de chaîne compréhensible.

Pour ajouter un bloc de fonctions COP, procédez comme suit :

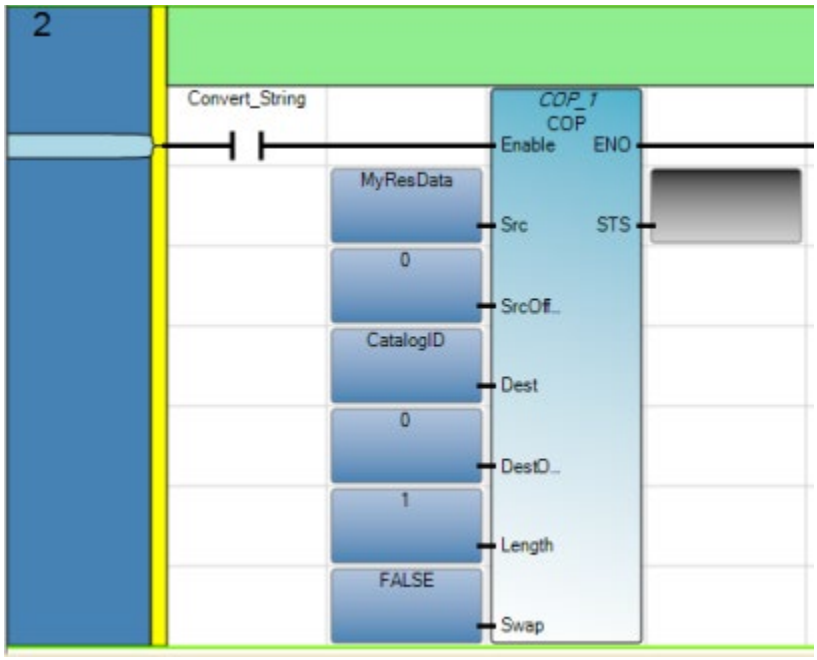
1. Dans la **Boîte à outils**, sélectionnez l'élément **Ligne de programme** et faites-le glisser pour le déposer directement sous la première ligne de programme de schéma à relais afin d'ajouter une deuxième ligne.
2. Ajoutez le bloc fonctionnel COP :
   - Dans la **Boîte à outils**, sélectionnez l'élément **Bloc** et faites-le glisser sur la deuxième ligne de programme de schéma à relais afin d'ouvrir le **Sélecteur de bloc d'instruction**.
   - Double-cliquez sur **COP** pour ajouter une instance du bloc fonctionnel à la logique à relais.
3. Ajouter des variables d'entrée locales pour COP :
   - Dans le **Gestionnaire de projet**, double-cliquez sur **Variables locales** pour afficher la page **Variables locales**.
   - Sur la page **Variables locales**, ajoutez les variables et les types de données répertoriés dans le tableau suivant.

Paramètre	Nom de la variable	Type de données
Src	MyResData	Tableau USINT
SrcOffset	0	UINT
Dest	CatalogID	Tableau STRING
DestOffset	0	UINT
Length	1	UINT
Swap	FALSE	BOOLÉEN

4. Pour la variable CatalogID, double-cliquez sur **Dimension** et modifiez la taille du tableau en [1..1].
5. Ajoutez un contact :
   - Dans la **Boîte à outils**, sélectionnez l'élément **Contact normalement ouvert** et faites-le glisser sur la gauche de l'entrée du bloc de fonctions COP sur la deuxième ligne de programme de schéma à relais.
   - Dans le **Sélecteur de variable**, sélectionnez la variable **Convert_String** pour le contact.

Résultat

La deuxième ligne de votre programme de logique à relais pour la messagerie MSG_CIPGENERIC doit être similaire à celle illustrée dans la page suivante.



Vérification de la configuration IP sur le contrôleur B

Observez la procédure suivante pour vérifier que les paramètres d'adresse IP sont corrects sur le contrôleur B.

Pour vérifier l'adresse IP, procédez comme suit :

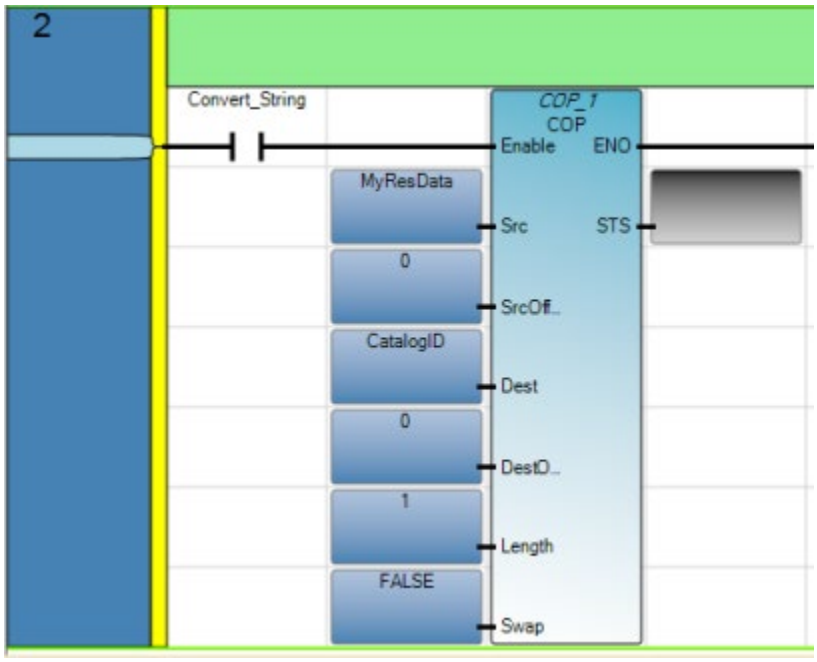
1. Ouvrez l'espace de travail d'application pour le contrôleur :
2. Dans le **Gestionnaire de projets**, double-cliquez sur le contrôleur pour ouvrir son espace de travail.
3. Dans l'espace de travail du contrôleur, développez la section **Ethernet** de l'**arborescence du Contrôleur**, puis cliquez sur Protocole Internet pour afficher la page de configuration du contrôleur.
4. Vérifiez que les paramètres de l'adresse IP sont corrects, tels qu'indiqués dans le tableau suivant.

Option de configuration IP	Valeur
Adresse IP	192.168.1.19

Option de configuration IP	Valeur
Masque de sous-réseau	255.255.255.0
Adresse de la passerelle	192.168.1.1

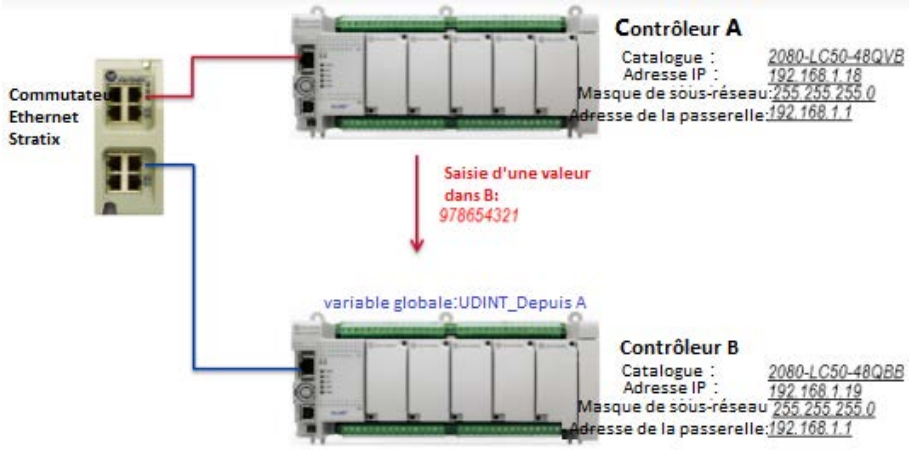
Résultats

Les options Protocole Internet de la page de configuration de votre contrôleur doivent être similaires à celles illustrés dans l'image suivante.



Exemple : création d'un programme de messagerie MSG_CIPSYMBOLIC pour écrire une valeur dans une variable

Cet exemple indique comment créer un programme de message pour écrire une valeur dans une variable globale du contrôleur B, à partir du contrôleur A.



Pour créer un programme de messagerie MSG_CIPSYMBOLIC qui permet d'écrire une valeur dans une variable globale sur un autre contrôleur, effectuez les tâches suivantes.

Non	Tâche
1	Ajout d'un bloc fonctionnel COP, de variables , et d'un contact (MSG_CIPSYMBOLIC) sur la page 224
2	Ajout d'un opérateur Égal à et d'une bobine sur la page 225
3	Ajout d'un bloc fonctionnel MSG_CIPSYMBOLIC, de variables et d'un contact sur la page 226

Non	Tâche
4	Vérification de la configuration IP sur le contrôleur B sur la page 222
5	Création d'une variable globale pour le contrôleur B sur la page 230
6	Examen de l'ensemble des résultats du programme sur la page 230

Ajout d'un bloc fonctionnel COP, de variables et d'un contact (MSG_CIPSYMBOLIC)

L'instruction COP est utilisée pour convertir les données saisies dans le type de données de la destination, afin qu'elles soient compatibles avec la variable du contrôleur.

Pour ajouter un bloc de fonctions COP, procédez comme suit :

1. Ajoutez un contrôleur :
   - Développez le dossier **Contrôleurs** et le dossier Micro850 pour afficher tous les contrôleurs Micro850.
   - Double-cliquez sur un contrôleur (2080-LC50-48QVB) pour l'ajouter au **Gestionnaire de projet**.
2. Ajout d'un programme de logique à relais
   - Dans le **Gestionnaire de projet**, cliquez avec le bouton droit de la souris sur **Programmes**, cliquez sur **Ajouter**, puis cliquez sur **Nouvelle LD : Logique à relais**.
   - Cliquez avec le bouton droit de la souris sur l'icône du langage à relais dans le **Gestionnaire de projet**, puis cliquez sur **Renommer**, puis saisissez **CIPSymbolicMessage**.
   - Double-cliquez sur le programme de langage à relais dans le **Gestionnaire de projet**, afin d'afficher le POU LD dans l'éditeur de langage.
3. Ajoutez un bloc fonctionnel COP :
   - Dans la **Boîte à outils**, sélectionnez l'élément **Bloc d'instruction** et faites-le glisser sur la première ligne de programme de schéma à relais afin d'ouvrir le **Sélecteur de bloc d'instruction**.
   - Dans le champ Recherche, saisissez **COP**, puis double-cliquez sur **COP** pour ajouter une instance du bloc fonctionnel à la logique à relais.
4. Ajout de variables COP et de valeurs initiales :
   - Dans le Programme Unitaire de langage à relais, double-cliquez sur **Variables locales** pour ouvrir la page des **variables locales**.
   - Sur la page **Variables**, ajoutez les variables et les types de données répertoriés dans le tableau ci-dessous.
5. Créez des tableaux :
   - Pour ValueToWrite, double-cliquez dans le champ **Dimension** et modifiez la taille du tableau [1..1].

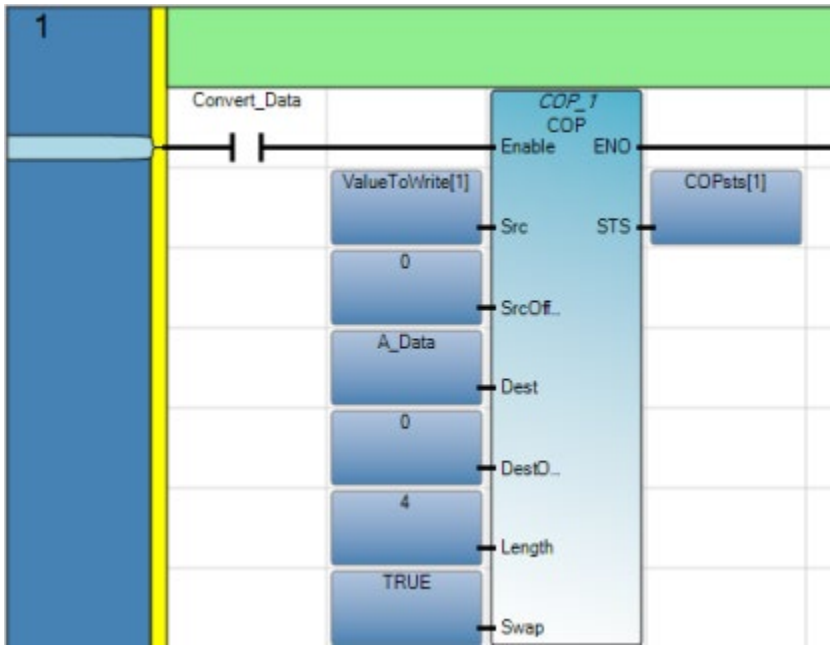
- Pour A_Data, double-cliquez dans le champ **Dimension** et modifiez la taille du tableau [1..4].

6. Saisissez les données indiquées dans la colonne **Valeur** du tableau ci-dessous dans le champ **Valeur initiale**, pour chaque variable.
7. Ajoutez un contact :
   - Dans la **Boîte à outils**, sélectionnez **Contact normalement ouvert** et faites-le glisser sur la gauche de l'entrée du bloc de fonctions COP, sur la première ligne de programme de schéma à relais.
   - Dans le **Sélecteur de variable**, affectez une variable appelée **Convert_Data** au contact.

Utilisez les variables définies dans le tableau pour le bloc de fonctions COP.

Paramètre	Nom de la variable	Type de données
Src	ValueToWrite	Tableau UDINT Valeur initiale : 987654321
SrcOffset	0	UINT
Dest	A_Data	Tableau USINT
DestOffset	0	UINT
Length	4	UINT
Swap	TRUE	BOOLÉEN
STS	COPsts	Tableau UINT

La première ligne de votre programme de logique à relais pour la messagerie MSG_CIPSYMBOLIC doit être similaire à celle illustrée dans la page suivante.



Ajout d'un opérateur Égal à et d'une bobine

L'instruction Égal à est utilisée pour déclencher l'écriture d'une valeur si la conversion du type de données a réussi. Pour ajouter un opérateur Égal à (=), les variables et une bobine, procédez comme suit.

Pour ajouter un opérateur Égal à, procédez comme suit :

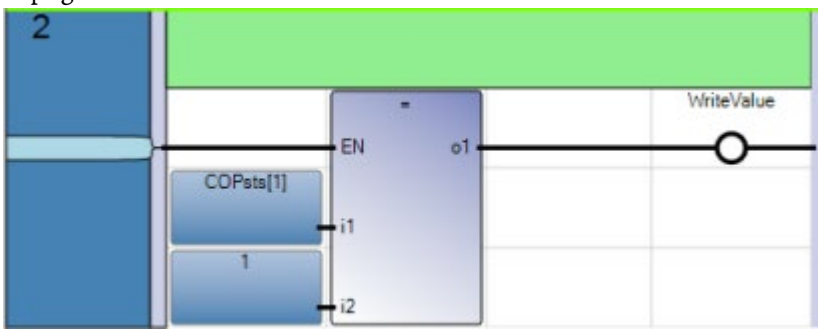
1. Dans la **Boîte à outils**, sélectionnez l'élément **Ligne de programme** et faites-le glisser pour le déposer directement sous la première ligne de programme de schéma à relais afin d'ajouter une deuxième ligne.
2. Ajoutez un opérateur Égal à :
   - Dans la **Boîte à outils**, sélectionnez l'élément **Bloc d'instruction** et faites-le glisser sur la deuxième ligne de programme de schéma à relais afin d'ouvrir le **Sélecteur de bloc d'instruction**.
   - Dans le champ Recherche, saisissez le signe « = » et double-cliquez sur **« = »** pour ajouter une instance de l'opérateur à la logique à relais.
3. Pour ajouter des variables Égal à, procédez comme suit :
   - Dans le programme unitaire LD, double-cliquez sur une variable pour afficher le **Sélecteur de variable**.
   - Dans le **Sélecteur de variable**, affectez les noms de variable tels qu'indiqués dans le tableau suivant.

Paramètre	Nom de la variable
i1	COPsts
i2	1

4. Pour ajouter une bobine à l'opérateur Égal à, procédez comme suit :

   Dans la **Boîte à outils**, sélectionnez **Bobine directe** et faites-le glisser sur la droite de la sortie de l'opérateur Égal à, sur la deuxième ligne de programme de schéma à relais.
   Dans le **Sélecteur de variable**, saisissez **WriteValue** dans le champ Nom de la bobine.

La deuxième ligne de votre programme de logique à relais pour la messagerie MSG_CIPGENERIC doit être similaire à celle illustrée dans la page suivante.



Ajout d'un bloc fonctionnel MSG_CIPSYMBOLIC, de variables et d'un contact

Pour ajouter un bloc de fonctions MSG_CIPSYMBOLIC, des variables d'entrée et un contact à une logique à relais, procédez comme suit.

Pour ajouter un bloc de fonctions et des variables, procédez comme suit :

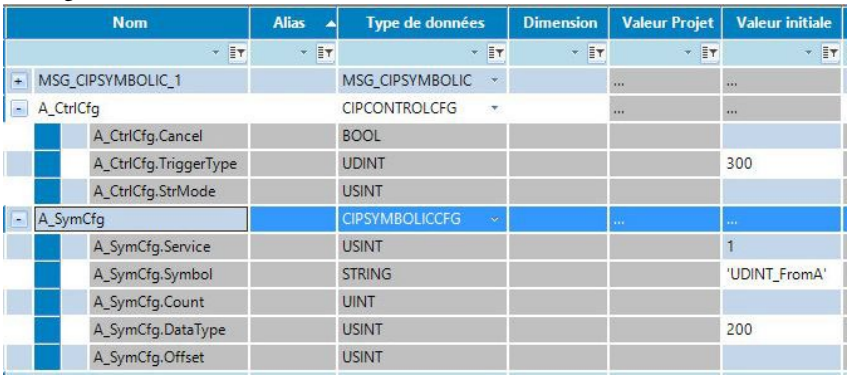
1. Dans la **Boîte à outils**, sélectionnez l'élément **Ligne de programme** et faites-le glisser, pour le déposer directement sous la deuxième ligne de programme de schéma à relais, afin d'ajouter une troisième ligne de programme.
2. Ajoutez le bloc fonctionnel MSG_CIPSYMBOLIC :
   - Dans la **Boîte à outils**, sélectionnez l'élément **Bloc d'instructions** et faites-le glisser sur la ligne de logique à relais, afin d'afficher le **Sélecteur de bloc d'instruction**.
   - Dans le champ Rechercher, saisissez **MSG** pour afficher les blocs fonctionnels de message.
   - Saisissez **MSG_WriteVariable** dans le champ **Instance**.
   - Double-cliquez sur **MSG_CIPSYMBOLIC** pour ajouter une instance du bloc fonctionnel MSG_WriteVariable à la logique à relais.
3. Affectez des noms de variable :
   - Dans le programme unitaire LD, double-cliquez sur une variable pour afficher le **Sélecteur de variable**.
   - Dans le **Sélecteur de variable**, affectez les noms de variable tels qu'indiqués dans le tableau suivant.
4. Pour la variable Données, double-cliquez sur **Dimension** et modifiez la taille du tableau en [1..4].
5. Configurez les valeurs initiales pour les variables locales, ajoutez les valeurs initiales CtrlCfg :
   - Dans la page **Variables locales**, développez le paramètre CtrlCfg pour afficher des paramètres complémentaires.
   - Saisissez les valeurs suivantes dans la colonne Valeur initiale pour chaque paramètre.

Paramètre	Valeur initiale
A_CtrlCfg.Cancel	Laisser vide
A_CtrlCfg.TriggerType	300
A_Ctrlcfg.StrMode	Laisser vide

6. Ajoutez les valeurs initiales SymCfg :
   - Dans la page **Variables locales**, développez le paramètre SymCfg pour afficher des paramètres complémentaires.
   - Saisissez les valeurs suivantes dans la colonne Valeur initiale pour chaque paramètre.

Paramètre	Valeur initiale
A_SymCfg.Service	1
A_SymCfg.Symbol	'UDINT_FromA'
A_SymCfg.Count	Laisser vide
A_SymCfg.DataType	200
A_SymCfg.Offset	Laisser vide

Le sélecteur Variables locales doit être similaire à celui illustré dans l'image suivante.

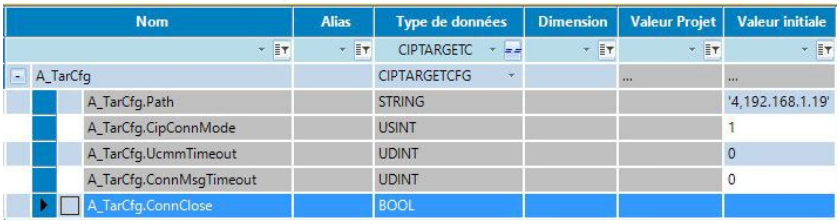


Nom	Alias	Type de données	Dimension	Valeur Projet	Valeur initiale
MSG_CIPSYMBOLIC_1		MSG_CIPSYMBOLIC		...	...
A_CtrlCfg		CIPCONTROLCFG		...	...
A_CtrlCfg.Cancel		BOOL			
A_CtrlCfg.TriggerType		UDINT			300
A_CtrlCfg.StrMode		USINT			
A_SymCfg		CIPSYMBOLICCFG		...	...
A_SymCfg.Service		USINT			1
A_SymCfg.Symbol		STRING			'UDINT_FromA'
A_SymCfg.Count		UINT			
A_SymCfg.DataType		USINT			200
A_SymCfg.Offset		USINT			

7. Configuration des valeurs initiales pour TargetCfg
   - Dans le POU de logique à relais, double-cliquez sur la variable A_TarCfg pour ouvrir le sélecteur des variables globales.
   - Développez le paramètre TargetCfg pour afficher des paramètres complémentaires.
   - Saisissez les valeurs suivantes dans la colonne Valeur initiale pour chaque paramètre.

Paramètre	Valeur initiale
A_TarCfg.Path	'4,192.168.1.19'
A_TarCfg.CipConnMode	1
A_TarCfg.UcmmTimeout	0
A_TarCfg.ConnMsgTimeout	0
A_TarCfg.ConnClose	Laisser vide

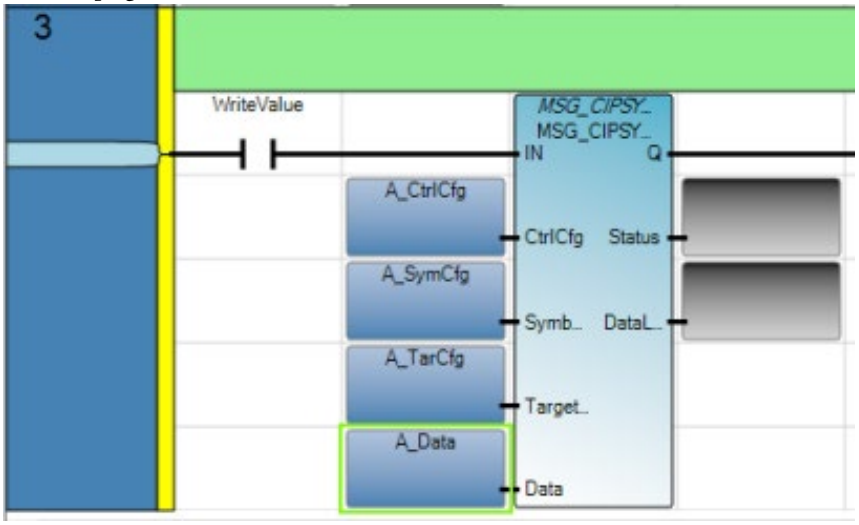
Le sélecteur de variables globales de l'utilisateur doit être similaire à celui illustré dans l'image suivante.



Nom	Alias	Type de données	Dimension	Valeur Projet	Valeur initiale
A_TarCfg		CIPTARGETCFG		...	...
A_TarCfg.Path		STRING			'4,192.168.1.19'
A_TarCfg.CipConnMode		USINT			1
A_TarCfg.UcmmTimeout		UDINT			0
A_TarCfg.ConnMsgTimeout		UDINT			0
A_TarCfg.ConnClose		BOOL			

8. Ajoutez un contact :
   - Dans la **Boîte à outils**, sélectionnez **Contact normalement ouvert** et faites-le glisser sur la gauche de l'entrée du bloc de fonctions MSG_CIPSYMBOLIC, dans la troisième ligne de logique à relais.
   - Dans le **Sélecteur de variable**, affectez la variable **WriteValue** au contact.

La troisième ligne de votre programme de logique à relais pour la messagerie MSG_CIPSYMBOLIC doit être similaire à celle illustrée dans la page suivante.



Vérification de la configuration IP sur le contrôleur B

Observez la procédure suivante pour vérifier que les paramètres d'adresse IP sont corrects sur le contrôleur B.

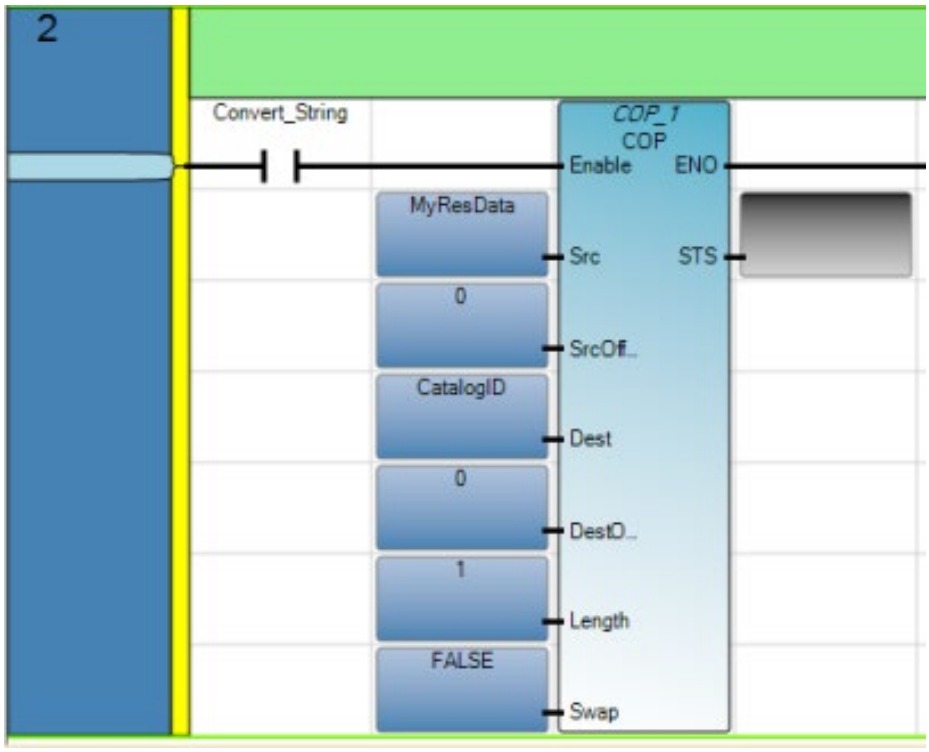
Pour vérifier l'adresse IP, procédez comme suit :

1. Ouvrez l'espace de travail d'application pour le contrôleur :
2. Dans le **Gestionnaire de projets**, double-cliquez sur le contrôleur pour ouvrir son espace de travail.
3. Dans l'espace de travail du contrôleur, développez la section **Ethernet** de l'**arborescence du Contrôleur**, puis cliquez sur Protocole Internet pour afficher la page de configuration du contrôleur.
4. Vérifiez que les paramètres de l'adresse IP sont corrects, tels qu'indiqués dans le tableau suivant.

Option de configuration IP	Valeur
Adresse IP	192.168.1.19
Masque de sous-réseau	255.255.255.0
Adresse de la passerelle	192.168.1.1

Résultats

Les options Protocole Internet de la page de configuration de votre contrôleur doivent être similaires à celles illustrés dans l'image suivante.



Création d'une variable globale pour le contrôleur B

Observez les étapes suivantes afin de créer une variable globale pour le contrôleur B.

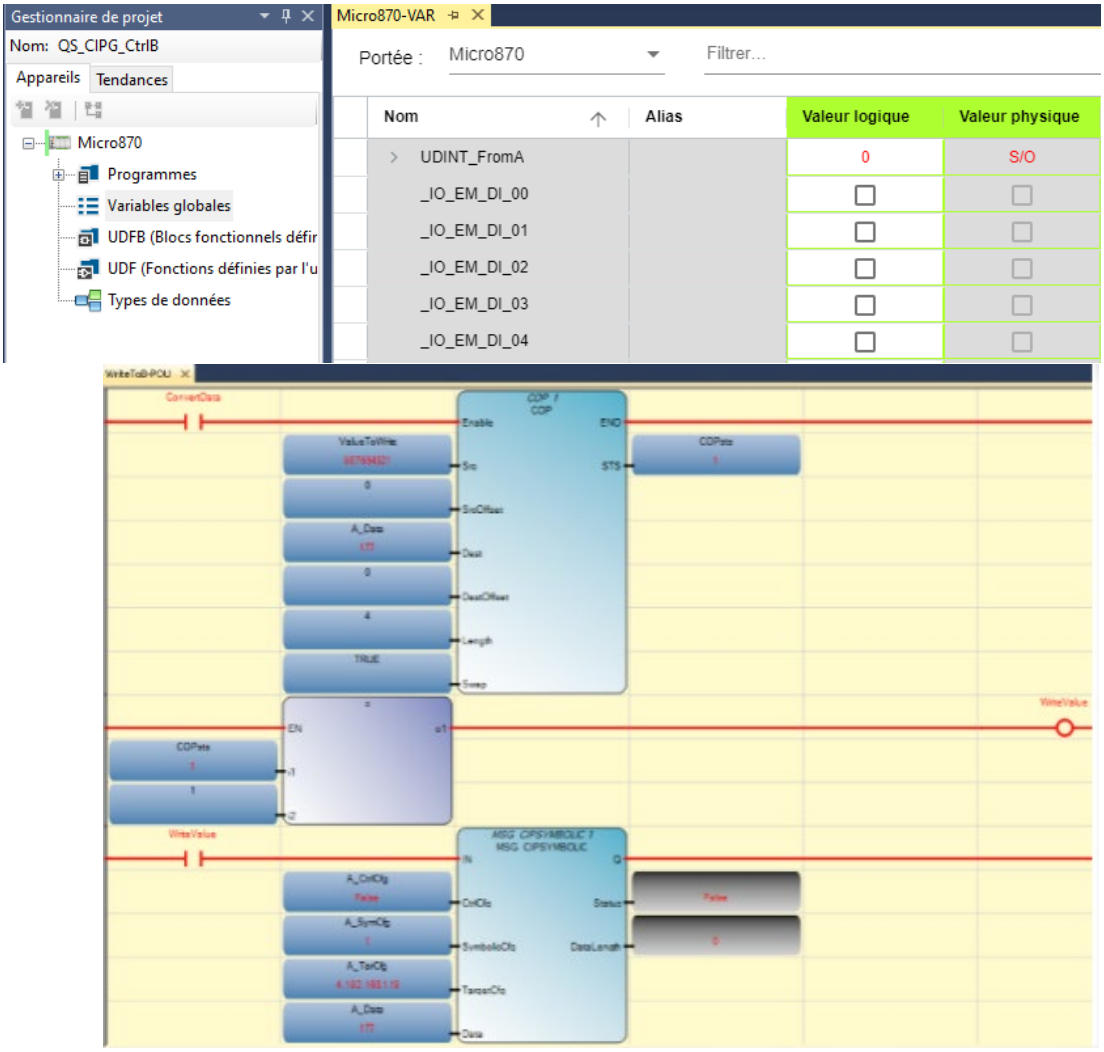
Pour créer une variable globale :

1. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables globales** pour afficher le sélecteur de **variables globales**.
2. Tapez UDINT_FromA dans la colonne **Nom**.
3. Configurez les autres paramètres :
   - Vérifiez que le type de données sélectionné est UDINT.
   - Vérifiez que l'attribut **Lecture/Écriture** est sélectionné.

Le sélecteur de **variables globales** doit être similaire à celui illustré dans l'image suivante.

Examen de l'ensemble des résultats du programme

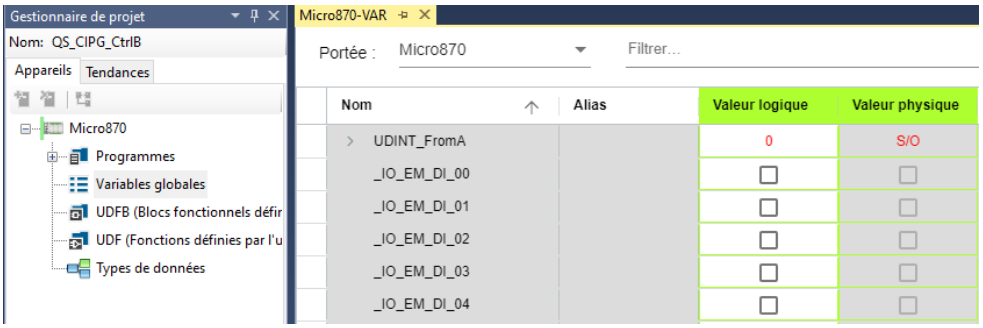
L'exemple suivant présente le programme de messagerie MSG_CIPSYMBOLIC complet, une fois qu'il a été exécuté.



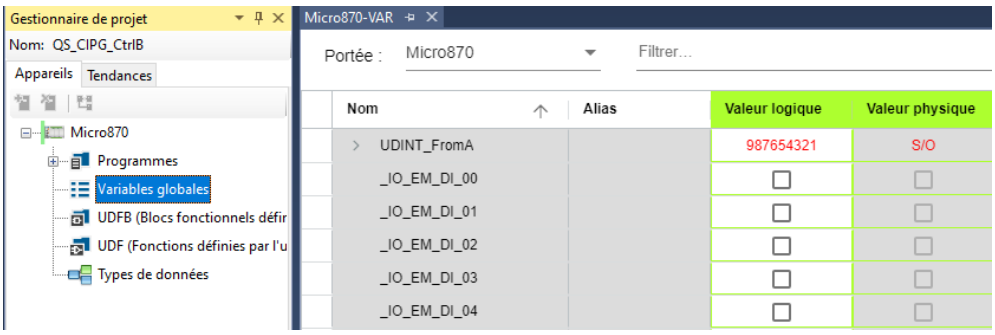
Résultats du contrôleur B

Les deux images suivantes illustrent les valeurs pour le contrôleur B, avant et après que le programme de messagerie a été exécuté.

Avant exécution du programme



Après exécution du programme

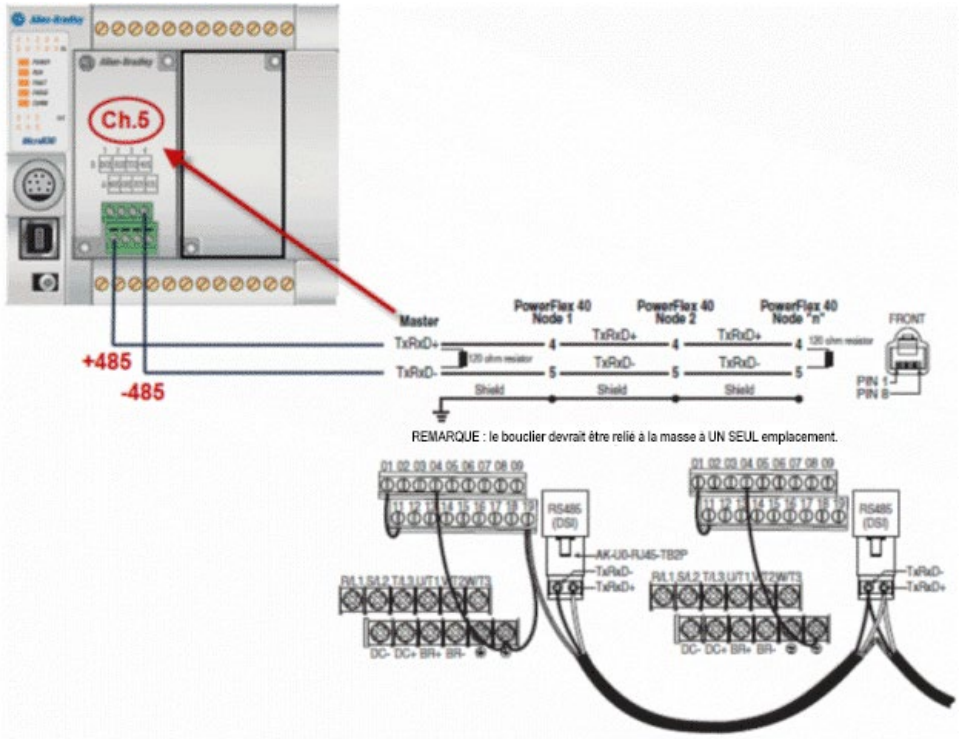


Exemple : configuration d'une communication Modbus pour lire à partir d'un variateur et écrire dans un variateur

Ces exemples décrivent comment configurer la communication Modbus pour lire des données d'état et écrire des données de contrôle dans un variateur PowerFlex 4 à l'aide de l'instruction MSG_MODBUS sur la page 175.

Câblage Micro830

Cet exemple tire parti d'un contrôleur Micro830 avec un module SERIALISOL relié au premier emplacement (Canal 5). Un seul variateur PowerFlex 40 est connecté, mais le schéma ci-dessous montre comment réaliser une connexion multipoint. Pour plus d'informations sur le câblage, consultez le manuel d'utilisation.



AK-U0-RJ45-TB2P est un connecteur RJ45 doté de 2 borniers pour les communications RS485

Exemple de lecture Modbus

L'instruction MSG_MODBUS suivante peut être utilisée pour lire les données de statut à partir du variateur PowerFlex 40.

Statut du variateur

La mention « 1807 » indique que le variateur est :

- Prêt (bit 0 activé)
- Actif (bit 1 activé)
- Avancer (bit 2 activé)
- Avancer en rotation (bit 3 activé)
- Statut de certaines entrées numériques du variateur

« 278 » signifie 27,8 Hz.

Consultez le manuel d'utilisation PowerFlex pour obtenir des informations supplémentaires sur les bits des mots de statut logique, sur les descriptions des codes d'erreur, sur les vitesses de référence et réelles et sur les autres codes de statut.

Configuration de la lecture MSG_MODBUS

L'image suivante illustre les options de variable pour le bloc d'instructions MSG_MODBUS, utilisé pour lire les données de statut à partir d'un variateur PowerFlex 40.

Nom	Type de don	Direction	Dimension
MSG_MODBUS_1	MSG_MODI	Var	
D2_lcfg	MODBUSLC	Var	
D2_lcfg.Channel	UINT	Var	
D2_lcfg.TriggerType	USINT	Var	
D2_lcfg.Cmd	USINT	Var	
D2_lcfg.ElementCnt	UINT	Var	
D2_Tcfg	MODBUSTA	Var	
D2_Tcfg.Addr	UDINT	Var	
D2_Tcfg.Node	USINT	Var	
D2_laddr	MODBUSLC	Var	
D2_laddr[1]	WORD	Var	
D2_laddr[2]	WORD	Var	
D2_laddr[3]	WORD	Var	
D2_laddr[4]	WORD	Var	
D2_laddr[5]	WORD	Var	

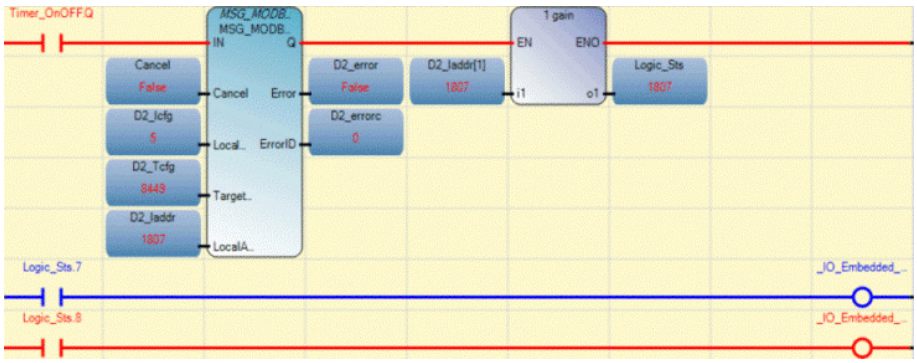
Variables de lecture MSG_MODBUS

Le tableau suivant indique les variables et les valeurs utilisées pour configurer l'instruction MSG_MODBUS afin qu'elle lise les données de statut à partir d'un variateur PowerFlex 4.

Variable	Valeur	Description
*.Channel	5	Canal 5 - emplacement du module SERIALISOL
*.TriggerType	0	Déclenchement sur transition Faux à Vrai
*.Cmd	3	Code de fonction Modbus 03 - Lecture des registres de conservation
*.ElementCnt	4	Longueur
*.Addr	8449	Adresse du mot de statut de la logique PowerFlex + 1
*.Node	2	Adresse du nœud PowerFlex
*_laddr[1]	{data}	Mot du statut logique PowerFlex
*_laddr[2]	{data}	Code d'erreur PowerFlex
*_laddr[3]	{data}	Vitesse de référence PowerFlex
*_laddr[4]	{data}	Retour de vitesse PowerFlex (vitesse réelle)

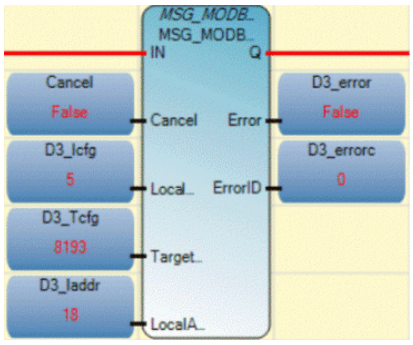
Exemple d'instruction MOV

Les images suivantes illustrent un exemple d'utilisation de l'instruction MOV, dans le but de déplacer la valeur de tableau *_I[1] dans un mot, ce qui vous permet d'accéder directement aux bits, individuellement.



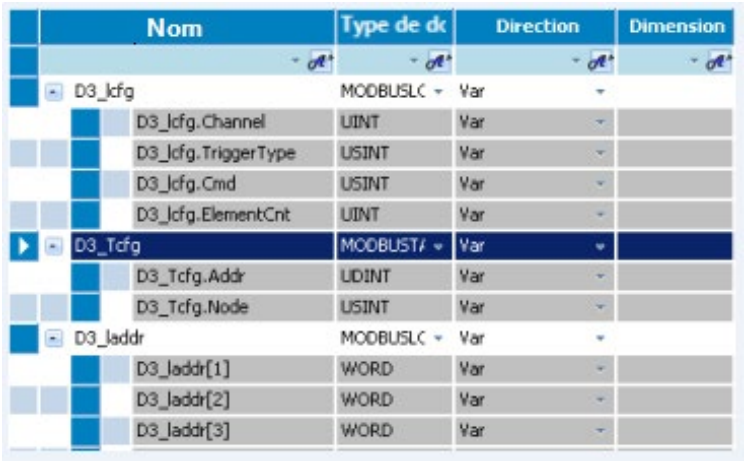
Exemple d'écriture Modbus

L'instruction MSG_MODBUS suivante est utilisée pour écrire les données de contrôle dans un variateur PowerFlex 40.



Configuration d'écriture MSG_MODBUS

L'image suivante illustre les variables et les valeurs utilisées pour configurer l'instruction MSG_MODBUS afin qu'elle écrive les données de contrôle dans un variateur PowerFlex 4.



Variables d'écriture MSG_MODBUS

Le tableau suivant présente la liste des variables et des valeurs et décrit le rôle de chaque variable.

Variable	Valeur	Description
*.Channel	5	Canal 5 - emplacement du module SERIALISOL
*.TriggerType	0	Déclenchement sur transition Faux à Vrai
*.Cmd	16	Code de fonction Modbus 16 - Ecriture des registres de conservation
*.ElementCnt	2	Longueur
*.Addr	8193	Adresse du mot de statut de la logique PowerFlex + 1
*.Node	2	Adresse du nœud PowerFlex
*_Iaddr[1]	{data}	Mot de la commande logique PowerFlex
*_Iaddr[2]	{data}	Mot de la vitesse de référence PowerFlex

Prise en charge du protocole de communication

Les blocs fonctionnels MSG_CIP prennent en charge différents protocoles de communication, tels que décrits dans la présente section.

Prise en charge du bloc de fonctions pour les protocoles de communication des messages.

Protocole de messagerie	Support de communication	Utilise ce bloc fonctionnel
Modbus/RTU, client et serveur	Via un port série configuré comme RTU Modbus.	MSG_MODBUS sur la page 175
Modbus/TCP, client et serveur	Par Ethernet au lieu d'un port série.	MSG_MODBUS2 sur la page 181
IP Ethernet, client et serveur	Via un canal Ethernet intégré.	MSG_CIPSYMBOLIC sur la page 171 MSG_CIPGENERIC sur la page 163
CIP en série, client et serveur	Câble Ethernet ou série.	MSG_CIPSYMBOLIC sur la page 171
ASCII	Via un port RS-232 configuré avec un pilote ASCII.	Voir Instructions de port série ASCII sur la page 103.

RTU Modbus

Modbus est un protocole de communication semi-duplex maître-esclave, qui permet à un même maître de communiquer avec 247 appareils esclaves (au maximum). Le maître de réseau Modbus lit et écrit les bits et les registres. Pour prendre en charge RTU Modbus, le port série doit être configuré en tant que RTU Modbus.

Pour plus d'informations sur le protocole Modbus, consultez les Spécifications du protocole Modbus (disponibles sur le site http://www.modbus.org).

Modbus/TCP

Le protocole de communication Modbus/TCP Server fait appel aux mêmes fonctions de mappage Modbus que RTU Modbus, mais il est pris en charge via Ethernet plutôt que sur un port série.

Le contrôleur Micro850 prend en charge jusqu'à 16 connexions Modbus TCP Server simultanées. Aucune configuration de protocole n'est requise, autre que la configuration de la table de mappage Modbus.

EtherNet/IP

Les contrôleurs Micro820 et Micro850 prennent en charge jusqu'à 16 connexions serveur EtherNet/IP simultanées par le biais d'un canal de communication Ethernet intégré sur la page 237. Ce canal peut être utilisé pour connecter un contrôleur Micro850 à divers appareils, via un réseau local, à un taux de transfert de 10/100 Mbits/s.

Protocole CIP (Common Industrial Protocol)

Le protocole CIP est un protocole de couche transport et application, utilisé pour l'envoi de messages sur les réseaux EtherNet/IP, ControlNet et DeviceNet et qui permet aux appareils IHM de se connecter facilement à un contrôleur Micro830 ou Micro850.

Messages CIP explicites

Le protocole CIP est utilisé pour créer des messages explicites. La fonction Explicit Messaging (configuration, collecte de données et diagnostics) est utilisée pour transférer les données n'exigeant pas une mise à jour en continu. Elle est prise en charge par le biais de la messagerie client CIP, via le bloc fonctionnel MSG_CIPGENERIC, et par le biais de la messagerie client CIP Symbolic via le bloc fonctionnel MSG_CIPSYMBOLIC.

Série CIP

La configuration CIP en série utilise le protocole DF1 en duplex intégral et offre une connexion point à point entre deux appareils. Elle allie la transparence des données (spécification ANSI - X3.28-1976 de l'American National Standards Institute, sous-catégorie D1) à une transmission bidirectionnelle simultanée avec réponses intégrées (sous-catégorie F1)

Les contrôleurs Micro830 et Micro850 prennent en charge la configuration Série CIP via une connexion RS-232 avec des équipements externes, tels que des ordinateurs utilisant le logiciel RSLinx Classic, les terminaux PanelView Component (micrologiciel versions 1.70 et supérieures) ou d'autres contrôleurs prenant en charge la configuration Série CIP en duplex intégral DF1, par exemple les contrôleurs ControlLogix et CompactLogix disposant de ports série intégrés.

Le pilote de port série peut être configuré comme CIP en série, RTU Modbus, ASCII ou Shutdown via l'arborescence de configuration de l'appareil.

Canaux de communication intégrés

Les contrôleurs Micro820, Micro830 et Micro850 possèdent les canaux de communication intégrés supplémentaires suivants :

- Un port mixte RS-232/485 non isolé
- Un port de programmation USB non isolé (Micro830 et Micro850 uniquement)
- Un port Ethernet RJ-45 (Micro820 et Micro850 uniquement)

Instructions de comparaison

Utilisez les instructions de comparaison pour comparer des valeurs entières, réelles, de temps, de date et de chaîne en vous basant sur une expression ou une instruction de comparaison donnée.

Instruction	Description
(=) Égal à sur la page 239	Compare la première entrée et la deuxième entrée pour déterminer l'égalité. Pour les types de données entières, réelles, de temps, de date et de chaîne.
(>) Supérieur à sur la page 242	Compare les valeurs d'entrée pour déterminer si la première est supérieure à la seconde.
(>=) Supérieur ou égal à sur la page 243	Compare les valeurs d'entrée pour déterminer si la première est supérieure ou égale à la seconde.
(<) Inférieur à sur la page 244	Compare les valeurs d'entrée pour déterminer si la première est inférieure à la seconde.
(<=) Inférieur ou égal à sur la page 246	Compare les valeurs d'entrée pour déterminer si la première est inférieure ou égale à la seconde.
(<>) Non égal sur la page 247	Compare les valeurs d'entrée pour déterminer si la première n'est pas égale à la seconde.

Équal

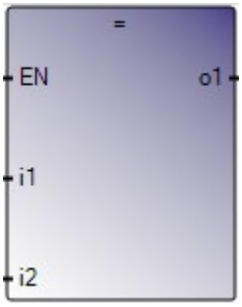
Effectue une opération qui compare la première entrée et la seconde entrée pour déterminer si celles-ci sont égales pour des valeurs entières, réelles, de temps, de date et de chaîne.

Détails de l'opération :

- Le test d'égalité des valeurs de temps n'est pas recommandé pour les blocs d'instruction TON, TP et TOF.
- Les données réelles ne sont pas recommandées lors du test d'égalité des valeurs. En effet, les chiffres des opérations mathématiques sont arrondis différemment de ceux répertoriés sur l'affichage de sortie des variables. Dès lors, deux valeurs de sortie peuvent être indiquées comme égales sur l'affichage, malgré un résultat négatif lors du test de comparaison. Par exemple, les valeurs 23,500001 et 23,499999 s'afficheront toutes les deux sous la forme 23,5 sur l'écran de saisie des variables, mais elles ne sont pas indiquées comme égales dans le contrôleur.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de la fonction VRAI : exécution de la comparaison d'égalité. FAUX : aucune comparaison. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toutes les entrées doivent être du même type de données. L'entrée Temps s'applique aux langages de texte structuré, de schéma à relais et de diagramme de bloc de fonctions. Le type de données REAL n'est pas recommandé.
i2	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL	

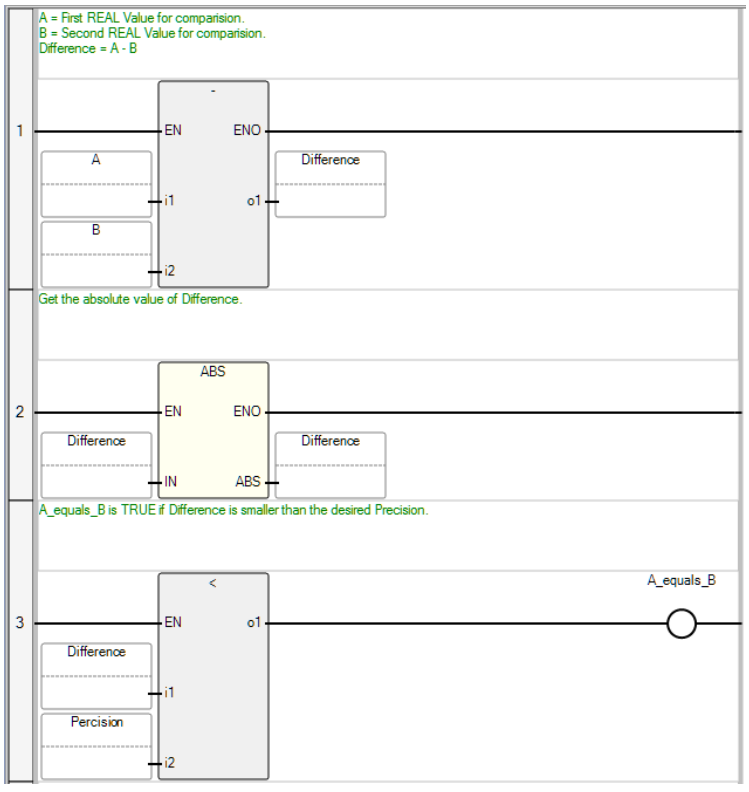
Paramètre	Type de paramètre	Type de données	Description
		LREAL TIME DATE STRING	
o1	Sortie	BOOL	TRUE si i1 = i2.

Exemple de comparaison de valeurs réelles avec les opérateurs de soustraction (-) ABS et d'infériorité (<)

Il est recommandé d'éviter d'utiliser le type de données REAL dans le cadre de la vérification de l'égalité de valeurs en raison des différences d'arrondi des chiffres. Deux valeurs de sortie peuvent être indiquées comme égales sur l'affichage de Connected Components Workbench, et générer un résultat négatif lors du test de comparaison.

Par exemple, les valeurs 23,500001 et 23,499999 s'afficheront toutes deux sous la forme 23,5 sur l'écran de saisie des variables, mais ne seront pas indiquées comme égales dans le contrôleur.

Pour vérifier l'égalité de deux valeurs REAL, vous pouvez utiliser une instruction Soustraction afin d'obtenir la différence entre ces valeurs, puis déterminer si cette différence est inférieure à une valeur de précision définie. L'exemple de programme LD ci-dessous illustre la comparaison de deux valeurs REAL.



Exemple de texte structuré Égal à (=)

(* Équivalence ST : *)

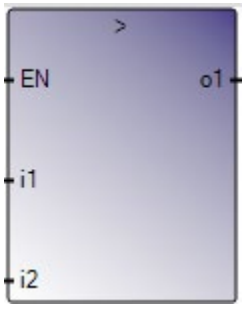
```
aresult := (10 = 25); (* aresult est FAUX *)
mresult := ('ab' = 'ab'); (* mresult est VRAI *)
```

Supérieur à

Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est supérieure à la seconde.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de la comparaison d'entrée. FAUX : aucune comparaison. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toutes les entrées doivent être du même type de données.
i2	Entrée	SINT USINT - BYTE INT	

Paramètre	Type de paramètre	Type de données	Description
		UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	
o1	Sortie	BOOL	VRAI si i1 > i2.

Exemple de texte structuré Supérieur à (>)

(* Équivalence ST : *)

```
aresult := (10 > 25); (* aresult est FAUX *)
mresult := ('ab' > 'a'); (* mresult est VRAI *)
```

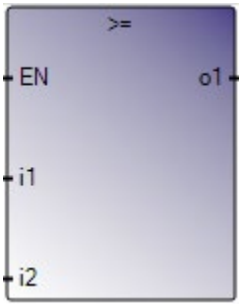
Supérieur ou égal à

Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est égale ou supérieure à la seconde.

Pour TON, TP et TOF, le test d'égalité des valeurs de temps n'est pas recommandé.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de la comparaison d'entrée. FAUX : aucune comparaison. Ne s'applique qu'aux programmes de langage à relais.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toutes les entrées doivent être du même type de données. L'entrée Temps s'applique aux langages de texte structuré, de schéma à relais et de diagramme de bloc de fonctions.
i2	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	
o1	Sortie	BOOL	VRAI si i1 >= i2.

Exemple de texte structuré Supérieur ou égal à (>=)

(* Équivalence ST : *)

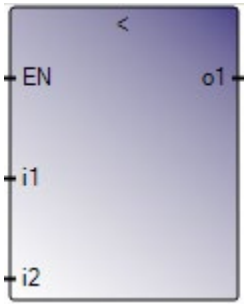
```
aresult := (10 >= 25); (* aresult est FAUX *)
mresult := ('ab' >= 'ab'); (* mresult est VRAI *)
```

Inférieur à

Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est inférieure à la seconde.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de la comparaison d'entrée. FAUX : aucune comparaison. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toutes les entrées doivent être du même type de données.
i2	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE	

Paramètre	Type de paramètre	Type de données	Description
		STRING	
o1	Sortie	BOOL	TRUE si i1 < i2.

Exemple de texte structuré Inférieur à (<)

(* Équivalence ST : *)

```
aresult := (10 < 25); (* aresult est VRAI *)
mresult := ('z' < 'B'); (* mresult est FAUX *)
```

(* Equivalence IL : *)

LD	10
LT	25
ST	aresult
LD	'z'
LT	'B'
ST	mresult

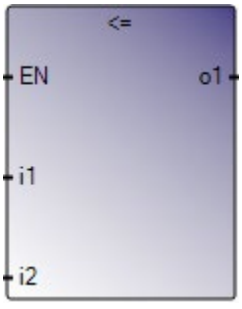
Inférieur ou égal à

Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est inférieure ou égale à la seconde.

Pour TON, TP et TOF, le test d'égalité des valeurs de temps n'est pas recommandé.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de la comparaison d'entrée. FAUX : aucune comparaison. Ne s'applique qu'aux programmes de langage à relais.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	Toutes les entrées doivent être du même type de données. L'entrée Temps s'applique aux langages de texte structuré, de schéma à relais et de diagramme de bloc de fonctions.
i2	Entrée	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	Sortie	BOOL	VRAI si i1 <= i2.

Exemple de texte structuré Inférieur ou égal à (<=)

(* Équivalence ST : *)

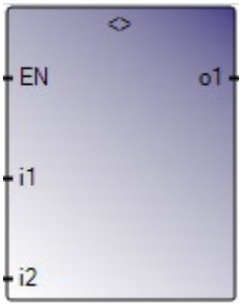
```
aresult := (10 <= 25); (* aresult est TRUE *)
mresult := ('ab' <= 'ab'); (* mresult est TRUE *)
```

Non égal

Compare les valeurs d'entrée entières, réelles, de temps, de date et de chaîne pour déterminer si la première est égale ou non à la seconde.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de comparaison actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT	Toutes les entrées doivent être du même type de données.

Paramètre	Type de paramètre	Type de données	Description
		WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	
i2	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	
o1	Sortie	BOOL	VRAI : si première variable <> seconde

Exemple de texte structuré d'inégalité (<>)

(* Équivalence ST : *)

```
aresult := (10 <> 25); (* aresult est VRAI *)
mresult := ('ab' <> 'ab'); (* mresult est FAUX *)
```

Instructions de compteur

Utilisez les instructions de compteur pour contrôler les opérations en fonction du nombre d'événements.

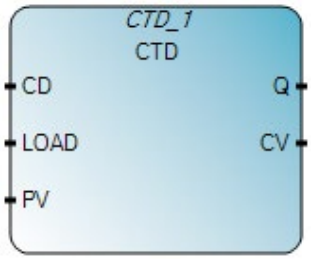
Instruction	Description
CTD sur la page 249	Décompte des entiers, 1 par 1, à partir d'une valeur donnée jusqu'à 0.
CTU sur la page 251	Décompte des entiers, 1 par 1, de 0 jusqu'à une valeur donnée.
CTUD sur la page 252	Décompte des entiers, 1 par 1, de 0 jusqu'à une valeur donnée ou d'une valeur donnée jusqu'à 0.

CTD (comptage décroissant)

Décompte des entiers, 1 par 1, à partir d'une valeur donnée jusqu'à 0.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

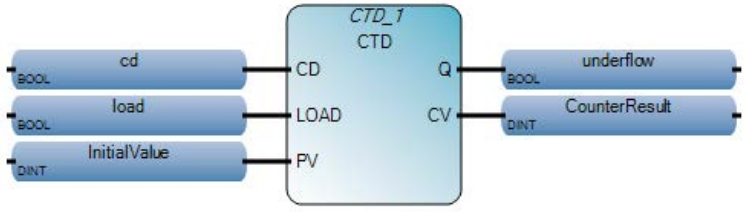
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



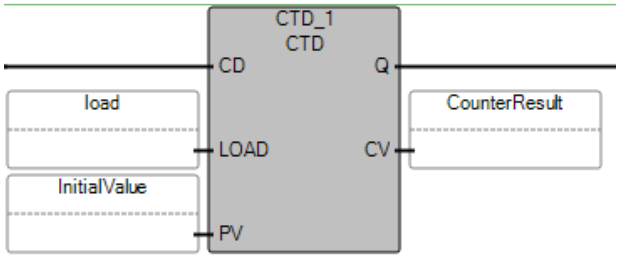
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
CD	Entrée	BOOL	Décompte descendant. VRAI : front montant détecté, comptage décroissant par incréments de un. FAUX : front descendant détecté, conserve la valeur de compteur à la même valeur.
LOAD	Entrée	BOOL	Chargement vérifie la valeur PV par rapport à la valeur du comptage décroissant. VRAI : définit CV = PV. FAUX : poursuit l'incrémentation du comptage décroissant par un.
PV	Entrée	DINT	Valeur maximale programmée du compteur.
Q	Sortie	BOOL	Indique si l'instruction du comptage décroissant a généré un nombre inférieur ou égal à la valeur maximale du compteur. VRAI : résultat du compteur <= 0 (condition de débordement). FAUX : résultat du compteur > 0.
CV	Sortie	DINT	Valeur actuelle du compteur.

Exemple de Diagramme de Bloc Fonction CTD



Exemple de schéma à relais CTD



Exemple de texte structuré CTD

```
1  InitialValue := 10;
2  CTD_1(cd, load, InitialValue);
```

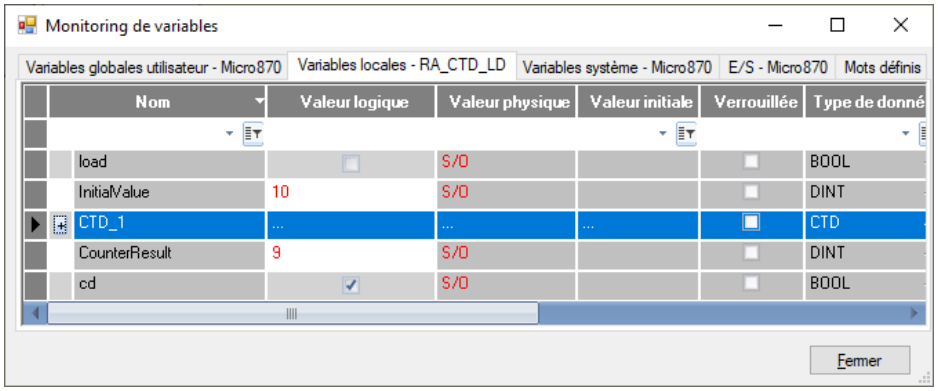
(* Équivalence ST : CTD1 est une instance du bloc *)

CTD1(trigger, load_cmd, 100) ;

débordement négatif := CTD1.Q ;

résultat := CTD1.CV ;

Résultats

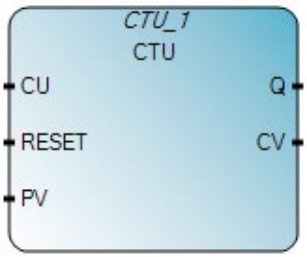


CTU (comptage croissant)

Décompte 1 par 1 de CTU (entiers) de 0 jusqu'à une valeur donnée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

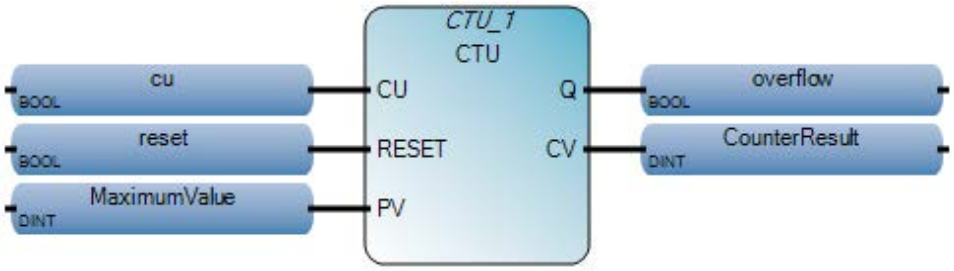
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



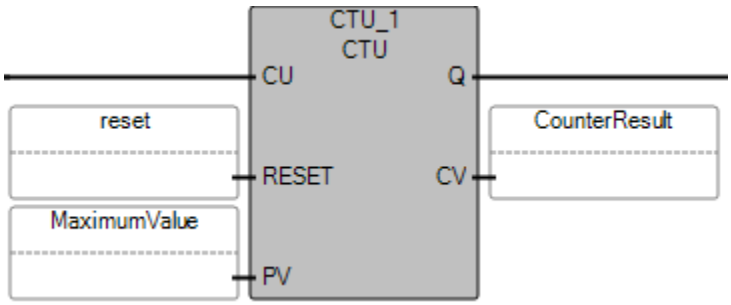
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
CU	Entrée	BOOL	Comptage croissant. VRAI : front montant détecté, comptage croissant par incréments de un. FAUX : front descendant détecté, conserve la valeur de compteur à la même valeur.
RESET	Entrée	BOOL	Réinitialisation vérifie la valeur PV par rapport à la valeur du comptage croissant. VRAI : définit la valeur CV sur zéro. FAUX : poursuit l'incrémentation du comptage croissant par un.
PV	Entrée	DINT	Valeur maximale programmée du compteur.
Q	Sortie	BOOL	Indique si l'instruction du comptage croissant a généré un nombre supérieur ou égal à la valeur maximale du compteur. VRAI : résultat du compteur => PV (condition de débordement). FAUX : résultat du compteur < PV
CV	Sortie	DINT	Résultat actuel du compteur.

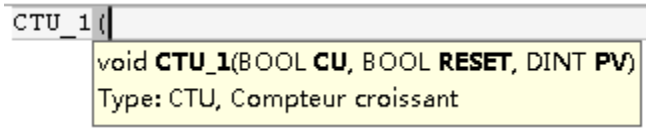
Exemple de Diagramme de Bloc Fonction CTU



Exemple de schéma à relais CTU



Exemple de texte structuré CTU

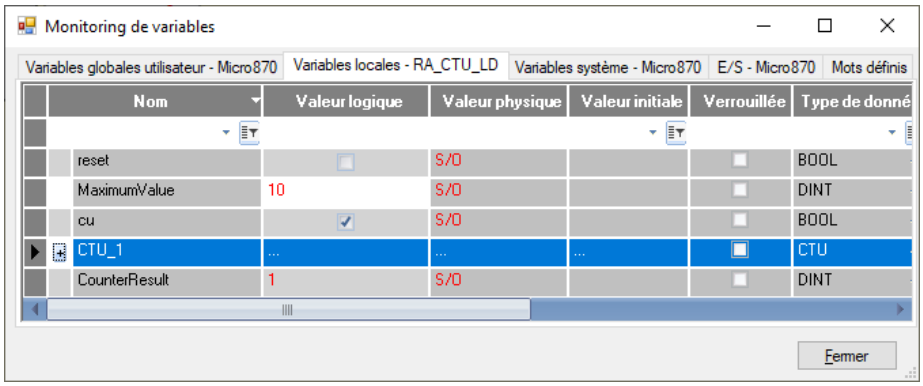



```
1  MaximumValue := 10;
2  CTU_1(cu, reset, MaximumValue);
```

(* Équivalence ST : CTU1 est une instance du bloc CTU*)

```
CTU1(déclencheur, NOT(auto_mode) ,100) ;
débordement := CTU1.Q ;
résultat := CTU1.CV ;
```

Résultats

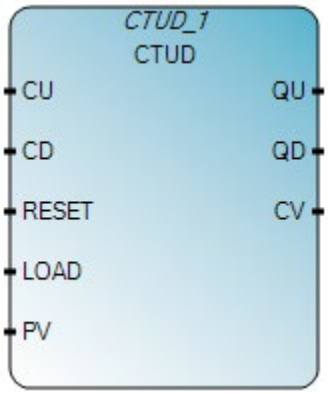


CTUD (comptage croissant/décroissant)

Décompte des entiers, 1 par 1, de 0 jusqu'à une valeur donnée ou d'une valeur donnée jusqu'à 0.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

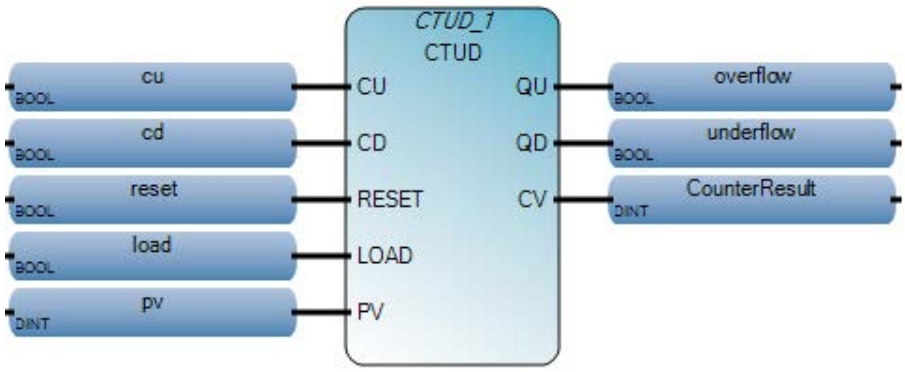
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



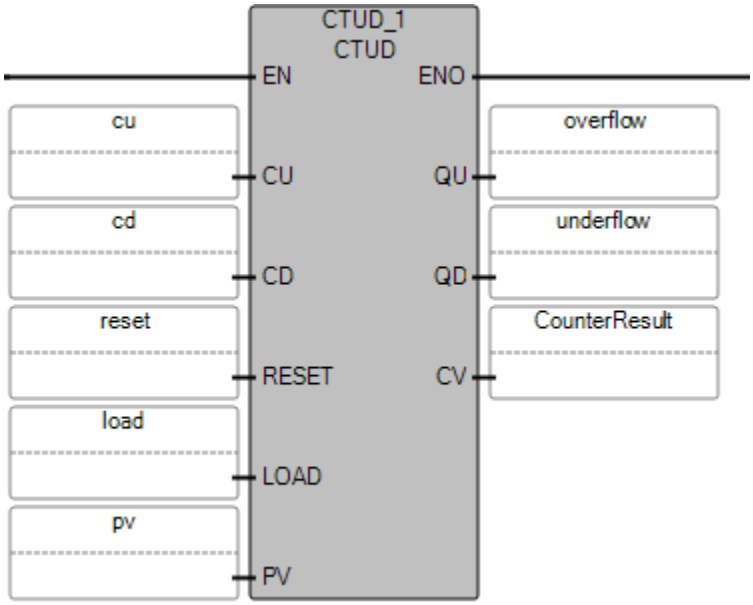
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
CU	Entrée	BOOL	VRAI : front montant détecté, comptage croissant.
CD	Entrée	BOOL	VRAI : front montant détecté, comptage décroissant.
RESET	Entrée	BOOL	Commande de réinitialisation du dominant (priorité la plus élevée lors de la détermination du comportement de l'instruction). (CV = 0 lorsque RESET est TRUE).
LOAD	Entrée	BOOL	Commande Chargement VRAI : définit CV = PV.
PV	Entrée	DINT	Valeur maximale programmée.
QU	Sortie	BOOL	Débordement. VRAI : lorsque CV >= PV.
QD	Sortie	BOOL	Soupassement. VRAI : lorsque CV <= 0.
CV	Sortie	DINT	Résultat du compteur.

Exemple de diagramme de bloc de fonctions CTUD



Exemple de logique à relais CTUD



Exemple de texte structuré CTUD

```
1  cu := TRUE;
2  cd := TRUE;
3  reset := FALSE;
4  load := FALSE;
5  pv := 10;
6  CTUD_1(cu, cd, reset, load, pv);
```

```
CTUD_1(
```

void **CTUD_1**(BOOL **CU**, BOOL **CD**, BOOL **RESET**, BOOL **LOAD**, DINT **PV**)
Type: CTUD, Compteur croissant-décroissant

(* Équivalence ST : nous supposons que CTUD1 est une instance de bloc*)

```
CTUD1(trigger1, trigger2, reset_cmd, load_cmd,100) ;
plein := CTUD1.QU ;
vide := CTUD1.QD ;
nb_elt := CTUD1.CV ;
```

Instructions de conversion des données

Utilisez les instructions de conversion de données pour convertir le type de données d'une variable en un type de données différent.

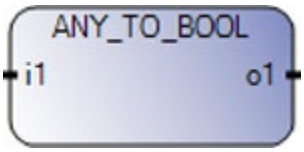
Instruction	Description
ANY_TO_BOOL sur la page 255	Convertit une valeur non booléenne en valeur booléenne.
ANY_TO_BYTE sur la page 256	Convertit une valeur en valeur BYTE.
ANY_TO_DATE sur la page 257	Convertit un type de données STRING, INTEGER, REAL ou TIME en type de données DATE.
ANY_TO_DINT sur la page 259	Convertit une valeur en entier double.
ANY_TO_DWORD sur la page 260	Convertit une valeur en valeur WORD double.
ANY_TO_INT sur la page 261	Convertit une valeur en entier.
ANY_TO_LINT sur la page 262	Convertit une valeur en entier long.
ANY_TO_LREAL sur la page 263	Convertit une valeur en valeur REAL longue.
ANY_TO_LWORD sur la page 264	Convertit une valeur en valeur WORD longue.
ANY_TO_REAL sur la page 265	Convertit une valeur en valeur REAL.
ANY_TO_SINT sur la page 266	Convertit une valeur en entier court.
ANY_TO_STRING sur la page 267	Convertit une valeur en valeur STRING.
ANY_TO_TIME sur la page 268	Convertit une valeur en type de données TIME.
ANY_TO_UDINT sur la page 269	Convertit une valeur en entier double non signé.
ANY_TO_UINT sur la page 270	Convertit une valeur en entier non signé.
ANY_TO_ULINT sur la page 271	Convertit une valeur en entier long non signé.
ANY_TO_USINT sur la page 272	Convertit une valeur en entier court non signé.
ANY_TO_WORD sur la page 273	Convertit une valeur en valeur WORD.

ANY_TO_BOOL

Convertit une valeur non booléenne en valeur booléenne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur booléenne.

Paramètre	Type de paramètre	Type de données	Description
			FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
i1	Entrée	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur non booléenne.
o1	Sortie	BOOL	Valeur booléenne.

Exemple de texte structuré ANY_TO_BOOL

(* Équivalence ST : *)

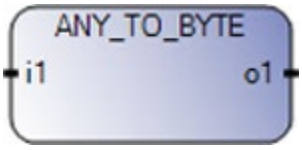
```
ares := ANY_TO_BOOL (10);          (* ares est VRAI *)
tres := ANY_TO_BOOL (t#0s);        (* tres est FAUX *)
mres := ANY_TO_BOOL ('FAUX');      (* mres est FAUX *)
```

ANY_TO_BYTE

Convertit une valeur en valeur BYTE de 8 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en OCTET. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	BOOL SINT USINT INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur autre que BYTE.
o1	Sortie	BYTE	Valeur BYTE de 8 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_BYTE

(* Équivalence ST : *)

```
bres := ANY_TO_BYTE (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_BYTE (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_BYTE ('0198');        (* mres est égal à 198 *)
```

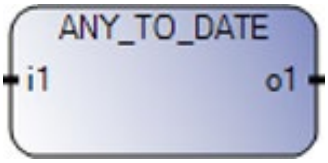
ANY_TO_DATE

Convertit un type de données STRING, INTEGER, REAL ou TIME en type de données DATE.

Les types de données BOOLEAN, BYTE et WORD sont prises en charge, mais ne sont pas recommandées.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

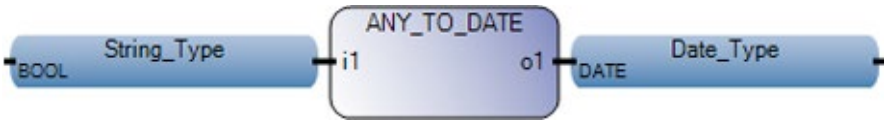


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

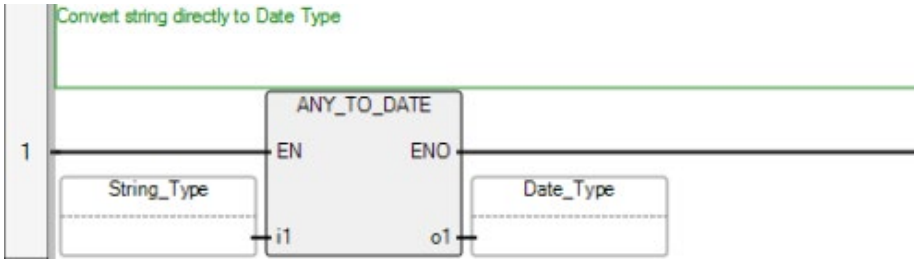
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en DATE. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	• Les chaînes sont directement converties en type de données DATE et doivent être au format AAAA-MM-JJ, DATE#AAAA-MM-JJ ou D#AAAA-MM-JJ. • INTEGERS et REAL, qui doivent être positifs, sont en unités de secondes et ajoutés à la date 1970-01-01. • Le temps est ajouté à la date 1970-01-01.
o1	Sortie	DATE	Valeur de la date convertie. Des erreurs pendant la conversion (comme une date négative) se traduisent par un résultat INVALID.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions ANY_TO_DATE

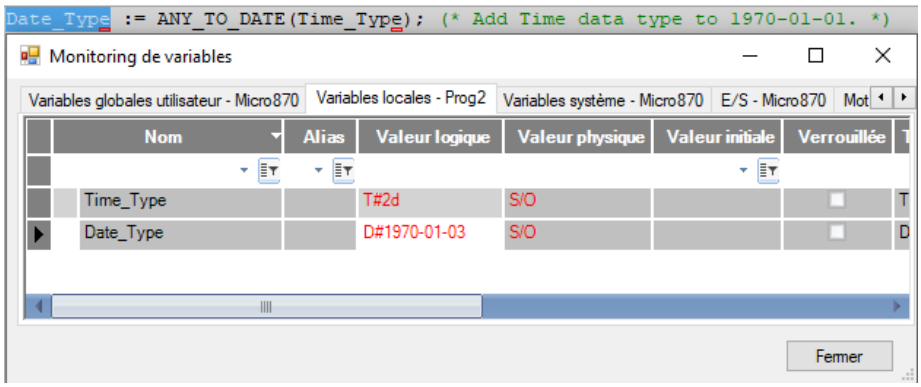
ANY_TO_DATE



Exemple de logique à relais ANY_TO_DATE



Exemple de texte structuré ANY_TO_DATE

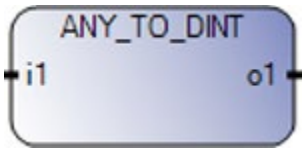


ANY_TO_DINT

Convertit une valeur en valeur entière double de 32 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier double à 32 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur autre qu'un nombre entier double.
o1	Sortie	DINT	Valeur INT double de 32 bits.

Paramètre	Type de paramètre	Type de données	Description
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_DINT

(* Équivalence ST : *)

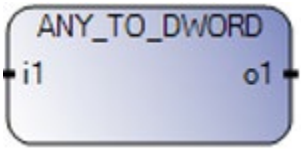
```
bres := ANY_TO_DINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_DINT (t#1s46ms);      (* tres est égal à 1046 *)
mres := ANY_TO_DINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_DWORD

Convertit une valeur en valeur WORD double de 32 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur Word double à 32 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur autre qu'un mot double.

Paramètre	Type de paramètre	Type de données	Description
o1	Sortie	DWORD	Valeur WORD double de 32 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_DWORD

(* Équivalence ST : *)

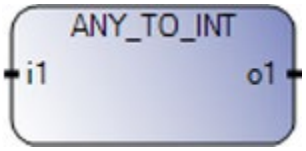
```
bres := ANY_TO_DWORD (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_DWORD (t#1s46ms);      (* tres est égal à 1046 *)
mres := ANY_TO_DWORD ('0198');        (* mres est égal à 198 *)
```

ANY_TO_INT

Convertit une valeur en valeur entière de 16 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier à 16 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Valeur entière autre que 16 bits, quelle qu'elle soit.

Paramètre	Type de paramètre	Type de données	Description
o1	Sortie	INT	Valeur INT de 16 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_INT

(* Équivalence ST : *)

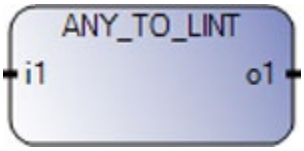
```
bres := ANY_TO_INT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_INT (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_INT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_LINT

Convertit une valeur en valeur entière longue de 64 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier long à 64 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE	Toute valeur autre qu'un nombre entier long.

Paramètre	Type de paramètre	Type de données	Description
		STRING	
o1	Sortie	LINT	Valeur INT longue de 64 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_LINT

(* Équivalence ST : *)

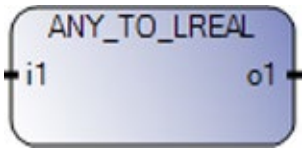
```
bres := ANY_TO_LINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_LINT (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_LINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_LREAL

Convertit toute valeur en valeur réelle longue.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur réelle longue. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT	Toute valeur autre qu'une valeur REAL longue.

Paramètre	Type de paramètre	Type de données	Description
		LWORD REAL TIME DATE STRING	
o1	Sortie	LREAL	Valeur réelle longue.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_LREAL

(* Équivalence ST : *)

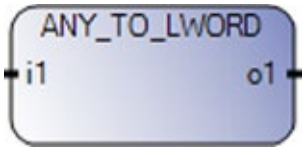
```
bres := ANY_TO_LREAL (vrai);          (* bres est égal à 1,0 *)
tres := ANY_TO_LREAL (t#1s46ms);      (* tres est égal à 1046,0 *)
ares := ANY_TO_LREAL (198);           (* ares est égale à 198,0 *)
```

ANY_TO_LWORD

Convertit une valeur en valeur WORD longue de 64 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur Word longue à 64 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT	Toute valeur autre qu'une valeur WORD longue.

Paramètre	Type de paramètre	Type de données	Description
		REAL LREAL IME DATE STRING	
o1	Sortie	LWORD	Valeur WORD longue de 64 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_LWORD

(* Équivalence ST : *)

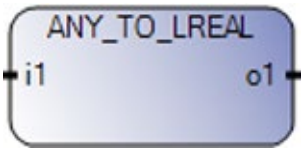
```
bres := ANY_TO_LWORD (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_LWORD (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_LWORD ('0198');        (* mres est égal à 198 *)
```

ANY_TO_REAL

Convertit une valeur en valeur réelle.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur réelle. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT	Toute valeur autre qu'une valeur REAL.

Paramètre	Type de paramètre	Type de données	Description
		LWORD LREAL TIME DATE STRING	
o1	Sortie	REAL	Valeur réelle.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_REAL

(* Équivalence ST : *)

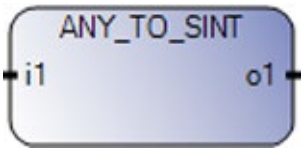
```
bres := ANY_TO_REAL (vrai);          (* bres est égal à 1,0 *)
tres := ANY_TO_REAL (t#1s46ms);      (* tres est égal à 1046,0 *)
ares := ANY_TO_REAL (198);           (* ares est égale à 198,0 *)
```

ANY_TO_SINT

Convertit une valeur en valeur entière courte.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier court à 8 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT	Toute valeur, sauf entier court.

Paramètre	Type de paramètre	Type de données	Description
		LWORD REAL LREAL TIME DATE STRING	
o1	Sortie	SINT	Valeur entière courte.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_SINT

(* Équivalence ST : *)

```
bres := ANY_TO_SINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_SINT (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_SINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_STRING

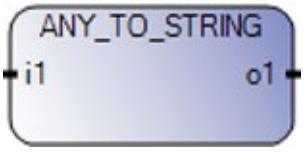
Convertit une valeur en valeur de chaîne.

Détails de l'opération :

- Lors de la conversion d'un type de données REAL en une chaîne, l'instruction ANY_TO_STRING utilise le format IEEE 754.
  - ANY_TO_STRING convertit 125,0 en 1,25000E+02
- Lorsque la longueur de la chaîne cible est 5 caractères :
  - ANY_TO_STRING convertit 125,0 en 1,25000E+02 et renvoie 1,250 dans la chaîne cible.
  - Envisagez de créer un bloc de fonctions défini par l'utilisateur pour convertir la notation exponentielle en nombre.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur de chaîne. FAUX : aucun calcul.

Paramètre	Type de paramètre	Type de données	Description
			Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE	Toute valeur autre qu'une valeur STRING.

o1	Sortie	STRING	Si IN est un booléen, « FALSE » ou « TRUE ».
			Si IN est un nombre entier ou réel, une représentation décimale.
			Si IN est une variable TIME, peut être :
			TIME time1
			STRING s1
			time1 :=13 ms;
			s1:=ANY_TO_STRING(time1);
			(* s1 = '0s13' *).

ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_STRING

(* Équivalence ST : *)

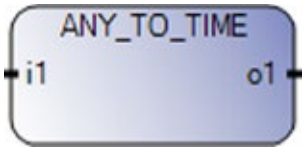
```
bres := ANY_TO_STRING (VRAI);          (* bres est 'VRAI' *)
ares := ANY_TO_STRING (125);           (* ares est égal à '125' *)
```

ANY_TO_TIME

Convertit une valeur autre qu'une valeur de temps ou de date en valeur de temps.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en TEMPS. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL STRING	Toute valeur positive autre qu'une valeur de temps ou de date. IN (ou partie de nombre entier d'IN si la variable est réelle) est le nombre de millisecondes. STRING (nombre de millisecondes, par exemple une valeur de 300 032 représente 5 minutes et 32 millisecondes).
o1	Sortie	TIME	Valeur de temps représentée par IN. Une valeur de 1193h2m47s295ms indique un temps non valide.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_TIME

(* Équivalence ST : *)

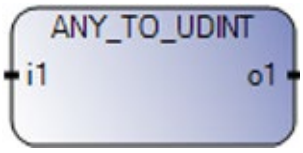
```
ares := ANY_TO_TIME (1256);          (* ares := t#1s256ms *)
rres := ANY_TO_TIME (1256.3);        (* rres := t#1s256ms *)
```

ANY_TO_UDINT

Convertit une valeur en valeur entière double non signée de 32 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier double à 32 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur autre qu'un nombre entier double non signé.
o1	Sortie	UDINT	Valeur INT double non signée de 32 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_UDINT

(* Équivalence ST : *)

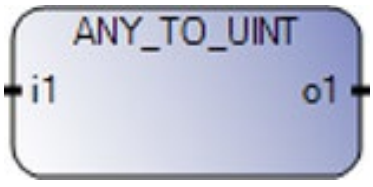
```
bres := ANY_TO_UDINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_UDINT (t#1s46ms);      (* tres est égal à 1046 *)
mres := ANY_TO_UDINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_UINT

Convertit une valeur en valeur entière non signée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier non signé à 16 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur sauf entier non signé.
o1	Sortie	UINT	Valeur entière non signée.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_UINT

(* Équivalence ST : *)

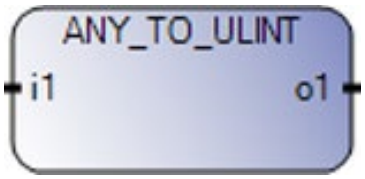
```
bres := ANY_TO_UINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_UINT (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_UINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_ULINT

Convertit une valeur en valeur entière longue non signée de 64 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier non signé long à 64 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT LWORD REAL LREAL TIME DATE STRING	Toute valeur autre qu'un nombre entier long non signé.
o1	Sortie	ULINT	Valeur INT longue non signée de 64 bits.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_ULINT

(* Équivalence ST : *)

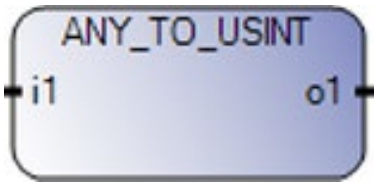
```
bres := ANY_TO_ULINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_ULINT (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_ULINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_USINT

Convertit une valeur en valeur entière courte non signée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en nombre entier non signé court à 8 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur, sauf entier court.
o1	Sortie	USINT	Valeur entière courte non signée.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_USINT

(* Équivalence ST : *)

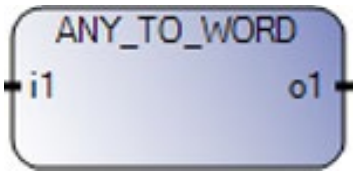
```
bres := ANY_TO_USINT (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_USINT (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_USINT ('0198');        (* mres est égal à 198 *)
```

ANY_TO_WORD

Convertit une valeur en valeur WORD 16 bits.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécute le calcul de conversion en valeur WORD à 16 bits. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
i1	Entrée	BOOL SINT USINT BYTE INT DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Toute valeur autre qu'une valeur WORD.
o1	Sortie	WORD	Une valeur WORD.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de texte structuré ANY_TO_WORD

(* Équivalence ST : *)

```
bres := ANY_TO_WORD (vrai);          (* bres est égal à 1 *)
tres := ANY_TO_WORD (t#0s46ms);      (* tres est égal à 46 *)
mres := ANY_TO_WORD ('0198');        (* mres est égal à 198 *)
```

Instructions de manipulation des données

Utilisez les instructions de manipulation des données pour modifier les données de sortie afin de changer l'état sans altérer le programme.

Instruction	Description
AVERAGE sur la page 275	Calcule une moyenne mobile sur un nombre d'échantillons définis.
COP sur la page 276	Copie les données binaires de l'élément source dans l'élément de destination.
MAX sur la page 283	Calcule la valeur maximale de deux entiers.
MIN sur la page 282	Calcule la valeur minimale de deux entiers.

AVERAGE

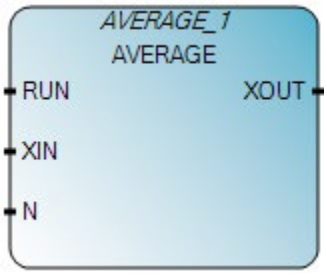
Calcule une moyenne mobile sur un nombre d'échantillons défini, puis stocke la valeur à chaque cycle.

Détails de l'opération :

- Le nombre défini d'échantillons (N) ne peut pas dépasser 127.
- Lorsque vous définissez ou changez la valeur de N, définissez RUN sur FAUX, puis de nouveau sur VRAI.
- Si la commande RUN est définie sur FALSE (mode de réinitialisation), la valeur de sortie est égale à la valeur d'entrée.
- Lorsque le nombre maximum de valeurs stockées est atteint, la première valeur stockée est remplacée par la dernière.
- l'utilisation de types de données à virgule flottante pourrait entraîner des calculs inexacts en raison des limites de l'arrondi inhérentes aux mathématiques à virgule flottante.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

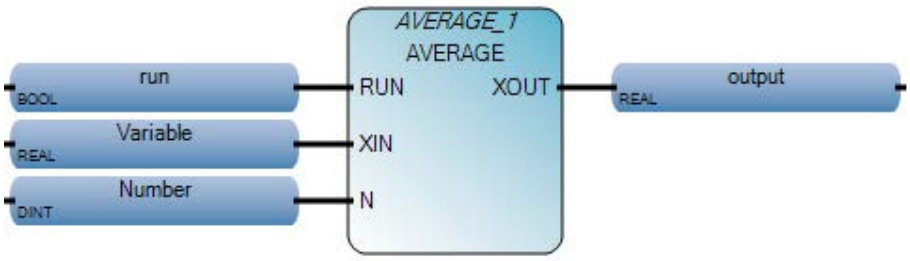
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



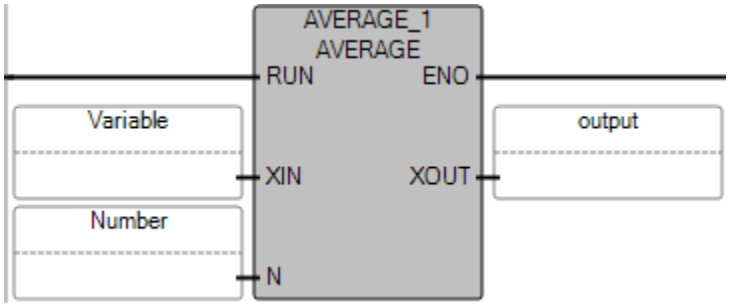
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
RUN	Entrée	BOOL	VRAI = exécution FAUX = réinitialisation
XIN	Entrée	REAL	Variable réelle, quelle qu'elle soit.
N	Entrée	DINT	Nombre d'échantillons défini par l'application.
XOUT	Sortie	REAL	Calcul de la moyenne de la valeur XIN.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions AVERAGE



Exemple de logique à relais AVERAGE



Exemple de texte structuré AVERAGE

```
1  AVERAGE_1(run, Variable, Number);
2  output := AVERAGE_1.XOUT;
```

(* Équivalence ST : AVERAGE1 est une instance d'un bloc AVERAGE *)

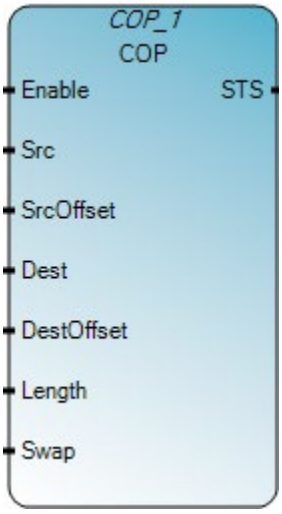
```
AVERAGE1((auto_mode & store_cmd), sensor_value, 100) ;
ave_value := AVERAGE1.XOUT ;
```

COP

Copie les données binaires de l'élément source dans l'élément de destination. L'élément source reste inchangé.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données		Description
Enable	Entrée	BOOL		Activation du bloc d'instruction. Le COP est activé en fonction du niveau. VRAI : réalisation d'une copie. FAUX : le bloc de fonctions est en attente.
Src	Entrée	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT	DWORD REAL TIME DATE STRING LWORD ULINT LINT LREAL	Élément initial à copier. Si la source est un type de données Chaîne, la destination doit être un type de données Chaîne ou un type de données USINT (UCHAR et BYTE). Dans le cas contraire, une divergence de type de données est signalée.
SrcOffset	Entrée	UINT		Le décalage de l'élément Source est utilisé avec les types de données Tableau pour identifier la position dans le tableau source à partir duquel copier des données. Définissez le décalage sur 0 : • S'il ne s'agit pas d'un type de données Tableau. OU • Pour effectuer la copie à partir du premier élément, pour un type de données Tableau.
Dest	Entrée	BOOL SINT USINT BYTE INT UINT WORD	DWORD REAL TIME DATE STRING LWORD ULINT	Élément initial à écraser avec la source. Si la destination est un type de données Chaîne, la source doit être un type de données Chaîne ou un type de données USINT (UCHAR et BYTE). Dans le cas contraire, une divergence de type de données est signalée.

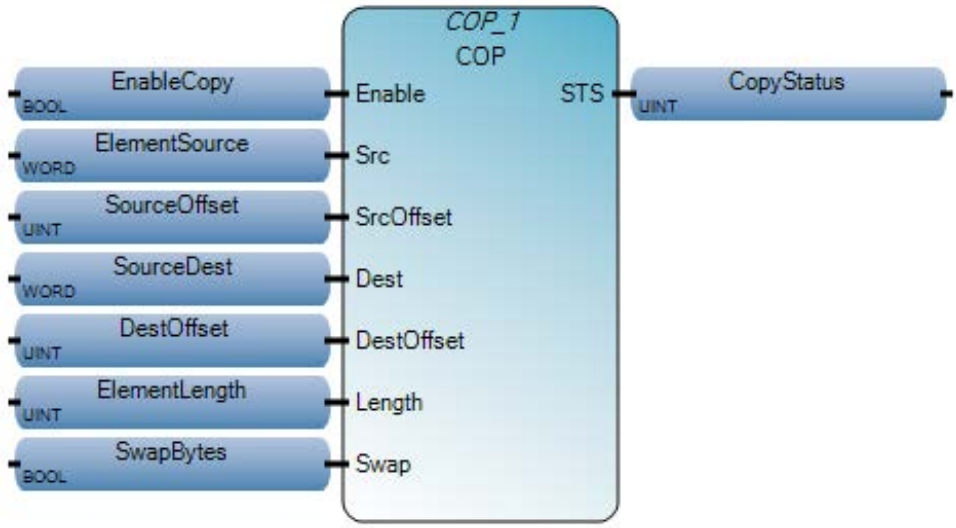
Paramètre	Type de paramètre	Type de données		Description
		DINT UDINT	LINT	
DestOffset	Entrée	UINT		Le décalage de l'élément Destination est utilisé avec les types de données Tableau pour identifier la position dans le tableau de destination vers lequel copier des données. Définissez le décalage sur 0 : • S'il ne s'agit pas d'un type de données Tableau. OU • Pour effectuer la copie à partir du premier élément, pour un type de données Tableau.
Length	Entrée	UINT		Nombre d'éléments Destination à copier. Lorsque la destination est un type de données Chaîne, il indique le nombre de chaînes à copier.
Swap	Entrée	BOOL		Utilisé pour échanger les données des éléments source et de destination, de sorte que les données de destination remplacent les données source, et que les données source remplacent les données de destination. VRAI : échange des octets selon le type de données. Aucune opération d'échange n'a lieu si : • Le type de données de la source ou de la destination est Chaîne, ou • La source et la destination ont toutes deux des données d'un octet de longueur.
Sts	Sortie	UINT		État de l'opération de copie. Les définitions du paramètre Sts sont définies dans les codes d'état COP.
ENO	Sortie	BOOL		Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Codes d'état COP (Sts)

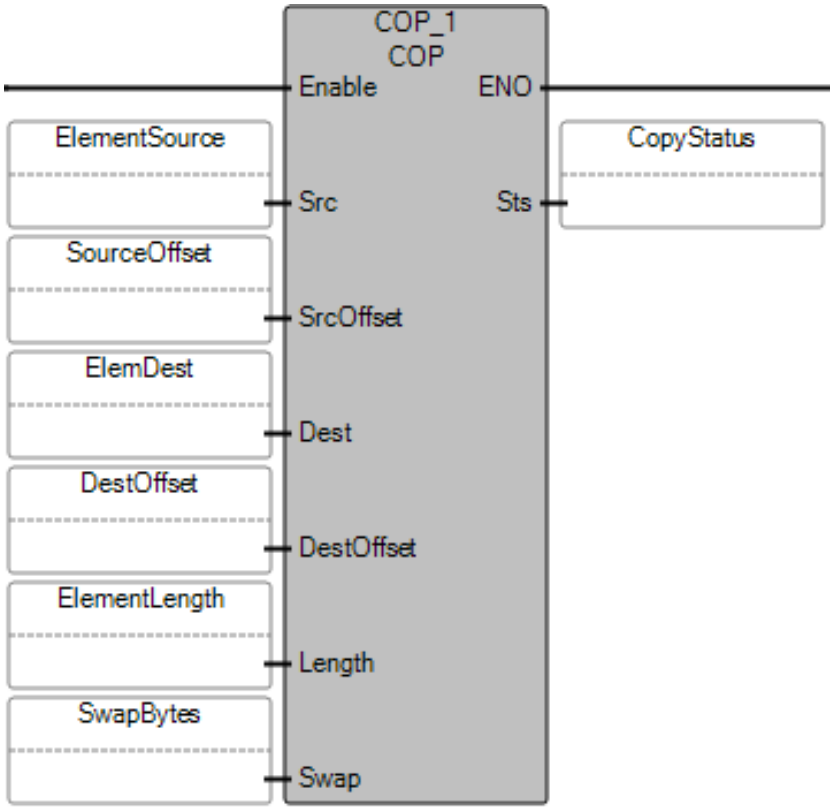
Le tableau suivant décrit les codes d'état COP.

Code d'état COP	Description de l'état
0x00	Aucune action n'est réalisée (non activée).
0x01	Réussite du bloc fonctionnel COP.
0x02	La destination dispose d'octets supplémentaires lors de la copie à partir de la chaîne.
0x03	Les données source sont tronquées.
0x04	La longueur de copie n'est pas valide.
0x05	Le type de données ne correspond pas lorsqu'un type de données Chaîne est présent soit à la source, soit à la destination.
0x06	La taille des données source est trop faible pour permettre la copie.
0x07	La taille des données source est trop élevée pour permettre la copie.
0x08	Le décalage des données source n'est pas valide.
0x09	Le décalage des données de destination n'est pas valide.
0x0A	Les données ne sont pas valides, soit à la source soit à la destination.

Exemple de Diagramme de Bloc Fonction COP



Exemple de langage à relais COP



Exemple de texte structuré COP

```
1  COP_1(EnableCopy, ElementSource, SourceOffset, SourceDest,
2    DestOffset, ElementLength, SwapBytes);
3  output :=COP_1.STS;
```

Copie vers un type de données différent

Lorsqu'une copie est effectuée de ou vers un type de données Chaîne, le format de chaîne courte ODVA est utilisé pour les données du tableau USINT.

Lorsque COP sur la page 276 est utilisé entre toute autre paire de types de données, l'opération de copie est valide, même si le type de données de la source est différent de celui de la destination, et même lorsque leur format n'est pas valide. La logique doit être validée au niveau de l'application.

D'un tableau USINT à un tableau Chaîne

Pour copier un tableau USINT dans un tableau Chaîne, les données du tableau USINT doivent être au format suivant :

- Octet1 : longueur de la première chaîne
- Octet2 : caractère du premier octet
- Octet3 : caractère du deuxième octet
- Octet n : caractère du dernier octet
- Octet (n+1) : longueur de la deuxième chaîne
- Octet (n+2) : caractère du premier octet pour la deuxième chaîne

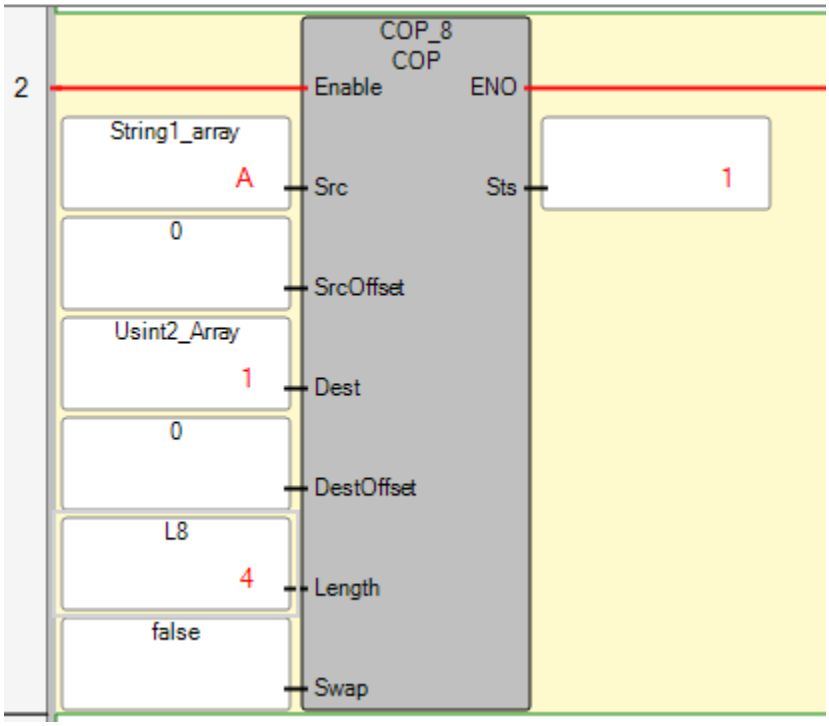
Exemple de tableau de chaîne COP

L'exemple suivant montre une instruction COP sur la page 276 copiant un tableau Chaîne dans un tableau Usint. L'instruction COP passe tous les éléments du tableau source présentant une longueur Zéro.

Dans le cadre de cet exemple :

- Une longueur de 4 octets est spécifiée.
- Le nombre d'éléments de destination à copier est de 4 octets.
- Tous les éléments de tableau présentant une longueur Zéro (vides) sont passés.
- L'instruction COP trouve un élément ne présentant pas une longueur Zéro dans le tableau Chaîne[1], qui est copié dans le tableau USINT de destination[1] avec 1. Le chiffre 1 est la longueur de la chaîne dans le tableau Chaîne[1] et le tableau USINT[2] présente une valeur de 65, qui est le code ASCII de « A ».
- L'instruction COP trouve un élément ne présentant pas une longueur Zéro dans le tableau Chaîne[10] qui est « a », qui est copié dans le tableau USINT de destination[3] avec 1. Le chiffre 1 est la longueur de la chaîne dans le tableau Chaîne[10] et le tableau USINT[4] présente une valeur de 97, qui est le code ASCII de « a ».

Langage à relais



Éléments de tableau avec des valeurs logiques

Nom	Alias	Valeur logique
- String_String_1.String1_array		...
String_String_1.String1_array[1]		A
String_String_1.String1_array[2]		
String_String_1.String1_array[3]		
String_String_1.String1_array[4]		
String_String_1.String1_array[5]		
String_String_1.String1_array[6]		
String_String_1.String1_array[7]		
String_String_1.String1_array[8]		
String_String_1.String1_array[9]		
String_String_1.String1_array[10]		a
- String_String_1.USint2_Array		...
String_String_1.USint2_Array[1]		1
String_String_1.USint2_Array[2]		65
String_String_1.USint2_Array[3]		1
String_String_1.USint2_Array[4]		97
String_String_1.USint2_Array[5]		0
String_String_1.USint2_Array[6]		0
String_String_1.USint2_Array[7]		0
String_String_1.USint2_Array[8]		0
String_String_1.USint2_Array[9]		0
String_String_1.USint2_Array[10]		0
String_String_1.USint2_Array[11]		0
String_String_1.USint2_Array[12]		0
String_String_1.USint2_Array[13]		0
String_String_1.L8		4
+ String_String_1.COP_8		...

MIN (minimum)

Calcule la valeur minimale de deux entiers.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

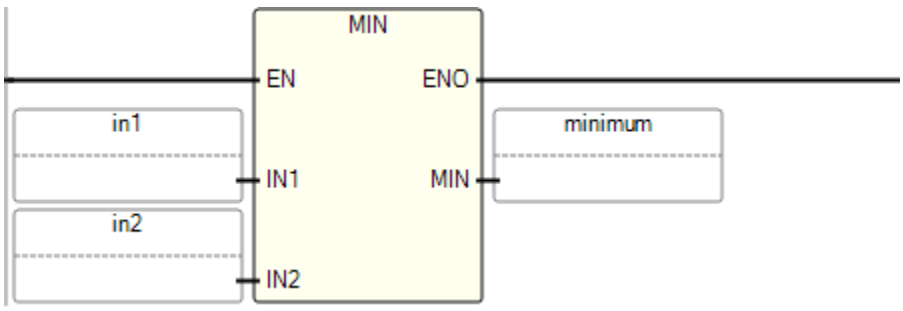
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

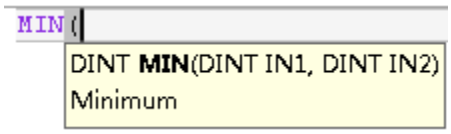
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la valeur entière minimale. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN1	Entrée	DINT	Valeur entière signée, quelle qu'elle soit.
IN2	Entrée	DINT	Ne peut pas être de type REAL.
MIN	Sortie	DINT	Valeur minimale des deux valeurs d'entrée.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction MIN

Exemple de schéma à relais MIN



Exemple de texte structuré MIN



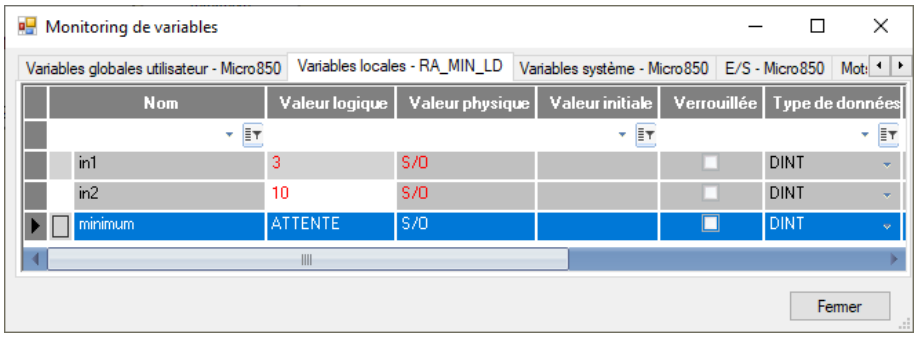

```
1  in1 := 3;
2  in2 := 10;
3  minimum := MIN(in1, in2);
```

(* Équivalence ST : *)

new_value := MAX (MIN (max_value, valeur), min_value) ;

(* limite la valeur à l'ensemble [min_value..max_value] *)

Résultats

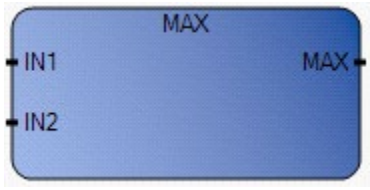


MAX (maximum)

Calcule la valeur maximale de deux entiers.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



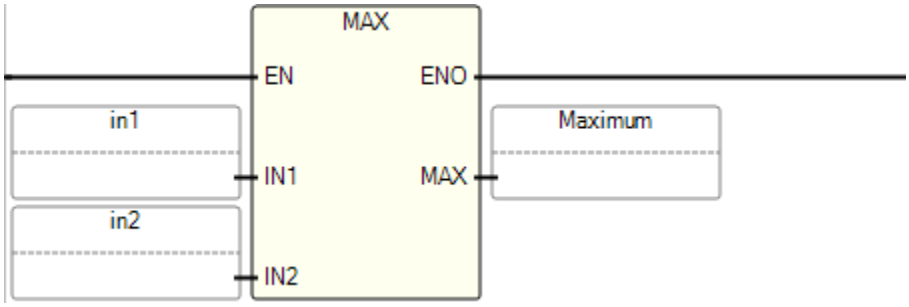
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul de la valeur entière maximale. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
IN1	Entrée	DINT	Valeur entière signée, quelle qu'elle soit.

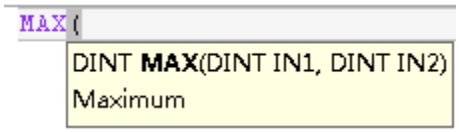
Paramètre	Type de paramètre	Type de données	Description
IN2	Entrée	DINT	Ne peut pas être de type REAL.
MAX	Sortie	DINT	Valeur maximale des deux valeurs d'entrée.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions MAX

Exemple de schéma à relais MAX



Exemple de texte structuré MAX



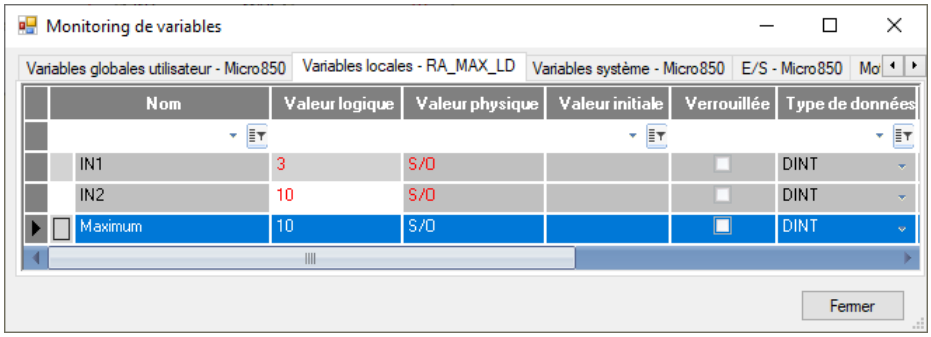

```
1  IN1 := 3;
2  IN2 := 10;
3  Maximum := MAX(IN1, IN2);
```

(* Équivalence ST : *)

new_value := MAX (MIN (max_value, valeur), min_value) ;

(* limite la valeur à l'ensemble [min_value..max_value] *)

Résultats


Monitoring de variables
Variables globales utilisateur - Micro850
Variables locales - RA_MAX_LD
Variables système - Micro850
E/S - Micro850
Nom
Valeur logique
Valeur physique
Valeur initiale
Verrouillée
Type de données
IN1
3
S/O
DINT
IN2
10
S/O
DINT
Maximum
10
S/O
DINT
Fermer

Instructions de compteur haute vitesse (HSC)

Utilisez les instructions de comptage rapide pour surveiller et contrôler le comptage rapide.

Instruction	Description
HSC sur la page 288	HSC applique les valeurs hautes prédéfinies, les valeurs basses prédéfinies et les valeurs de source de sortie au compteur haute vitesse.
HSC_SET_STS sur la page 304	HSC_SET_STS définit ou réinitialise manuellement l'état de décompte HSC.

Qu'est-ce qu'un compteur haute vitesse ?

Un compteur haute vitesse détecte et décompte les impulsions étroites (rapides), puis émet des instructions spécialisées pour lancer des opérations de contrôle lorsque le nombre détecté dépasse la valeur prédéfinie. Ces opérations de contrôle incluent l'exécution automatique et immédiate de la routine d'interruption du compteur haute vitesse et la mise à jour immédiate des sorties, sur la base du modèle de masque et de source configuré.

Fonctionnalités du compteur haute vitesse

Les instructions HSC présentent des exigences de hautes performances, c'est pourquoi leur exécution est assurée par des circuits personnalisés fonctionnant en parallèle du processeur principal du système. Les fonctionnalités avancées de comptage rapide (HSC) incluent :

- Contrôle direct des sorties en haute vitesse à 100 kHz
- Données de type Nombre entier signé 32 bits (plage ± 2 147 483 647)
- Valeurs programmables prédéfinies haute et basse
- Points de consigne de débordement et de soupassement
- Processus d'interruption automatique basé sur le décompte cumulé
- Paramètres d'exécution modifiables (à partir du programme de contrôle utilisateur) pour l'instruction HSC

Prise en charge du compteur HSC par le contrôleur Micro800

Tous les contrôleurs Micro830, Micro850 et Micro870, excepté le modèle 2080-LCxx-AWB, prennent en charge jusqu'à six entrées HSC. La fonctionnalité HSC est mise en œuvre sur les contrôleurs Micro800 à l'aide d'un compteur matériel haute vitesse (entrées intégrées au contrôleur) et de l'instruction HSC de l'application. L'instruction HSC sur la page 287 configure le compteur matériel haute vitesse et met à jour l'accumulateur d'image.

IMPORTANT La fonction HSC peut être utilisée uniquement avec l'E/S intégrée du contrôleur. Elle ne peut être utilisée avec des modules d'E/S d'extension.

HSC (Comptage rapide)

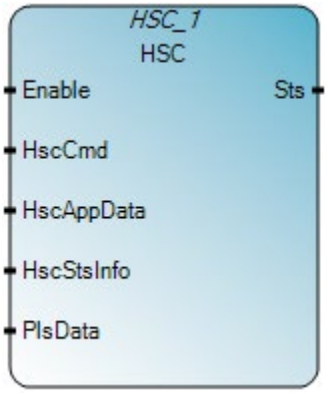
HSC applique les valeurs hautes prédéfinies, les valeurs basses prédéfinies et les valeurs de source de sortie au compteur haute vitesse.

Détails de l'opération :

- L'interrupteur à came rotatif (PLS) est activé en définissant le paramètre HSCAppData.PLSEnable sur True.
- Le paramètre PLSPosition est réinitialisé après qu'un cycle complet est terminé et que la valeur de HSCSTS.HP est atteinte.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

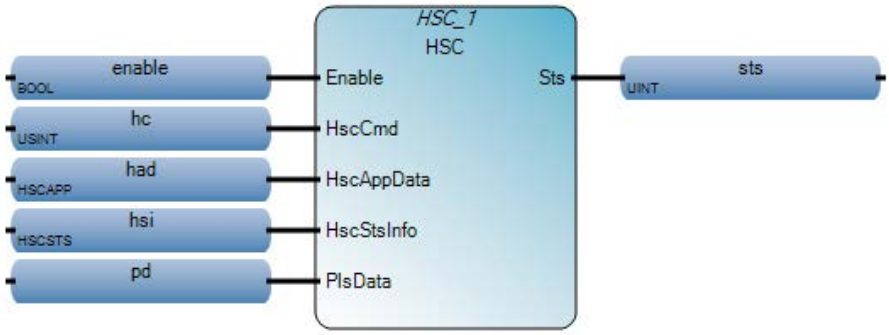


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

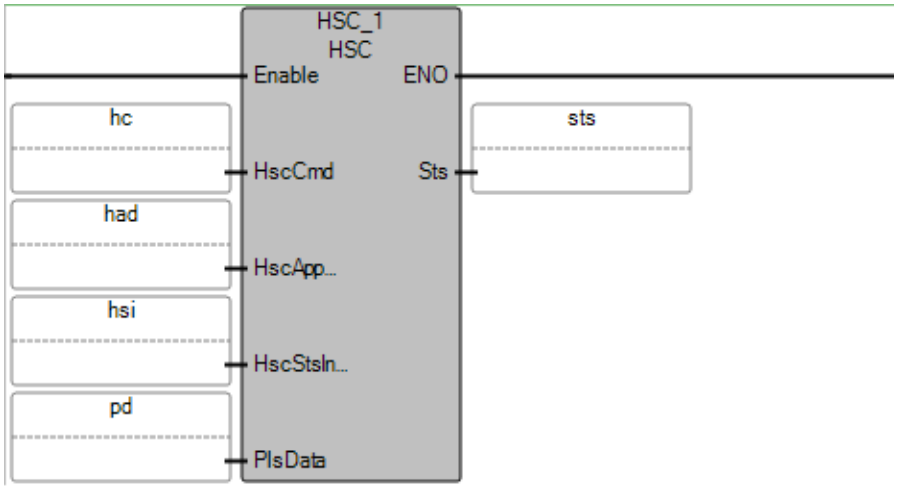
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	État de la ligne de programme du bloc d'instruction. VRAI : l'incrémentation du temporisateur commence. FAUX : le bloc de fonctions est en attente. Il est recommandé de ne pas utiliser le paramètre EN avec le bloc de fonctions HSC, car l'incrémentation du temporisateur se poursuit lorsqu'EN est défini sur « FAUX ». Ne s'applique qu'aux programmes de langage à relais.
Enable	Entrée	BOOL	Activer le bloc d'instruction. VRAI : exécution de l'opération HSC spécifiée dans les paramètres de commande HSC. FAUX : aucune commande HSC n'est émise.
HscCmd	Entrée	USINT	Émet des commandes vers le HSC.
HSCAppData	Entrée	HSCAPP sur la page 290	Configuration de l'application HSC, généralement nécessaire une seule fois.
HSCStsInfo	Entrée	HSCSTS sur la page 294	État HSC dynamique, mis à jour de façon continue lors du décompte HSC.
PlsData	Entrée	DINT UDINT	Structure des données du commutateur limite programmable (PLS)

Paramètre	Type de paramètre	Type de données	Description
Sts	Sortie	UINT	État d'exécution HSC. Codes de statut HSC : • 0x00 : aucune action n'est réalisée (non activée). • 0x01 : exécution HSC réussie. • 0x02 : commande HSC non valide. • 0x03 : ID HSC hors plage. • 0x04 : erreur de configuration HSC.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

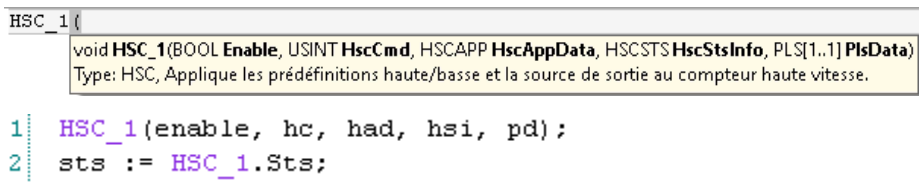
Exemple de Diagramme de Bloc Fonction HSC



Exemple de langage à relais HSC



Exemple de texte structuré HSC



Valeurs HSCCmd

Le tableau suivant décrit les commandes HSC pour chaque valeur de commande HSC.

Commande HSC	Description de la commande
0x01	Exécution HSC • Démarrage HSC (si HSC est en mode Inactif et que la ligne est activée). • Mise à jour des informations de statut HSC (si HSC est en mode Exécuter et que la ligne est activée).
0x02	Arrêt HSC : arrêt d'un décompte HSC (si HSC est en mode Exécuter et que la ligne est activée).
0x03	Chargement/définition HSC : rechargement des données d'application HSC (si la ligne est activée) pour 6 éléments d'entrée. HPSetting, LPSetting, HPOutput, LPOutput, OFSetting et UFSetting. **Remarque :** cette commande ne recharge pas l'élément d'entrée suivant : Accumulateur HSC
0x04	Réinitialisation de l'accumulateur HSC (si la ligne est activée).

Résultats des commandes HSC

Valeur de commande	Résultat	Conditions
HscCmd =1	Démarre le mécanisme HSC ; le compteur HSC passe en mode exécution.	La définition du paramètre d'entrée Enable sur Faux ne vous permet pas d'arrêter le décompte lorsque le mode d'exécution est activé. HscCmd =2 doit être émis pour stopper le décompte.
	Le mécanisme HSC met automatiquement à jour les valeurs.	HSC AppData.Accumalator est mis à jour avec HSC Sts.Accumulator
HscCmd =4 (reset)	Définit la valeur HSC Acc sur la valeur HSC AppData.Accumalator.	HscCmd =4 ne permet pas d'arrêter le décompte HSC. Si le décompte HSC est activé lors de l'émission de la commande HscCmd =4, il est possible que certains décomptes soient perdus. Pour définir HSC Acc sur une valeur donnée lors du décompte, écrivez la valeur HSC AppData.Accumalator immédiatement avant l'émission de HscCmd =4.

Type de données HSCAPP

Utilisez le type de données HSCAPP pour définir le paramètre HSCAppData dans l'instruction HSC. Les paramètres du type de données HSCApp sont utilisés pour définir les données de configuration HSC.

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données HSCAPP.

Paramètre	Type de données	Format des données	Accès au programme utilisateur	Description
PLSEnable	BOOL	bit	Lecture/Écriture	Active et désactive la fonction Interrupteur à came rotatif (PLS) du comptage rapide.
HSCID	UINT	mot	Lecture/Écriture	Définit le compteur HSC.
HSCMode	UINT	mot	Lecture/Écriture	Définit le mode HSC.
Accumulator	DINT	mot long	Lecture/Écriture	Valeur d'accumulateur initiale. HSCApp.Accumulator définit la valeur d'accumulateur initiale lorsque le comptage rapide démarre. Lorsque le compteur HSC est en mode Décompte, l'accumulateur est automatiquement mis à jour par le sous-système HSC afin de refléter la valeur d'accumulateur HSC réelle.
HPSetting	DINT	mot long	Lecture/Écriture	Paramétrage de la valeur haute prédéfinie. Le paramètre HSCApp.HPSetting détermine le point de consigne haut (en nombre) qui définit à quel moment le sous-système HSC génère une

Paramètre	Type de données	Format des données	Accès au programme utilisateur	Description
				interruption. Les données chargées dans la valeur haute prédéfinie doivent être inférieures ou égales aux données contenues dans le paramètre de débordement (HSCAPP.OFSetting) ou une erreur HSC est générée.
LPSetting	DINT	mot long	Lecture/Écriture	Paramétrage de la valeur basse prédéfinie. HSCApp.HPSetting détermine le point de consigne bas (en nombre) qui définit à quel moment le sous-système HSC génère une interruption. Les données chargées dans la valeur basse prédéfinie doivent être supérieures ou égales aux données contenues dans le paramètre de débordement négatif (HSCAPP.UFSetting) ou une erreur HSC est générée. Si la valeur de débordement négatif et la valeur basse prédéfinie sont négatives, cette dernière doit être constituée d'un chiffre dont la valeur absolue est inférieure à celle du débordement négatif.
OFSetting	DINT	mot long	Lecture/Écriture	Paramétrage du débordement. Le paramètre de débordement HSCApp.OFSetting définit la limite de décompte haute pour le compteur. Si la valeur cumulée du compteur monte au-dessus de la valeur spécifiée dans le paramètre OFSetting, une interruption de débordement est générée. Dans un tel cas, le sous-système HSC réinitialise la valeur cumulée et la redéfinit sur la valeur de débordement négatif ; puis le compteur poursuit le décompte à partir de cette valeur de débordement négatif (le décompte n'est pas perdu lors de cette transition). Les valeurs OFSetting doivent répondre aux critères suivants : • Elles doivent se situer entre -2 147 483 648 et 2 147 483 647. • Elles doivent être supérieures à la valeur de débordement négatif. • Elles doivent être supérieures ou égales aux données contenues dans la valeur haute prédéfinie (HSCAPP.HPSetting) ou une erreur HSC est générée.
UFSetting	DINT	mot long	Lecture/Écriture	Paramétrage du débordement négatif. Le paramètre de débordement négatif HSCApp.UFSetting définit la limite de décompte basse pour le compteur. Si la valeur cumulée du compteur descend en dessous de la valeur spécifiée dans le paramètre UFSetting, une interruption de débordement négatif est générée. Dans un tel cas, le sous-système HSC réinitialise la valeur cumulée et la redéfinit sur la valeur de débordement ; puis le compteur relance le décompte à partir de cette valeur de débordement (le décompte n'est pas perdu lors de cette transition). Les valeurs UFSetting doivent répondre aux critères suivants : • Elles doivent se situer entre -2 147 483 648 et 2 147 483 647. • Elles doivent être inférieures à la valeur de débordement. • Elles doivent être inférieures ou égales aux données contenues dans la valeur basse prédéfinie (HSCAPP.LPSetting) ou une erreur HSC est générée.
OutputMask	UDINT	mot	Lecture/Écriture	Masque de sortie. HSCApp.OutputMask définit les sorties intégrées, sur le contrôleur, que le comptage rapide peut contrôler directement. Le sous-système HSC peut, sans interaction du programme de contrôle, activer et désactiver les sorties sur la base des valeurs prédéfinies haute et basse de l'accumulateur HSC. Le modèle de bit stocké dans HSCApp.OutputMask indique quelles sont les sorties contrôlées par HSC et lesquelles ne le sont pas. Le modèle de bit HSCAPP.OutputMask correspond aux bits de sorties du

Paramètre	Type de données	Format des données	Accès au programme utilisateur	Description
				contrôleur et peut être configuré uniquement durant l'installation initiale. Les bits configurés sur 1 sont activés et peuvent être activés ou désactivés par le sous-système HSC. Les bits configurés sur 0 ne peuvent être activés ou désactivés par le sous-système HSC. Par exemple, pour utiliser HSC afin de contrôler les sorties 0, 1, 3, spécifiez ce qui suit : • HscAppData.OutputMask = 2#1011 ou • HscAppData.OutputMask = 11
HPOutput	UDINT	mot long	Lecture/Écriture	Paramétrage de sortie 32 bits pour atteindre la valeur haute prédéfinie. HSCApp.HPOutput définit l'état (1 = activé / 0 = désactivé) des sorties du contrôleur, lorsque la valeur haute prédéfinie est atteinte. Pour plus d'informations sur la manière d'activer et de désactiver directement les sorties en fonction de la valeur haute prédéfinie. Configurez le modèle de bit de sortie haute durant l'installation initiale ou utilisez le bloc de fonctions HSC pour charger les nouveaux paramètres lorsque le contrôleur fonctionne.
LPOutput	UDINT	mot long	Lecture/Écriture	Paramétrage de sortie 32 bits pour atteindre la valeur basse prédéfinie. HSCApp.LPOutput définit l'état (1 = activé / 0 = désactivé) des sorties du contrôleur, lorsque la valeur basse prédéfinie est atteinte. Pour plus d'informations sur la manière d'activer et de désactiver directement les sorties en fonction de la valeur basse prédéfinie. Configurez le modèle de bit de sortie basse durant l'installation initiale ou utilisez le bloc de fonctions HSC pour charger les nouveaux paramètres lorsque le contrôleur fonctionne.

Paramètres HSCApp ou paramètres PLSData

Lorsque la fonction PLS est activée, les paramètres HSCApp pertinents sont remplacés par les paramètres PLSData correspondants, tel qu'indiqué dans le tableau suivant.

Paramètre HSCApp	Paramètre PLSData
HSCAPP.HpSetting	HSCHP
HSCAPP.LpSetting	HSCLP
HSCAPP.HPOutput	HSCHPOutput
HSCAPP.LPOutput	HSCLPOutput

HSCApp.HSCID

Le paramètre HSCApp.HSCID identifie le comptage rapide.

Le tableau suivant répertorie les valeurs pour HSCID :

Sélection de la sortie	Bit	Description
Premier mot des données de la fonction HSC	15-13	Type de module HSC : • 0x00 - Intégré. • 0x01 - Expansion. • 0x02 - Port enfichable.

	12-8	ID d'emplacement du module : • 0x00 - Intégré. • 0x01-0x1F - ID du module d'expansion. • 0x01-0x05 - ID du port enfichable.
	7-0	ID HSC dans le module : • 0x00-0x0F - Intégré. • 0x00-0x07 - ID HSC pour l'expansion. • 0x00-0x07 - ID HSC pour le port enfichable. dans la version initiale de Connected Components Workbench, seuls les ID 0x00-0x05 sont pris en charge.

HSCApp.HSCMode

Le paramètre HSCApp.HSCMode définit l'un des dix modes de décompte pour le comptage rapide. La valeur de mode est configurée via l'appareil de programmation et est accessible dans le programme de contrôle.

Pour obtenir des informations complémentaires sur les modes de fonctionnement HSC et l'affectation des entrées, consultez les rubriques Entrées HSC et Schéma de câblage du *Manuel d'utilisation des contrôleurs programmables Micro830 et Micro850*.

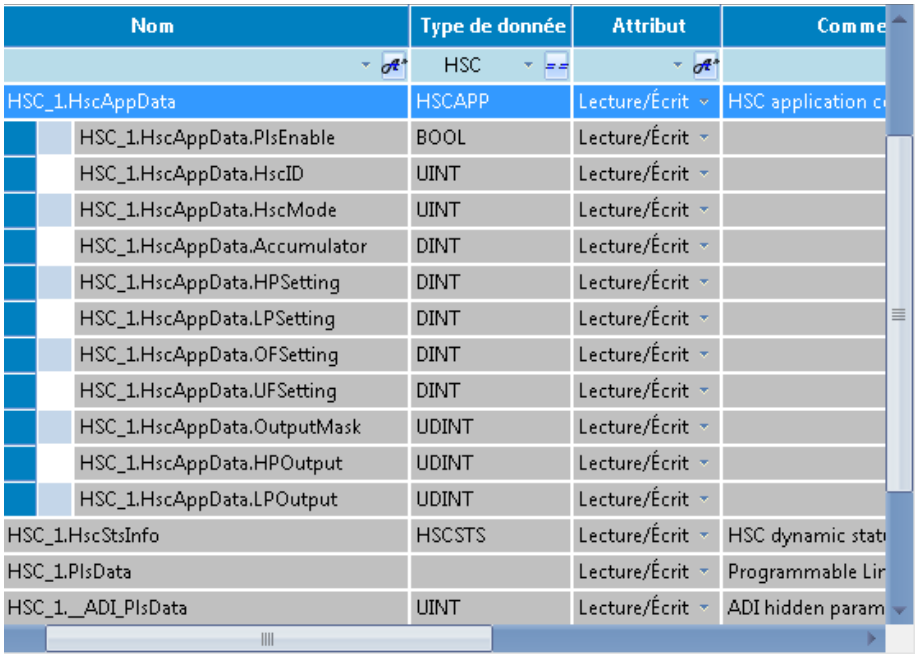
Les modes de fonctionnement HSC, le système principal et le sous-système HSC prennent en charge des modes différents.

- Les compteurs haute vitesse principaux prennent en charge dix modes d'exécution.
- Les sous-compteurs prennent en charge cinq modes de fonctionnement (modes 0, 2, 4, 6 et 8).
- Si le comptage rapide principal est défini sur le mode 1, 3, 5, 7 ou 9, le sous-comptage rapide est désactivé.

HSCMode	Mode de décompte
0	Compteur ascendant. L'accumulateur est immédiatement réinitialisé (0) lorsqu'il atteint la valeur haute prédéfinie. Aucune valeur basse ne peut être prédéfinie pour ce mode.
1	Compteur ascendant avec réinitialisation et conservation externes. L'accumulateur est immédiatement réinitialisé (0) lorsqu'il atteint la valeur haute prédéfinie. Aucune valeur basse ne peut être prédéfinie pour ce mode.
2	Compteur avec direction externe.
3	Compteur avec direction, réinitialisation et conservation externes.
4	Compteur double entrée (ascendante et descendante).
5	Compteur double entrée (ascendante et descendante) avec réinitialisation et conservation externes.
6	Encodeur en quadrature (entrées de phase A et B).
7	Encodeur en quadrature (entrées de phase A et B) avec réinitialisation et conservation externes.
8	Encodeur en quadrature X4 (entrées de phase A et B).
9	Encodeur en quadrature X4 (entrées de phase A et B) avec réinitialisation et conservation externes.

Exemples de paramètre HSCAppData

L'image suivante présente les paramètres HSCAppData du **Sélecteur de variable**.



Nom	Type de donnée	Attribut	Comme
	HSC		
HSC_1.HscAppData	HSCAPP	Lecture/Écrit	HSC application c
HSC_1.HscAppData.PlsEnable	BOOL	Lecture/Écrit	
HSC_1.HscAppData.HscID	UINT	Lecture/Écrit	
HSC_1.HscAppData.HscMode	UINT	Lecture/Écrit	
HSC_1.HscAppData.Accumulator	DINT	Lecture/Écrit	
HSC_1.HscAppData.HPSetting	DINT	Lecture/Écrit	
HSC_1.HscAppData.LPSetting	DINT	Lecture/Écrit	
HSC_1.HscAppData.OFSetting	DINT	Lecture/Écrit	
HSC_1.HscAppData.UFSetting	DINT	Lecture/Écrit	
HSC_1.HscAppData.OutputMask	UDINT	Lecture/Écrit	
HSC_1.HscAppData.HPOutput	UDINT	Lecture/Écrit	
HSC_1.HscAppData.LPOutput	UDINT	Lecture/Écrit	
HSC_1.HscStsInfo	HSCSTS	Lecture/Écrit	HSC dynamic statu
HSC_1.PlsData		Lecture/Écrit	Programmable Lir
HSC_1.__ADI_PlsData	UINT	Lecture/Écrit	ADI hidden param

Type de données HSCSTS

Le paramètre HSCSTSInfo (type de données HSCSTS) affiche le statut du compteur haute vitesse.

Actions de statut HSCSTSInfo

Lors du décompte HSC, les actions de statut HSC suivantes ont lieu.

- Si le bloc fonctionnel HSC effectue un décompte avec la commande 0x01, le statut HSC est mis à jour en continu.
- Si une erreur se produit, l'indicateur Error_Detected est activé et un code d'erreur est défini.

Paramètres HSCSTSInfo

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur	Description
CountEnable	BOOL	0...9	Lecture seule	Décompte activé.
ErrorDetected	BOOL	0...9	Lecture/Écriture	Une valeur différente de zéro indique la détection d'une erreur.
CountUpFlag	BOOL	0...9	Lecture seule	Indicateur de comptage croissant.
CountDwnFlag	BOOL	2...9	Lecture seule	Indicateur de comptage décroissant.
Mode1Done	BOOL	0 ou 1	Lecture/Écriture	HSC est en Mode 1A ou Mode 1B ; l'accumulateur compte jusqu'à la valeur HP.
OVF	BOOL	0...9	Lecture/Écriture	Un débordement a été détecté.
UNF	BOOL	0...9	Lecture/Écriture	Un débordement négatif a été détecté.
CountDir	BOOL	0...9	Lecture seule	1 : comptage croissant ; 0 : comptage décroissant.
HPReached	BOOL	2...9	Lecture/Écriture	Valeur haute prédéfinie atteinte.
LPReached	BOOL	2...9	Lecture/Écriture	Valeur basse prédéfinie atteinte.

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur	Description
OFCauseInter	BOOL	0...9	Lecture/Écriture	Le débordement a provoqué une interruption HSC.
UFCauseInter	BOOL	2...9	Lecture/Écriture	Le débordement négatif a provoqué une interruption HSC.
HPCauseInter	BOOL	0...9	Lecture/Écriture	La valeur haute prédéfinie a été atteinte, ce qui a provoqué une interruption HSC.
LPCauseInter	BOOL	2...9	Lecture/Écriture	La valeur basse prédéfinie a été atteinte, ce qui a provoqué une interruption HSC.
PlsPosition	UINT	0...9	Lecture seule	Position du commutateur limite programmable (PLS) Le paramètre PLSPosition est réinitialisé après qu'un cycle complet est terminé et que la valeur de HP est atteinte.
ErrorCode	UINT	0...9	Lecture/Écriture	Affiche les codes d'erreur détectés par le sous-système HSC.
Accumulator	DINT		Lecture/Écriture	Lecture de l'accumulateur réel.
HP	DINT		Lecture seule	Paramétrage de la dernière valeur haute prédéfinie.
LP	DINT		Lecture seule	Paramétrage de la dernière valeur basse prédéfinie.
HPOutput	UDINT		Lecture/Écriture	Paramétrage de la dernière valeur de sortie haute prédéfinie.
LPOutput	UDINT		Lecture/Écriture	Paramétrage de la dernière valeur de sortie basse prédéfinie.

Détail des paramètres HSCSTSInfo

Les paramètres HSCSTSInfo (type de données HSCSTS) sont utilisés pour déterminer le statut du compteur haute vitesse.

CountEnable

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.CountEnable	BOOL	0...9	Lecture seule

Indique le statut du compteur haute vitesse, que le décompte soit activé (1) ou désactivé (0, par défaut).

ErrorDetected

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.ErrorDetected	BOOL	0...9	Lecture/Écriture

Détecte si une erreur est présente au niveau du sous-système HSC. Les erreurs de configuration sont celles qui sont le plus souvent représentées par ErrorDetectedr. Lorsque ce bit est défini sur 1, recherchez le code d'erreur concerné dans le paramètre HSCSTS.ErrorCode, qui est géré par le contrôleur. Vous pouvez définir sur 0 le bit ErrorDetected, si nécessaire.

CountUpFlag

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.CountUpFlag	BOOL	0...9	Lecture seule

Utilisé avec tous les compteurs HSC (modes 0 à 9). Si le HSCSTS.CountEnable est défini sur 1, le bit Comptage croissant (Count Up) l'est également. Si le HSCSTS.CountEnable est défini sur 0, le bit Comptage croissant l'est également.

CountDownFlag

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.CountDownFlag	BOOL	2...9	Lecture seule

Utilisé avec les compteurs bidirectionnels (modes 2 à 9). Si le HSCSTS.CountEnable est défini sur 1, le bit Comptage décroissant (Count Down) l'est également. Si le HSCSTS.CountEnable est défini sur 0, le bit Comptage décroissant l'est également.

Mode1Done

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.Mode1Done	BOOL	0 ou 1	Lecture/Écriture

Le sous-système HSC définit l'indicateur de statut HSCSTS.Mode1Done sur 1 lorsque le compteur HSC est configuré pour le comportement Mode 0 ou Mode 1, et que l'accumulateur atteint la valeur haute prédéfinie.

OVF

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.OVF	BOOL	0...9	Lecture/Écriture

Le sous-système HSC définit l'indicateur de statut HSCSTS.OVF sur 1 lorsque la valeur cumulée (HSCSTS.Accumulator) a dépassé la variable de débordement (HSCAPP.OFSetting). Ce bit est transitionnel et défini par le sous-système HSC. C'est le programme de contrôle qui décide ou non d'utiliser, de surveiller et d'effacer (0) la condition de débordement.

Les conditions de débordement ne génèrent pas de défaut de contrôleur.

UNF

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.UNF	BOOL	0...9	Lecture/Écriture

Le sous-système HSC définit l'indicateur de statut HSCSTS.UNF sur 1 lorsque la valeur cumulée (HSCSTS.Accumulator) a dépassé la variable de débordement négatif (HSCAPP.UFSetting). Ce bit est transitionnel et défini par le sous-système HSC. C'est le programme de contrôle qui décide ou non d'utiliser, de suivre et d'effacer (0) la condition de débordement négatif.

Les conditions de débordement négatif ne génèrent pas de défaut de contrôleur.

CountDir

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.CountDir	BOOL	0...9	Lecture seule

Le sous-système HSC contrôle l'indicateur de statut HSCSTS.CountDir. Lorsque la valeur de l'accumulateur HSC augmente, l'indicateur de direction est défini sur 1. Lorsque la valeur de l'accumulateur HSC baisse, l'indicateur de direction est défini sur 0.

Si la valeur cumulée s'arrête, le bit de direction conserve sa valeur. L'indicateur de direction change uniquement lorsque la valeur cumulée s'inverse.

Ce bit est mis à jour en continu par le sous-système HSC lorsque le contrôleur est en mode exécution.

HPReached

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.HPReached	BOOL	2...9	Lecture/Écriture

Le sous-système HSC définit l'indicateur de statut HSCSTS.HPReached sur 1 lorsque la valeur cumulée (HSCSTS.Accumulator) est supérieure ou égale à la variable haute prédéfinie (HSCAPP.HPSetting).

Ce bit est mis à jour en continu par le sous-système HSC lorsque le contrôleur est en mode exécution. L'écriture dans cet élément n'est pas recommandée.

LPReached

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.LPReached	BOOL	2...9	Lecture seule

Le sous-système HSC définit l'indicateur de statut HSCSTS.LPReached sur 1 lorsque la valeur cumulée (HSCSTS.Accumulator) est inférieure ou égale à la variable basse prédéfinie (HSCAPP.LPSetting).

Ce bit est mis à jour en continu par le sous-système HSC lorsque le contrôleur est en mode exécution. L'écriture dans cet élément n'est pas recommandée.

OFCauseInter

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.OFCauseInter	BOOL	0...9	Lecture/Écriture

Le bit de statut Interruption de débordement est défini sur 1 lorsque la valeur de l'accumulateur HSC dépasse la valeur de débordement et que l'interruption HSC est déclenchée. Ce bit peut être utilisé dans le programme de contrôle pour déterminer si la variable de débordement a entraîné l'interruption HSC. Si le programme de contrôle doit exécuter une action de contrôle particulière sur la base du débordement, ce bit est utilisé comme logique conditionnelle.

Ce bit peut être défini sur 0 par le programme de contrôle, mais également par le sous-système HSC lorsque les conditions suivantes sont détectées :

- Exécution d'une interruption sur valeur basse prédéfinie
- Exécution d'une interruption sur valeur haute prédéfinie
- Exécution d'une interruption sur débordement négatif

UFCauseInter

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.UFCauseInter	BOOL	2...9	Lecture/Écriture

Le bit de statut Interruption de débordement négatif est défini sur 1 lorsque la valeur de l'accumulateur HSC dépasse la valeur de débordement négatif et que l'interruption HSC est déclenchée. Ce bit peut être utilisé dans le programme de contrôle pour déterminer si la condition de débordement négatif a entraîné l'interruption HSC. Si le programme de contrôle doit exécuter une action de contrôle particulière sur la base du débordement négatif, ce bit est utilisé comme logique conditionnelle.

Ce bit peut être défini sur 0 par le programme de contrôle, mais également par le sous-système HSC lorsque les conditions suivantes sont détectées :

- Présence d'une interruption sur valeur basse prédéfinie
- Présence d'une interruption sur valeur haute prédéfinie
- Présence d'une interruption sur débordement

HPCauseInter

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.HPCauseInter	BOOL	0...9	Lecture/Écriture

Le bit de statut Interruption sur valeur haute prédéfinie est défini sur 1 lorsque la valeur de l'accumulateur HSC atteint la valeur haute prédéfinie et que l'interruption HSC est déclenchée. Ce bit peut être utilisé dans le programme de contrôle pour déterminer si la condition de valeur haute prédéfinie a entraîné l'interruption HSC. Si le programme de contrôle doit exécuter une action de contrôle particulière sur la base de la valeur haute prédéfinie, ce bit est utilisé comme logique conditionnelle.

Ce bit peut être défini sur 0 par le programme de contrôle, mais également par le sous-système HSC lorsque les conditions suivantes sont détectées :

- Présence d'une interruption sur valeur basse prédéfinie
- Présence d'une interruption sur débordement négatif
- Présence d'une interruption sur débordement

LPCauseInter

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.LPCauseInter	BOOL	2...9	Lecture/Écriture

Le bit de statut Interruption sur valeur basse prédéfinie est défini sur 1 lorsque la valeur de l'accumulateur HSC atteint la valeur basse prédéfinie et que l'interruption HSC est déclenchée. Ce bit peut être utilisé dans le programme de contrôle pour déterminer si la condition de valeur basse prédéfinie a entraîné l'interruption HSC. Si le programme de contrôle doit exécuter une action de contrôle particulière sur la base de la valeur basse prédéfinie, ce bit est utilisé comme logique conditionnelle.

Ce bit peut être défini sur 0 par le programme de contrôle, mais également par le sous-système HSC lorsque les conditions suivantes sont détectées :

- Présence d'une interruption sur valeur haute prédéfinie

- Présence d'une interruption sur débordement négatif
- Présence d'une interruption sur débordement

PlsPosition

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.PLSPosition	UINT	0...9	Lecture seule

Lorsque le compteur HSC est en mode Décompte et que le Commutateur limite programmable (PLS) est activé, ce paramètre indique quel élément PLS est utilisé pour la configuration HSC en cours.

ErrorCode

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.ErrorCode	BOOL	0...9	Lecture seule

Affiche les codes d'erreur détectés par le sous-système HSC.

Sous-élément du code d'erreur	Code d'erreur de décompte HSC	Accès au programme utilisateur
Bit 15-8 (octet de poids fort)	0-255	Une valeur différente de zéro associée à l'octet de poids fort indique que l'erreur HSC est due au paramétrage des données PLS. La valeur de l'octet de poids fort indique quel élément des données PLS est à l'origine de l'erreur.
Bit 7-0 (octet de poids faible)	0x00	Aucune erreur.
	0x01	Mode de décompte HSC non valide.
	0x02	Valeur haute prédéfinie non valide.
	0x03	Débordement non valide.
	0x04	Débordement négatif non valide
	0x05	Aucune donnée PLS.

Accumulator

Paramètre	Type de données	Accès au programme utilisateur
HSCApp.Accumulator	DINT	Lecture/Écriture

Définit la valeur d'accumulateur initiale lorsque le compteur haute vitesse démarre. Lorsque le compteur HSC est en mode Décompte, l'accumulateur est automatiquement mis à jour par le sous-système HSC afin de refléter la valeur d'accumulateur HSC réelle.

HP

Paramètre	Type de données	Accès au programme utilisateur
HSCSTS.HP	DINT	Lecture seule

HSCSTS.HP est le point de consigne haut (en nombre) qui définit à quel moment le sous-système HSC génère une interruption.

Les données chargées dans la valeur haute prédéfinie doivent être inférieures ou égales aux données contenues dans le paramètre de débordement (HSCAPP.OFSetting) ou une erreur HSC est générée.

Il s'agit de la dernière valeur haute prédéfinie, qui peut être mise à jour par la fonction PLS depuis le bloc de données PLS.

LP

Paramètre	Type de données	Mode HSC	Accès au programme utilisateur
HSCSTS.LP	DINT		Lecture seule

HSCSTS.LP est le point de consigne bas (en nombre) qui définit à quel moment le sous-système HSC génère une interruption.

Les données chargées dans la valeur basse prédéfinie doivent être supérieures ou égales aux données contenues dans le paramètre de débordement négatif (HSCAPP.UFSetting) ou une erreur HSC est générée. Si la valeur de débordement négatif et la valeur basse prédéfinie sont négatives, la valeur absolue de cette dernière doit être inférieure.

Il s'agit de la dernière valeur basse prédéfinie, qui peut être mise à jour par la fonction PLS depuis le bloc de données PLS.

HPOutput

Paramètre	Type de données	Accès au programme utilisateur
HSCApp.HPOutput	UDINT	Lecture/Écriture

Définit l'état (1 = activé / 0 = désactivé) des sorties du contrôleur, lorsque la valeur haute prédéfinie est atteinte. Pour plus d'informations sur la manière d'activer et de désactiver directement les sorties en fonction de la valeur haute prédéfinie.

Vous pouvez configurer le modèle de bit de sortie haute durant l'installation initiale, ou bien utiliser le bloc fonctionnel HSC pour charger les nouveaux paramètres lorsque le contrôleur est en fonctionnement.

LPOutput

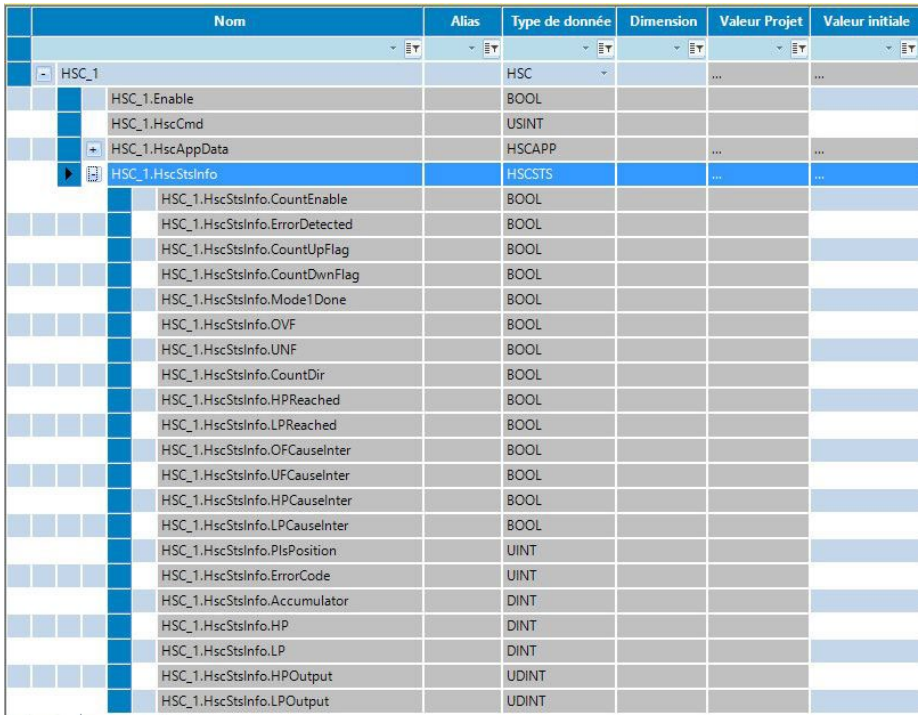
Paramètre	Type de données	Accès au programme utilisateur
HSCApp.LPOutput	UDINT	Lecture/Écriture

LPOutput (HSCApp.LPOutput) définit l'état (1 = activé / 0 = désactivé) des sorties du contrôleur, lorsque la valeur basse prédéfinie est atteinte. Pour plus d'informations sur la manière d'activer et de désactiver directement les sorties en fonction de la valeur basse prédéfinie.

Vous pouvez configurer le modèle de bit de sortie basse durant l'installation initiale, ou bien utiliser le bloc fonctionnel HSC pour charger les nouveaux paramètres lorsque le contrôleur est en fonctionnement.

Exemples de paramètre HSCSTSInfo

L'image suivante présente les paramètres HSCStsInfo du **Sélecteur de variable**.



Nom	Alias	Type de donnée	Dimension	Valeur Projet	Valeur initiale
HSC_1		HSC		...	...
HSC_1.Enable		BOOL			
HSC_1.HscCmd		USINT			
HSC_1.HscAppData		HSCAPP		...	...
HSC_1.HscStsInfo		HSCSTS		...	...
HSC_1.HscStsInfo.CountEnable		BOOL			
HSC_1.HscStsInfo.ErrorDetected		BOOL			
HSC_1.HscStsInfo.CountUpFlag		BOOL			
HSC_1.HscStsInfo.CountDwnFlag		BOOL			
HSC_1.HscStsInfo.Mode1Done		BOOL			
HSC_1.HscStsInfo.OVF		BOOL			
HSC_1.HscStsInfo.UNF		BOOL			
HSC_1.HscStsInfo.CountDir		BOOL			
HSC_1.HscStsInfo.HPReached		BOOL			
HSC_1.HscStsInfo.LPReached		BOOL			
HSC_1.HscStsInfo.OFCauseInter		BOOL			
HSC_1.HscStsInfo.UFCauseInter		BOOL			
HSC_1.HscStsInfo.HPCauseInter		BOOL			
HSC_1.HscStsInfo.LPCauseInter		BOOL			
HSC_1.HscStsInfo.PlsPosition		UINT			
HSC_1.HscStsInfo.ErrorCode		UINT			
HSC_1.HscStsInfo.Accumulator		DINT			
HSC_1.HscStsInfo.HP		DINT			
HSC_1.HscStsInfo.LP		DINT			
HSC_1.HscStsInfo.HPOutput		UDINT			
HSC_1.HscStsInfo.LPOutput		UDINT			

Type de données PLS

Le paramètre PLSData (type de données PLS) est utilisé pour configurer l'interrupteur à came rotatif sur la page 311.

Éléments de structure PLSData

La structure des données PLS se présente sous la forme d'un tableau flexible, composé des éléments suivants.

Élément	Ordre des éléments	Type de données	Description de l'élément
HSCHP	Mot 0 - 1	DINT	Valeur haute prédéfinie
HSCLP	Mot 2 - 3	DINT	Valeur basse prédéfinie
HSCHPOutput	Mot 4 - 5	UDINT	Données de sortie hautes
HSCLPOutput	Mot 6 - 7	UDINT	Données de sortie basses

Le nombre total d'éléments pour une structure de données PLS ne doit pas être supérieur à 255.

Paramètres PLSData

Le tableau suivant répertorie le détail des paramètres PLSData.

Élément	Type de données	Format des données	Mode HSC	Accès au programme utilisateur	Description
HSCHP	DINT	Entier signé 32 bits	0	Lecture/Écriture	Valeur haute prédéfinie
HSCLP	DINT	Entier signé 32 bits	0	Lecture/Écriture	Valeur basse prédéfinie

Élément	Type de données	Format des données	Mode HSC	Accès au programme utilisateur	Description
HSCHPOutput	UDINT	Binaire 32 bits	0	Lecture/Écriture	Données de sortie hautes
HSCLPOutput	UDINT	Binaire 32 bits	0	Lecture/Écriture	Données de sortie basses

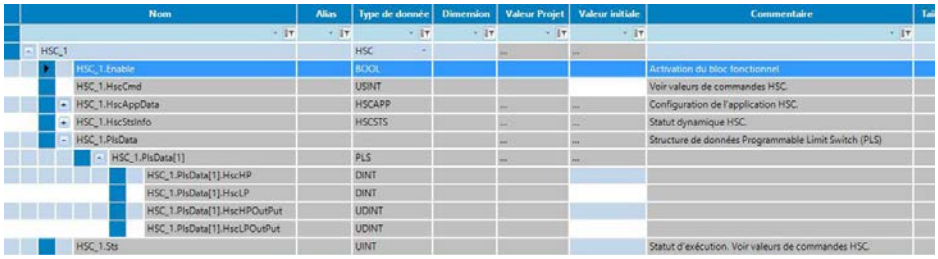
Paramètres HSCApp ou paramètres PLSData

Lorsque la fonction PLS est activée, les paramètres HSCApp pertinents sont remplacés par les paramètres PLSData correspondants, tel qu'indiqué dans le tableau suivant.

Paramètre HSCApp	Paramètre PLSData
HSCAPP.HpSetting	HSCHP
HSCAPP.LpSetting	HSCLP
HSCAPP.HPOutput	HSCHPOutput
HSCAPP.LPOutput	HSCLPOutput

Exemples de paramètre PLSData

La figure suivante présente les paramètres PLSData du **Sélecteur de variable**.



Type de données HSCE_CHANNEL

Cela décrit le type de données HSCE_CHANNEL :

```
@typedef struct struct_HSCE_channel
{

USINT    ModuleType;
USINT    SlotID;
USINT    HSCID;
} HSCE_CHANNEL;
```

Voici la description :

Octet	Description
Type de module	0x00 : Intégré 0x01 : Extension (découpée) 0x02 : Port universel
ID d'emplacement	0x00 : Intégré 0x01-0x1F : ID du module d'extension (découpé) 0x01-0x05 : ID de port universel
HSCID	0x00-0x0F : Intégré 0x00-0x07 : ID HSC sur la page 288 pour l'extension 0x00-0x01 : ID HSC pour le port universel

Type de données HSCE_STS

Voici la description du type de données HSCE_STS :

Paramètre	Type de données	Mode HSC	Description
CountEnable	BOOL	0...13	Décompte activé.
ErrorDetected	BOOL	0...13	Une valeur différente de zéro indique la détection d'une erreur.
CountUpFlag	BOOL	0...13	Indicateur de comptage croissant.
CountDwnFlag	BOOL	2...13	Indicateur de comptage décroissant.
Mode1Done	BOOL	0 ou 1	HSC est en Mode 1A ou Mode 1B ; l'accumulateur compte jusqu'à la valeur HP.
OVF	BOOL	0...13	Un débordement a été détecté.
UNF	BOOL	0...13	Un débordement négatif a été détecté.
CountDir	BOOL	0...13	1 : comptage croissant ; 0 : comptage décroissant.
HPReached	BOOL	2...13	Valeur haute prédéfinie atteinte.
LPReached	BOOL	2...13	Valeur basse prédéfinie atteinte.
OFCauseInter	BOOL	0...13	Le débordement a provoqué une interruption HSC sur la page 288.
UFCauseInter	BOOL	2...13	Le débordement négatif a provoqué une interruption HSC.
HPCauseInter	BOOL	0...13	La valeur haute prédéfinie a été atteinte, ce qui a provoqué une interruption HSC.
LPCauseInter	BOOL	2...13	La valeur basse prédéfinie a été atteinte, ce qui a provoqué une interruption HSC.
StateInfo	USINT		Informations d'état de la machine du compteur HSCE
PlsPosition	UINT	0...9	Position du commutateur limite programmable (PLS) Le paramètre PLSPosition est réinitialisé après qu'un cycle complet est terminé et que la valeur de HP est atteinte.
ErrorCode	UINT	0...13	Affiche les codes d'erreur détectés par le sous-système HSC.
Accumulator	DINT		Lecture de l'accumulateur réel.
HP	DINT		Paramétrage de la dernière valeur haute prédéfinie.
LP	DINT		Paramétrage de la dernière valeur basse prédéfinie.
HPOutput	UDINT		Paramétrage de la dernière valeur de sortie haute prédéfinie.
LPOutput	UDINT		Paramétrage de la dernière valeur de sortie basse prédéfinie.

Type de données PLS_HSCE

La fonction Interrupteur à came rotatif est un ensemble supplémentaire de modes de fonctionnement pour le comptage rapide. Lors du fonctionnement dans ces modes, les valeurs des données prédéfinies et de sortie sont mises à jour à l'aide des données fournies par l'utilisateur à chaque fois que l'une des valeurs prédéfinies est atteinte. Ces modules sont programmés en fournissant un fichier PLS qui contient les ensembles de données à utiliser. La structure de données PLS_HSCE est un tableau flexible dont chaque élément est défini comme suit :

Élément	Type de données	Description de l'élément
HighPreset	LINT	Paramètre de valeur haute prédéfinie
LowPreset	LINT	Paramètre de valeur basse prédéfinie
HiPresetOutput	UDINT	Données de sortie de la valeur haute prédéfinie
LoPresetOutput	UDINT	Données de sortie de la valeur basse prédéfinie

```
Structure // pour un élément PLS
typedef struct PLS_HSCE_EleStruct
{
LINT HighPreset; // Valeur haute prédéfinie HSC
LINT LowPreset; // Valeur basse prédéfinie HSC
```

```
UDINT HiPresetOutput; // Sortie de valeur haute prédéfinie
HSC
UDINT LoPresetOutput; // Sortie de valeur basse prédéfine
HSC
} PLS_HSCE_EleStruct;
```

Le nombre total d'éléments pour une donnée PLS_HSCE ne doit pas être supérieur à 24 pour un module HSC.

HSC_SET_STS (état défini du comptage rapide)

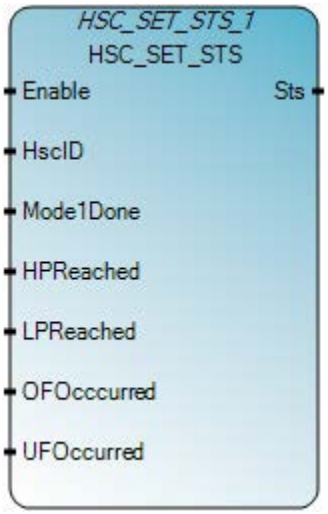
HSC_SET_STS définit ou réinitialise manuellement l'état de décompte HSC.

Détails de l'opération :

- Le bloc de fonctions HSC doit être arrêté (aucun décompte) pour que le bloc de fonctions HSC_SET_STC puisse définir ou réinitialiser l'état HTS. Si la fonction HSC n'est pas arrêtée, les paramètres d'entrée continuent à être mis à jour, et toute modification apportée à l'aide du bloc de fonctions HSC_SET_STS est ignorée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : définition/réinitialisation de l'état HSC. FAUX : l'état HSC ne change pas.
HscID	Entrée	UINT	Définit ou réinitialise manuellement l'état HSC.
Mode1Done	Entrée	BOOL	Comptage du mode 1A ou 1B est complété. Ce bit peut être défini ou réinitialisé lorsqu'aucun décompte HSC n'est en cours.
HPReached	Entrée	BOOL	Valeur haute prédéfinie atteinte. Ce bit peut être défini ou réinitialisé lorsqu'aucun décompte HSC n'est en cours.
LPReached	Entrée	BOOL	Valeur basse prédéfinie atteinte. Ce bit peut être défini ou réinitialisé lorsqu'aucun décompte HSC n'est en cours.

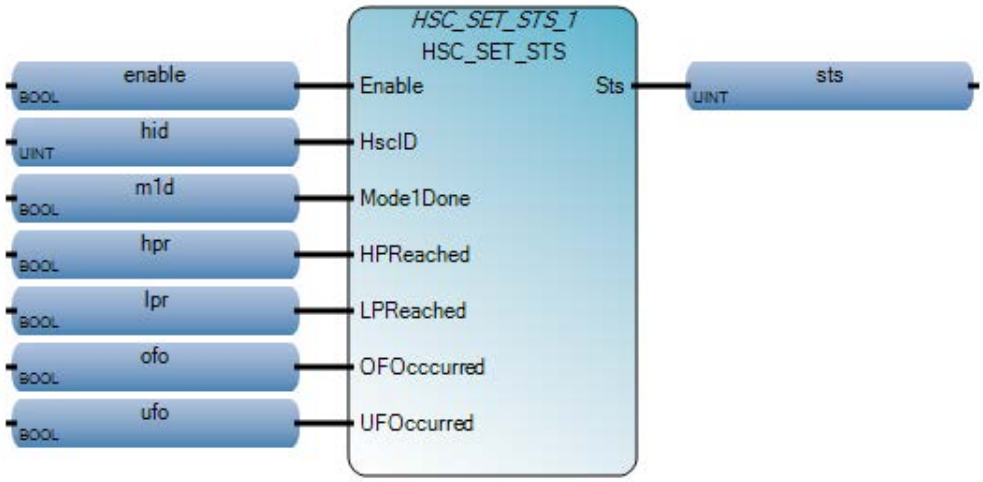
Paramètre	Type de paramètre	Type de données	Description
OFOccurred	Entrée	BOOL	Un débordement est survenu. Ce bit peut être défini ou réinitialisé lorsqu'aucun décompte HSC n'est en cours.
UFOccurred	Entrée	BOOL	Un débordement négatif est survenu. Ce bit peut être défini ou réinitialisé lorsqu'aucun décompte HSC n'est en cours.
Sts	Sortie	UINT	Les codes d'état sont définis dans les codes d'état HSC (Sts).
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Codes d'état HSC (Sts)

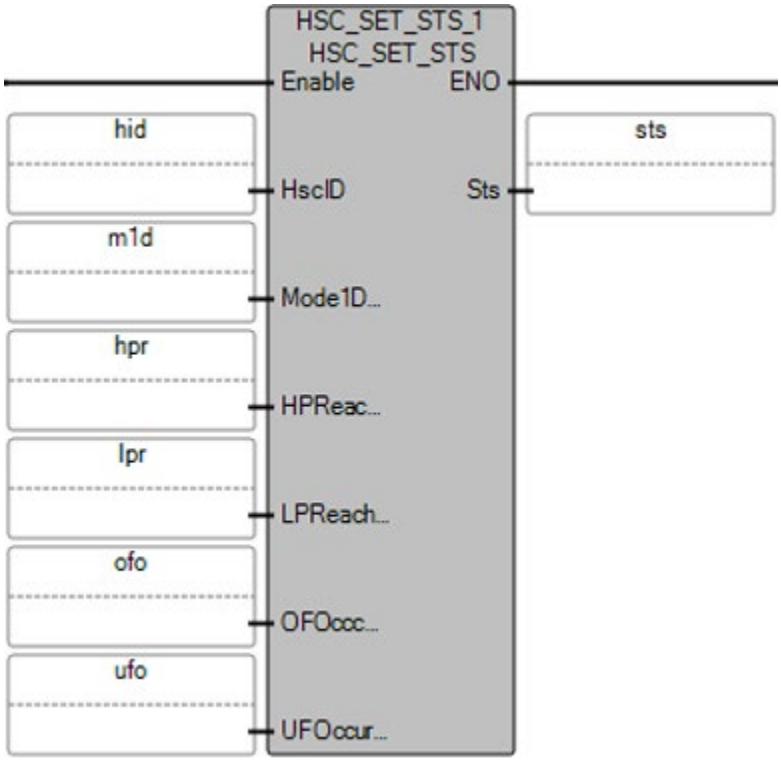
Le tableau suivant décrit les codes d'état pour le bloc de fonctions HSC.

Code d'état	Description de l'état
0x00	Aucune action n'est réalisée (non activée).
0x01	Exécution HSC réussie.
0x02	Commande HSC non valide.
0x03	ID HSC hors plage.
0x04	Erreur de configuration HSC.

Exemple de Diagramme de Bloc Fonction HSC_SET_STS



Exemple de langage à relais HSC_SET_STS



Exemple **de texte structuré HSC_SET_STS**

```
HSC_SET_STS_1(
void HSC_SET_STS_1(BOOL Enable, UINT HscID, BOOL Mode1Done, BOOL HPReached, BOOL LPReached, BOOL OFOcccurred, BOOL UFOccurred)
Type: HSC_SET_STS, Applique/réapplique manuellement le statut HSC.
1  HSC_SET_STS_1(enable, hid, m1d, hpr, lpr, ofo, ufo);
2  sts := HSC_SET_STS_1.Sts;
```

Utilisation des instructions de comptage rapide

Cette section fournit des détails précis et des exemples sur l'utilisation des instructions de compteur haute vitesse dans les programmes logiques, notamment :

Mise à jour des données d'application HSC sur la page 306

Configuration d'une interruption par l'utilisateur du comptage rapide (HSC) sur la page 307

Configuration d'un interrupteur à came rotatif (PLS) sur la page 311

Exemple : Création d'un compteur haute vitesse (HSC) sur la page 312

Exemple : Ajout d'une fonction Commutateur limite programmable (PLS) sur la page 321

Exemple : Commutateur limite programmable (PLS) activé sur la page 322

Mise à jour des données d'application HSC

La configuration HSC est définie dans les données d'application HSC et n'est généralement configurée qu'une seule fois, avant la programmation de l'instruction HSC. Les modifications apportées aux données d'application

HSC (paramètre HSCAppData) sont ignorées lorsque le HSC est en cours de décompte.

Pour modifier la configuration HSC

1. Mettez à jour HSCAppData.
2. Appelez l'instruction HSC sur la page 288 avec la commande 0x03 (définir/recharger).

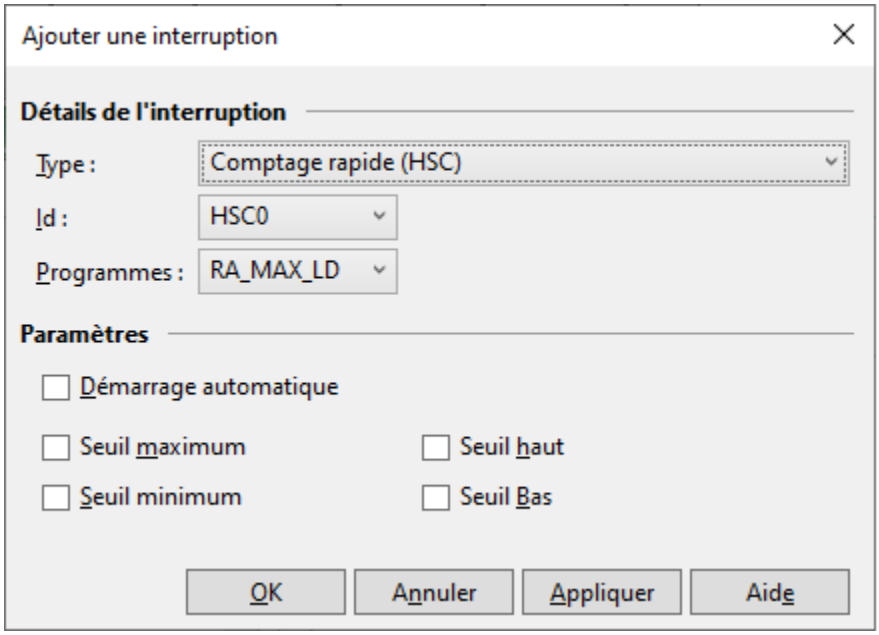
Boîte de dialogue Interruption par l'utilisateur de compteur haute vitesse (HSC)

Ouverture de la boîte de dialogue Interruption par l'utilisateur de compteur haute vitesse (HSC)

Dans Type d'interruption, sélectionnez **Interruption par l'utilisateur de compteur haute vitesse (HSC)**.

Servez-vous de la boîte de dialogue d'interruption HSC pour :

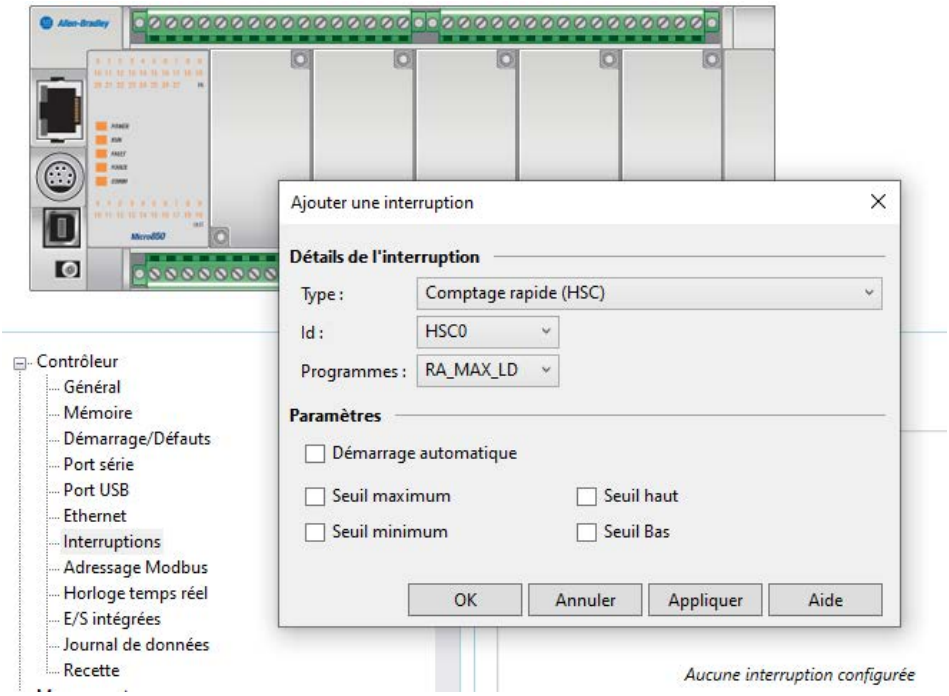
- Configurer les propriétés d'interruption sur la page 307, telles que l'ID et le programme avec lesquels l'utiliser
- Configurer les paramètres d'interruption



Configuration d'une interruption par l'utilisateur du comptage rapide (HSC)

Au niveau du contrôleur, une interruption par l'utilisateur entraîne la suspension de la tâche en cours, le lancement d'une tâche différente et la reprise de la tâche suspendue au point où elle s'était arrêtée.

Les contrôleurs Micro830, Micro850 et Micro870 prennent en charge jusqu'à six interruptions HSC par l'utilisateur ; celles-ci peuvent être utilisées pour exécuter la logique utilisateur sélectionnée lors d'un événement préconfiguré.



Ajout et configuration d'une interruption utilisateur du compteur haute vitesse (HSC)

Pour ajouter et configurer une interruption HSC sur la page 307 à partir de l'espace de travail de configuration du contrôleur, procédez comme suit.

Pour ajouter une interruption HSC, procédez comme suit :

1. Dans le **Gestionnaire de projets**, double-cliquez sur le contrôleur pour ouvrir son espace de travail.
2. Dans l'**Arborescence du contrôleur**, cliquez sur **Interruptions** pour afficher la page de configuration **Interruptions**.
3. Cliquez avec le bouton droit de la souris sur une ligne vide, puis cliquez sur **Ajouter** pour ouvrir la boîte de dialogue **Propriétés d'interruption**.
4. Pour configurer une interruption HSC, procédez comme suit :
   - Dans Type d'interruption, sélectionnez **Interruption par l'utilisateur de compteur haute vitesse (HSC)**.
   - Sélectionnez les propriétés d'interruption HSC sur la page 308.
   - Sélectionnez les paramètres d'interruption HSC sur la page 310.
5. Fermez la boîte de dialogue **Propriétés d'interruption**.

Propriétés d'interruption HSC

Les bits de statut des propriétés d'interruption HSC indiquent le statut activé/désactivé, le statut d'exécution et si la condition d'interruption a été ou non perdue.

Activation d'une interruption utilisateur (HSC0.Enabled)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.Enabled	bit	0...9	Lecture seule

Le bit Activé est utilisé pour indiquer le statut d'activation ou de désactivation de l'interruption HSC.

Exécution d'une interruption utilisateur (HSC0.EX)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.Ex	bit	0...9	Lecture seule

Le bit EX (exécution d'une interruption utilisateur) est défini sur 1 lorsque le sous-système HSC commence le traitement de la sous-routine HSC, en raison de l'une des conditions suivantes :

- Valeur basse prédéfinie atteinte
- Valeur haute prédéfinie atteinte
- Débordement, comptage croissant jusqu'à la valeur de débordement
- Débordement négatif, comptage décroissant jusqu'à la valeur de débordement négatif

Le bit EX HSC peut être utilisé dans le programme de contrôle comme logique conditionnelle afin de détecter si une interruption HSC est en cours d'exécution.

Le sous-système HSC définit le EX sur 0 lorsque le contrôleur termine le traitement de la sous-routine HSC.

Interruption utilisateur en attente (HSC0.PE)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.PE	bit	0...9	Lecture seule

L'indicateur de statut PE (interruption utilisateur en attente) signale qu'une interruption est en attente. Le bit de statut PE peut être surveillé ou utilisé à des fins logiques par le programme de contrôle, si vous souhaitez être averti lorsqu'une sous-routine ne peut être exécutée immédiatement. Le bit PE est géré par le contrôleur et défini sur 1 ou sur 0 automatiquement.

Perte d'interruption utilisateur (HSC0.LS)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.LS	bit	0...9	Lecture seule

L'indicateur de statut LS (perte d'interruption utilisateur) signale qu'une interruption a été perdue. Le contrôleur peut traiter une condition d'interruption utilisateur active et gérer une condition d'interruption utilisateur en attente avant d'activer le bit de perte d'interruption.

Le bit LS est défini par le contrôleur. C'est le programme de contrôle qui décide ou non d'utiliser et de surveiller une condition de perte d'interruption.

Paramètres d'interruption HSC

Les paramètres d'interruption HSC sont utilisés pour configurer les options de masque et de démarrage.

Démarrage automatique (HSC0.AS)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.AS	bit	0...9	Lecture seule

Le démarrage automatique (Auto Start) est configuré avec l'appareil de programmation et stocké dans le programme utilisateur. Le bit de démarrage automatique indique si la fonction d'interruption HSC démarre automatiquement lorsque le contrôleur passe en mode exécution ou test.

Masque de débordement (HSC0.MV)

Le bit de contrôle **MV** (masque de débordement) permet d'activer (autoriser) ou de désactiver (interdire) l'interruption du débordement. Si le bit est défini sur o et qu'une condition **Débordement** est détectée par le compteur **HSC**, l'interruption utilisateur **HSC** n'est pas exécutée.

Le bit **MV** est contrôlé par programme utilisateur et conserve sa valeur tout au long du cycle de mise sous tension. Le programme utilisateur doit activer et désactiver le bit **MV**.

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.MV	bit	0...9	Lecture seule

Masque de débordement négatif (HSC0.MN)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.MN	bit	2...9	Lecture seule

Le bit de contrôle MN (masque de soupassement) permet d'activer (autoriser) ou de désactiver (interdire) l'interruption du soupassement. Si le bit est défini sur o et qu'une condition Débordement négatif est détectée par le compteur HSC, l'interruption par l'utilisateur HSC n'est pas exécutée.

Le bit MN est contrôlé par le programme utilisateur et conserve sa valeur lors de l'arrêt et de la remise sous tension. Le programme utilisateur doit activer et désactiver le bit MN.

Masque de valeur haute prédéfinie (HSC0.MH)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.MH	bit	0...9	Lecture seule

Le bit de contrôle MH (seuil haut) permet d'activer (autoriser) ou de désactiver (interdire) l'interruption de la valeur prédéfinie haute. Si le bit est défini sur 0 et qu'une condition Valeur prédéfinie haute atteinte est détectée par le compteur HSC, l'interruption par l'utilisateur HSC n'est pas exécutée.

Le bit MH est contrôlé par le programme utilisateur et conserve sa valeur lors de l'arrêt et de la remise sous tension. Le programme utilisateur doit activer et désactiver le bit MH.

Masque de valeur basse prédéfinie (HSC0.ML)

Paramètre	Format des données	Modes HSC	Accès au programme utilisateur
HSC0.ML	bit	2...9	Lecture seule

Le bit de contrôle ML (seuil bas) permet d'activer (autoriser) ou de désactiver (interdire) une interruption de valeur prédéfinie basse. Si le bit est défini sur 0 et qu'une condition Valeur prédéfinie basse atteinte est détectée par le compteur HSC, l'interruption par l'utilisateur HSC n'est pas exécutée.

Le bit ML est contrôlé par programme utilisateur et conserve sa valeur tout au long du cycle de mise sous tension. Le programme utilisateur doit activer et désactiver le bit ML.

Configuration d'un interrupteur à came rotatif (PLS)

Le compteur haute vitesse sur la page 288 dispose de modes de fonctionnement complémentaires pour la mise en œuvre d'un interrupteur à came rotatif (Programmable Limit Switch, PLS). La fonction PLS est utilisée pour configurer le compteur haute vitesse afin qu'il joue le rôle de PLS (commutateur limite programmable) ou de commutateur de came rotative. La fonction PLS prend en charge jusqu'à 255 paires de valeurs prédéfinies hautes et basses ; elle peut être utilisée lorsque vous avez besoin de plusieurs paires de valeurs prédéfinies.

Activation du PLS dans le HSC

Le mode PLS fonctionne uniquement en tandem avec le HSC du contrôleur Micro800, et doit être activé au niveau de l'instruction HSC en définissant le paramètre HSCAppData.PLSEnable sur True.

Le paramètre PLSPosition est réinitialisé après qu'un cycle complet est terminé et que la valeur de HSCSTS.HP est atteinte. La réinitialisation de l'instruction HSC ou la définition du paramètre PLSPositon sur 0 ne réinitialise pas PLSPosition.

Exécution de compteur HSC lorsque le PLS est activé

La fonction PLS est compatible avec les autres fonctionnalités HSC, y compris sa capacité à déterminer quels événements HSC génèrent une interruption utilisateur.

Lorsque la fonction PLS est activée sur la page 322 et que le contrôleur est en mode exécution, le HSC dénombre les impulsions entrantes, et les événements suivants se produisent.

- Lorsque le décompte atteint la première valeur prédéfinie (HSCHP ou HSCLP) dans les données PLS, les données de source de sortie (HSCHPOutput ou HSCLPOutput) sont écrites par le biais du masque HSC (HSCAPP.OutputMask).
- À ce moment-là, les valeurs prédéfinies suivantes (HSCHP et HSCLP) dans les données PLS deviennent actives.
- Lorsque le décompte HSC atteint la nouvelle valeur prédéfinie, les nouvelles données de sortie sont écrites par le biais du masque HSC.
- Le traitement se poursuit ainsi jusqu'à ce que le dernier élément du bloc de données PLS soit chargé.
- L'élément actif du bloc de données PLS est alors remis à zéro.
- Ce comportement est appelé « fonctionnement circulaire ».

Différence de valeur PLS prédéfinie entre le HSC intégré et le HSC du module enfichable

Les comportements des valeurs prédéfinies HSCHP PLS et HSCLP sont différents entre le HSC intégré et le HSC du module enfichable. Le bit Valeur prédéfinie haute HSC intégré sera uniquement défini lorsque le dernier PLS est exécuté, alors le bit Valeur prédéfinie haute HSC du module enfichable sera défini lorsque le premier PLS est exécuté. Par exemple :

- HSC PLS0-PLS23 intégré :

  Le bit Valeur prédéfinie haute sera défini lorsque PLS23HP est égal à la valeur de l'accumulateur
  Le bit Valeur prédéfinie basse sera défini lorsque PLS23LP est égal à la valeur de l'accumulateur

- HSC PLS0-PLS23 du module enfichable :

  Le bit Valeur prédéfinie haute sera défini lorsque PLS0HP est égal à la valeur de l'accumulateur
  Le bit Valeur prédéfinie basse sera défini lorsque PLS0LP est égal à la valeur de l'accumulateur

Exemple : Création d'un programme de compteur haute vitesse (HSC)

Cet exemple explique comment créer un programme de compteur haute vitesse (HSC) sur la page 288, utilisant un encodeur en quadrature et disposant d'une fonction d'interrupteur à came rotatif (Programmable Limit Switch, PLS).

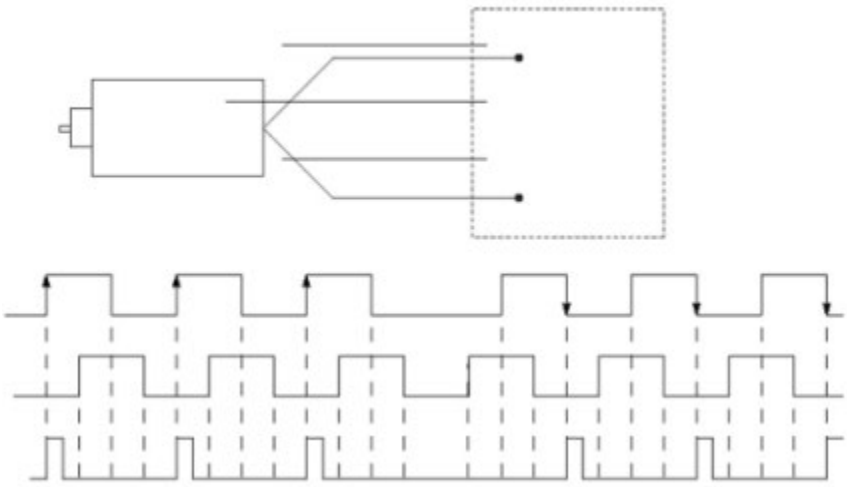
Encodeur en quadrature utilisé dans l'exemple

L'exemple de programme HSC (compteur haute vitesse) utilise un bloc fonctionnel HSC et un encodeur en quadrature avec entrées de phase A et B.

L'encodeur détermine le sens de rotation et la position des équipements rotatifs, par exemple un tour. Le compteur bidirectionnel compte la rotation de l'encodeur en quadrature.

L'encodeur en quadrature suivant est connecté aux entrées 0 et 1. Le sens de décompte est déterminé par l'angle de phase entre A et B :

- Si A est devant B, le compteur augmente.
- Si B est devant A, le compteur baisse.



Création d'un programme de comptage rapide (HSC)

Effectuez les tâches suivantes pour créer, construire et tester le programme HSC, puis ajoutez une fonction PLS (Commutateur limite programmable).

Étape	Tâche
1	Création d'une logique à relais et ajout de variables sur la page 313
2	Affectation de valeurs aux variables HSC sur la page 316
3	Affectation de valeurs aux variables et construction du programme sur la page 317
4	Test du programme et utilisation du compteur haute vitesse (HSC) sur la page 319
5	Ajout d'une fonction Commutateur limite programmable (PLS) sur la page 321

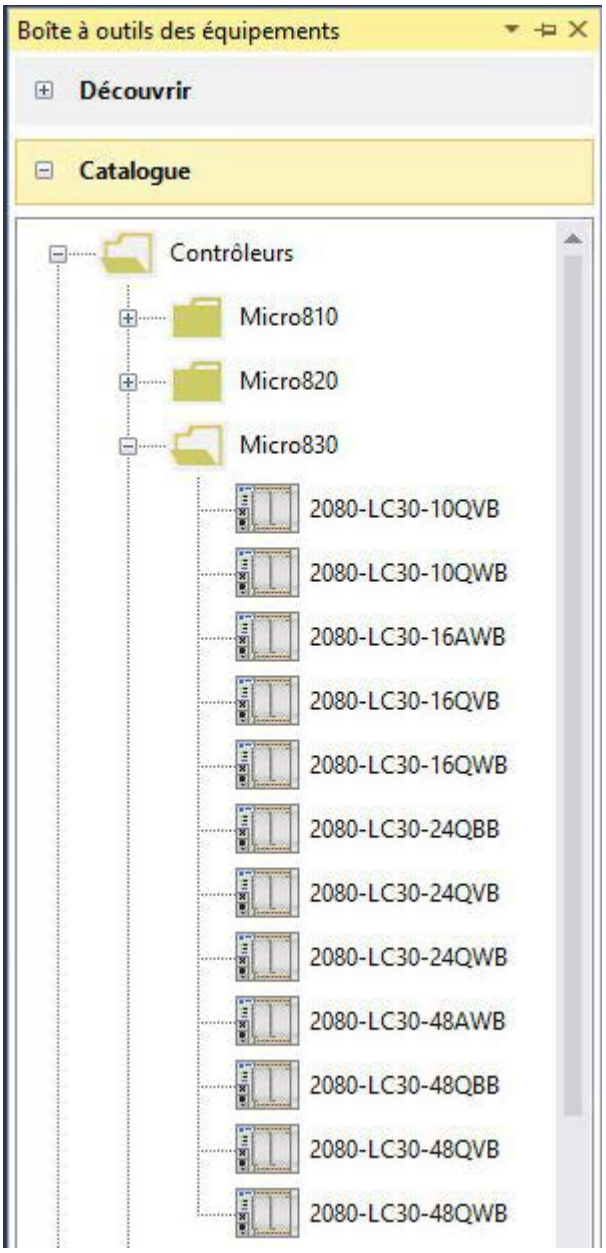
Création d'une logique à relais et ajout de variables

Créez une logique à relais et ajoutez des variables locales dans la ligne. Cet exemple de programme est basé sur un contrôleur 2080-LC50-24QVB. Le HSC sur la page 288 est pris en charge sur tous les contrôleurs Micro830 et Micro850, excepté les contrôleurs de type 2080-LCxx-xxAWB.

Pour créer une logique à relais et ajouter des variables, procédez comme suit :

1. Dans la **Boîte à outils des appareils**, développez l'onglet **Catalogue** pour afficher les dossiers d'appareil.

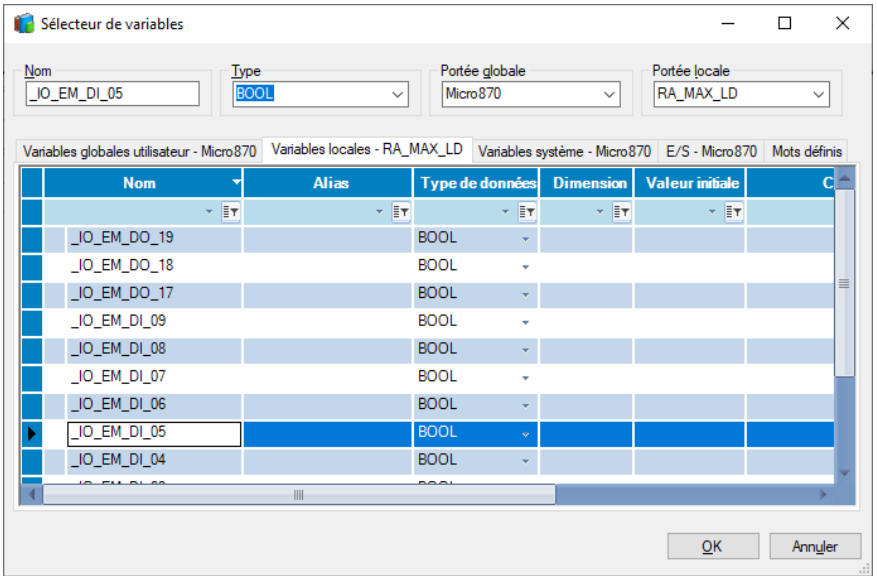
2. Développez le dossier Contrôleurs et le dossier Micro830 afin d'afficher tous les contrôleurs Micro830. Double-cliquez sur un contrôleur (2080-LC50-24QVB) pour l'ajouter à le **Gestionnaire de projet**.



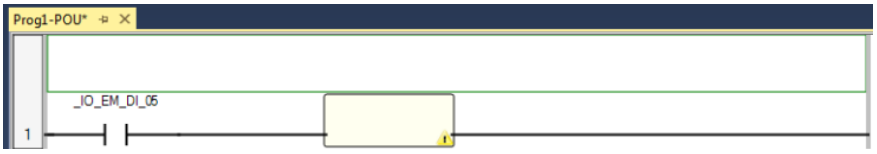
3. Dans l'**Organisateur de projets**, cliquez avec le bouton droit de la souris sur **Programmes**, cliquez sur **Ajouter,** puis cliquez sur **Nouvelle : Logique à relais** pour ajouter un nouveau programme de logique à relais.
4. Cliquez avec le bouton droit de la souris sur **UntitledLD** et sélectionnez **Ouvrir**.
5. Dans la boîte de dialogue **Boîte à outils**.
   - Double-cliquez sur **Contact normalement ouvert** pour l'ajouter à la ligne. Ou bien,

- Déplacez un élément **Contact normalement ouvert** en le faisant glisser jusqu'à la ligne.

6. Affectez une variable au contact normalement ouvert :
   - Double-cliquez sur le contact normalement ouvert pour afficher le **Sélecteur de variable**, puis cliquez sur l'onglet **E/S - Micro830**.
   - Cliquez sur **_IO_EM_DI_05**, puis sur **OK** pour affecter le contact normalement ouvert à l'entrée 5.

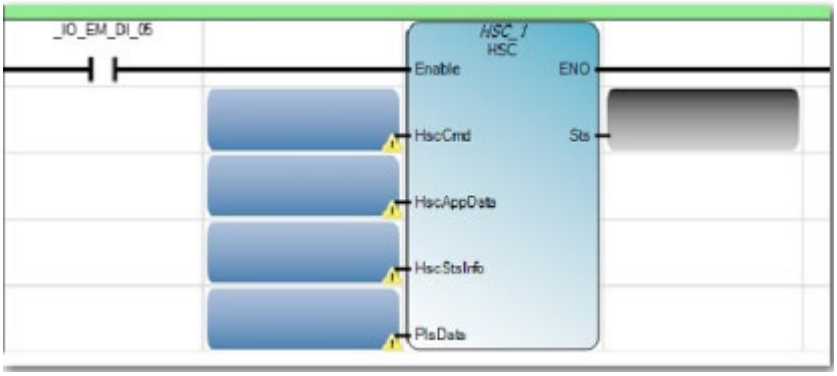


7. Dans la boîte de dialogue **Boîte à outils**, sélectionnez un bloc de fonctions et faites-le glisser vers la droite du contact normalement ouvert, comme illustré dans l'image suivante.



8. Double-cliquez sur le bloc de fonctions pour ouvrir le **Sélecteur de bloc d'instruction**.
9. Dans le **Sélecteur de bloc d'instruction**, sélectionnez **HSC** et cliquez sur **OK**.

10. Vérifiez que la ligne de logique à relais est similaire à l'exemple suivant.

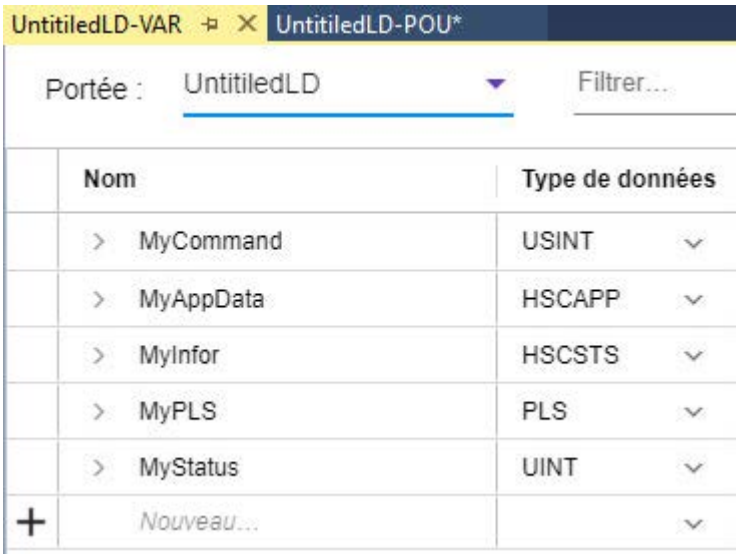


11. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables locales** pour afficher la page **Variables**.
12. Sur la page **Variables**, ajoutez les variables et les types de données suivants.

Nom de variable	Type de données
MyCommand	USINT
MyAppData	HSCAPP
MyInfo	HSCSTS
MyPLS	PLS
MyStatus	UINT

Résultat

La page **Variables** ressemble à l'image ci-dessous :



Affectation de valeurs aux variables HSC

Après avoir ajouté des variables, observez les étapes suivantes pour ajouter des valeurs à ces variables à l'aide de la colonne Valeur initiale du **Sélecteur de variable**. Un programme standard utilise généralement une routine pour affecter une valeur aux variables.

Pour affecter des valeurs aux variables HSC, procédez comme suit :

1. Développez la section **MyAppData** pour afficher toutes les variables.
2. Définissez la valeur du mode HSC :
   - Dans le champ **Valeur initiale** de la variable MyAppData.HSCMode, saisissez 6.
   - Consultez HSCMode dans la rubrique Type de données HSCAPP sur la page 290 pour en savoir plus sur la description de chaque valeur.
3. Affectez également une valeur aux autres variables MyAppData comme indiqué dans la figure suivante.
   - Dans le champ **Valeur initiale**, saisissez la valeur souhaitée.
   - Consultez Type de données HSCAPP sur la page 290 pour en savoir plus sur la description de chaque valeur.

Nom	Alias	Type de donnée	Dimension	Valeur Projet	Valeur initiale
MyAppData		HSCAPP		...	...
MyAppData.PlsEnable		BOOL			FALSE
MyAppData.HscID		UINT			0
MyAppData.HscMode		UINT			6
MyAppData.Accumulator		DINT			
MyAppData.HPSetting		DINT			40
MyAppData.LPSetting		DINT			-40
MyAppData.OFSetting		DINT			50
MyAppData.UFSetting		DINT			-50
MyAppData.OutputMask		UDINT			3
MyAppData.HPOutput		UDINT			1
MyAppData.LPOutput		UDINT			2

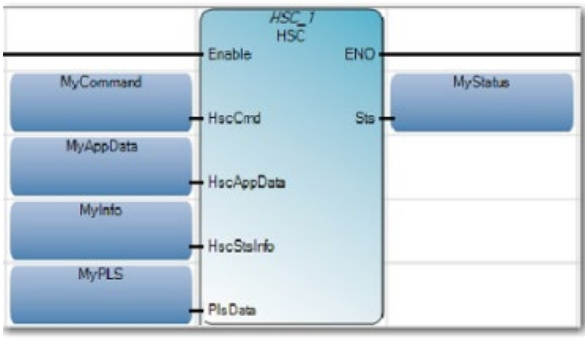
4. Définissez la valeur de la commande HSC :
   - Dans le champ **Valeur initiale** de la variable MyCommand, saisissez 1.
   - Consultez la rubrique Valeurs HSCCmd sur la page 290 pour plus d'informations sur les valeurs de commande.

Affectation de valeurs aux variables et construction du programme

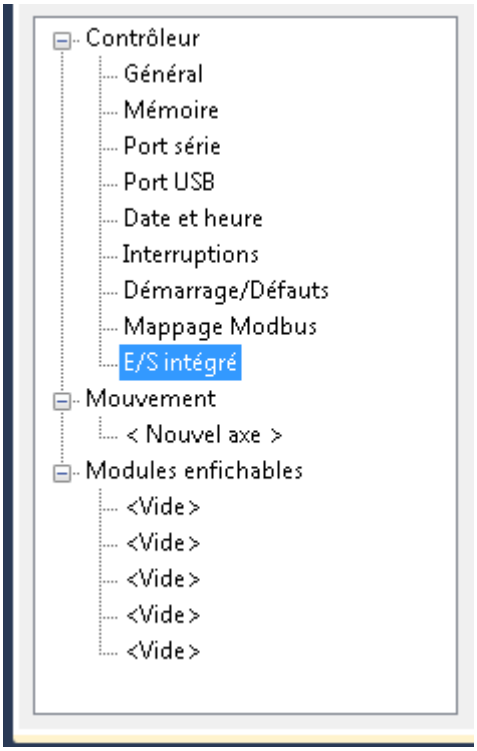
Après avoir saisi une valeur pour chaque variable HSC, observez les étapes suivantes pour affecter les variables au bloc fonctionnel, et construire le programme.

Pour affecter des variables et construire le programme, procédez comme suit :

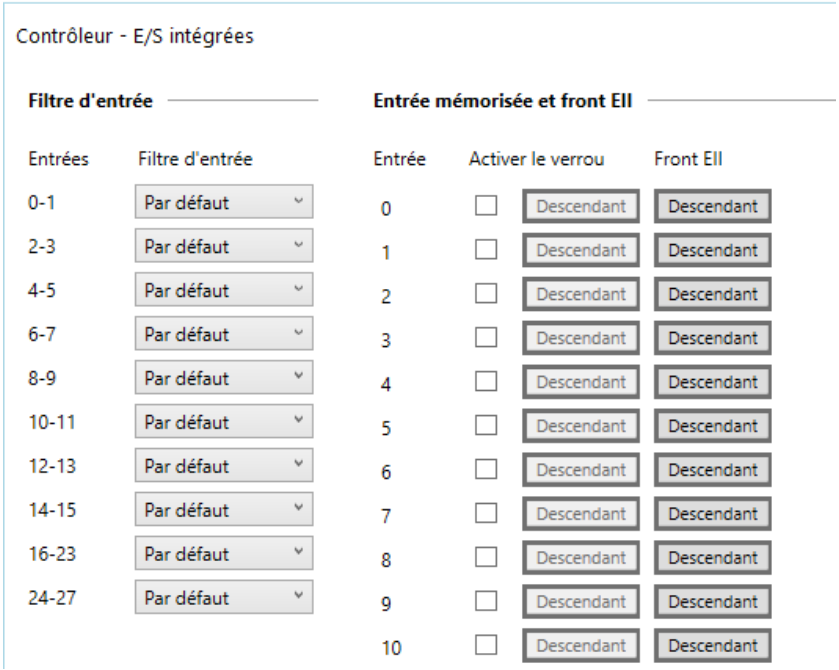
1. Dans l'Éditeur de schéma à relais, affectez chaque variable à l'élément de bloc de fonctions HSC comme indiqué.



2. Dans le **Gestionnaire de projet**, cliquez sur le contrôleur pour afficher l'**Arborescence du contrôleur**.



3. Dans l'**Arborescence du contrôleur**, cliquez sur **E/S intégrées** et sélectionnez des filtres d'entrée pour l'encodeur.



4. Vérifiez que l'encodeur est connecté au contrôleur Micro830.
5. Démarrez le contrôleur Micro830 et connectez-le à votre ordinateur.
6. Construisez le programme, puis chargez-le sur le contrôleur.

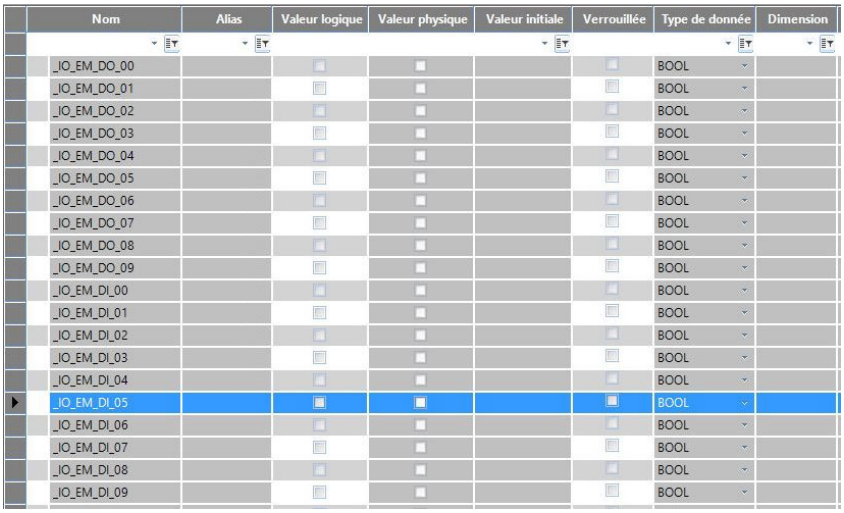
Test du programme et utilisation du compteur haute vitesse (HSC)

Après avoir chargé le programme HSC sur le contrôleur, vous pouvez le tester puis exécuter le compteur haute vitesse.

Pour tester le programme :

1. Connectez-vous au contrôleur.
2. Dans le **gestionnaire de projet**, double-cliquez sur le programme HSC, puis sur **Variables locales**.

   Vous pouvez consulter les valeurs des deux sorties HSC : STS (MyStatus) et HSCSTS (MyInfo).
3. Double-cliquez sur le contact normalement ouvert _IO_EM_DI_05 pour afficher la fenêtre **Sélecteur de variable**.
4. Cliquez sur l'onglet **E/S Micro830**, puis sur la ligne **_IO_EM_DI_05**.
5. Sélectionnez **Verrouiller** et **Valeur logique** pour forcer l'entrée sur la position **Activée**.



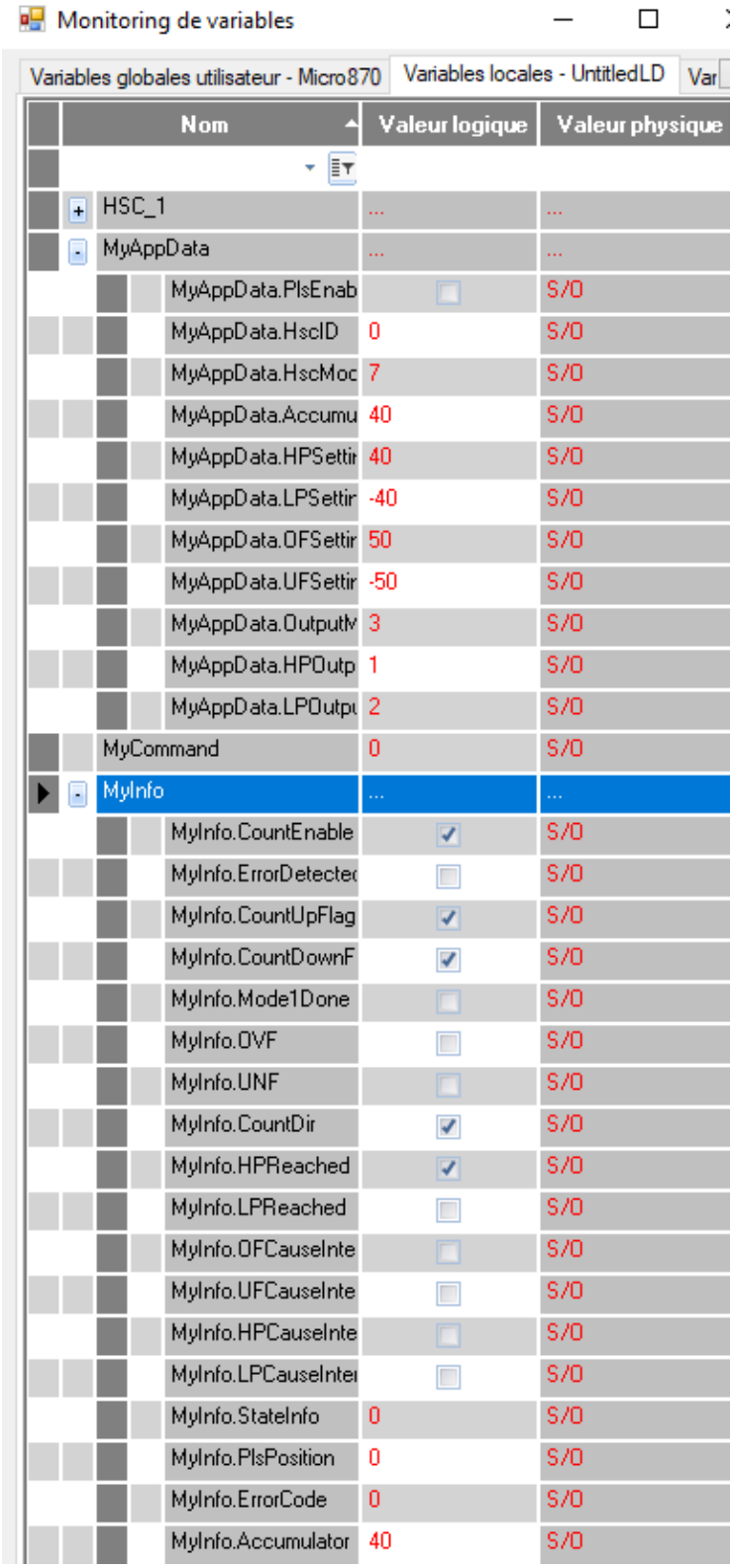
Nom	Alias	Valeur logique	Valeur physique	Valeur initiale	Verrouillée	Type de donnée	Dimension
_IO_EM_DO_00						BOOL	
_IO_EM_DO_01						BOOL	
_IO_EM_DO_02						BOOL	
_IO_EM_DO_03						BOOL	
_IO_EM_DO_04						BOOL	
_IO_EM_DO_05						BOOL	
_IO_EM_DO_06						BOOL	
_IO_EM_DO_07						BOOL	
_IO_EM_DO_08						BOOL	
_IO_EM_DO_09						BOOL	
_IO_EM_DI_00						BOOL	
_IO_EM_DI_01						BOOL	
_IO_EM_DI_02						BOOL	
_IO_EM_DI_03						BOOL	
_IO_EM_DI_04						BOOL	
_IO_EM_DI_05						BOOL	
_IO_EM_DI_06						BOOL	
_IO_EM_DI_07						BOOL	
_IO_EM_DI_08						BOOL	
_IO_EM_DI_09						BOOL	

6. Pour voir les résultats, cliquez sur l'onglet **Variables locales** pour afficher les modifications de variable.
7. Développez la liste des variables **MyAppData** et **MyInfo**.
8. Activez l'encodeur pour afficher le comptage croissant/décroissant du compteur. Par exemple, si l'encodeur est accouplé à un arbre de moteur, mettez en marche le moteur pour déclencher le décompte HSC.
9. Vérifiez que la Valeur logique de la variable MyStatus est définie sur 1, ce qui indique que le HSC est en fonctionnement.
10. Consultez la valeur de compteur dans MyInfo.Accumulator.

    Consultez la section HSC (compteur haute vitesse) sur la page 288 pour obtenir la liste complète des codes d'état.

Résultats

Dans cet exemple, une fois que MyInfo.Accumulator atteint une valeur haute prédéfinie de 40, la sortie 0 s'active, ainsi que l'indicateur HPReached. Si MyInfo.Accumulator atteint une valeur basse prédéfinie de -40, la sortie 1 s'active, ainsi que l'indicateur LPReached.



Add a Programmable Limit Switch (PLS) function

Cet exemple explique comment ajouter une fonction Interrupteur à came rotatif (PLS) au programme HSC.

Valeurs des variables pour les paramètres du compteur :

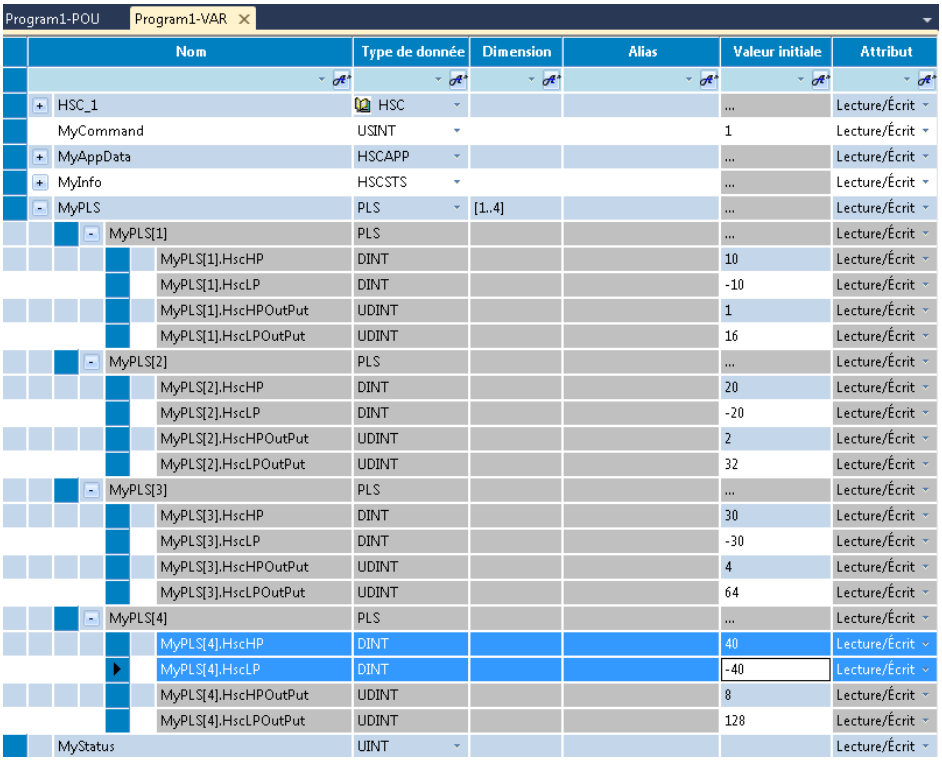
- **MyAppData.PlsEnable** est utilisé pour activer ou désactiver les paramètres PLS. Il doit être définit sur FAUX (désactivé) si la variable MyAppData est utilisée.
- **MyAppData.HscID** est utilisé pour spécifier quelles entrées intégrées sont utilisées en fonction du mode et du type d'application. consultez les rubriques Entrées HSC et Schéma de câblage pour connaître les différents ID pouvant être utilisés ainsi que les entrées intégrées et leurs caractéristiques.
- Si l'ID 0 est utilisé, l'ID 1 ne peut pas être utilisé sur le même contrôleur car les entrées sont utilisées par la Réinitialisation et Conservation.
- **MyAppData.HscMode** est utilisée pour indiquer le type d'opération utilisée par HSC pour compter. Voir Mode HSC (HSCAPP.HSCMode) sur la page 290.

Pour activer la fonction PLS, procédez comme suit :

1. Dans le **Gestionnaire de projet**, double-cliquez sur **Variables locales** pour afficher la page **Variables**.
2. Activez la fonction PLS :
   - Dans le champ Valeur initiale de la variable MyAppData.PlsEnable, sélectionnez Vrai.
3. Configurez les paramètres de débordement et de débordement négatif :
   - Dans le champ Valeur initiale de la variable MyAppData.OFSetting, saisissez 50.
   - Dans le champ Valeur initiale de la variable MyAppData.UFSetting, saisissez -50.
4. (facultatif) Configurez le masque de sortie dans le cas d'une sortie.

   Résultats pour cet exemple :
   - La variable PLS a une dimension de [1..4]. Cela signifie que HSC peut disposer de quatre paires de valeurs prédéfinies hautes et basses.
   - Les valeurs hautes prédéfinies doivent être inférieures à OFSetting et les valeurs basses prédéfinies être supérieures à UFSetting.

- Les valeurs HscHPOutPut et HscLPOutPut déterminent quelles sorties sont activées lorsque la valeur prédéfinie haute ou basse est atteinte.



Program1-POU | Program1-VAR

Nom	Type de donnée	Dimension	Alias	Valeur initiale	Attribut
HSC_1	HSC			...	Lecture/Écrit
MyCommand	USINT			1	Lecture/Écrit
MyAppData	HSCAPP			...	Lecture/Écrit
MyInfo	HSCSTS			...	Lecture/Écrit
MyPLS	PLS	[1..4]		...	Lecture/Écrit
MyPLS[1]	PLS			...	Lecture/Écrit
MyPLS[1].HscHP	DINT			10	Lecture/Écrit
MyPLS[1].HscLP	DINT			-10	Lecture/Écrit
MyPLS[1].HscHPOutPut	UDINT			1	Lecture/Écrit
MyPLS[1].HscLPOutPut	UDINT			16	Lecture/Écrit
MyPLS[2]	PLS			...	Lecture/Écrit
MyPLS[2].HscHP	DINT			20	Lecture/Écrit
MyPLS[2].HscLP	DINT			-20	Lecture/Écrit
MyPLS[2].HscHPOutPut	UDINT			2	Lecture/Écrit
MyPLS[2].HscLPOutPut	UDINT			32	Lecture/Écrit
MyPLS[3]	PLS			...	Lecture/Écrit
MyPLS[3].HscHP	DINT			30	Lecture/Écrit
MyPLS[3].HscLP	DINT			-30	Lecture/Écrit
MyPLS[3].HscHPOutPut	UDINT			4	Lecture/Écrit
MyPLS[3].HscLPOutPut	UDINT			64	Lecture/Écrit
MyPLS[4]	PLS			...	Lecture/Écrit
MyPLS[4].HscHP	DINT			40	Lecture/Écrit
MyPLS[4].HscLP	DINT			-40	Lecture/Écrit
MyPLS[4].HscHPOutPut	UDINT			8	Lecture/Écrit
MyPLS[4].HscLPOutPut	UDINT			128	Lecture/Écrit
MyStatus	UINT				Lecture/Écrit

Exemple : Commutateur limite programmable (PLS) activé

Cet exemple décrit les résultats obtenus lorsque la fonction PLS est activée à l'aide de valeurs de paramètre HSC sur la page 288 et PLSData particulières.

Valeur des paramètres HSC

Cet exemple suppose que les paramètres HSC sont définis sur les valeurs suivantes :

- HSCApp.OutputMask = 31
- HSCApp.HSCMode = 0
- Le compteur HSC contrôle les sorties intégrées 0 à 4 uniquement.

Valeur des paramètres PLSData

Cet exemple suppose que les paramètres PLSData pour la variable (HSC_PLS) sont configurés de la manière suivante.

Nom	Type de donnée	Dimension	Valeur initiale
HSC_1	HSC		...
HSC_PLS	PLS	[1..4]	...
HSC_PLS[1]	PLS		...
HSC_PLS[1].HscHP	DINT		250
HSC_PLS[1].HscLP	DINT		-2
HSC_PLS[1].HscHPOutPut	UDINT		3
HSC_PLS[1].HscLPOutPut	UDINT		0
HSC_PLS[2]	PLS		...
HSC_PLS[2].HscHP	DINT		500
HSC_PLS[2].HscLP	DINT		-2
HSC_PLS[2].HscHPOutPut	UDINT		7
HSC_PLS[2].HscLPOutPut	UDINT		0
HSC_PLS[3]	PLS		...
HSC_PLS[3].HscHP	DINT		750
HSC_PLS[3].HscLP	DINT		-2
HSC_PLS[3].HscHPOutPut	UDINT		15
HSC_PLS[3].HscLPOutPut	UDINT		0
HSC_PLS[4]	PLS		...
HSC_PLS[4].HscHP	DINT		1000
HSC_PLS[4].HscLP	DINT		-2
HSC_PLS[4].HscHPOutPut	UDINT		31
HSC_PLS[4].HscLPOutPut	UDINT		0

Résultats avec PLS

Dans cet exemple, les événements suivants se produisent.

- Lorsque la logique à relais est exécutée pour la première fois : HSCSTS.Accumulator = 1, ce qui signifie que toutes les sorties sont désactivées.
- Lorsque HSCSTS.Accumulator = 250, HSC_PLS[1].HSCHPOutput est envoyé via HSCAPP.OutputMask et alimente les sorties 0 et 1.
- L'envoi de la sortie de valeur haute prédéfinie via le masque de sortie se répète lorsque HSCSTS.Accumulator atteint 500, 750 et 1 000, et le contrôleur alimente les sorties 0 à 2, 0 à 3 et 0 à 4, respectivement.
- Une fois l'opération terminée, le cycle est réinitialisé et se répète à partir de HSCSTS.HP = 250.
- Lorsque le cycle complet est terminé et que la valeur HSCSTS.HP est atteinte, le paramètre PLSPositon est réinitialisé.

Instructions HSCE

Utilisez les instructions HSCE pour surveiller et contrôler le comptage rapide.

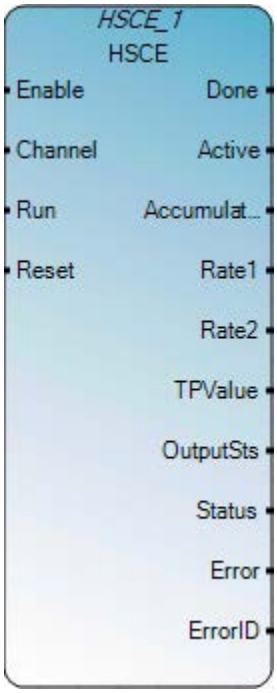
Instruction	Description
HSCE sur la page 325	HSCE démarre, arrête et lit la valeur de l'accumulateur.
HSCE_CFG sur la page 328	HSCE_CFG est la configuration de comptage rapide.
HSCE_CFG_PLS sur la page 330	HSCE_CFG_PLS est la configuration PLS de comptage rapide.
HSCE_READ_STS sur la page 332	HSCE_READ_STS lit l'état du comptage rapide.
HSCE_SET_STS sur la page 325	HSCE_SET_STS définit/réinitialise manuellement l'état de comptage rapide.

HSCE

HSCE est utilisé pour le contrôle et la lecture du compteur HSC.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.

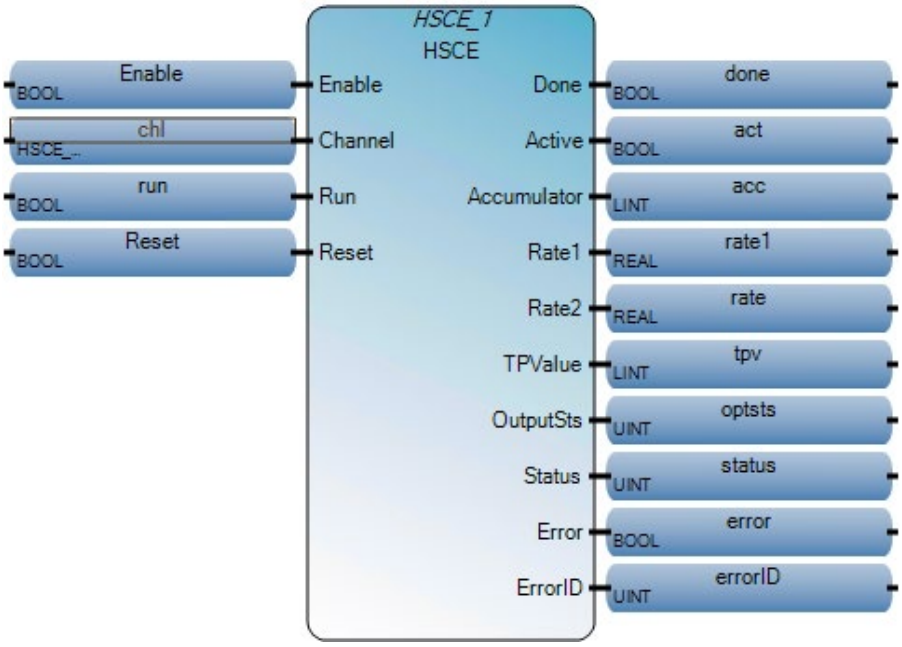


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

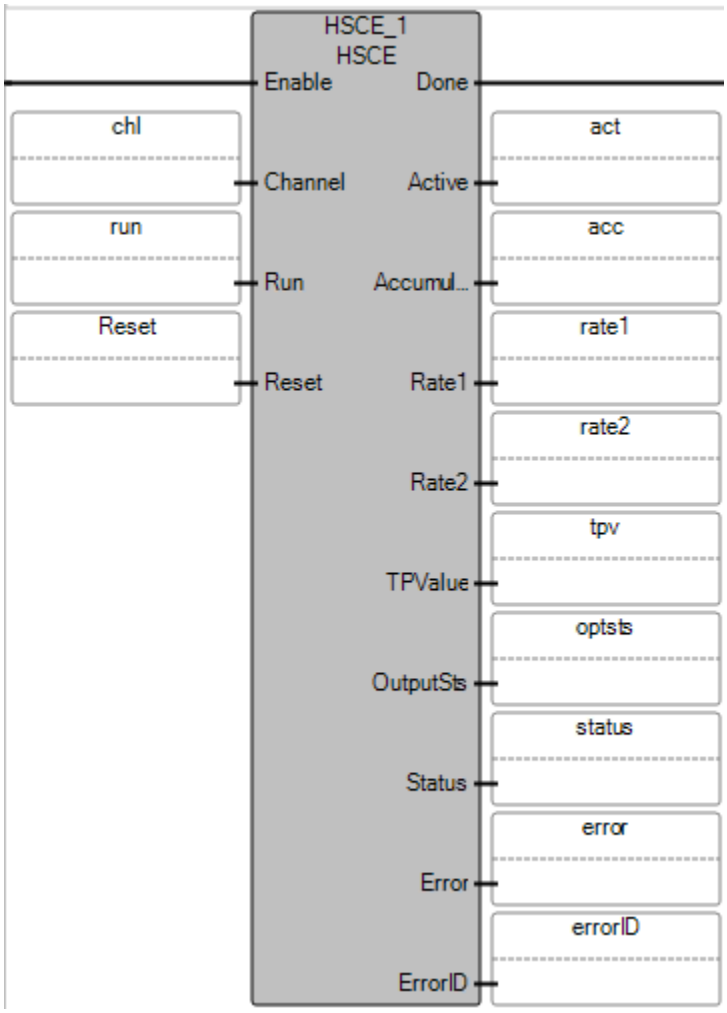
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	VRAI : le HSCE amorce le bloc de fonctions et le module enfichable HSC. Accumulateur = InitAcc. L'état du HSCE sera réinitialisé. Rate1 = 0, Rate2 = 0.

Paramètre	Type de paramètre	Type de données	Description
			FAUX : l'accumulateur est mis à jour par lecture du module enfichable. Done, Active, Error = FAUX quand ErrorID = 0, Rate1, Rate2 = 0.
Channel	Entrée	HSCE_CHANNEL	Le canal HSCE.
Run	Entrée	BOOL	Pour que le HSCE compte l'état opérationnel. VRAI : le HSCE compte les impulsions. FAUX : le HSCE arrête de compter.
Reset	Entrée	BOOL	Vrai : toutes les sorties et l'accumulateur sont effacés. Les valeurs Rate1 et Rate2 sont également redéfinies sur 0. Effacez l'état du module enfichable HSC. La priorité de l'entrée de réinitialisation est supérieure à celle de l'entrée d'exécution.
Done	Sortie	BOOL	Vrai : lorsque HSCE Enable est défini sur Vrai et qu'aucune erreur n'est détectée. Faux : lorsque HSCE Enable est défini sur Vrai, mais que Run est défini sur Faux.
Active	Sortie	BOOL	Vrai : lorsque HSCE Enable est défini sur Vrai, que Done est défini sur Vrai et que Run est défini sur Faux. Faux : lorsque HSCE Enable est défini sur Vrai, mais que Run est défini sur Faux.
Rate1	Sortie	REAL	Le taux d'impulsion actuel en unité utilisateur par seconde (méthode par impulsion).
Rate2	Sortie	REAL	Le taux d'impulsion actuel en unité utilisateur par seconde (méthode cyclique).
TPValue	Sortie	LINT	La valeur de l'accumulateur est capturée lorsque la sonde tactile est déclenchée.
OutputSts	Sortie	UINT	L'état de sortie physique et virtuelle du module enfichable, uniquement applicable pour le compteur HSC 0. Bit 0 : sortie 0 (état de la sortie physique) Bit 1 à 15 : sortie 1 à sortie 15 (état de la sortie virtuelle)
Status	Sortie	UINT	Informations sur l'état HSC.
Error	Sortie	BOOL	Indique une erreur.
ErrorID	Sortie	UINT	Lorsqu'une erreur se produit, ErrorID contient le code d'erreur.

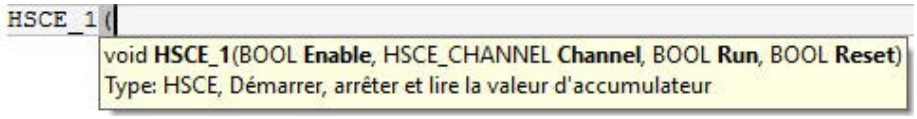
Exemple de diagramme de bloc de fonctions HSCE



Exemple de langage à relais HSCE



Exemple de structure de texte HSCE



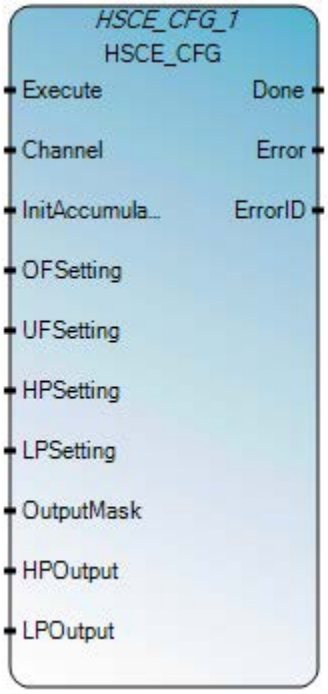

```
1  HSCE_1(Enable,chl,Run,Reset);
2  Done_HSCE := HSCE_1.Done;
3  Active_HSCE := HSCE_1.Active;
4  Accumulator_HSCE := HSCE_1.Accumulator;
5  Rate1_HSCE := HSCE_1.Rate1;
6  Rate2_HSCE := HSCE_1.Rate2;
7  TPValue_HSCE := HSCE_1.TPValue;
8  Outputsts_HSCE := HSCE_1.OutputSts;
9  Status_HSCE := HSCE_1.Status;
10 Error_HSCE := HSCE_1.Error;
11 ErrorID_HSCE := HSCE_1.ErrorID;
```

HSCE_CFG

HSCE_CFG est utilisé pour configurer le comptage rapide.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

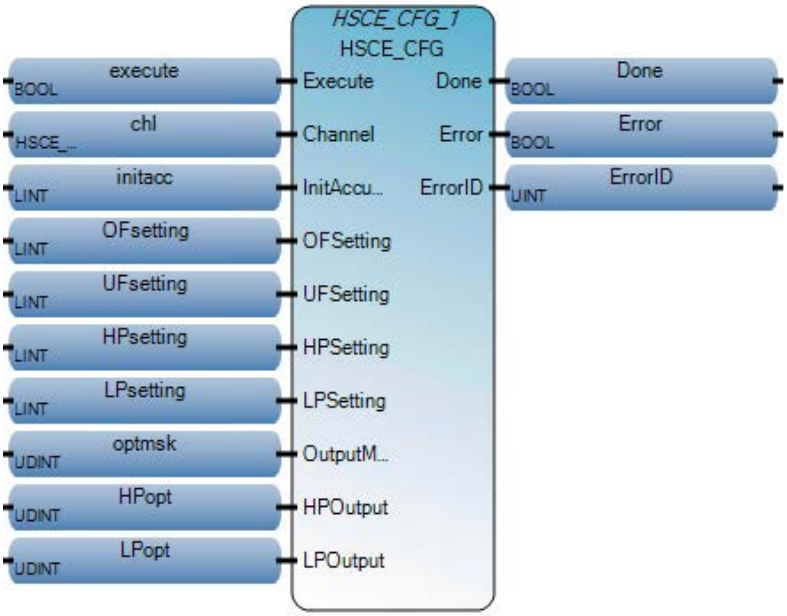
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



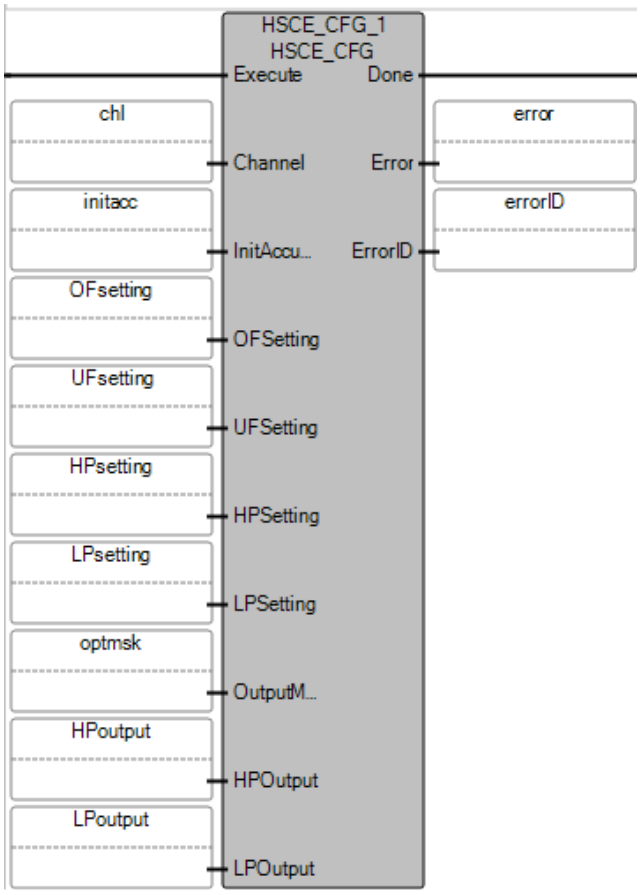
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Le front montant initie la configuration HSC. (HSCE Enable doit être égal à FAUX) Le front descendant efface la valeur de sortie.
Channel	Entrée	HSCE_CHANNEL	Le canal HSCE.
InitAccumulator	Entrée	LINT	Valeur initiale de l'accumulateur.
OFSetting	Entrée	LINT	Valeur limite de débordement du compteur.
UFSetting	Entrée	LINT	Valeur limite de soupassement du compteur.
HPSetting	Entrée	LINT	Valeur élevée prédéfinie de HSCE.
LPSetting	Entrée	LINT	Valeur basse prédéfinie de HSCE.
PLS_Offset	Entrée	USINT	Décalage au démarrage dans le tableau de données PLS.
OutputMask	Entrée	USINT	Masque de sortie pour fonctionnalité PLS.
HPOutput	Entrée	UDINT	État des sorties élevées prédéfinies.
LPOutput	Entrée	UDINT	État des sorties basses prédéfinies.
Done	Sortie	BOOL	Réussite de l'action de configuration HSC (initiée par cette instruction).
Error	Sortie	BOOL	Indique une erreur.
ErrorID	Sortie	UINT	Lorsqu'une erreur se produit, ErrorID contient le code d'erreur.

Exemple de diagramme de bloc de fonctions HSCE_CGF



Exemple de langage à relais HSCE_CGF



Exemple de structure de texte HSCE_CGF

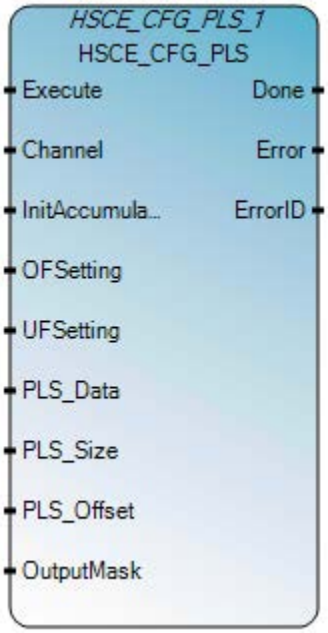
```
HSCE_CFG_1
void HSCE_CFG_1(BOOL Execute, HSCE_CHANNEL Channel, LINT InitAccumulator, LINT OFSetting, LINT UFSetting, LINT HPSetting, LINT LPSetting, UDINT OutputMask, UDINT HPOutput, UDINT LPOutput)
Type: HSCE_CFG, Configuration du comptage rapide

1  HSCE_CFG_1(execute,chl,initacc,OFsetting,UFsetting,HPsetting,LPsetting,optmsk,HPopt,LPopt);
2  Done_HSCE_CFG :=HSCE_CFG_1.Done;
3  Error_HSCE_CFG :=HSCE_CFG_1.Error;
4  ErrorID_HSCE_CFG := HSCE_CFG_1.ErrorID;
```

HSCE_CFG_PLS

Cette instruction est utilisée pour la configuration HSC avec interrupteur à came rotatif (PLS). Cette fonction est un ensemble supplémentaire de modes de fonctionnement pour le comptage rapide. Lors du fonctionnement dans ces modes, les valeurs des données prédéfinies et de sortie sont mises à jour à l'aide des données fournies par l'utilisateur à chaque fois que l'une des valeurs prédéfinies est atteinte. Ces modules sont programmés en fournissant un fichier PLS qui contient les ensembles de données à utiliser.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.

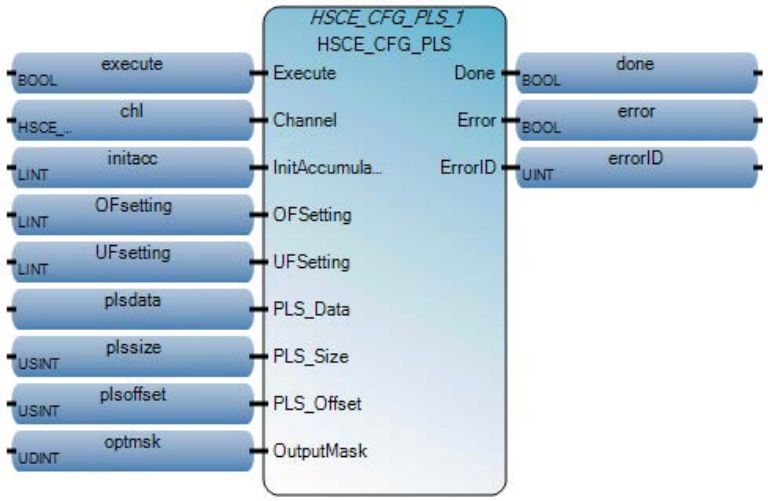


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

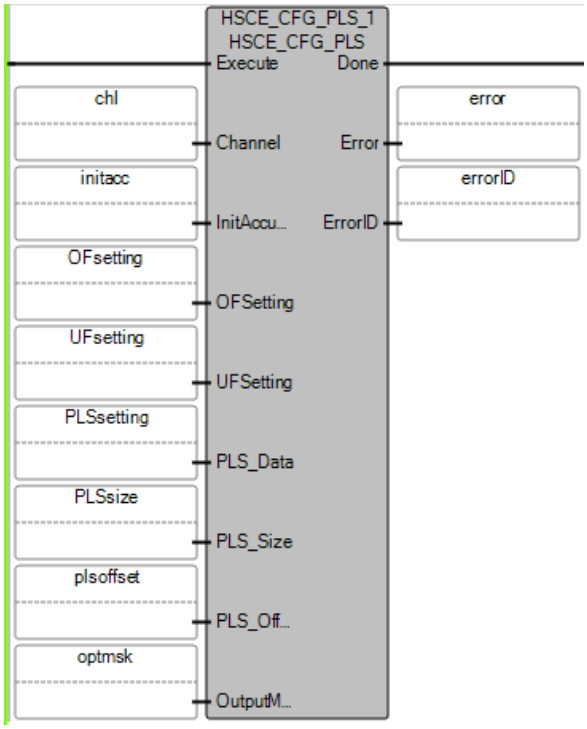
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Le front montant initie la configuration HSC. (HSCE Enable doit être égal à FAUX) Le front descendant efface la valeur de sortie.
Channel	Entrée	HSCE_CHANNEL	Le canal HSCE.
InitAccumulator	Entrée	LINT	Valeur initiale de l'accumulateur.
OFSetting	Entrée	LINT	Valeur limite de débordement du compteur.
UFSetting	Entrée	LINT	Valeur limite de soupassement du compteur.
PLS_Data	Entrée	PLS2	Tableau de PLS (PLS_54)

PLS_Size	Entrée	USINT	Taille des données PLS, la valeur maximale est 24 pour le module enfichable.
PLS_Offset	Entrée	USINT	Décalage au démarrage dans le tableau de données PLS.
OutputMask	Entrée	USINT	Masque de sortie pour fonctionnalité PLS.
Done	Sortie	BOOL	Réussite de l'action de configuration HSC (initiée par cette instruction).
Error	Sortie	BOOL	Indique une erreur.
ErrorID	Sortie	UINT	Lorsqu'une erreur se produit, ErrorID contient le code d'erreur.

Exemple de diagramme de bloc de fonctions HSCE_CGF_PLS



Exemple de langage à relais HSCE_CGF_PLS



Exemple de structure de texte HSCE_CGF_PLS

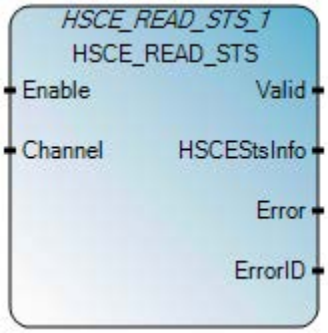
```
HSCE_CFG_PLS_1(Execute,chl,initacc,OFsetting,UFsetting,PLS_data,PLS_Size,PLS_Offset,optmsk);
Done_HSCE_CFG_PLS :=HSCE_CFG_PLS_1.Done;
Error_HSCE_CFG_PLS :=HSCE_CFG_PLS_1.Error;
ErrorID_HSCE_CFG_PLS := HSCE_CFG_PLS_1.ErrorID;
```

HSCE_READ_STS

L'instruction est utilisée pour lire l'état HSC actuel.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

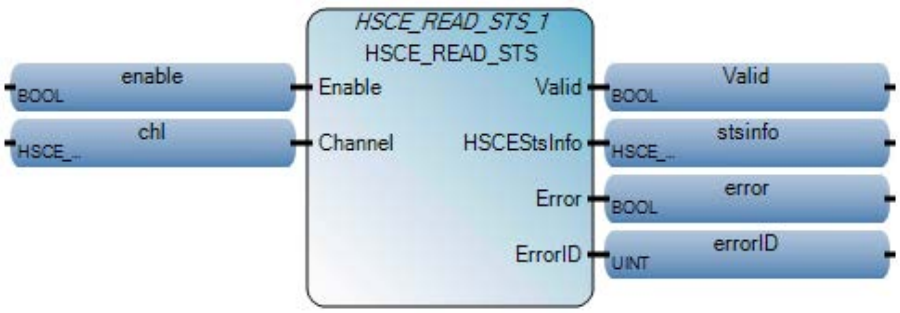
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



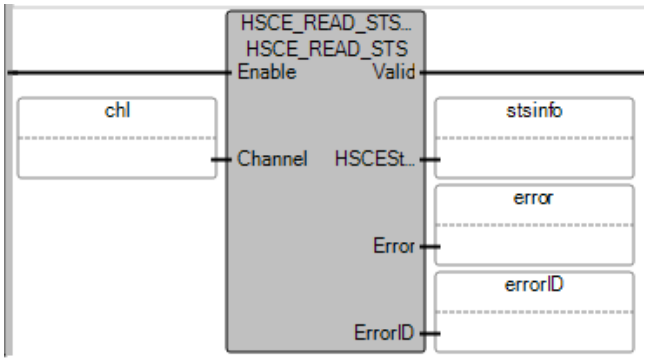
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Si Enable est Vrai, HSC2StsInfo est mis à jour.
Channel	Entrée	HSCE_CHANNEL	Le canal HSCE.
Valid	Sortie	BOOL	HSC2StsInfo est valide s'il est VRAI.
HSCEStsInfo	Entrée	LINT	Valeur limite de débordement du compteur.
Error	Sortie	BOOL	Indique une erreur.
ErrorID	Sortie	UINT	Lorsqu'une erreur se produit, ErrorID contient le code d'erreur.

Exemple de diagramme de bloc de fonctions HSCE_READ_STS



Exemple de langage à relais HSCE_READ_STS



Exemple de structure de texte HSCE_READ_STS

```
HSCE_READ_STS_1(
```

void **HSCE_READ_STS_1**(BOOL **Enable**, HSCE_CHANNEL **Channel**)
Type: HSCE_READ_STS, Lire le statut du comptage rapide

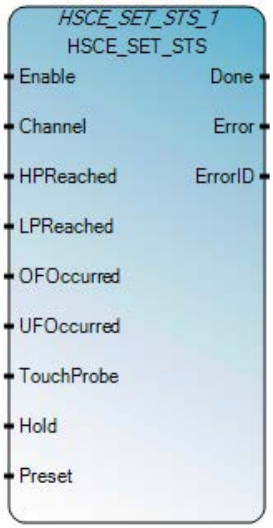
```
1  HSCE_READ_STS_1(Enable,Channel);
2  Valid_HSCE_READ_STS :=HSCE_READ_STS_1.Valid;
3  HSCEstsInfo_HSCE_READ_STS := HSCE_READ_STS_1.HSCEStsInfo;
4  Error_HSCE_READ_STS := HSCE_READ_STS_1.Error;
5  ErrorID HSCE READ STS := HSCE READ STS 1.ErrorID;
```

HSCE_SET_STS

L'instruction définit et réinitialise manuellement les indicateurs d'état de décompte HSC. Le bloc de fonctions HSC doit être arrêté pour que le bloc de fonctions HSCE_SET_STC puisse définir ou réinitialiser son état HTS. Si le bloc de fonctions HSC n'est pas arrêté ou si le canal HSC n'est pas configuré, le bloc de fonctions HSC2_SET_STS génère une erreur.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

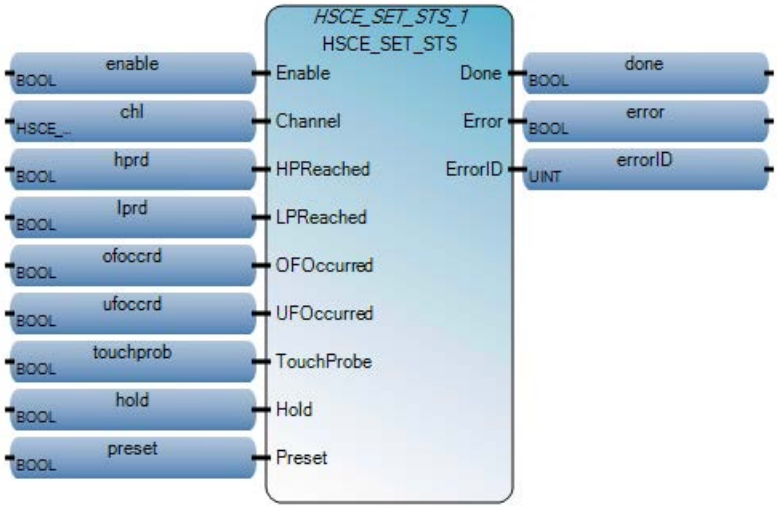
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



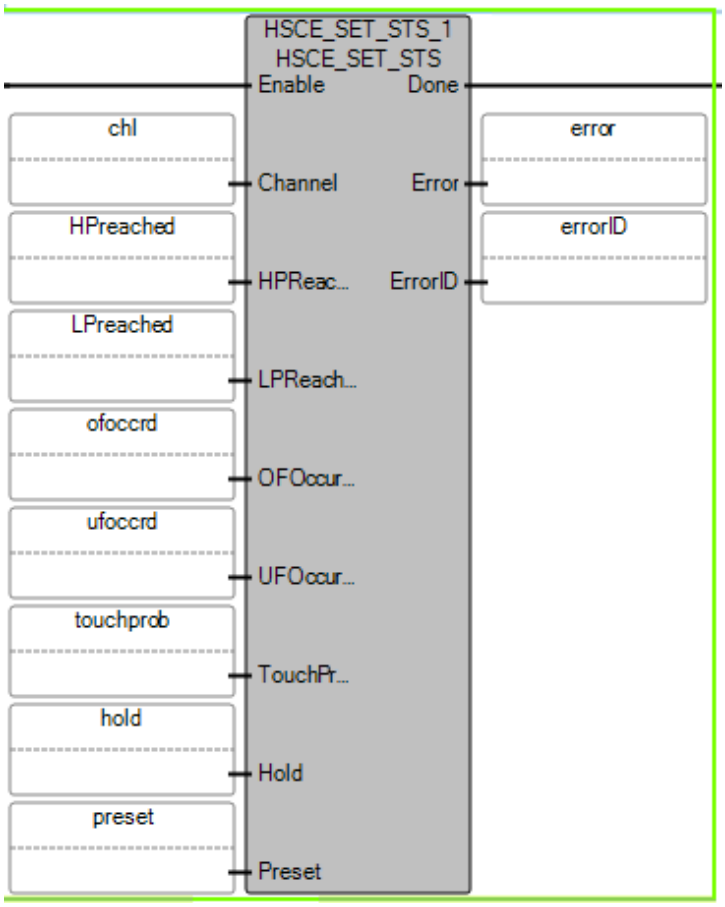
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc de fonctions. VRAI : définition/réinitialisation de l'état HSC. FAUX : l'état HSC ne change pas.
Channel	Entrée	HSCE_CHANNEL	Le canal HSCE.
HPReached	Sortie	BOOL	Lorsque la valeur élevée prédéfinie est atteinte, ce bit est défini sur VRAI par le module enfichable. Ce bit est défini ou redéfini par le bloc de fonctions HSCE_SET_STS.
LPReached	Entrée	BOOL	Lorsque la valeur basse prédéfinie est atteinte, ce bit est défini sur VRAI par le module enfichable. Ce bit est défini ou redéfini par le bloc de fonctions HSCE_SET_STS.
OFOccurred	Entrée	BOOL	Lorsque le débordement se produit, ce bit est défini sur VRAI par le module enfichable. Ce bit est défini ou redéfini par le bloc de fonctions HSCE_SET_STS.
UFOccurred	Entrée	BOOL	Lorsque le soupassement se produit, ce bit est défini sur VRAI par le module enfichable. Ce bit est défini ou redéfini par le bloc de fonctions HSCE_SET_STS.
TouchProbe	Entrée	BOOL	Lorsque la sonde tactile est déclenchée, ce bit est défini sur VRAI par le module enfichable.
Hold	Entrée	BOOL	Lorsque la conservation HSC est déclenchée, ce bit est défini sur VRAI par le module enfichable. Ce bit est défini ou redéfini par le bloc de fonctions HSCE_SET_STS. Ce paramètre d'entrée est uniquement efficace pour le compteur 0. Le compteur 1 est toujours réinitialisé.
Preset	Entrée	BOOL	Lorsque la réinitialisation Z ACC est déclenchée, ce bit est défini sur VRAI par le module enfichable. Ce bit est défini ou redéfini par le bloc de fonctions HSCE_SET_STS. Ce paramètre d'entrée est uniquement efficace pour le compteur 0. Le compteur 1 est toujours réinitialisé.
Done	Sortie	BOOL	Réussite de l'action de configuration HSC (initiée par cette instruction).
Error	Sortie	BOOL	Indique une erreur.
ErrorID	Sortie	UINT	Lorsqu'une erreur se produit, ErrorID contient le code d'erreur.

Exemple de diagramme de bloc de fonctions HSCE_SET_STS



Exemple de langage à relais HSCE_SET_STS



Exemple de structure de texte HSCE_SET_STS

```
HSCE_SET_STS_1
void HSCE_SET_STS_1(BOOL Enable, HSCE_CHANNEL Channel, BOOL HPReached, BOOL LPReached, BOOL OFOccurred, BOOL UFOccurred, BOOL TouchProbe, BOOL Hold, BOOL Preset)
Type: HSCE_SET_STS, Définir manuellement/réinitialiser le statut du comptage rapide

1  HSCE_SET_STS_1(Enable, chl, hprd, ofocccrd, ufoccrd, touchprob, hold, preset);
2  Done_HSCE_SET_STS := HSCE_SET_STS_1.Done;
3  Error_HSCE_SET_STS := HSCE_SET_STS_1.Error;
4  ErrorID_HSCE_SET_STS := HSCE_SET_STS_1.ErrorID;
```

Codes d'erreur HSCE

Le tableau suivant décrit les codes d'erreur d'état pour les instructions HSCE sur la page 325 :

Code ErrorID	Description de l'erreur	Action corrective
0	L'instruction a correctement terminé l'opération.	
1	Fichiers de configuration HSC non valides.	Contactez le représentant local du support technique Rockwell Automation. Pour obtenir des informations sur les coordonnées, consultez le site suivant : http://www.rockwellautomation.com/support
2	Type de module HSC non valide.	Corrigez le type de module. Par exemple, sélectionnez « enfichable » comme type de module.
3	ID d'emplacement HSC non valide.	Corrigez l'ID d'emplacement dans le canal d'entrée du bloc de fonctions.
4	ID HSC non valide.	Corrigez l'ID HSC. Par exemple, définissez 0 pour le compteur 0.

Code ErrorID	Description de l'erreur	Action corrective
5	Mode non valide pour le canal.	Contactez le représentant local du support technique Rockwell Automation.
6	Taille PLS non valide.	La taille PLS devrait être inférieure ou égale à 24, et la taille du tableau de données PLS devrait être supérieure ou égale à la taille PLS.
7	Décalage PLS non valide.	Le décalage PLS et la taille PLS devraient être compris dans la taille du tableau de données PLS.
8	Valeur InitAccumulator non valide.	Corrigez la valeur InitAccumulator. Elle pourrait dépasser la limite ou sortir de la plage de la limite basse prédéfinie (LP) ou haute prédéfinie (HP) (LP ≤ InitACC ≤ HP).
9	LP non valide.	Corrigez la valeur LP. Elle pourrait dépasser la limite.
10	OF non valide.	Corrigez la valeur OF.
11	UF non valide.	Corrigez la valeur UF.
12	HP non valide.	Corrigez la valeur HP.
13	Aucune configuration pour HSCE.	Contactez le représentant local du support technique Rockwell Automation.
14	État HSCE non valide.	Vérifiez le bloc de fonctions HSCE pour contrôler l'état de ce bloc de fonctions.
15	Module enfichable non valide	Vérifiez le module enfichable pour vous assurer qu'il s'agit d'un module HSC.
16	HSCE est en cours d'exécution.	Lorsque HSC effectue un comptage, aucun état de configuration et de paramètre n'est autorisé. L'exécution de HSCE_CFG sur la page 328 et HSCE_CFG_PLS sur la page 330 a lieu pendant celle de HSCE.
17	Erreur d'arrêt du module enfichable HSC.	Contactez le représentant local du support technique Rockwell Automation.
18	Erreur de mise à jour de l'écriture du débit.	Contactez le représentant local du support technique Rockwell Automation.
19	Erreur d'écriture du nombre d'impulsions.	Contactez le représentant local du support technique Rockwell Automation.
20	Erreur d'écriture de l'ACC d'application.	Contactez le représentant local du support technique Rockwell Automation.
21	Erreur d'écriture de l'ACC.	Contactez le représentant local du support technique Rockwell Automation.
22	Erreur d'écriture de l'application.	Contactez le représentant local du support technique Rockwell Automation.
23	Erreur d'écriture du nombre de PLS.	Contactez le représentant local du support technique Rockwell Automation.
24	Erreur d'écriture de soupassement.	Contactez le représentant local du support technique Rockwell Automation.
25	Erreur d'écriture de débordement.	Contactez le représentant local du support technique Rockwell Automation.
26	Erreur d'écriture de la valeur basse prédéfinie.	Contactez le représentant local du support technique Rockwell Automation.
27	Erreur d'écriture de la valeur haute prédéfinie.	Contactez le représentant local du support technique Rockwell Automation.
28	Erreur d'écriture de la valeur basse prédéfinie externe.	Contactez le représentant local du support technique Rockwell Automation.
29	Erreur d'écriture de la valeur haute prédéfinie externe.	Contactez le représentant local du support technique Rockwell Automation.
30	Erreur d'écriture du masque externe.	Contactez le représentant local du support technique Rockwell Automation.
31	Erreur d'écriture de la valeur basse prédéfinie externe PLS.	Contactez le représentant local du support technique Rockwell Automation.
32	Erreur d'écriture de la valeur haute prédéfinie externe PLS.	Contactez le représentant local du support technique Rockwell Automation.
33	Erreur d'écriture de la valeur basse prédéfinie PLS.	Contactez le représentant local du support technique Rockwell Automation.
34	Erreur d'écriture de la valeur haute prédéfinie externe PLS.	Contactez le représentant local du support technique Rockwell Automation.
35	Erreur d'écriture du décalage PLS.	Contactez le représentant local du support technique Rockwell Automation.
36	Erreur d'écriture du nombre PLS.	Contactez le représentant local du support technique Rockwell Automation.
37	Erreur de lecture de l'état.	Contactez le représentant local du support technique Rockwell Automation.
38	Erreur de lecture de la valeur haute prédéfinie.	Contactez le représentant local du support technique Rockwell Automation.
39	Erreur de lecture de la valeur basse prédéfinie.	Contactez le représentant local du support technique Rockwell Automation.
40	Erreur de lecture de l'ACC.	Contactez le représentant local du support technique Rockwell Automation.
41	Erreur de lecture du nombre de la largeur d'impulsion.	Contactez le représentant local du support technique Rockwell Automation.
42	Erreur de lecture de la largeur d'impulsion.	Contactez le représentant local du support technique Rockwell Automation.

Code ErrorID	Description de l'erreur	Action corrective
43	Erreur de lecture du nombre d'impulsions.	Contactez le représentant local du support technique Rockwell Automation.
44	Erreur de lecture du débit de mise à jour.	Contactez le représentant local du support technique Rockwell Automation.
45	Erreur d'écriture de l'état.	Contactez le représentant local du support technique Rockwell Automation.
46	Erreur de lecture de la valeur basse prédéfinie externe.	Contactez le représentant local du support technique Rockwell Automation.
47	Erreur de lecture de la valeur haute prédéfinie externe.	Contactez le représentant local du support technique Rockwell Automation.
48	Erreur de lecture du nombre PLS.	Contactez le représentant local du support technique Rockwell Automation.
49	S.O.	Contactez le représentant local du support technique Rockwell Automation.
50	Erreur d'écriture de l'état d'application.	Contactez le représentant local du support technique Rockwell Automation.
51	Canal d'entrée non valide.	Contactez le représentant local du support technique Rockwell Automation.
52	Erreur de lecture de la sonde tactile.	Contactez le représentant local du support technique Rockwell Automation.
53	Erreur d'écriture de l'ACC de réinitialisation.	Contactez le représentant local du support technique Rockwell Automation.
54	Erreur de démarrage HSC.	Contactez le représentant local du support technique Rockwell Automation.
55	Le compteur est désactivé.	Le compteur HSC est désactivé. Vérifiez la configuration du canal pour vous assurer que le compteur HSC est activé.
56	Valeur de masque de sortie non valide.	Vérifiez si le masque de sortie est compris dans la plage valide. Pour un module enfichable HSC, la plage est comprise entre 0 et 65535.
57	Sortie de valeur haute prédéfinie non valide.	Vérifiez si la sortie de la valeur haute prédéfinie est comprise dans la plage valide. Pour un module enfichable HSC, la plage est comprise entre 0 et 65535.
58	Sortie de valeur basse prédéfinie non valide.	Vérifiez si la sortie de la valeur basse prédéfinie est comprise dans la plage valide. Pour un module enfichable HSC, la plage est comprise entre 0 et 65535.
59	Version UPM non prise en charge.	Vérifiez la configuration de la version du module enfichable HSC.
60	Aucun module HSC ajouté.	Vérifiez si le module enfichable actuel est un module HSC.
61	Aucun module enfichable HSCE n'est configuré.	Contactez le représentant local du support technique Rockwell Automation.
62	Une erreur d'écriture de module UPM survient lors du passage du module enfichable HSC en mode « Ne pas exécuter ».	Contactez le représentant local du support technique Rockwell Automation.
63	Une erreur d'écriture de module UPM survient lors du passage du module enfichable HSC en mode « Exécuter ».	Contactez le représentant local du support technique Rockwell Automation.
64	Une erreur de lecture de module UPM survient lors de la lecture de l'état de sortie.	Contactez le représentant local du support technique Rockwell Automation.

Instructions d'entrée/sortie

Utilisez les instructions d'entrée/de sortie pour lire ou écrire des données sur ou depuis un contrôleur ou un module, en utilisant les signaux envoyés à un appareil physiquement connecté à un contrôleur logique programmable. Les relais d'entrée transfèrent les signaux aux relais internes, et les relais de sortie les transfèrent aux appareils de sortie externes.

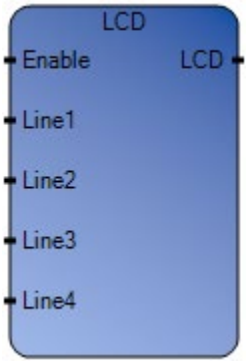
Instruction	Description
LCD sur la page 340	Micro810 uniquement. Affiche une chaîne ou un nombre sur un écran LCD.
LCD_BKLT_REM sur la page 342	Définit les paramètres de rétro-éclairage du LCD à distance dans un programme utilisateur.
LCD_REM sur la page 343	Affiche les messages définis par l'utilisateur pour le LCD à distance.
RHC sur la page 346	Lit une valeur de l'horloge haute vitesse dans le contrôleur Micro800.
RPC sur la page 348	Lit la somme de contrôle du programme utilisateur à partir du contrôleur ou du module mémoire.
DLG sur la page 349	Écrit des valeurs de variables depuis le moteur d'exécution dans un fichier de journalisation des données sur une carte SD.
IIM sur la page 350	Met à jour les entrées avant la scrutation de sortie normale.
IOM sur la page 352	Met à jour les sorties avant la scrutation de sortie normale.
KEY_READ sur la page 355	Micro810 uniquement. Lit l'état des touches sur le module LCD facultatif lorsque l'affichage utilisateur est actif.
KEY_READ_REM sur la page 357	Micro820 uniquement. Lit l'état des touches sur le module LCD à distance facultatif lorsque l'affichage utilisateur est actif.
MM_INFO sur la page 359	Lit les informations d'en-tête du module mémoire.
MODULE_INFO sur la page 362	Lit les informations du module à partir d'un module enfichable ou d'extension, à l'exception du module mémoire 2080-MEMBAK-RTC.
PLUGIN_INFO sur la page 373	Lit les informations du module à partir d'un module enfichable ou d'extension générique, à l'exception du module mémoire 2080-MEMBAK-RTC.
PLUGIN_READ sur la page 375	Lit les données à partir d'un module enfichable générique, à l'exception du module mémoire 2080-MEMBAK-RTC.
PLUGIN_RESET sur la page 377	Réinitialise le matériel pour un module enfichable générique, à l'exception du module mémoire 2080-MEMBAK-RTC.
PLUGIN_WRITE sur la page 379	Écrit des données dans un module enfichable générique, à l'exception du module mémoire 2080-MEMBAK-RTC.
RCP sur la page 381	Lit et écrit les données de recette depuis et vers une carte mémoire SD.
RTC_READ sur la page 383	Lit les informations du module d'horloge en temps réel (RTC).
RTC_SET sur la page 385	Définit les données d'horloge en temps réel pour les informations du module RTC.
SYS_INFO sur la page 387	Lit le bloc de données d'état pour le contrôleur Micro800.
TRIMPOT_READ sur la page 390	Lit la valeur de potentiomètre d'ajustement à partir d'un potentiomètre d'ajustement spécifique.

LCD

Affiche une chaîne ou un nombre sur un écran LCD facultatif.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

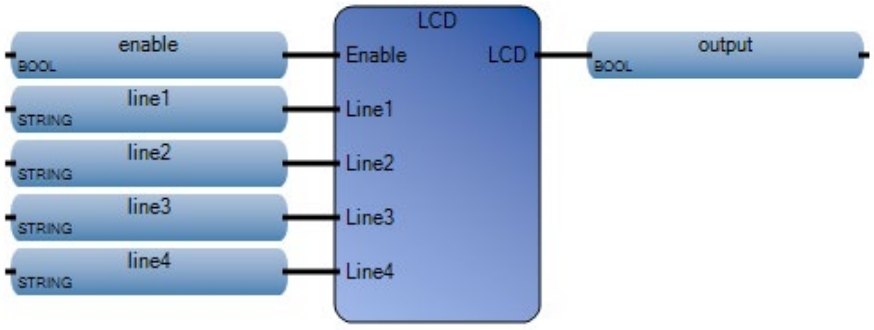
Cette instruction s'applique aux contrôleurs Micro810.



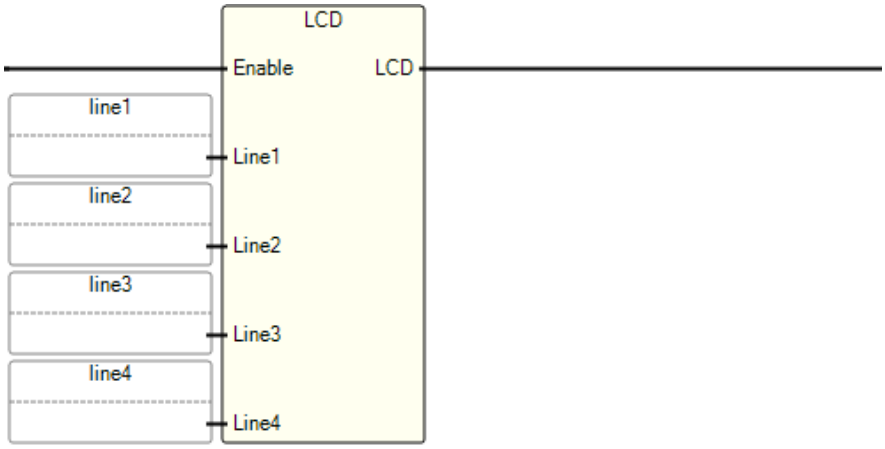
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : le module LCD bascule sur l'écran défini par l'utilisateur (des chaînes s'affichent sur l'écran LCD) au lieu de l'écran d'état des E/S. FAUX : le module LCD affiche le contenu de l'écran d'état des E/S.
Line1	Entrée	STRING	Chaîne affichée sur la ligne 1 du module LCD.
Line2	Entrée	STRING	Chaîne affichée sur la ligne 2 du module LCD.
Line3	Entrée	STRING	Chaîne affichée sur la ligne 3 du module LCD.
Line4	Entrée	STRING	Chaîne affichée sur la ligne 4 du module LCD.
LCD	Sortie	BOOL	VRAI : fonction activée.

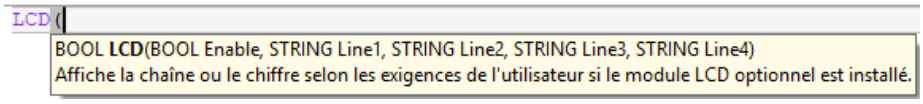
Exemple de Diagramme de Bloc Fonction LCD



Exemple de schéma à relais LCD



Exemple de texte structuré LCD

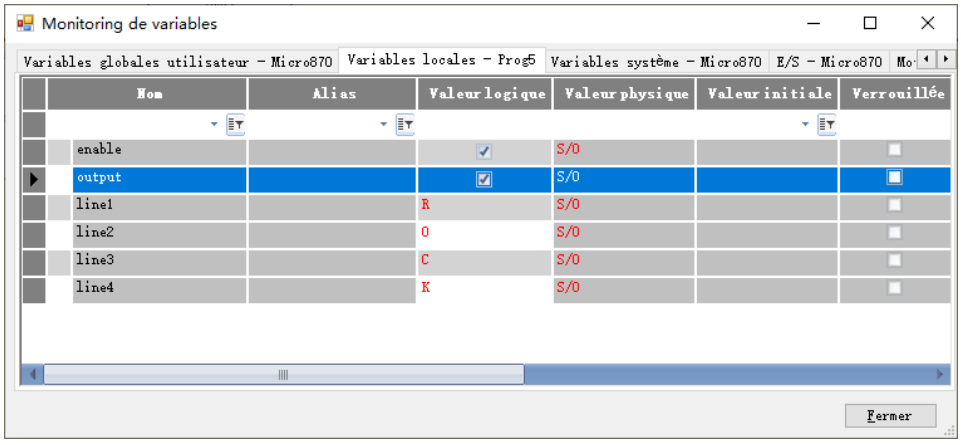



```
1  enable := TRUE;
2  line1 := 'R';
3  line2 := 'O';
4  line3 := 'C';
5  line4 := 'K';
6  output := LCD(enable, line1, line2, line3, line4);
```

(* Équivalence ST : *)

TESTOUTPUT := LCD(LCDENABLE, LINE1, LINE2, LINE3, LINE4) ;

Résultats



LCD_BKLT_REM (rétro-éclairage LCD à distance)

Définit les paramètres de rétro-éclairage du LCD à distance dans un programme utilisateur.

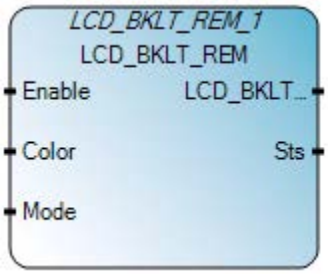
Détails de l'opération :

Les paramètres de rétro-éclairage définis dans LCD_BKLT_REM sont utilisés lorsque l'écran LCD à distance est :

- un écran défini par l'utilisateur en utilisant LCD_REM.
- l'écran des états des E/S par défaut.
- Pour tous les autres écrans, les paramètres de rétro-éclairage utilisés sont ceux définis à l'aide du menu sur le LCD à distance.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820.

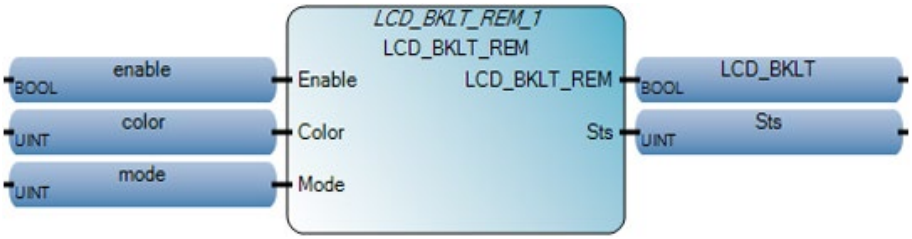


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

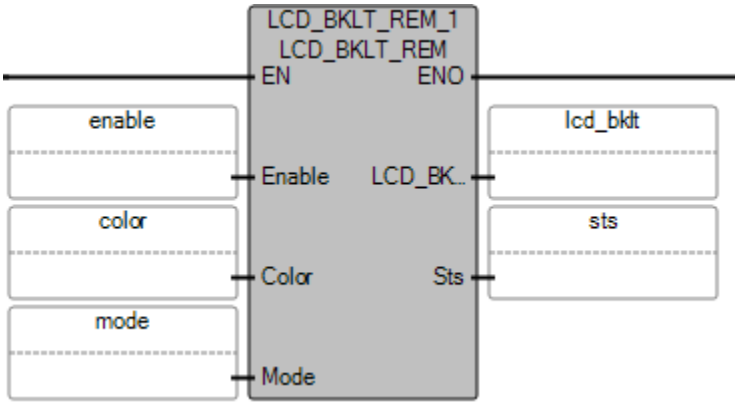
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de REM_LCD_BKLT, écrasant les paramètres de rétro-éclairage actuels. FAUX : REM_LCD_BKLT est désactivé et les paramètres du menu LCD à distance prennent effet.
Color	Entrée	UINT	Code couleur du rétro-éclairage • 0: Blanc • 1: Bleu • 2: Rouge • 3: Vert • 4-65535 : Réservé
Mode	Entrée	UINT	• 0 : Désactivé en permanence • 1: Activé en permanence • 2: Flash (1 seconde d'intervalle) • 3-65535 : Réservé
LCD_BKLT_REM	Sortie	BOOL	VRAI : instruction exécutée avec succès. FAUX : une erreur est survenue lors de l'exécution de l'instruction.
Sts	Sortie	UINT	Statut de l'opération LCD distant. Codes d'état LCD_BKLT_REM : • 0 - L'entrée d'activation est définie sur Faux. • 1 - Réussite. • 2 - LCD à distance non détecté. Peut se produire lorsque : • Le LCD à distance n'est pas connecté physiquement au contrôleur ou le câblage est incorrect.

Paramètre	Type de paramètre	Type de données	Description
			• Les paramètres port série sont différents de ce qui est requis pour le LCD distant. • 3 - Erreur de connexion. Peut se produire en cas d'erreur de machine à l'état interne, comme une incompatibilité entre la version du micrologiciel du contrôleur et la version du micrologiciel du LCD à distance. • 4 - Code couleur non valide. • 5 - Mode non valide. • 6-65535 - Réservé.

Exemples de Diagramme de Bloc Fonction LCD_BKLT_REM



Exemple de logique à relais LCD_BKLT_REM



Exemple de texte structuré LCD_BKLT_REM

```
LCD_BKLT_REM_4(
```

void **LCD_BKLT_REM_4**(BOOL **Enable**, UINT **Color**, UINT **Mode**)
Type: LCD_BKLT_REM, Fixe les paramètres de rétro-éclairage du LCD à distance.

```
1  LCD_BKLT_REM_1 (EN, Enable, Color, Mode);
2  output := LCD_BKLT_REM_1.ENO
3  LCD_BKLT_REM_1 := LCD_BKLT_REM_1.LCD_BKLT_REM
4  sts_lcd_rem := LCD_REM_1.Sts
```

LCD_REM (LCD à distance)

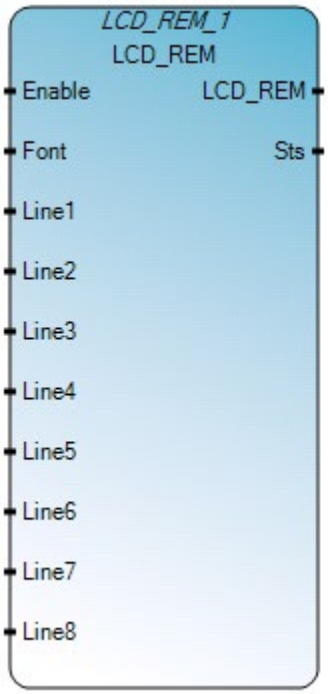
Affiche les messages définis par l'utilisateur pour le LCD à distance.

Détails de l'opération :

- Pour les lignes 1 à 8, la longueur maximale de la chaîne est de 24 caractères.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820.

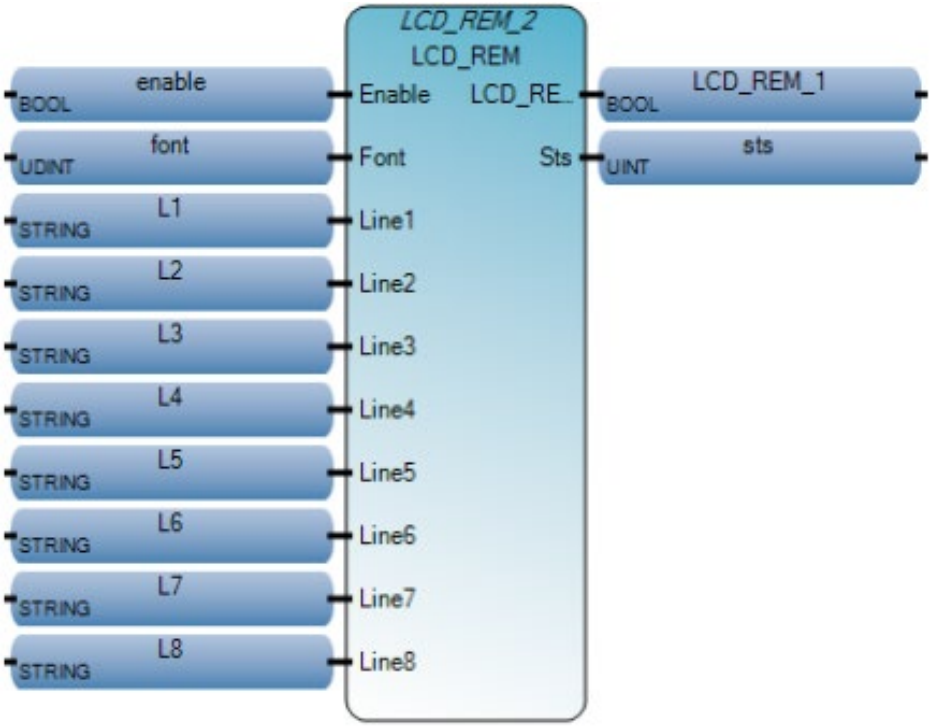


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

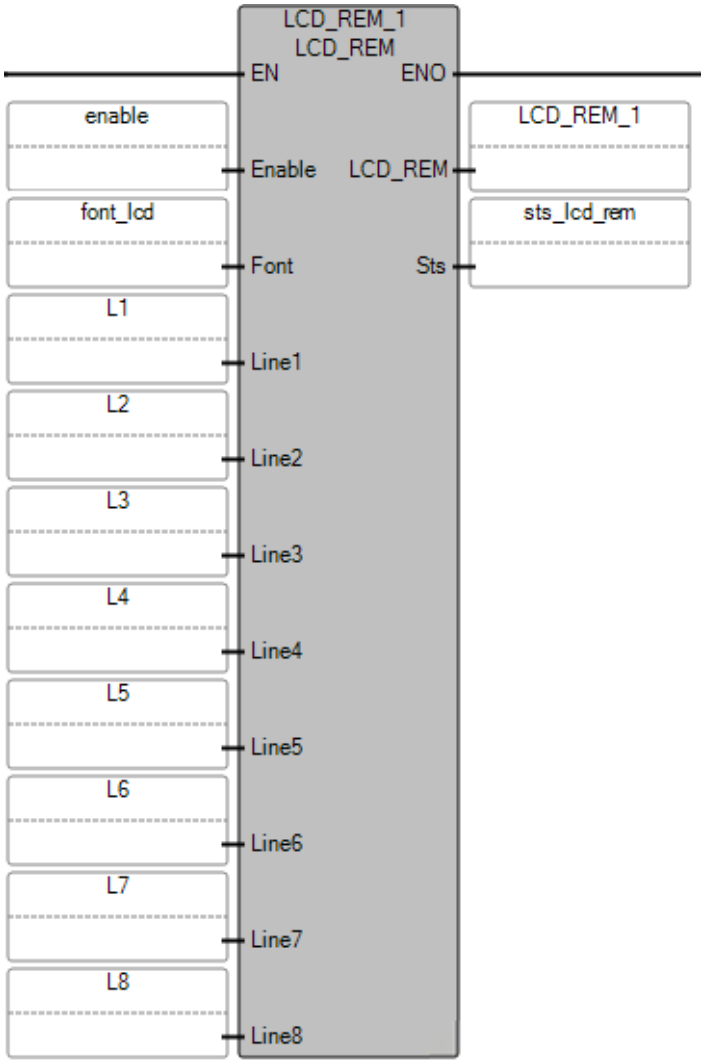
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Active le bloc d'instruction. VRAI : le LCD à distance bascule sur l'écran défini par l'utilisateur à partir de l'écran d'état des E/S. FAUX : le LCD à distance revient à l'écran d'état des E/S.
Font	Entrée	UDINT	Taille de la police du message de démarrage : • 0: Par défaut (grand – 8 x 16) • 1: Petit (8 x 8) • 2: Grand (8 x 16) • 3: Très grand (16 x 16) • À partir de 4 : Réservé La taille du LCD à distance est de 192 x 64 pixels. Lorsque la taille de la police pour le message de démarrage est petite, le LCD à distance affiche : • les chaînes des lignes 1 à 8. • un maximum de 24 caractères par ligne. Lorsque la taille de la police pour le message de démarrage est grande ou très grande, le LCD à distance affiche : • les chaînes des lignes 1 à 4. • ignore les chaînes des lignes 5 à 8. • un maximum de 12 caractères par ligne.
Line1	Entrée	Chaîne	Chaîne à afficher sur la ligne 1 du module LCD. Pour les lignes 1 à 8, la longueur maximale de la chaîne est de 24 caractères.

Paramètre	Type de paramètre	Type de données	Description
Line2	Entrée	Chaîne	Chaîne affichée sur la ligne 2 du module LCD.
Line3	Entrée	Chaîne	Chaîne affichée sur la ligne 3 du module LCD.
Line4	Entrée	Chaîne	Chaîne affichée sur la ligne 4 du module LCD.
Line5	Entrée	Chaîne	Chaîne affichée sur la ligne 5 du module LCD.
Line6	Entrée	Chaîne	Chaîne affichée sur la ligne 6 du module LCD.
Line7	Entrée	Chaîne	Chaîne affichée sur la ligne 7 du module LCD.
Line8	Entrée	Chaîne	Chaîne affichée sur la ligne 8 du module LCD.
LCD_REM	Sortie	BOOL	Activation du bloc de fonctions. Lorsque Enable = VRAI, l'affichage utilisateur est actif. Lorsque Enable = FAUX, l'affichage menu/statut E/S est actif.
Sts	Sortie	UINT	Statut de l'opération LCD distant. Codes d'état LCD_REM : • 0 - L'entrée d'activation est définie sur Faux. • 1 - Message utilisateur affiché avec succès. • 2 - LCD à distance non détecté. • 3 - Erreur de connexion. Peut se produire lorsque : • Le LCD distant n'est pas connecté physiquement au contrôleur (ou le câblage est incorrect). • Les paramètres port série sont différents de ce qui est requis pour le LCD distant. • 4 - Code de police non valide. • 5 - 5-65535 - Réservé.

Exemple de Diagramme de Bloc Fonction LCD_REM



Exemple de logique à relais LCD_REM



Exemple de texte structuré LCD_REM

```
LCD_REM_1(
    void LCD_REM_1(BOOL Enable, UDINT Font, STRING Line1, STRING Line2, STRING Line3, STRING Line4, STRING Line5, STRING Line6, STRING Line7, STRING Line8)
    Type: LCD_REM, Affiche les chaînes utilisateur sur le LCD à distance lorsque connecté.

1  LCD_REM_1 (EN, Enable, Font, Line1, Line2, Line3, Line4, Line5, Line6, Line7, Line8);
2  output := LCD_REM_1.ENO
3  LCD_REM_1 := LCD_REM_1.LCD_REM
4  sts_lcd_rem := LCD_REM_1.Sts
```

RHC (lecture de l'horloge haute vitesse)

Lit une valeur de l'horloge haute vitesse dans le contrôleur Micro800.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

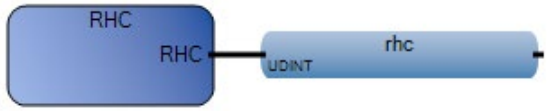
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : lecture de l'horloge haute vitesse. FAUX : aucune opération. S'applique aux programmes de schémas à relais.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
RHC	Sortie	UDINT	La valeur de l'horloge haute vitesse.

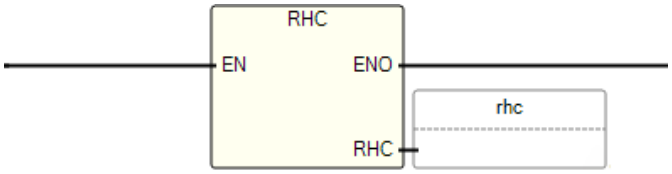
Résolution de l'horloge haute vitesse

Type du contrôleur	Incréments	Référence temporelle	Résolution
Micro810	4 toutes les 40 microsecondes	10 microsecondes	40 microsecondes
Micro820 Micro830 Micro850	1 toutes les 10 microsecondes	10 microsecondes	10 microsecondes

Exemple de Diagramme de Bloc Fonction RHS



Exemple de schéma à relais RHS



Exemple de texte structuré RHS

```
RHC(
    UDINT RHC()
    Lecture d'une horloge haute vitesse.
1   rhc := RHC();
```

(* Équivalence ST : *)

TESTOUTPUT2 := RHC() ;

RPC (lire la somme de contrôle du programme)

Lit la somme de contrôle du programme utilisateur à partir du contrôleur ou du module mémoire.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

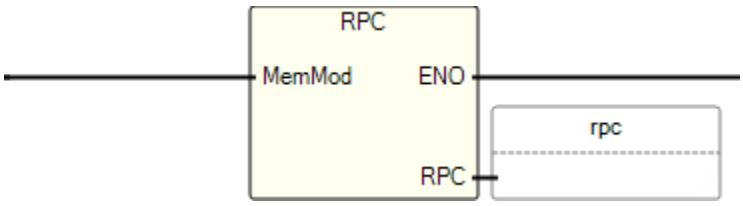
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
MemMod	Entrée	BOOL	VRAI : la valeur est extraite du module mémoire. FAUX : la valeur est extraite du contrôleur Micro800.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
RPC	Sortie	UDINT	Valeur de la somme de contrôle du programme utilisateur spécifié.

Exemple de diagramme de bloc de fonctions RPC

Exemple de langage à relais RPC



Exemple de texte structuré RPC

```
RPC(
    UDINT RPC(BOOL MemMod)
    Lecture de la somme de contrôle (checksum) du programme utilisateur.

1  MemMod := TRUE;
2  rpc := RPC(MemMod);
```

```
(* Équivalence ST : *)

TESTOUTPUT2 := RPC(TESTINPUT) ;
```

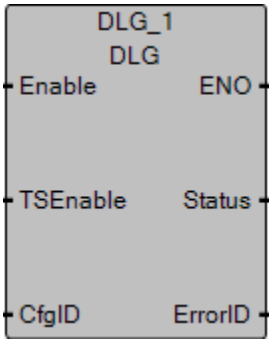
DLG (journal de données)

Écrit des valeurs de variables depuis le moteur d'exécution dans un fichier de journalisation des données sur une carte SD.

En ce qui concerne l'écriture dans un journal de données, 50 dossiers de groupes maximum sont autorisés par jour. Chaque dossier de groupe contient un maximum de 50 fichiers, d'une taille comprise entre 4k et 8k.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

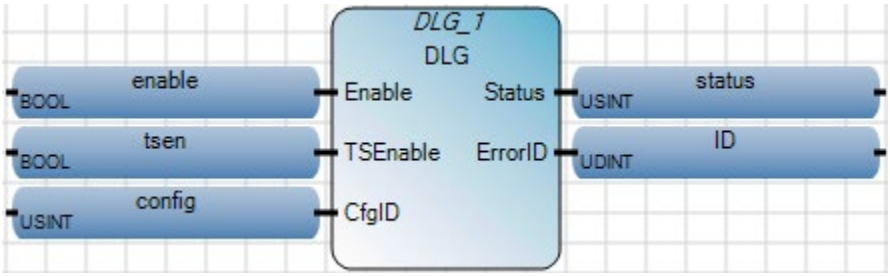
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation de l'écriture de la journalisation des données. VRAI : front montant activé détecté, début de l'opération de journalisation des données lorsque les opérations d'instruction précédentes sont terminées. FAUX : front montant non détecté.
TSEnable	Entrée	BOOL	VRAI : indicateur d'activation d'horodatage date et heure.
CfgID	Entrée	USINT	Numéro d'ID de VA de configuration de journalisation des données.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
Status	Sortie	USINT	État actuel de l'instruction. Codes d'état de journalisation des données : • 0 - En attente • 1 - En cours • 2 - Réussite, indique la réussite de la journalisation des données. • 3 - Error, indique que la journalisation des données s'est terminée avec des erreurs.
ErrorID	Sortie	UDINT	Code d'erreur numérique unique pour DLG.

Codes d'erreur DLG

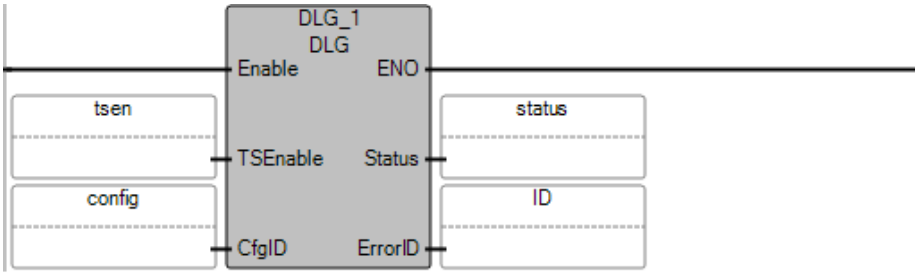
Code d'erreur	Nom de l'erreur	Commentaires
0	DLG_ERR_NONE	Aucune erreur.

Code d'erreur	Nom de l'erreur	Commentaires
1	DLG_ERR_NO_SDCARD	La carte SD est absente.
2	DLG_ERR_RESERVED	Réservé.
3	DLG_ERR_DATAFILE_ACCESS	Erreur d'accès au fichier de journalisation des données.
4	DLG_ERR_CFG_ABSENT	Le fichier de configuration de la journalisation des données est absent.
5	DLG_ERR_CFG_ID	L'ID de configuration est absent du fichier de configuration de la journalisation des données
6	DLG_ERR_RESOURCE_BUSY	L'opération de journalisation des données liée à cet ID de journalisation des données est utilisée par une autre opération FB.
7	DLG_ERR_CFG_FORMAT	Le format du fichier de configuration de la journalisation des données n'est pas valide.
8	DLG_ERR_RTC	Horloge en temps réel non valide.
9	DLG_ERR_UNKNOWN	Une erreur non spécifiée s'est produite.

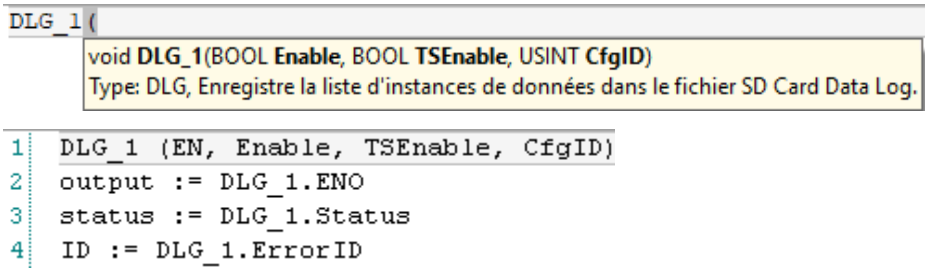
Exemple de Diagramme de Bloc Fonction DLG



Exemple de langage à relais DLG



Exemple de texte structuré DLG



IIM (entrée immédiate)

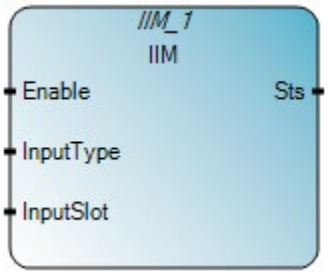
Met à jour les entrées avant la scrutation de sortie normale.

Détails de l'opération :

- Généralement utilisé au début d'un programme d'interruption pour sélectionner ou masquer les entrées qui sont immédiatement scrutées pour obtenir les entrées actuelles.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

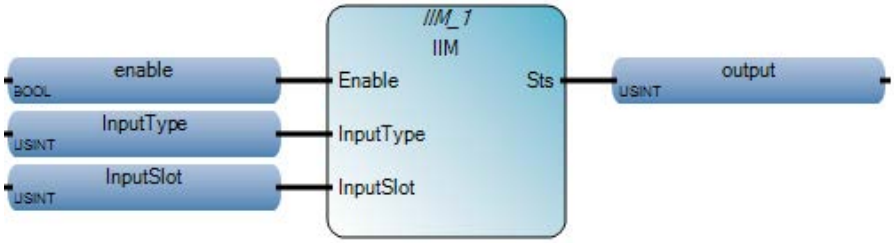
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870.



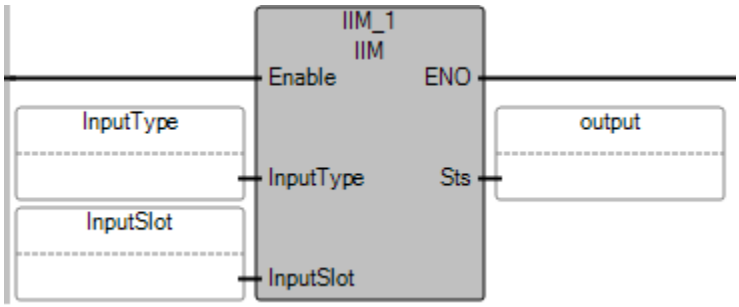
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du bloc d'instruction. FAUX : aucune exécution.
InputType	Entrée	USINT	Identifie le type de l'entrée : 0 - Entrée intégrée 1 - Entrée enfichable
InputSlot	Entrée	USINT	Identifie l'emplacement de l'entrée. 0 - Entrée intégrée 1,2,3,4,5 - Numéro d'emplacement du module enfichable. (Les emplacements sont numérotés de gauche à droite, en commençant par le numéro 1.) Pour les entrées intégrées, toujours 0. Pour les entrées enfichables, les emplacements d'entrée correspondent aux emplacements 1, 2, 3, 4 et 5 (numéro d'emplacement enfichable, avec l'emplacement le plus à gauche = 1).
Sts	Sortie	USINT	État de la scrutation d'entrée immédiate. Codes d'état IIM (Sts) : • 0x00 : non activé (aucune action déclenchée). • 0x01 : réussite de la scrutation d'entrée/de sortie. • 0x02 : type d'entrée/de sortie non valide. • 0x03 : emplacement d'entrée/de sortie non valide.
ENO	Sortie	BOOL	Activation de la sortie. VRAI : entrée mise à jour. FAUX : entrée non mise à jour. Ne s'applique qu'aux programmes de langage à relais.

Exemple de Diagramme de Bloc Fonction IIM



Exemple de langage à relais IIM



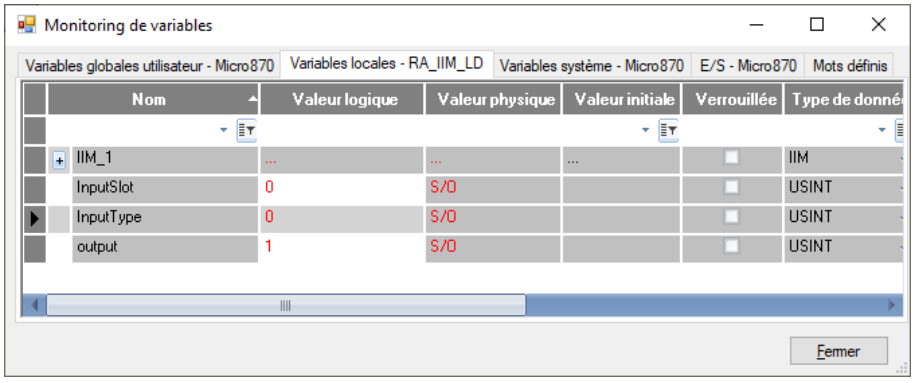
Exemple de texte structuré IIM

```
IIM_1(
```

void **IIM_1**(BOOL **Enable**, USINT **InputType**, USINT **InputSlot**)
Type: IIM, Mise à jour des entrées avant l'analyse normale des entrées.

```
1  enable := TRUE;
2  InputType := 0;
3  InputSlot := 0;
4  IIM_1(enable, InputType, InputSlot);
5  output := IIM_1.Sts;
```

Résultats



IOM (sortie immédiate)

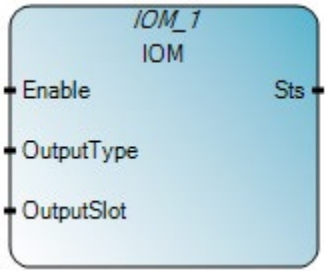
Mise à jour des sorties avant la scrutation normale des sorties.

Détails de l'opération :

- Généralement à la fin d'un programme d'interruption pour sélectionner ou masquer les sorties qui sont immédiatement scrutées et mises à jour.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

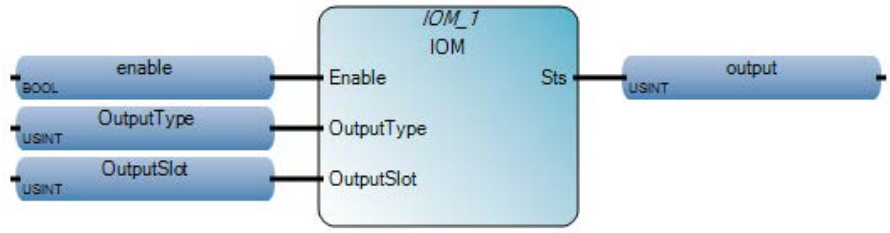
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870.



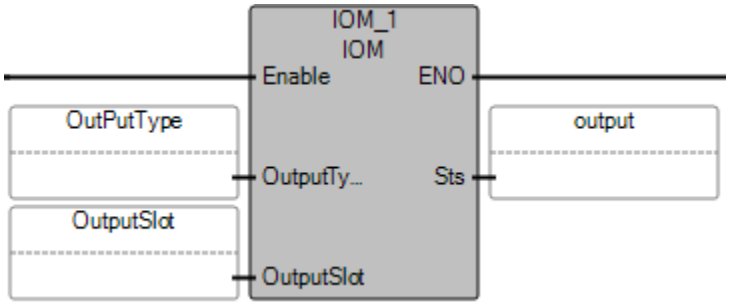
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de l'instruction. FAUX : aucune exécution.
OutputType	Entrée	USINT	Permet d'identifier le type de sortie : 0 - Sortie intégrée 1 - Sortie enfichable
OutputSlot	Entrée	USINT	Permet d'identifier l'emplacement de sortie : 0 - Sortie intégrée 1,2,3,4,5 - Numéro d'emplacement du module enfichable. (Les emplacements sont numérotés de gauche à droite, en commençant par le numéro 1.) Pour les sorties intégrées, toujours 0. Pour les sorties enfichables, les emplacements de sortie correspondent aux emplacements 1, 2, 3, 4 et 5 (numéro d'emplacement enfichable, avec l'emplacement le plus à gauche = 1).
Sts	Sortie	USINT	État de la scrutation de sortie immédiate. Codes d'état IOM (Sts) : • 0x00 : non activé (aucune action déclenchée). • 0x01 : réussite de la scrutation d'entrée/de sortie. • 0x02 : type d'entrée/de sortie non valide. • 0x03 : emplacement d'entrée/de sortie non valide.
ENO	Sortie	BOOL	Activation de la sortie. VRAI : sortie mise à jour. FAUX : sortie non mise à jour. Ne s'applique qu'aux programmes de langage à relais.

Exemple de Diagramme de Bloc Fonction IOM



Exemple de langage à relais IOM



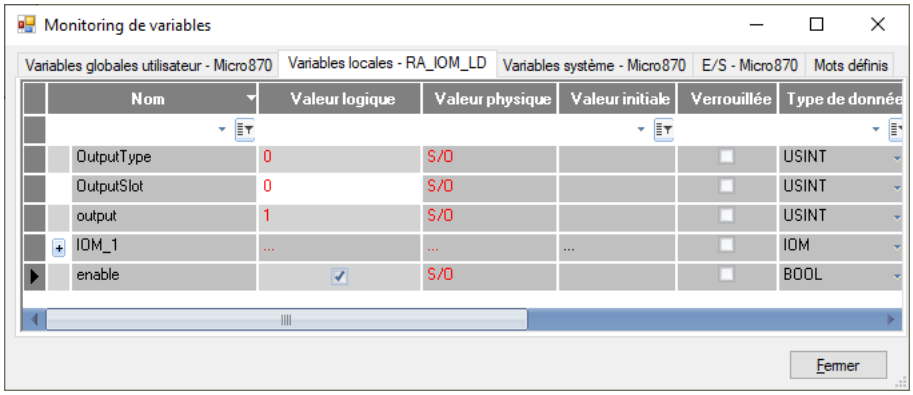
Exemple de texte structuré IOM

```
IOM_1(
```

void **IOM_1**(BOOL **Enable**, USINT **OutputType**, USINT **OutputSlot**)
Type: IOM, Mise à jour des sorties avant l'analyse normale des sorties.

```
1  enable := TRUE;
2  OutputType := 0;
3  OutputSlot := 0;
4  IOM_1(enable, OutputType, OutputSlot);
5  output := IOM_1.Sts;
```

Résultats

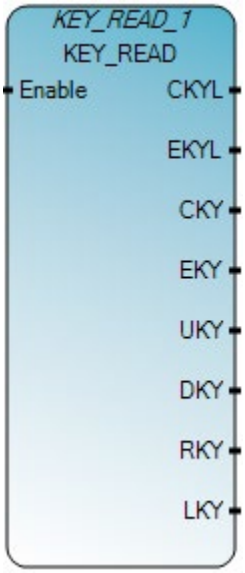


KEY_READ (lit des touches sur LCD)

Lit l'état des touches sur le module LCD facultatif lorsque l'affichage utilisateur est actif.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

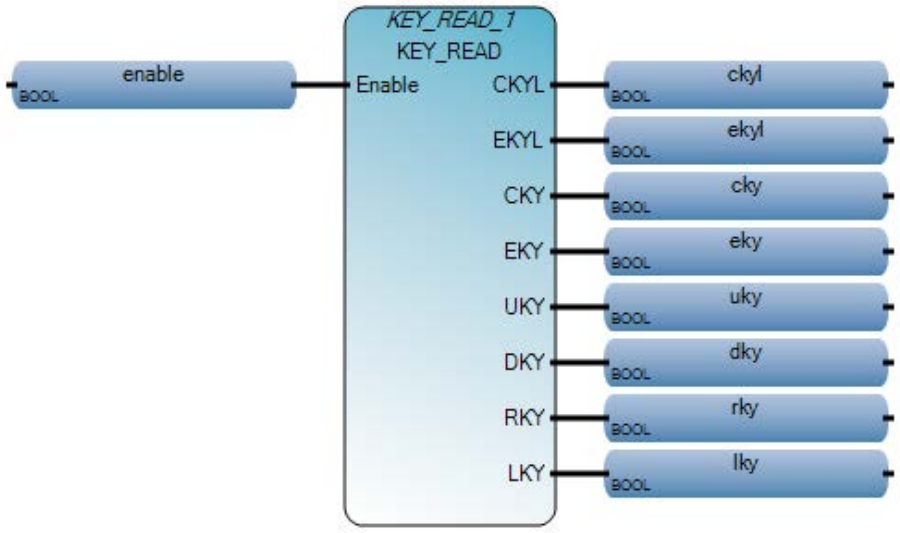
Cette instruction s'applique au contrôleur Micro810.



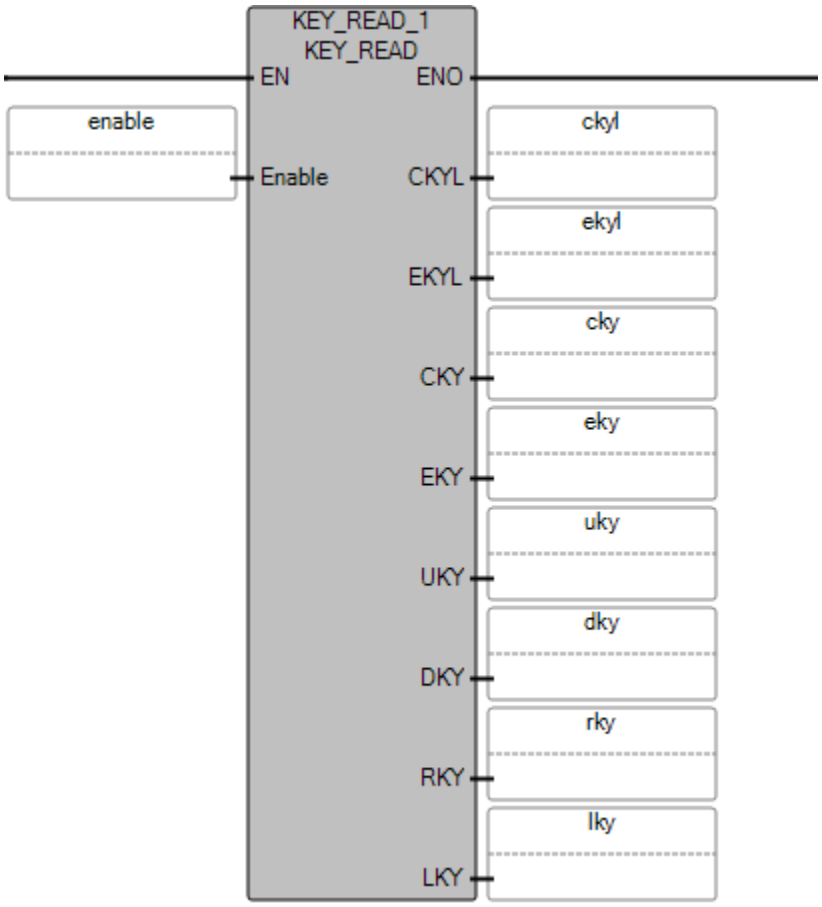
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Active le bloc d'instruction. VRAI : active la lecture des touches du clavier sur le LCD à distance. FAUX : désactive la lecture des touches sur le LCD à distance.
CKYL	Sortie	BOOL	TRUE : appui sur la touche ÉCHAP pendant plus de 2 secondes.
EKYL	Sortie	BOOL	TRUE : appui sur la touche OK pendant plus de 2 secondes.
CKY	Sortie	BOOL	TRUE : appui sur la touche ÉCHAP.
EKY	Sortie	BOOL	TRUE : appui sur la touche OK.
UKY	Sortie	BOOL	TRUE : appui sur la touche HAUT.
DKY	Sortie	BOOL	TRUE : appui sur la touche BAS.
LKY	Sortie	BOOL	TRUE : appui sur la touche Gauche.
RKY	Sortie	BOOL	TRUE : appui sur la touche Droite.

Exemple de Diagramme de Bloc Fonction KEY_READ



Exemple de logique à relais KEY_READ



Exemple de texte structuré KEY_READ

```
KEY_READ_1(
```

void **KEY_READ_1**(BOOL **Enable**)
Type: KEY_READ, Lecture du statut de la clé sur le module LCD optionnel.

```
1  KEY_READ_1(enable);
2  ckyl := KEY_READ_1.CKYL;
3  ekyl := KEY_READ_1.EKYL;
4  cky := KEY_READ_1.CKY;
5  eky := KEY_READ_1.EKY;
6  uky := KEY_READ_1.UKY;
7  dky := KEY_READ_1.DKY;
8  lky := KEY_READ_1.LKY;
9  rky := KEY_READ_1.RKY;
```

(* Équivalence ST : *)

```
KEY_READ_1(KEYENABLE) ;
KEY_EKYL := KEY_READ_1.EKYL ;
KEY_CKY := KEY_READ_1.CKY ;
KEY_EKY := KEY_READ_1.EKY ;
KEY_UKY := KEY_READ_1.UKY ;
KEY_DKY := KEY_READ_1.DKY ;
KEY_RKY := KEY_READ_1.RKY ;
KEY_LKY := KEY_READ_1.LKY ;
```

KEY_READ_REM (lecture des touches pour un LCD à distance)

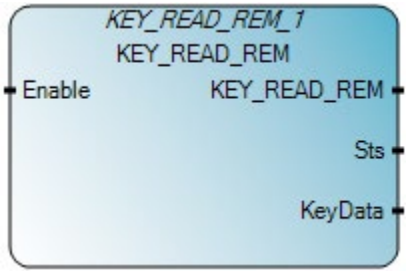
Lit l'état des touches sur le module LCD à distance facultatif lorsque l'affichage utilisateur est actif.

Détails de l'opération :

- Utilisez l'instruction LCD_REM pour activer l'affichage utilisateur sur le module LCD à distance. Si l'affichage utilisateur n'est pas actif, une erreur se produit lors de l'exécution de KEY_READ_REM.
- La propriété du bouton P dans le fichier de fonction LCD s'active ; sinon tous les états de touche indiquent « FALSE ».
- Seules les pressions de touche uniques sont prises en charge pour l'instruction KEY_READ_REM ; les combinaisons à deux touches ne sont pas prises en charge.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

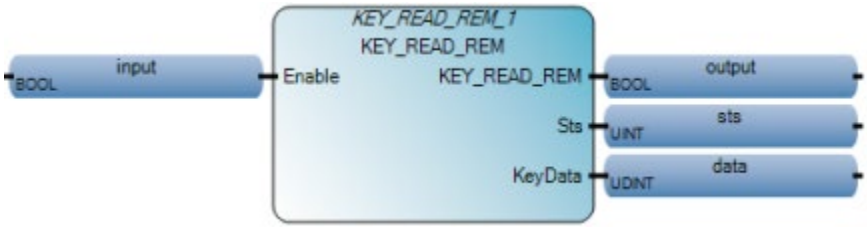
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : activation FAUX : désactivation
KEY_READ_REM	Sortie	BOOL	VRAI : les données de touche du LCD à distance ont été lues correctement. FAUX : l'activation est définie sur Faux, erreur de lecture. Les données de touche du LCD distant ou de l'affichage utilisateur ne sont pas actives.
Sts	Sortie	UINT	Statut de l'opération KEY_READ_REM. Codes d'état KEY_READ_REM : • 0 - L'entrée d'activation est définie sur Faux. • 1 - Les données de touche sont lues correctement. • 2 - LCD à distance non détecté. Peut se produire lorsque : • Le LCD distant n'est pas connecté physiquement au contrôleur (ou le câblage est incorrect). • Les paramètres port série sont différents de ce qui est requis pour le LCD distant. • 3 - Erreur de connexion. Peut se produire en cas d'erreur de machine à l'état interne. Cause possible : une incompatibilité entre la version du micrologiciel du contrôleur et la version du micrologiciel du LCD à distance. • 4 - L'affichage utilisateur n'est pas actif. • 5-65535 - Réservé.
KeyData	Sortie	UDINT	Données du clavier de LCD distant. Les définitions KeyData sont définies dans le tableau des champs de bits KeyData.

Tableau de champs de bits KeyData

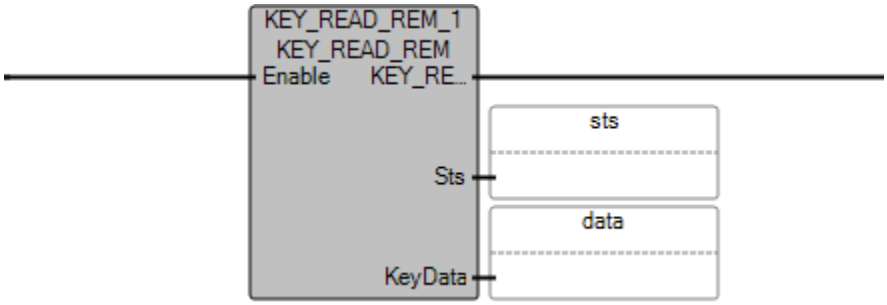
Ce tableau permet de déterminer les champs de bits KeyData pour KEY_READ_REM.

N° de bit dans KeyData	Nom	Description du paramètre
0	UKY	TRUE = appui sur la touche HAUT.
1	DKY	TRUE = appui sur la touche BAS.
2	LKY	TRUE = appui sur la touche GAUCHE.
3	RKY	TRUE = appui sur la touche DROITE.
4	F1KY	TRUE = appui sur la touche F1.
5	F2KY	TRUE = appui sur la touche F2.
6	F3KY	TRUE = appui sur la touche F3.
7	F4KY	TRUE = appui sur la touche F4.
8	F5KY	TRUE = appui sur la touche F5.
9	F6KY	TRUE = appui sur la touche F6.
10	EKY	TRUE = appui sur la touche ENTRÉE.
11	CKY	TRUE = appui sur la touche ÉCHAP.
12	EKYL	TRUE = appui sur la touche ENTRÉE pendant plus de 2 secondes.
13	CKYL	TRUE = appui sur la touche ÉCHAP pendant plus de 2 secondes.
14-31	--	Réservé.

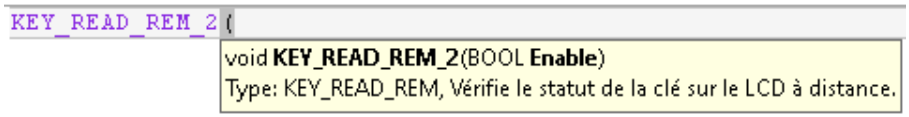
Exemple de Diagramme de Bloc Fonction KEY_READ_REM



Exemple de logique à relais KEY_READ_REM



Exemple de texte structuré KEY_READ_REM




```
1  KEY_READ_REM (Enable);
2  output := KEY_READ_REM.ENO
3  sts := KEY_READ_REM.Sts
4  data := KEY_READ_REM.KeyData
```

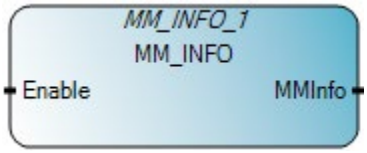
MM_INFO (informations du module mémoire)

Lit les informations d'en-tête du module mémoire. En cas d'absence de module mémoire, toutes les valeurs sont définies sur zéro (0).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).

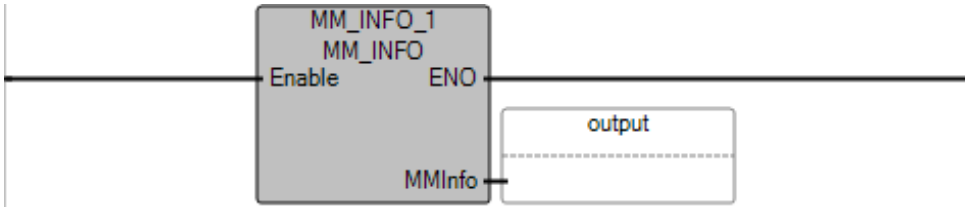


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

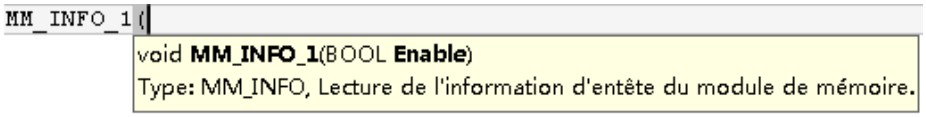
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lecture des informations d'en-tête du module mémoire. FAUX : aucune opération de lecture n'est réalisée et les informations du module mémoire de sortie ne sont pas valides.
MMInfo	Sortie	MMINFO	Les informations du module mémoire sont définies dans le type de données MMINFO sur la page 361.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions MM_INFO

Exemple de langage à relais MM_INFO



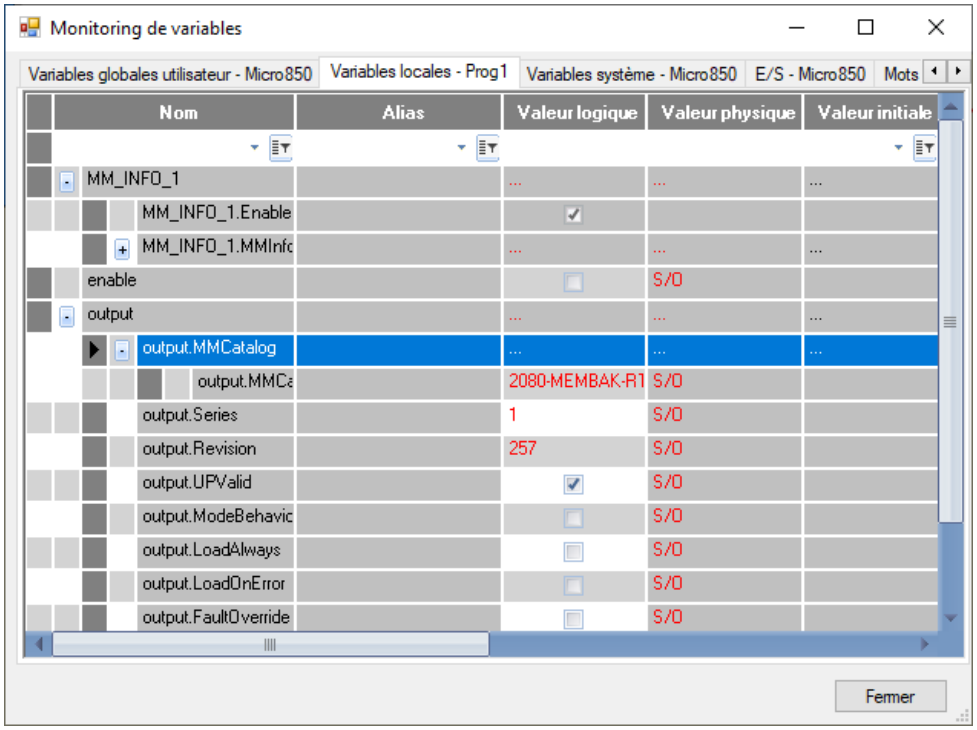
Exemple de texte structuré MM_INFO



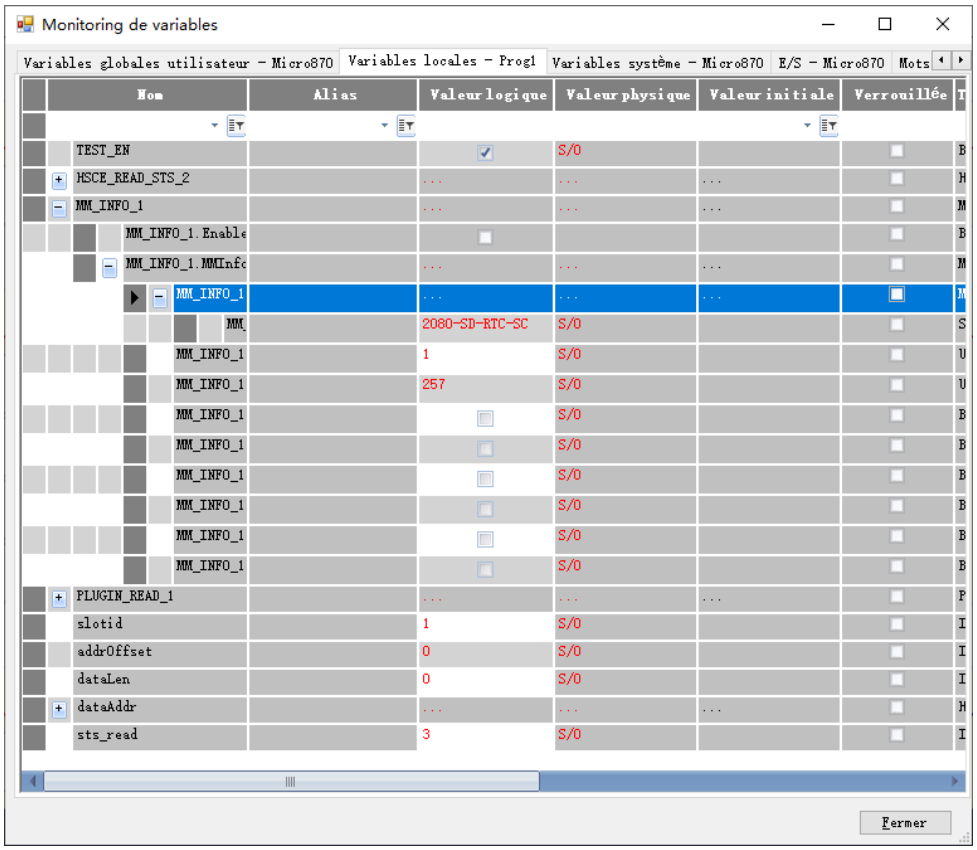

```
1  MM_INFO_1(enable);
2  output := MM_INFO_1.MMInfo;
```

Résultats

Pour les contrôleurs utilisant 2080-MEMBAK-RTC :



Pour les contrôleurs utilisant une carte SD :



Type de données MMINFO

Le tableau suivant décrit les paramètres du type de données MMINFO.

Paramètre	Type de données	Description
MMCatalog	MMCATNUM	Numéro catalogue du module mémoire. Lorsqu'une instruction MM_INFO sur la page 359 est utilisée sur des contrôleurs avec une carte SD, le MMCatalog est « SD CARD ».
Series	UINT	Série du module mémoire. Lorsqu'une instruction MM_INFO est utilisée sur des contrôleurs avec une carte SD, la série est 1.
Revision	UINT	Version du module mémoire. Lorsqu'une instruction MM_INFO est utilisée sur des contrôleurs avec une carte SD, la version est 257.
UPValid	BOOL	Le programme utilisateur est présent (TRUE : un projet potentiellement valide a été identifié). Remarque : Même lorsque la valeur est TRUE, il est possible que le projet soit identifié comme non valide durant le chargement ou la restauration si des fichiers individuels sont manquants ou corrompus.
ModeBehavior	BOOL	Comportement du mode (TRUE : passage à RUN lors de la mise sous tension).
LoadAlways	BOOL	Restauration du module mémoire avec les données du contrôleur à chaque mise sous tension.
LoadOnError	BOOL	Restauration du module mémoire avec les données du contrôleur si une erreur survient lors de la mise sous tension.
FaultOverride	BOOL	Annulation du défaut lors de la mise sous tension.
MMPresent	BOOL	Le module mémoire est présent.

MODULE_INFO

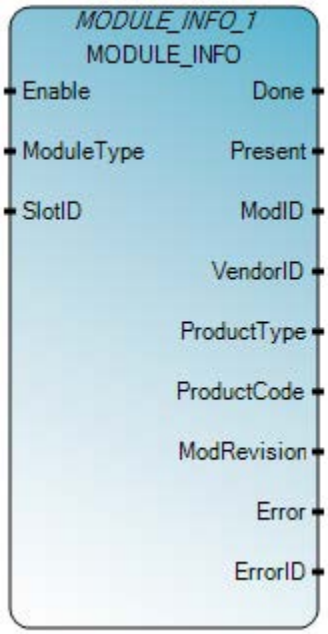
Lit les informations de module à partir d'un module enfichable ou d'un module d'extension.

Détails de l'opération :

- Les informations du module enfichable sont lues lors de l'exécution.
- Le module enfichable de mémoire 2080-MEMBAK-RTC n'est pas pris en charge.
- Les informations du module d'extension sont lues lorsque le module est mis sous tension.
- Lorsqu'un module enfichable ou un module d'extension n'est pas défini par un ID de module, un type de produit ou un code de produit, l'opération MODULE_INFO renvoie 0 comme paramètre de sortie respectif.
- Les informations d'identification des modules enfichables et des modules d'extension sont définies par Allen-Bradley et sont fournies ci-dessous dans le cadre de la description MODULE_INFO.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870. Les modules d'extension sont uniquement pris en charge sur les contrôleurs Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de l'opération de lecture MODULE_INFO. FAUX : aucune exécution de l'opération de lecture. Toutes les valeurs de données de sortie sont réinitialisées sur 0.
ModuleType	Entrée	USINT	Identifie le type de module : • 1 - Module d'extension 2085. • 2 - Module enfichable 2080.
SlotID	Entrée	USINT	Le numéro d'emplacement où le module enfichable ou le module d'extension se trouve. Les ID d'emplacement sont : 1, 2, 3, 4, 5 L'emplacement 1 se trouve le plus à gauche.
Done	Sortie	BOOL	VRAI : opération terminée avec succès. FAUX : l'opération ne s'exécute pas ou une erreur s'est produite.
Present	Sortie	BOOL	Détecte le module enfichable ou le module d'extension dans l'emplacement du contrôleur. VRAI : le module est présent physiquement. FAUX : le module n'est pas présent physiquement.
ModID	Sortie	UINT	L'identification du module dans l'emplacement du contrôleur. • Les modules enfichables sont définis avec un identifiant de module unique. • Les modules d'extension ne sont pas définis avec un identifiant de module unique, ModID renvoie 0.
VendorID	Sortie	UINT	L'ID du fournisseur des modules enfichables ou des modules d'extension. Pour les produits Allen-Bradley, l'ID du fournisseur est 1.
ProductType	Sortie	UINT	Type de produit du module enfichable ou du module d'extension.
ProductCode	Sortie	UINT	Le code de produit du module enfichable ou du module d'extension.
ModRevision	Sortie	UINT	Les informations de version du module enfichable ou du module d'extension.

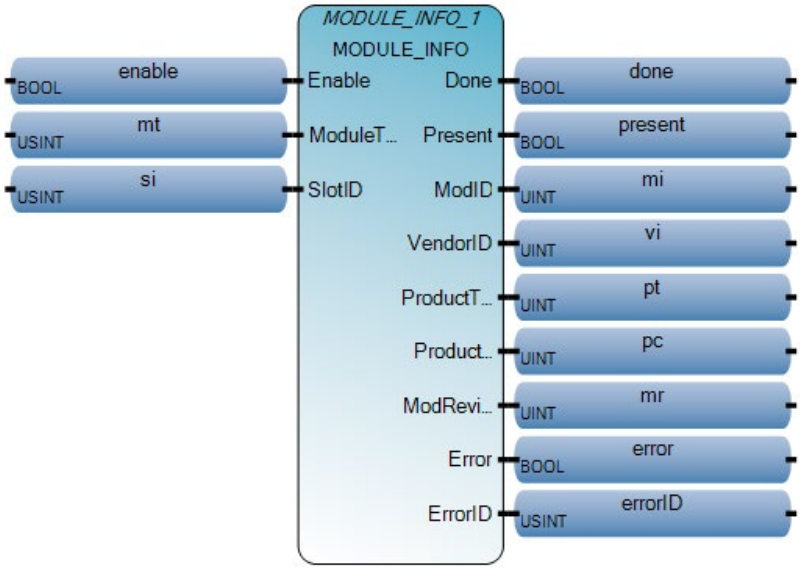
Paramètre	Type de paramètre	Type de données	Description
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	USINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur MODULE_INFO.

Codes d'erreur MODULE_INFO

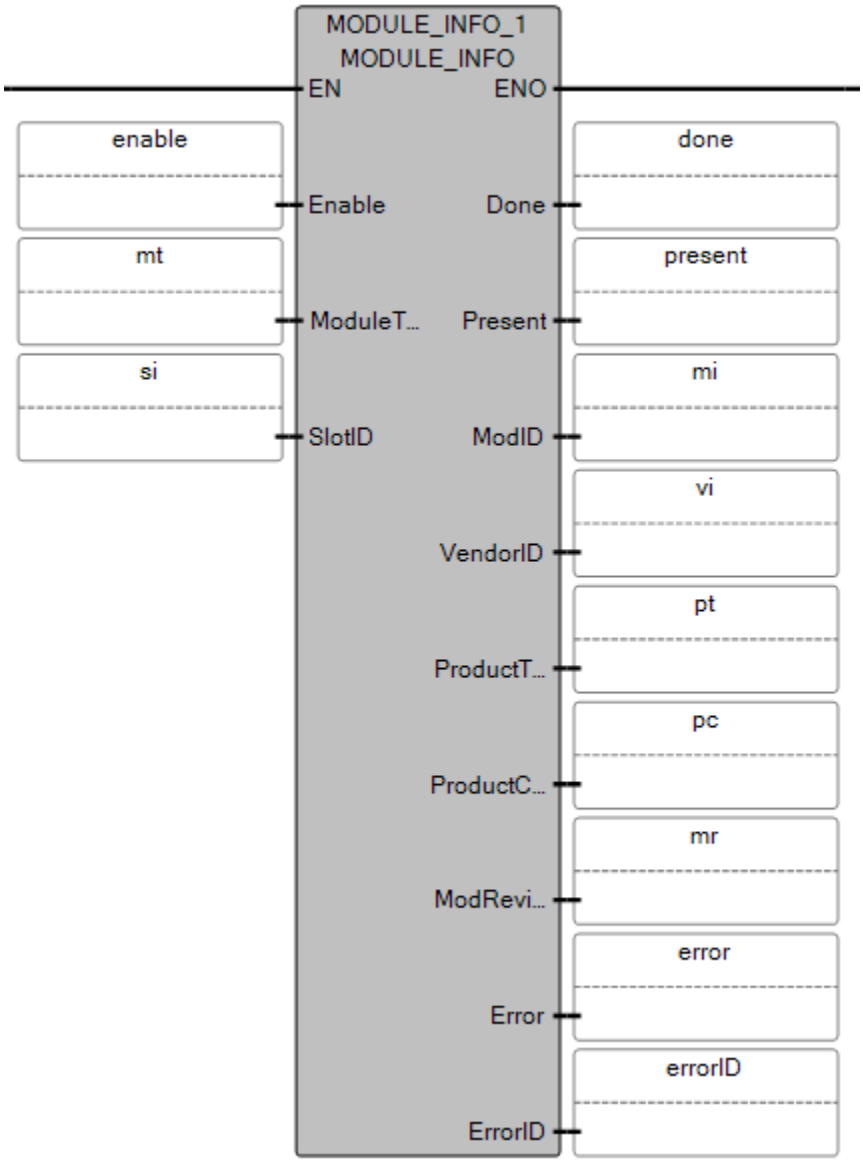
Utilisez ce tableau pour déterminer les codes d'erreur MODULE_INFO et les descriptions.

Code d'erreur	Description de l'erreur
1	Type de module non valide. Modifiez le type de module en un type valide. Les types de module valides sont les suivants : 1 - Module d'extension 2085. Uniquement pris en charge sur les contrôleurs Micro850. 2 - Module enfichable 2080. Pris en charge sur les contrôleurs Micro820, Micro830 et Micro850.
2	Numéro d'emplacement non valide. Modifiez le numéro d'emplacement en un numéro valide.
3	Type de module d'extension non valide.
4	Erreur fatale du module d'extension.
5	Les informations de lecture du module enfichable ne sont pas prises en charge.
6	Une erreur de lecture du module enfichable s'est produite pendant la lecture de l'ID du module.
7	Une erreur de lecture du module enfichable s'est produite pendant la lecture de l'ID du fournisseur.
8	Erreur de lecture du module enfichable pendant la lecture du type de produit.
9	Une erreur de lecture du module enfichable s'est produite pendant la lecture du code du produit.
10	Une erreur de lecture du module enfichable s'est produite pendant la lecture de la version du module.

Exemple de diagramme de bloc de fonctions MODULE_INFO



Exemple de langage à relais MODULE_INFO



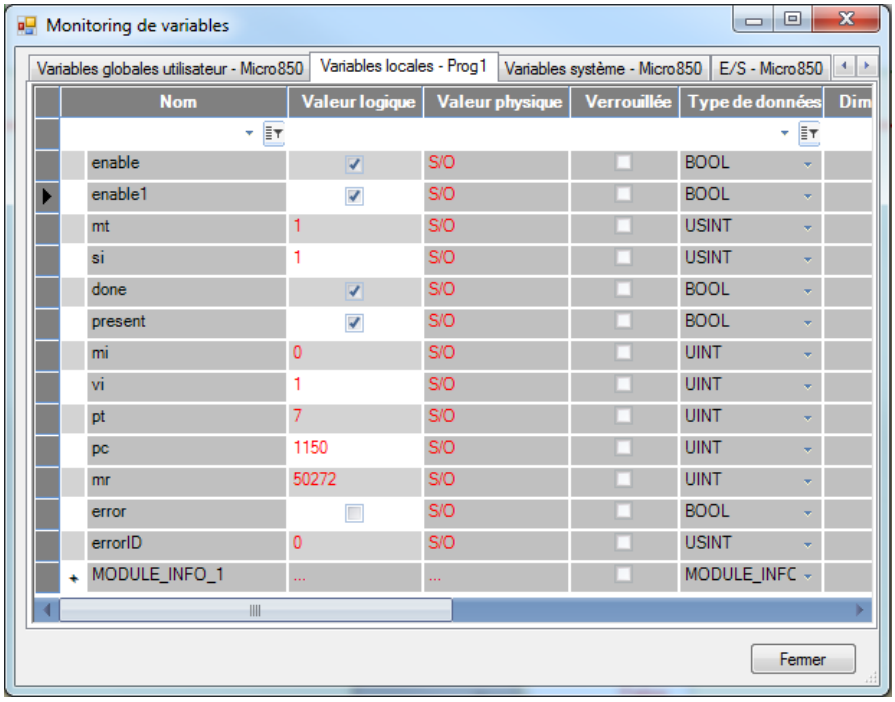
Exemple de texte structuré MODULE_INFO

```
1  MODULE_INFO_1(enable, mt, si);
2  done := MODULE_INFO_1.Done;
3  present :=MODULE_INFO_1.Present;
4  mi :=MODULE_INFO_1.ModID;
5  vi :=MODULE_INFO_1.VendorID;
6  pt :=MODULE_INFO_1.ProductType;
7  pc :=MODULE_INFO_1.ProductCode;
8  mr :=MODULE_INFO_1.ModRevision;
9  error :=MODULE_INFO_1.Error;
10 errorID :=MODULE_INFO_1.ErrorID;
```

```
MODULE_INFO_1(
```

void **MODULE_INFO_1**(BOOL **Enable**, USINT **ModuleType**, USINT **SlotID**)
(Aucun commentaire)

Résultats



MODULE_INFO - Informations relatives au module enfichable et au module d'extension

Les informations ci-dessous fournissent le type de module d'extension, l'ID de module, l'ID de fournisseur, les codes de produit et les descriptions Word de la version pour les modules d'extension, tel que défini par Allen-Bradley.

Informations du module enfichable

Utilisez ce tableau pour déterminer les informations du module enfichable défini par Allen-Bradley.

Module enfichable	Type de module enfichable	ID du module	ID du fournisseur	Type de produit	Code de produit
2080-IF2	Analogique	96	1	10	32
2080-IF4	Analogique	98	1	10	33
2080-OF2	Analogique	100	1	10	34
2080-TC2	Analogique	102	1	10	35
2080-RTD2	Analogique	104	1	10	36
2080-DNET20	Communication	34	1	12	249
2080-SERIALISOL	Communication	32	1	-	-
2080-IQ4	Numérique	192	1	7	192
2080-OB4	Numérique	193	1	7	193
2080-OV4	Numérique	194	1	7	194
2080-IQ4OB4	Numérique	195	1	7	195
2080-IQ4OV4	Numérique	196	1	7	196
2080-OW4I	Numérique	197	1	7	197
2080-MOT-HSC	Spécialité	48	1	43	48
2080-TRIMPOT6	Spécialité	72	1	-	-

Informations du module d'expansion

Utilisez ce tableau pour déterminer les informations du module d'extension défini par Allen-Bradley.

Numéro de modèle	Type de module d'extension	ID du module	ID du fournisseur	Type de produit	Code de produit
2085-IF4	Analogique	-	1	10	208
2085-IF8	Analogique	-	1	10	206
2085-IRT4	Analogique	-	1	10	213
2085-OF4	Analogique	-	1	10	214
2085-IA8	Numérique	-	1	7	1148
2085-IM8	Numérique	-	1	7	1152
2085-IQ16	Numérique	-	1	7	1144
2085-IQ32T	Numérique	-	1	7	1145
2085-OA8	Numérique	-	1	7	1149
2085-OB16	Numérique	-	1	7	1146
2085-OV16	Numérique	-	1	7	1147
2085-OW16	Numérique	-	1	7	1151
2085-OW8	Numérique	-	1	7	1150

Descriptions Word de la version du module d'extension

Utilisez ce tableau pour déterminer les informations Word de la version pour les modules d'extension Allen-Bradley.

Bit	Nom	Description
15 14	Vitesse (bauds) maximale (1:0)	Ces octets identifient la fréquence maximale pour la vitesse (bauds) maximale (1:0). 00 (bin) est 2 Mbps 01 (bin) est 4 Mbps 10 (bin) est 8 Mbps 11 (bin) est 16 Mbps
13 12 11 10	Révision mineure (3:0)	Révision mineure (3:0) désigne la révision mineure du produit. Ce champ indique la révision mineure du numéro de référence Catalogue désignée par l'ID du fournisseur, le type de produit, le code de produit, la série et la révision majeure. La révision mineure est comprise entre 0 et 15.
9 8 7 6 5	Révision majeure (4:0)	Révision majeure (4:0) désigne la révision majeure du produit. Ce champ indique la révision majeure du numéro de référence Catalogue désignée par l'ID du fournisseur, le type de produit, le code de produit et la série. La version majeure est comprise entre 0 et 31.

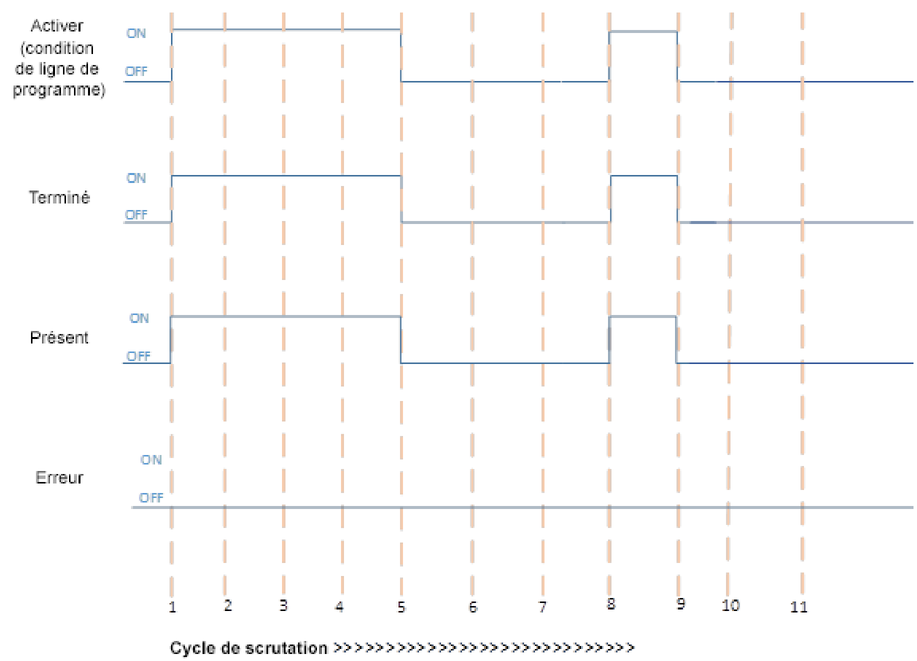
4	Série (4:0)	SÉRIE (4:0) désigne la série du produit. Ce champ indique la lettre de la série du numéro de référence Catalogue désignée par l'ID du fournisseur, le type de produit et le code de produit.

Série (4:0)	Série	Série (4:0)	Série	Série (4:0)	Série	Série (4:0)	Série
0	A	8	I	16	Q	24	Y
1	B	9	J	17	R	25	Z
2	C	10	K	18	S	26	AA
3	D	11	L	19	T	27	AB
4	E	12	M	20	U	28	CA
5	F	13	N	21	V	29	AD
6	G	14	O	22	W	30	AE
7	H	15	P	23	X	31	>AE

Diagrammes de temporisation d'instructions MODULE_INFO

Les exemples de diagrammes de temporisation suivants décrivent les scénarios d'exécution pour l'instruction MODULE_INFO sur la page 362.

Exécution réussie lorsqu'un module est physiquement présent

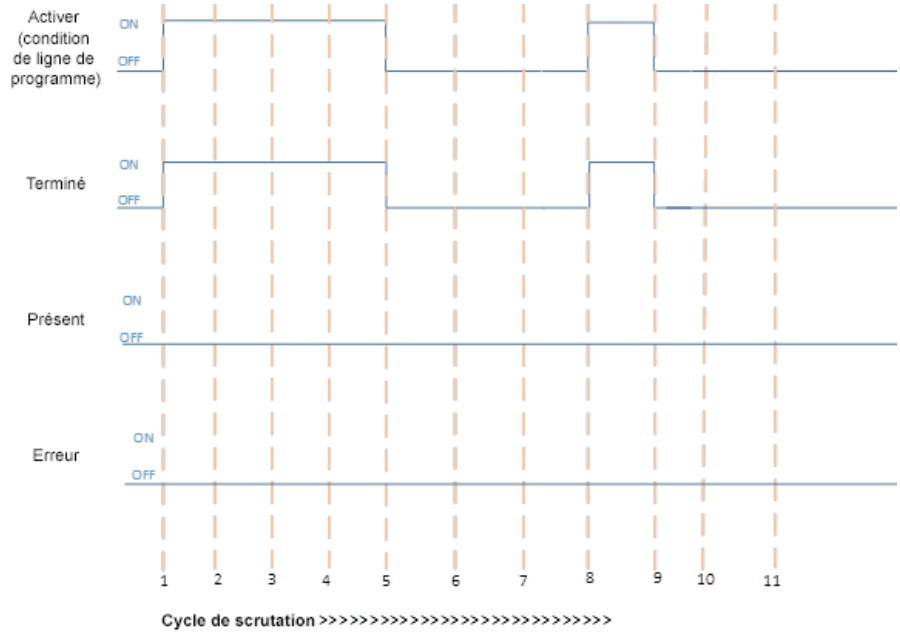


Utilisez ce tableau pour déterminer les valeurs du paramètre MODULE_INFO pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Le Type de module et l'ID d'emplacement sont valides. Un module physique est présent. • Les bits de sortie Terminé et Présent sont VRAI.

Cycle de scrutation	Description
	• Le bit de sortie Erreur est FAUX. • Mettez à jour les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module en conséquence.
2,3,4	Aucun changement de la condition de ligne.
5. 9	La condition de ligne de programme devient FAUX lorsque le bit Activer est FAUX. Tous les paramètres de sortie sont effacés.
6, 7, 10, 11	Aucun changement de la condition de ligne. Le bit Activer est FAUX. Tous les paramètres de sortie sont effacés.

Exécution réussie lorsque le module n'est pas physiquement présent

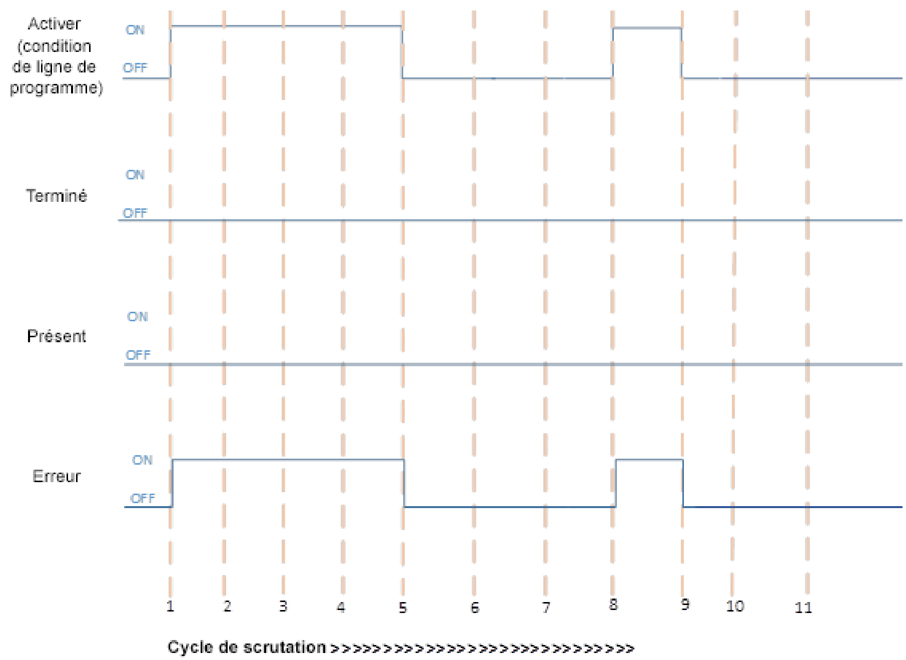


Utilisez ce tableau pour déterminer les valeurs du paramètre MODULE_INFO pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Le Type de module et l'ID d'emplacement sont valides. Aucun module physique n'est présent. • Le bit de sortie Terminé est VRAI. • Les bits de sortie Erreur et Présent sont FAUX. • Mettez à jour les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module en conséquence.
2,3,4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée sont valides, et aucun module physique n'est présent. • Mettez à jour les paramètres de sortie en conséquence.
5. 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Tous les paramètres de sortie sont effacés.

Cycle de scrutation	Description
6, 7, 10, 11	Aucun changement de la condition de ligne. Le bit Activer est FAUX. Tous les paramètres de sortie sont effacés.

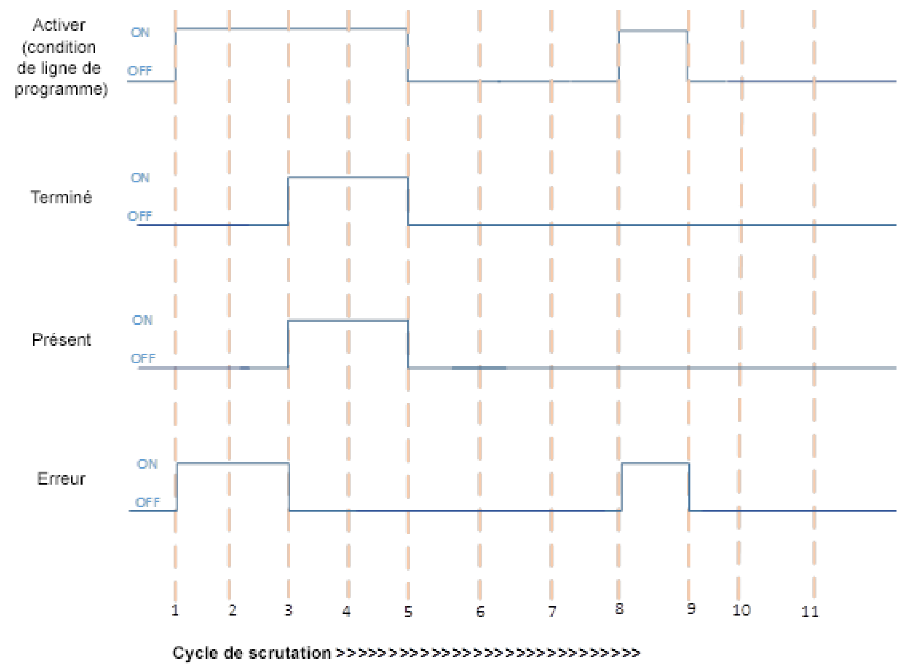
Exécution de MODULE_INFO avec erreur



Utilisez ce tableau pour déterminer les valeurs du paramètre MODULE_INFO pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Le Type de module et l'ID d'emplacement sont valides. Aucun module physique n'est présent. • Les bits de sortie Terminé et Présent sont VRAI. • Le bit de sortie Erreur est VRAI. • Les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module sont effacées.
2,3,4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Le Type de module, l'ID d'emplacement ou les deux paramètres ne sont pas valides. • Mettez à jour les paramètres de sortie en conséquence.
5. 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Tous les paramètres de sortie sont effacés.
6, 7, 10, 11	Aucun changement de la condition de ligne. Le bit Activer est FAUX. Tous les paramètres de sortie sont effacés.

Exécution de MODULE_INFO réussie sans erreur, pas de module physique

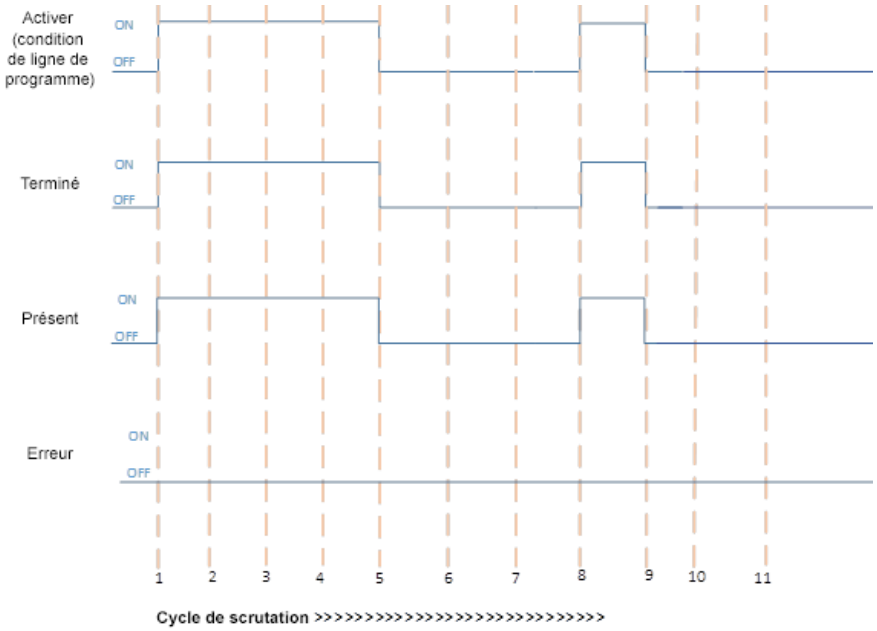


Utilisez ce tableau pour déterminer les valeurs du paramètre MODULE_INFO pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Le Type de module, l'ID d'emplacement ou les deux paramètres ne sont pas valides. Aucun module physique n'est présent. • Les bits de sortie Terminé et Présent sont FAUX. • Le bit de sortie Erreur est VRAI. • Les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module sont effacées.
2	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Le Type de module, l'ID d'emplacement ou les deux paramètres ne sont pas valides. • Mettez à jour les paramètres de sortie en conséquence.
3, 4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Le Type de module et l'ID d'emplacement sont tous les deux valides. Le module est physiquement présent. • Les bits de sortie Terminé et Présent sont VRAI. • Le bit de sortie Erreur est FAUX. • Mettez à jour les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module en conséquence.
5. 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Tous les paramètres de sortie sont effacés.

Cycle de scrutation	Description
6, 7, 10, 11	Aucun changement de la condition de ligne. Le bit Activer est FAUX. Tous les paramètres de sortie sont effacés.

Exécution de MODULE_INFO réussie avec erreur lorsque le module physique est présent



Utilisez ce tableau pour déterminer les valeurs du paramètre MODULE_INFO pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Le Type de module, l'ID d'emplacement ou les deux paramètres sont valides. Le module est physiquement présent. • Les bits de sortie Terminé et Présent sont VRAI. • Le bit de sortie Erreur est FAUX. • Mettez à jour les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module en conséquence.
2	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée du module sont valides, et le module est physiquement présent. • Mettez à jour les paramètres de sortie en conséquence.
3, 4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Le Type de module et l'ID d'emplacement sont tous les deux non valides. Le module est physiquement présent. • Les bits de sortie Terminé et Présent sont FAUX. • Le bit de sortie Erreur est VRAI et effacé.

Cycle de scrutation	Description
	• Mettez à jour les informations liées au module pour l'ID de module, l'ID de fournisseur, le type de produit, le code de produit et la version du module en conséquence.
5. 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Tous les paramètres de sortie sont effacés.
6, 7, 10, 11	Aucun changement de la condition de ligne. Le bit Activer est FAUX. Tous les paramètres de sortie sont effacés.

PLUGIN_INFO (informations sur le module enfichable)

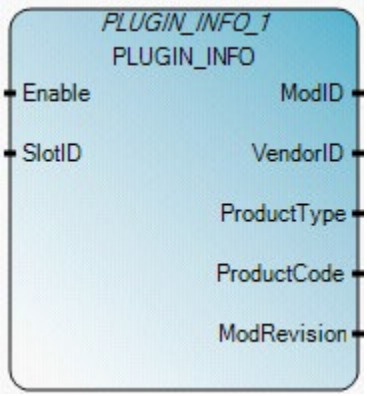
Lit les informations sur un module enfichable ou d'extension générique.

Détails de l'opération :

- Dans Connected Components Workbench 10 ou versions ultérieures, l'instruction PLUGIN_INFO peut lire les informations sur un module enfichable ou d'extension générique, à l'exception du module 2080-MEMBAK-RTC.
- Si aucun module enfichable ou d'extension générique n'est présent, toutes les valeurs sont redéfinies sur zéro (0).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.

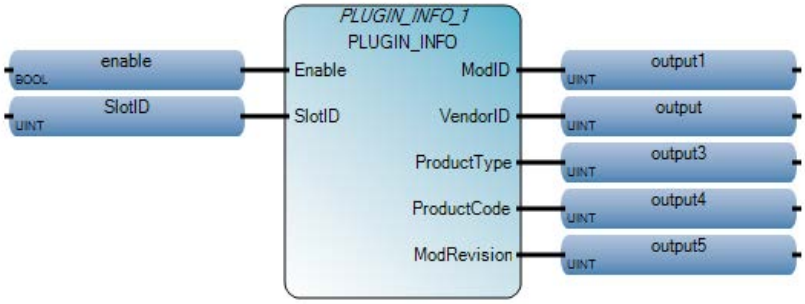


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

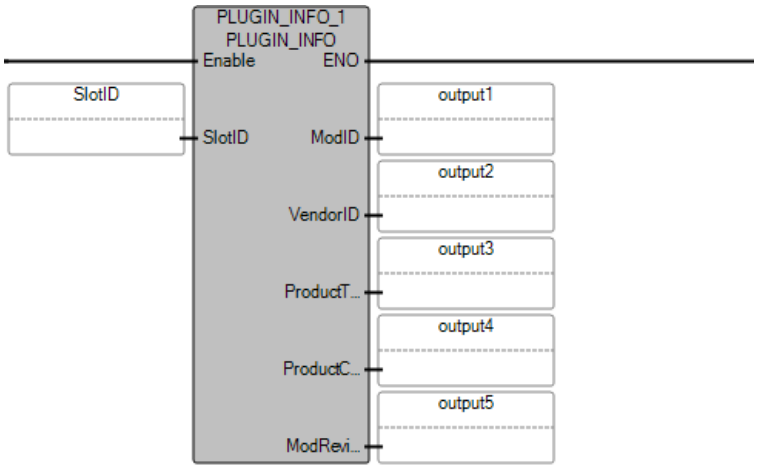
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de la lecture des informations sur le module enfichable ou d'extension. FAUX : le bloc d'instruction n'est pas exécuté. Toutes les valeurs de données de sortie sont réinitialisées sur 0.
SlotID	Entrée	UINT	Numéro d'emplacement du module enfichable : ID d'emplacement = 1,2,3,4,5 (En commençant par le premier emplacement à gauche = 1).

Paramètre	Type de paramètre	Type de données	Description
			Numéro d'emplacement du module d'extension : ID d'emplacement = 101, 102, 103, 104 (En commençant par le premier emplacement à gauche = 101).
ModID	Sortie	UINT	ID physique du module générique enfichable. • Si le module d'extension n'est pas pris en charge, ModID = 0xFFF • Si aucun module enfichable ou d'extension n'est présent, ModID = 0x0000
VendorID	Sortie	UINT	L'ID du fournisseur des modules enfichables ou d'extension génériques. Pour les produits Allen Bradley, l'ID du fournisseur = 1. Si aucun module enfichable ou d'extension n'est présent, VendorID = 0x0000
ProductType	Sortie	UINT	Type de produit correspondant au module enfichable ou d'extension générique. Si aucun module enfichable ou d'extension n'est présent, ProductType = 0x0000
ProductCode	Sortie	UINT	Code produit correspondant au module enfichable ou d'extension générique. Si aucun module enfichable ou d'extension n'est présent, ProductCode = 0x0000
ModRevision	Sortie	UINT	Informations relatives à la version du module enfichable ou d'extension générique. Si aucun module enfichable ou d'extension n'est présent, ModRevision = 0x0000
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais. Un schéma à relais ajoute automatiquement la sortie ENO.

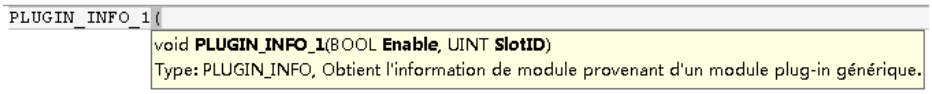
Exemple de diagramme de bloc de fonctions PLUGIN_INFO



Exemple de langage à relais PLUGIN_INFO

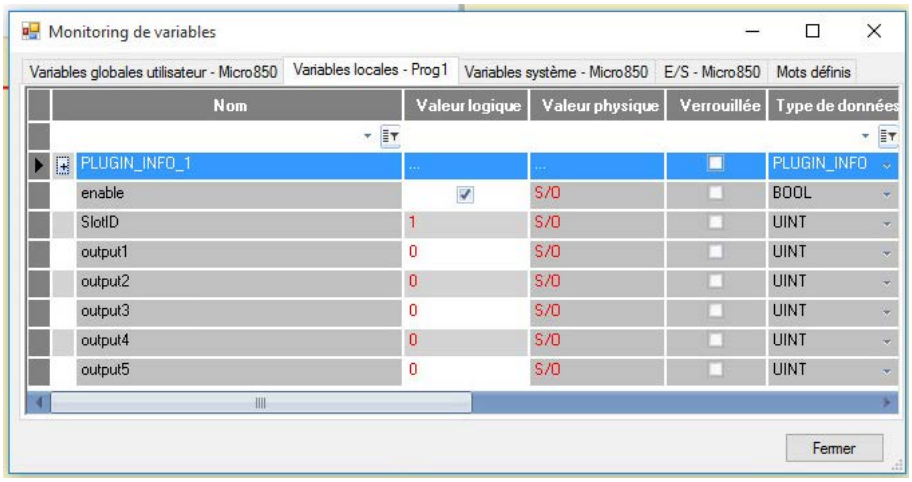


Exemple de texte structuré PLUGIN_INFO




```
1  SlotID := 1;
2  PLUGIN_INFO_1(enable, SlotID);
3  output1 := PLUGIN_INFO_1.ModID;
4  output2 := PLUGIN_INFO_1.VendorID;
5  output3 := PLUGIN_INFO_1.ProductType;
6  output4 := PLUGIN_INFO_1.ProductCode;
7  output5 := PLUGIN_INFO_1.ModRevision;
```

Résultats



PLUGIN_READ (lecture de module enfichable)

Lit les données à partir d'un module enfichable générique.

Détails de l'opération :

- N'importe quel module enfichable à l'exception des modules 2080-MEMBAK-RTC.
- En cas d'absence de module générique enfichable, toutes les valeurs sont définies sur zéro (0).

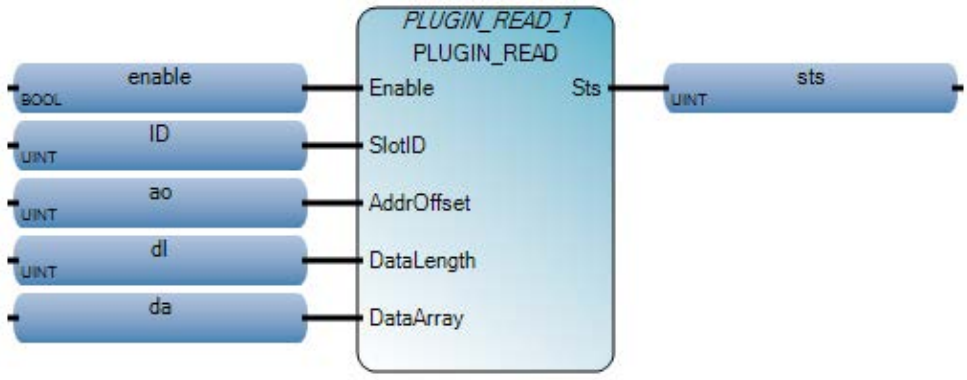
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.

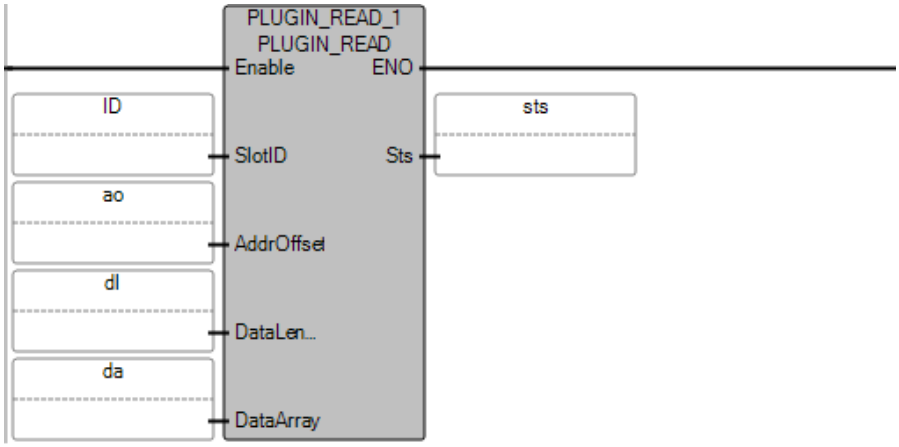
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de la lecture UPM. FAUX : aucune opération de lecture n'est réalisée et les données du tableau ne sont pas valides.
SlotID	Entrée	UINT	Numéro d'emplacement du module enfichable. ID d'emplacement = 1, 2, 3, 4, 5 (avec emplacement le plus à gauche = 1).
Offset	Entrée	UINT	Décalage de l'adresse des premières données à lire, calculé à partir du premier octet du module générique enfichable.
DataLength	Entrée	UINT	Nombre d'octets à lire
DataArray	Entrée	USINT	Tableau utilisé pour stocker les données lues à partir du module générique enfichable.
Sts	Sortie	UINT	Codes d'état pour PLUGIN_READ. Codes d'état (Sts) : • 0x00 - Bloc de fonctions non activé (aucune opération). • 0x01 - Réussite de l'opération du module enfichable. • 0x02 - Échec de l'opération du module enfichable en raison d'un ID d'emplacement non valide. • 0x03 - Échec de l'opération du module enfichable, car le module générique enfichable n'est pas valide. • 0x04 - Échec de l'opération du module enfichable en raison de données hors plage. • 0x05 - Échec de l'opération du module enfichable en raison d'une erreur de parité lors de l'accès aux données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

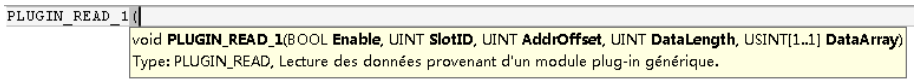
Exemple de diagramme de bloc de fonctions PLUGIN_READ



Exemple de langage à relais PLUGIN_READ



Exemple de texte structuré PLUGIN_READ




```
1  PLUGIN_READ_1(enable, ID, ao, dl, da);
2  sts := PLUGIN_READ_1.Sts;
```

PLUGIN_RESET (réinitialisation du module enfichable)

Réinitialise un matériel de module générique enfichable, quel qu'il soit, à l'exception des modules 2080-MEMBAK-RTC. Après la réinitialisation du matériel, le module générique enfichable est prêt à être configuré et exploité.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

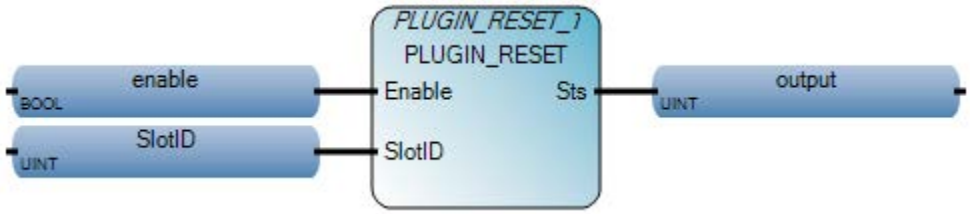
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

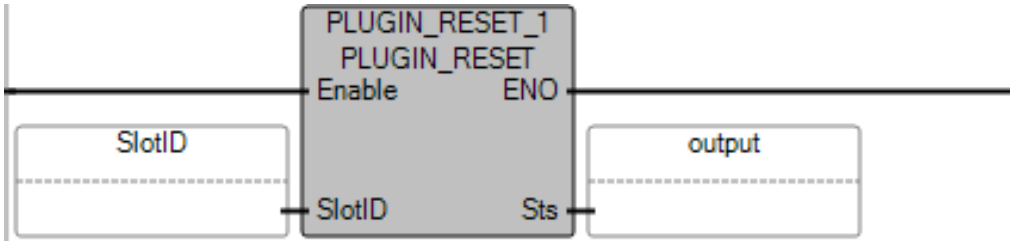
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de la réinitialisation du module enfichable. FAUX : aucune opération de réinitialisation.
SlotID	Entrée	UINT	Numéro d'emplacement du module enfichable. ID d'emplacement = 1, 2, 3, 4, 5 (avec emplacement le plus à gauche = 1).
Sts	Sortie	UINT	Codes d'état PLUGIN_RESET. Codes d'état (Sts) : • 0x00 - Bloc de fonctions non activé (aucune opération).

Paramètre	Type de paramètre	Type de données	Description
			• 0x01 - Réussite de l'opération du module enfichable. • 0x02 - Échec de l'opération du module enfichable en raison d'un ID d'emplacement non valide. • 0x03 - Échec de l'opération du module enfichable, car le module générique enfichable n'est pas valide. La configuration du module 2080-MOT-HSC est en mode d'instructions de comptage rapide. • 0x04 - Échec de l'opération du module enfichable en raison de données hors plage. • 0x05 - Échec de l'opération du module enfichable en raison d'une erreur de parité lors de l'accès aux données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

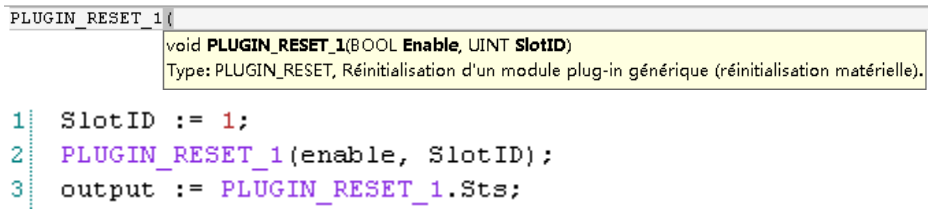
Exemple de Diagramme de Bloc Fonction PLUGIN_RESET



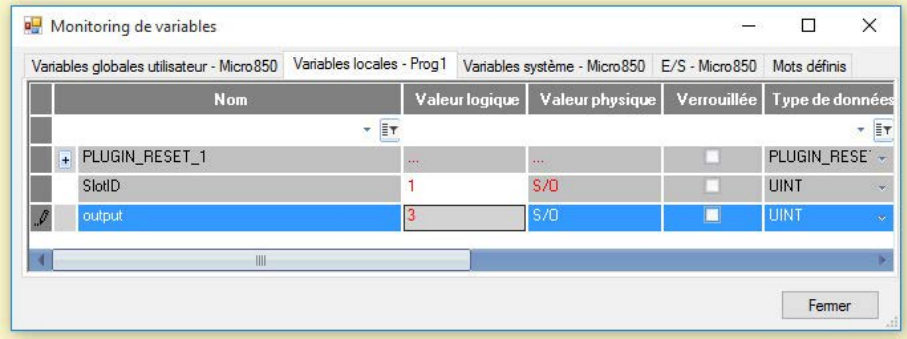
Exemple de langage à relais PLUGIN_RESET



Exemple de texte structuré PLUGIN_RESET



Résultats

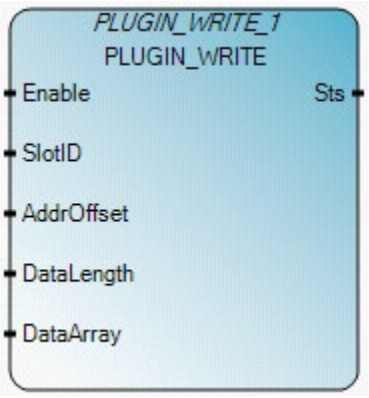


PLUGIN_WRITE (écriture vers un module enfichable)

Écrit un bloc de données vers un matériel de module générique enfichable, quel qu'il soit, à l'exception des modules 2080-MEMBAK-RTC.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.

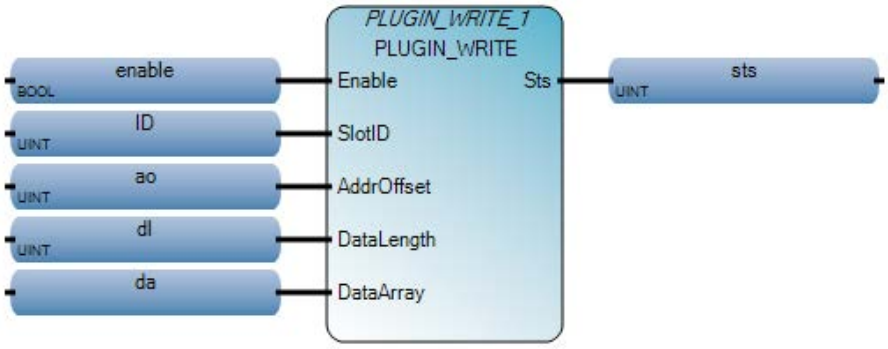


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

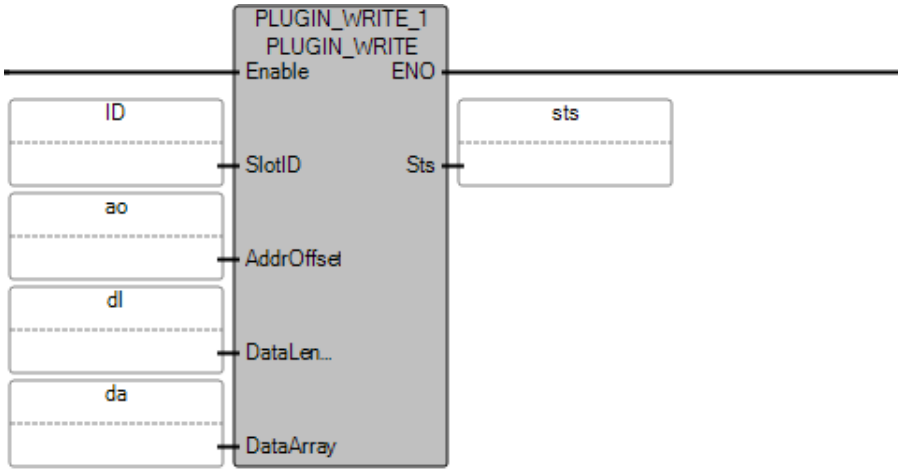
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de l'écriture vers le module enfichable. FAUX : aucune opération d'écriture.
SlotID	Entrée	UINT	Numéro d'emplacement du module enfichable. ID d'emplacement = 1, 2, 3, 4, 5 (avec emplacement le plus à gauche = 1).
AddrOffset	Entrée	UINT	Décalage de l'adresse des premières données à écrire, calculé à partir du premier octet du module générique enfichable.
DataLength	Entrée	UINT	Nombre d'octets à écrire
DataArray	Entrée	USINT	Données à écrire dans le module générique enfichable.
Sts	Sortie	UINT	Codes d'état PLUGIN_WRITE. Codes d'état (Sts) : • 0x00 - Bloc de fonctions non activé (aucune opération). • 0x01 - Réussite de l'opération du module enfichable. • 0x02 - Échec de l'opération du module enfichable en raison d'un ID d'emplacement non valide. • 0x03 - Échec de l'opération du module enfichable, car le module générique

Paramètre	Type de paramètre	Type de données	Description
			enfichable n'est pas valide. • 0x04 - Échec de l'opération du module enfichable en raison de données hors plage. • 0x05 - Échec de l'opération du module enfichable en raison d'une erreur de parité lors de l'accès aux données.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

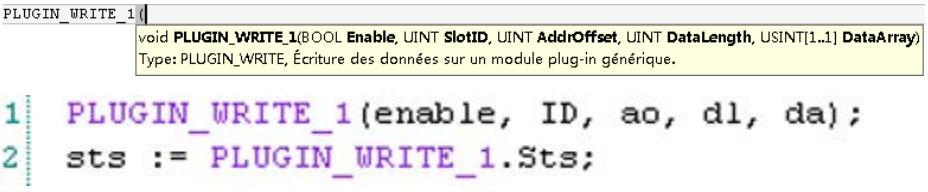
Exemple de diagramme de bloc de fonctions PLUGIN_WRITE



Exemple de langage à relais PLUGIN_WRITE



Exemple de texte structuré PLUGIN_WRITE

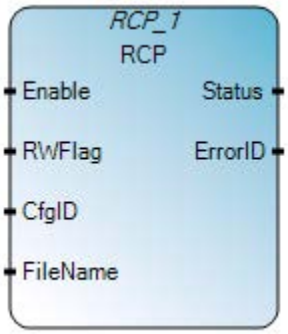


RCP (recette)

Permet de lire la valeur de données d'une variable dans le fichier de données de recette qui se trouve dans le dossier de fichiers de données de recette sur la carte SD et de mettre à jour la valeur dans le moteur d'exécution. Écrit les valeurs de la variable avec le moteur d'exécution dans un fichier de données de recette sur la carte SD.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

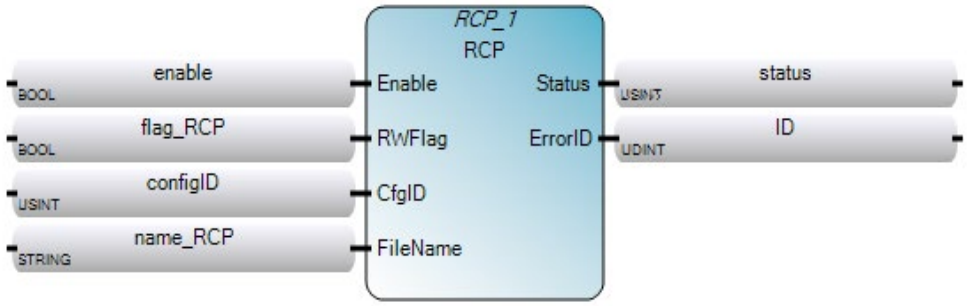
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction de lecture/écriture de recette. VRAI : front montant détecté, exécution de l'instruction de recette si l'opération précédente est terminée. FAUX : front montant non détecté, ne pas exécuter l'instruction de recette.
RWFlag	Entrée	BOOL	VRAI : RWFlag (opération d'écriture). La recette écrit les valeurs de la variable avec le moteur d'exécution dans un fichier de données de recette sur la carte SD. FAUX : RWFlag (opération de lecture). La recette lit les valeurs de la variable à partir de la carte SD et met à jour la valeur de variable correspondante sur le moteur d'exécution.
CfgID	Entrée	USINT	Numéro 1 à 10 d'ID de VA de configuration de recette.
FileName	Entrée	STRING	Nom de fichier de données de recette (30 caractères maximum).
Status	Sortie	USINT	État actuel du bloc d'instruction de recette. Codes d'état RCP : • 0 En attente • 1 En cours • 2 Réussite, terminé sans erreur. • 3 Erreur, terminé avec erreur.
ErrorID	Sortie	UDINT	Le code d'erreur RCP numérique. Les définitions sont définies dans les codes d'erreur RCP.

Codes d'erreur RCP

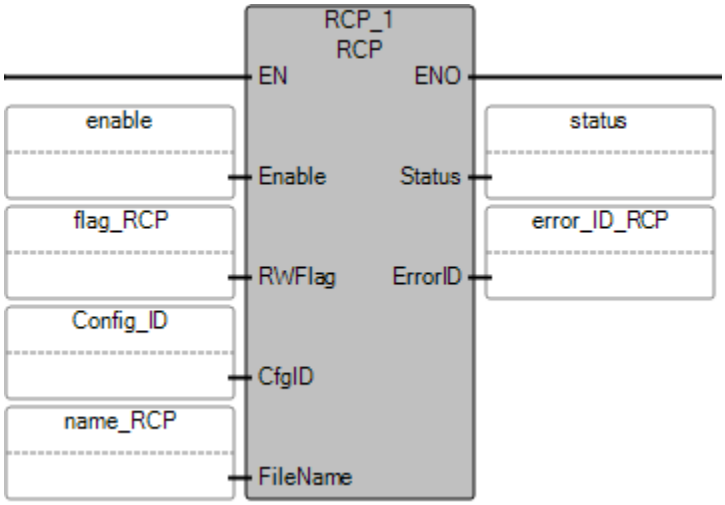
Code d'erreur	Nom de l'erreur
0	RCP_ERR_NONE
1	RCP_ERR_NO_SDCARD

Code d'erreur	Nom de l'erreur
2	RCP_ERR_DATAFILE_FULL
3	RCP_ERR_DATAFILE_ACCESS La carte SD est identifiée comme : • Cassée. • Pleine. • Lecture seule.
4	RCP_ERR_CFG_ABSENT
5	RCP_ERR_CFG_ID
6	RCP_ERR_RESOURCE_BUSY
7	RCP_ERR_CFG_FORMAT
8	RCP_ERR_RESERVED Réservé pour une future expansion possible.
9	RCP_ERR_UNKNOWN
10	RCP_ERR_DATAFILE_NAME
11	RCP_ERR_DATAFOLDER_INVALID
12	RCP_ERR_DATAFILE_ABSENT
13	RCP_ERR_DATAFILE_FORMAT
14	RCP_ERR_DATAFILE_SIZE Le fichier de données de recette est trop volumineux (> 4 ko).

Exemple de Diagramme de Bloc Fonction RCP



Exemple de logique à relais RCP



Exemple de texte structuré RCP

```
RCP_1(
```

void **RCP_1**(BOOL **Enable**, BOOL **RWFlag**, USINT **CfgID**, STRING **FileName**)
Type: RCP, Enregistre/Restaure une liste de données à/d'un fichier SD Card Recipe.

```
1  RCP_1 (EN, Enable, RWFlag, CfgID, FileName);
2  output := RCP_1.ENO
3  status := RCP_1.Status
4  error_ID_RCP := RCP_1.ErrorID
```

RTC_READ (lecture de l'horloge en temps réel)

Lit les informations du module d'horloge en temps réel (RTC).

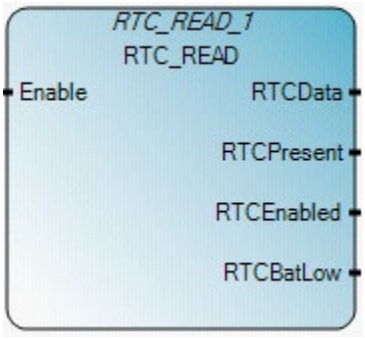
Détails de l'opération :

- Contrôleur Micro810 ou Micro820 avec RTC intégrée :
  - RTCBatLow est toujours défini sur zéro (0).
  - RTCEnabled est toujours défini sur un (1).
- Lorsque l'horloge en temps réel intégrée est déchargée ou que sa mémoire a été vidée en raison d'une panne électrique :
  - RTCData est défini sur 2000/1/1/0/0/0.
  - RTCEnabled est défini sur un (1).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de la lecture des informations RTC. FAUX : aucune opération de lecture n'est réalisée et les données RTC de sortie ne sont pas valides.
RTCData	Sortie	RTC	Information de données RTC: aa/mm/jj, hh/mm/ss, semaine. La sortie RTCData est définie à l'aide du type de données RTC.

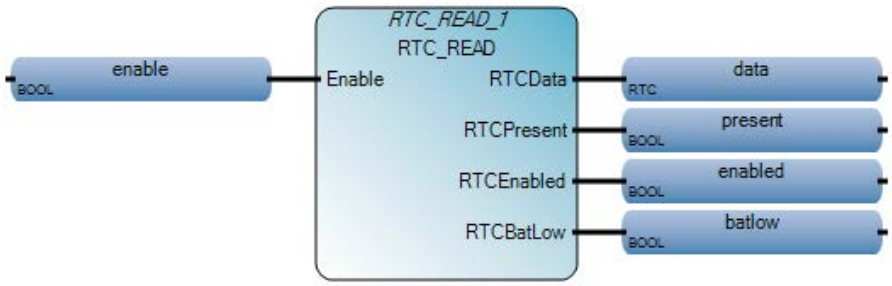
RTCPresent	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est branché. FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC n'est pas branché.
RTCEnabled	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est activé (temporisation). FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC est désactivé (aucune temporisation).
RTCBatLow	Sortie	BOOL	VRAI : le niveau de la batterie RTC est faible. FAUX : le niveau de la batterie RTC n'est pas faible.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Type de données RTC

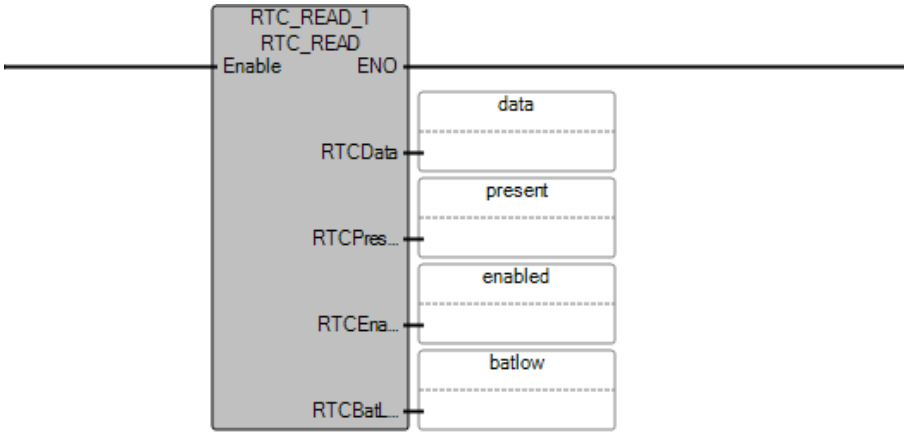
Ce tableau permet de déterminer les valeurs de paramètre pour le type de données RTC.

Paramètre	Type de données	Description
Year	UINT	Paramétrage de l'année pour l'horloge RTC. Valeur 16 bits ; les valeurs correctes sont comprises entre 2000 (01 Jan, 00:00:00) et 2098 (31 Déc, 23:59:59)
Month	UINT	Paramétrage du mois pour l'horloge RTC.
Day	UINT	Paramétrage du jour pour l'horloge RTC.
Hour	UINT	Paramétrage de l'heure pour l'horloge RTC.
Minute	UINT	Paramétrage de la minute pour l'horloge RTC.
Second	UINT	Paramétrage de la seconde pour l'horloge RTC.
DayOfWeek	UINT	Paramétrage du jour de la semaine pour l'horloge RTC. Ce paramètre n'est pas pris en compte pour RTC_SET.

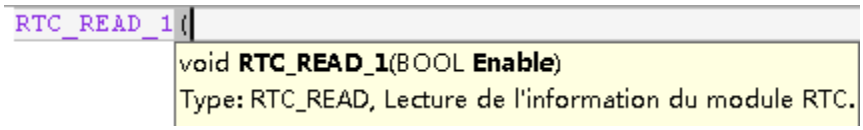
Exemple de diagramme de bloc de fonctions RTC_READ



Exemple de langage à relais RTC_READ



Exemple de texte structuré RTC_READ




```
1  RTC_READ_1(enable);
2  data := RTC_READ_1.RTCData;
3  present := RTC_READ_1.RTCPresent;
4  enabled := RTC_READ_1.RTCEnabled;
5  batlow := RTC_READ_1.RTCBatLow;
```

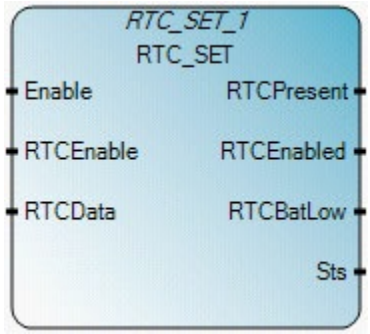
RTC_SET (définir l'horloge en temps réel)

Définit les données RTC (horloge en temps réel) pour les informations du module RTC.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description

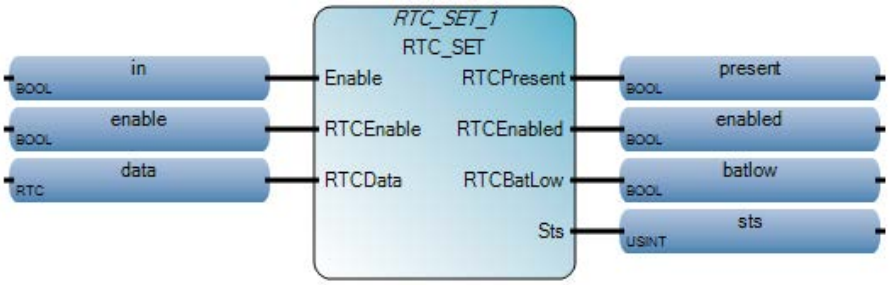
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de RTC_SET avec les informations RTC de l'entrée. En général, s'exécute uniquement pour 1 scrutation de programme lors de la mise à jour de RTC. FAUX : n'exécute pas RTC_SET. Défini sur FAUX pour exécuter RTC normalement.
RTCEnable	Entrée	BOOL	VRAI : activation de l'horloge RTC avec les données RTC spécifiées. FALSE - Désactivation de l'horloge RTC.
RTCData	Entrée	RTC	Informations sur les données RTC : aa/mm/jj, hh/mm/ss, semaine, tel que défini dans le type de données RTC. RTCData est ignoré lorsque RTCEnable = 0.
RTCPresent	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est branché. FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC n'est pas branché.
RTCEnabled	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est activé (temporisation). FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC est désactivé (aucune temporisation).
RTCBatLow	Sortie	BOOL	VRAI : le niveau de la batterie RTC est faible. FAUX : le niveau de la batterie RTC n'est pas faible.
Sts	Sortie	USINT	État de l'opération de lecture. Valeurs d'état RTC_Set (Sts) : • 0x00 - Bloc de fonctions non activé (aucune opération). • 0x01 - Réussite de l'opération de la définition RTC. • 0x02 - Échec de l'opération de la définition RTC.

Type de données RTC

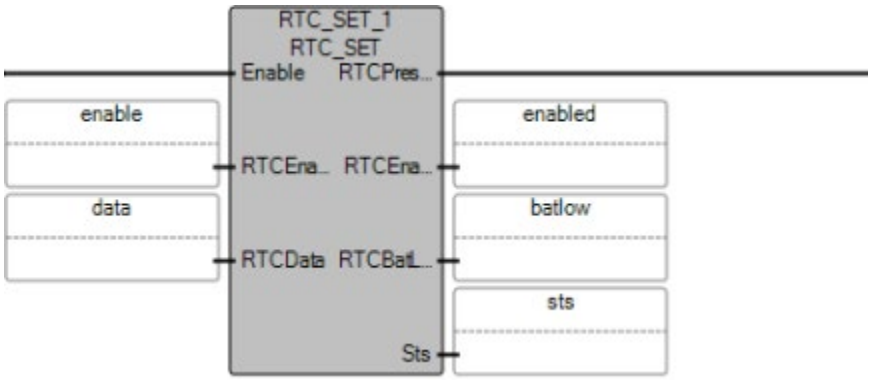
Ce tableau permet de déterminer les valeurs de paramètre pour le type de données RTC.

Paramètre	Type de données	Description
Year	UINT	Paramétrage de l'année pour l'horloge RTC. Valeur 16 bits ; les valeurs correctes sont comprises entre 2000 (01 Jan, 00:00:00) et 2098 (31 Déc, 23:59:59)
Month	UINT	Paramétrage du mois pour l'horloge RTC.
Day	UINT	Paramétrage du jour pour l'horloge RTC.
Hour	UINT	Paramétrage de l'heure pour l'horloge RTC.
Minute	UINT	Paramétrage de la minute pour l'horloge RTC.
Second	UINT	Paramétrage de la seconde pour l'horloge RTC.
DayOfWeek	UINT	Paramétrage du jour de la semaine pour l'horloge RTC. Ce paramètre n'est pas pris en compte pour RTC_SET.

Exemple de diagramme de bloc de fonctions RTC_SET



Exemple de langage à relais RTC_SET



Exemple de texte structuré RTC_SET

```
RTC_SET_1(
```

void **RTC_SET_1**(BOOL **Enable**, BOOL **RTCEnable**, RTC **RTCData**)
Type: RTC_SET, Fixe les données RTC au module RTC.

```
1  RTC_SET_1(in, enable, data);
2  present := RTC_SET_1.RTCPresent;
3  enabled := RTC_SET_1.RTCEnabled;
4  batlow := RTC_SET_1.RTCBatLow;
5  sts := RTC_SET_1.Sts;
```

SYS_INFO

Lit le bloc de données d'état pour le contrôleur Micro800.

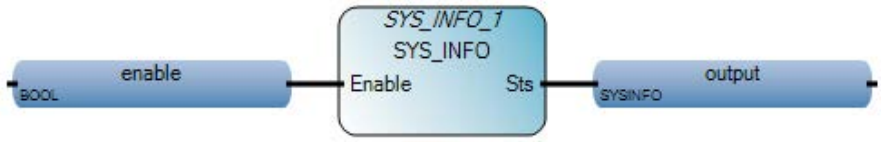
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

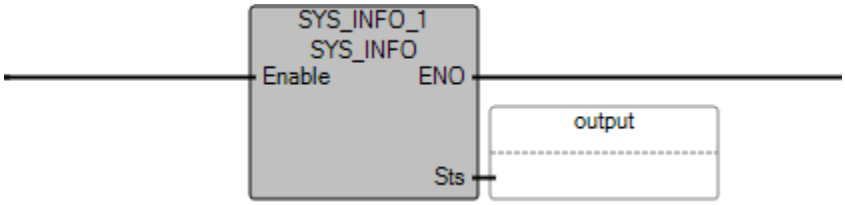
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de l'opération de lecture. FAUX : n'exécute pas la fonction.
Sts	Sortie	SYSINFO	Bloc de données correspondant au statut du système. La sortie Sts est définie dans le type de données SYS_INFO sur la page 389.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions SYS_INFO



Exemple de langage à relais SYS_INFO



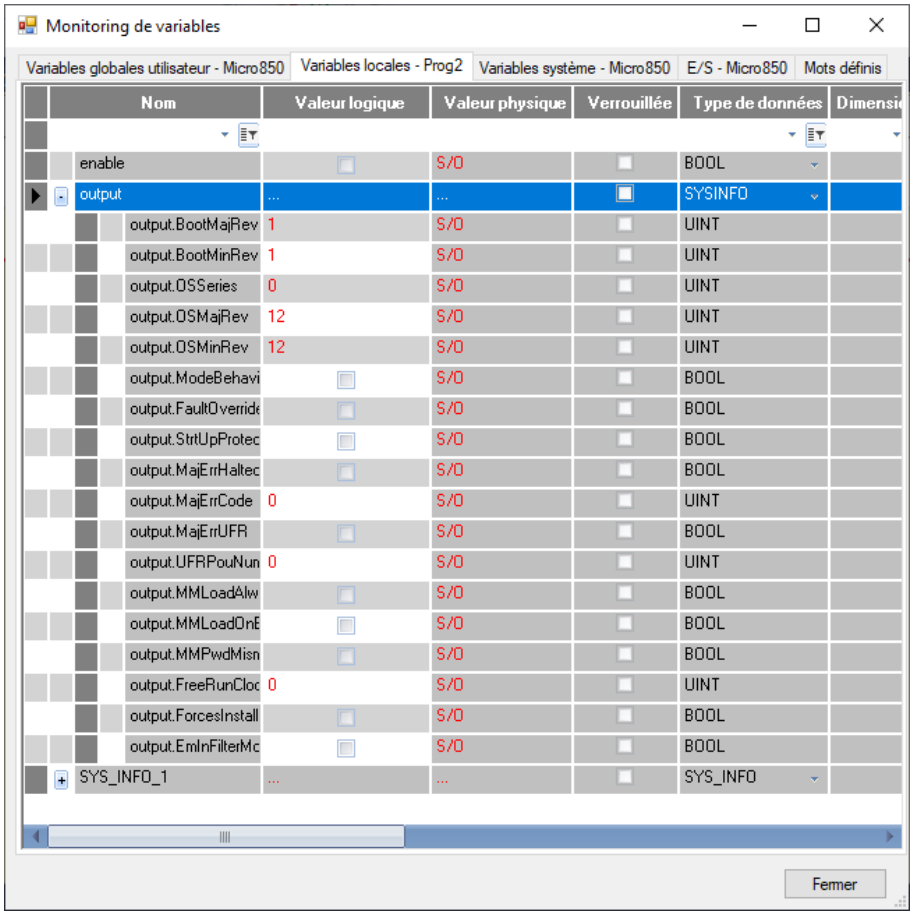
Exemple de texte structuré SYS_INFO

```
SYS_INFO_1(
```

void **SYS_INFO_1**(BOOL **Enable**)
Type: SYS_INFO, Lecture du statut du système Micro800.

```
1  SYS_INFO_1(enable);
2  output := SYS_INFO_1.Sts;
```

Résultats



Type de données SYS_INFO

Le tableau suivant décrit le type de données SYSINFO.

Paramètre	Type de données	Description
BootMajRev	UINT	Révision majeure du démarrage.
BootMinRev	UINT	Révision mineure du démarrage.
Operating System Series	UINT	Système d'exploitation - série : 0 indique un appareil de série A 1 indique un appareil de série B
OSMajRev	UINT	Révision majeure du système d'exploitation.
OSMinRev	UINT	Révision mineure du système d'exploitation.
ModeBehaviour	BOOL	Comportement du mode (TRUE : passage à RUN lors de la mise sous tension).
FaultOverride	BOOL	Annulation de défaut (TRUE : annulation de l'erreur lors de la mise sous tension).
StrtUpProtect	BOOL	Protection au démarrage (TRUE : exécution du programme de protection au démarrage lors de la mise sous tension). Pour version ultérieure.
MajErrHalted	BOOL	Erreur majeure stoppée (TRUE : erreur majeure stoppée).
MajErrCode	UINT	Code de défaut majeur.
MajErrUFR	BOOL	Défaut majeur lors de la routine de défaut utilisateur. Pour version ultérieure.
UFRPouNum	UINT	Numéro du programme de la routine de défaut utilisateur.

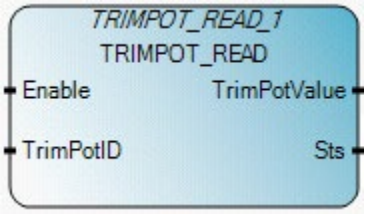
Paramètre	Type de données	Description
MMLoadAlways	BOOL	Restauration du module mémoire avec les données du contrôleur à chaque mise sous tension (TRUE : restauration).
MMLoadOnError	BOOL	Restauration du module mémoire avec les données du contrôleur si une erreur survient lors de la mise sous tension (TRUE : restauration).
MMPwdMismatch	BOOL	Discordance du mot de passe du module mémoire (TRUE : discordance entre le mot de passe du contrôleur et celui du module mémoire).
FreeRunClock	UINT	Horloge d'exécution libre qui s'incrémente toutes les 100 microsecondes de 0 à 65 535 avant de revenir à 0. Vous pouvez utiliser l'horloge, accessible globalement, si vous avez besoin d'une résolution supérieure au temporisateur standard de 1 milliseconde. Prise en charge uniquement par les contrôleurs Micro830 et Micro850. La valeur pour les contrôleurs Micro810 reste 0.
ForcesInstall	BOOL	Forçage activé (TRUE : Activé).
EMINFilterMod	BOOL	Filtre intégré modifié (TRUE : modifié).

TRIMPOT_READ (lecture du potentiomètre d'ajustement)

Lit la valeur de potentiomètre d'ajustement à partir d'un potentiomètre d'ajustement spécifique.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

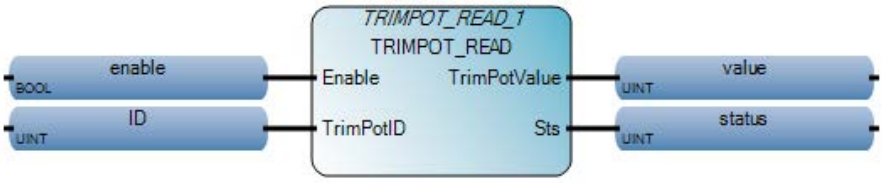
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870.



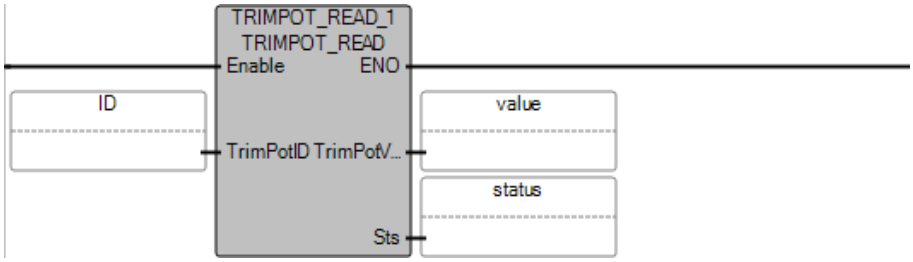
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de la lecture du potentiomètre d'ajustement. FAUX : aucune opération de lecture n'est réalisée, et la valeur de potentiomètre d'ajustement de sortie n'est pas valide.
TrimPotID	Entrée	UINT	ID du potentiomètre d'ajustement à lire. TrimPotID est défini dans les définitions d'ID de potentiomètre d'ajustement sur la page 391.
TrimPotValue	Sortie	UINT	Valeur courante de Trimpot.
Sts	Sortie	UINT	Statut de l'opération de lecture du potentiomètre d'ajustement. Codes d'état du potentiomètre d'ajustement (Sts) : • 0x00 - Bloc de fonctions non activé (aucune opération de lecture/écriture). • 0x01 - Réussite de l'opération de lecture/écriture. • 0x02 - Échec de l'opération de lecture/écriture en raison d'un ID de potentiomètre d'ajustement non valide. • 0x03 - Échec de l'opération d'écriture en raison d'une valeur hors plage.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions TRIMPOT



Exemple de langage à relais TRIMPOT



Exemple de texte structuré TRIMPOT

```
TRIMPOT_READ_1(
```

void **TRIMPOT_READ_1**(BOOL **Enable**, UINT **TrimPotID**)
Type: TRIMPOT_READ, Lecture de la valeur Trimpot provenant d'un Trimpot spécifique.

```
1  TRIMPOT_READ_1(enable, ID);
2  value := TRIMPOT_READ_1.TrimPotValue;
3  status := TRIMPOT_READ_1.Sts;
```

Définition de l'ID de potentiomètre d'ajustement

Le tableau suivant décrit la définition de l'ID de potentiomètre d'ajustement utilisée dans l'instruction TRIMPOT_read sur la page 390.

Sélection de la sortie	Bit	Description
Définition de l'ID de potentiomètre d'ajustement	15 - 13	Type de module correspondant au potentiomètre d'ajustement : • 0x00 - Intégré. • 0x01 - Expansion. • 0x02 - Port enfichable.
	12 - 8	ID d'emplacement du module : • 0x00 - Intégré. • 0x01-0x1F - ID du module d'expansion. • 0x01-0x05 - ID du port enfichable.
	7 - 4	Type de potentiomètre d'ajustement : • 0x00 - Réservé. • 0x01 - Potentiomètre d'ajustement numérique de type 1 (Module LCD 1). • 0x02 - Potentiomètre d'ajustement mécanique (Module 1).

Sélection de la sortie	Bit	Description
	3 - 0	ID de potentiomètre d'ajustement dans le module : • 0x00-0x0F - Intégré. • 0x00-0x07 - ID du potentiomètre d'ajustement pour l'expansion. • 0x00-0x07 - ID du potentiomètre d'ajustement pour le port enfichable. L'ID de potentiomètre d'ajustement commence à 0.

Instructions d'interruption

Utilisez les instructions d'interruption par l'utilisateur pour signaler au processeur qu'un événement doit être pris en compte. Généralement, le signal d'interruption sert lors des situations à haute priorité, exigeant une interruption du code que le processeur est en train d'exécuter.

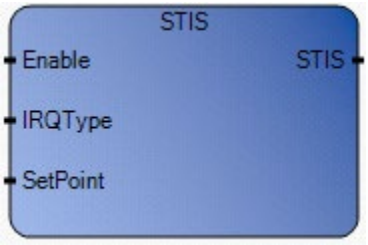
Fonction	Description
STIS sur la page 393	Démarre le temporisateur STI à partir du programme de contrôle plutôt que manuellement.
UIC sur la page 395	Efface le bit de perte de l'interruption par l'utilisateur sélectionnée.
UID sur la page 396	Désactive une interruption par l'utilisateur spécifique.
UIE sur la page 398	Active une entrée utilisateur spécifique.
UIF sur la page 400	Vide ou supprime une entrée utilisateur en attente.

STIS (sélectionner un démarrage temporisé)

Démarre le temporisateur STI (programmable en Interruption temporisée) à partir du programme de contrôle plutôt qu'automatiquement.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

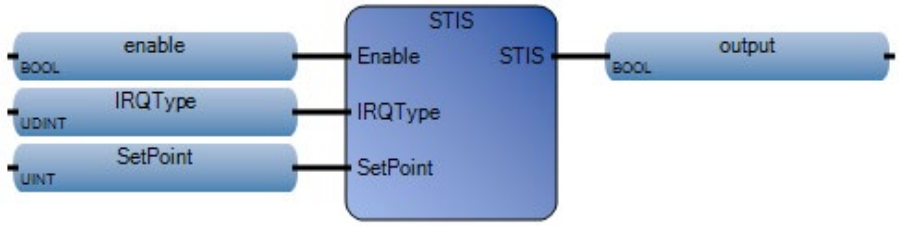


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

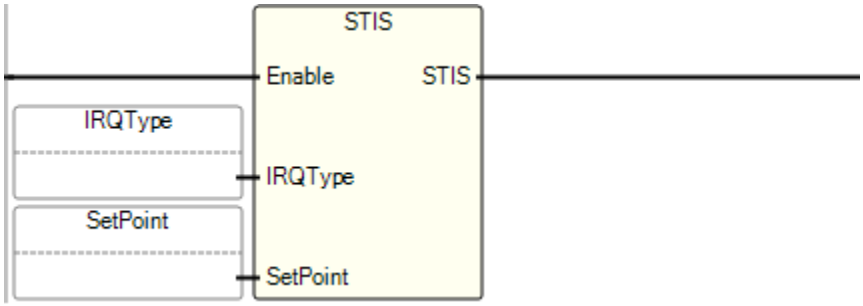
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : démarre le temporisateur STI à partir du programme de contrôle. FAUX : n'exécute pas la fonction.
IRQType	Entrée	UDINT	Utilise les mots STI définis. - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3
SetPoint	Entrée	UINT	Délai d'expiration (en ms) avant l'exécution du paramètre programmable en

Paramètre	Type de paramètre	Type de données	Description
			Interruption temporisée (STI). Une valeur de 0 désactive la fonction STIS. Une valeur comprise entre 1 et 65 535 active la fonction STIS.
STIS	Sortie	BOOL	Statut de la ligne (identique au paramètre Enable).

Exemple de Diagramme de Bloc Fonction STIS



Exemple de logique à relais STIS



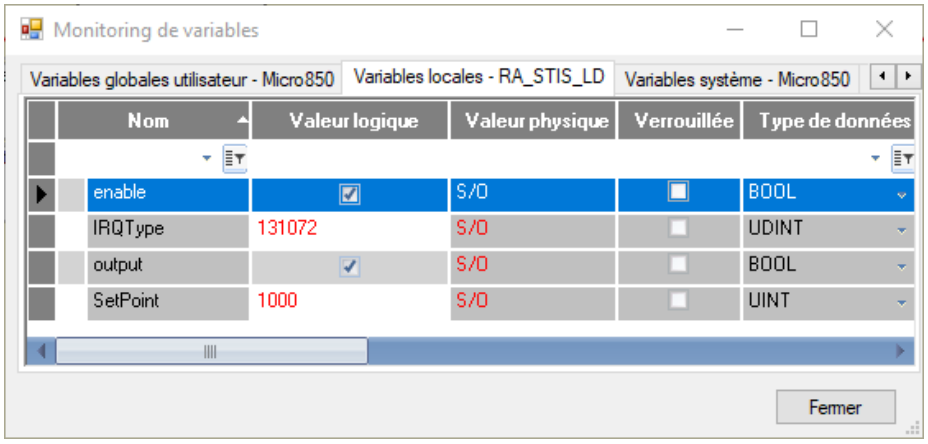
Exemple de texte structuré STIS

```
1  enable := TRUE;
2  IRQType := IRQ_STI2;
3  SetPoint := 1000;
4  output := STIS(enable, IRQType, SetPoint);
```

(* Équivalence ST : *)

TESTOUTPUT := STIS(TESTENABLE, 2, 1000) ;

Résultats



UIC (effacer le bit de perte d'interruption)

Efface le bit de perte de l'interruption par l'utilisateur sélectionnée.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

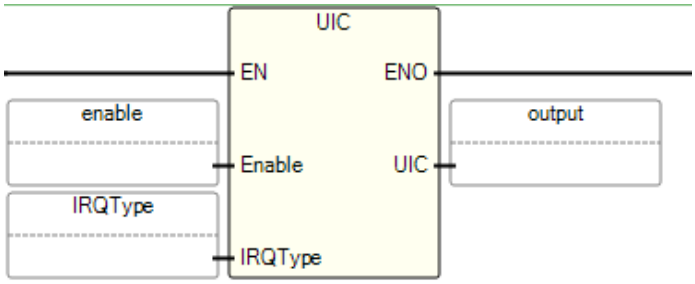
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description	
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : démarre l'opération de suppression de bit. FAUX : n'exécute pas la fonction.	
IRQType	Entrée	UDINT	Utilise les mots STI définis. - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIC	Sortie	BOOL	Statut de la ligne (identique au paramètre Enable).	

Exemple de Diagramme de Bloc Fonction UIC

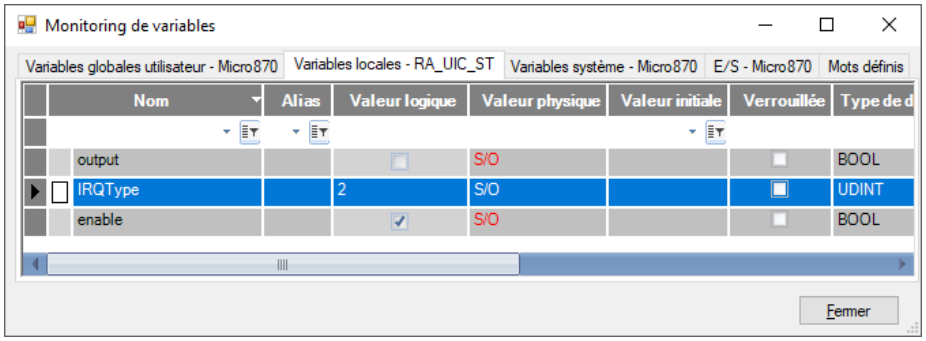
Exemple de logique à relais UIC



Exemple de texte structuré UIC

```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIC (enable, IRQType);
```

Résultats



UID (désactiver l'interruption)

Désactive une interruption par l'utilisateur spécifique.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

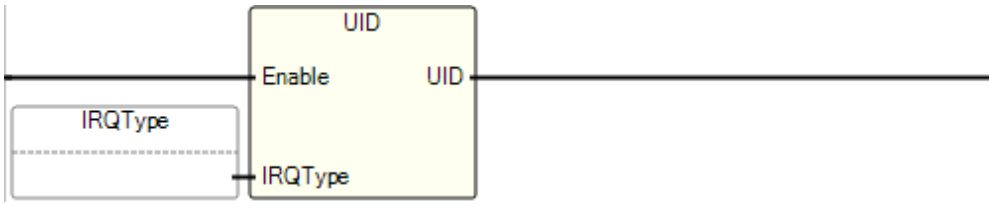
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

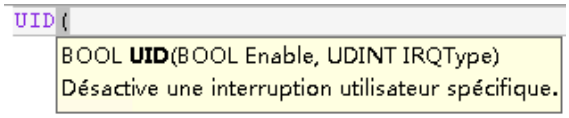
Paramètre	Type de paramètre	Type de données	Description	
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : démarre l'opération de désactivation de bit. FAUX : n'exécute pas la fonction.	
IRQType	Entrée	UDINT	Utilise les mots STI définis. - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UID	Sortie	BOOL	Statut de la ligne (identique au paramètre Enable).	

Exemple de Diagramme de Bloc Fonction UID

Exemple de logique à relais UID



Exemple de texte structuré UID

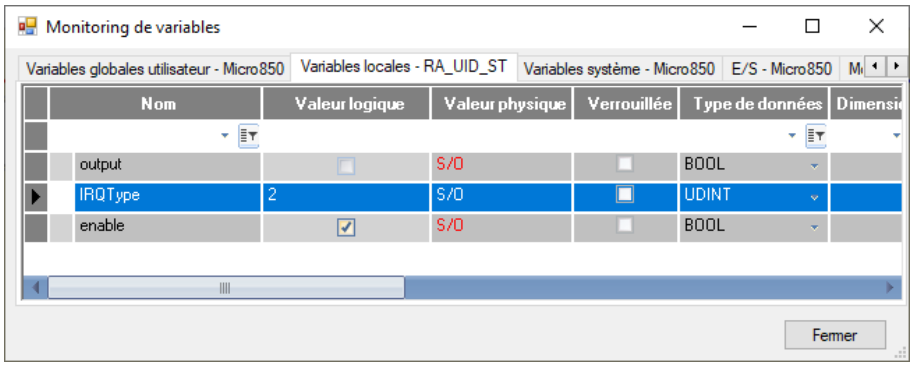



```
1  enable := TRUE;
2  IRQType := 2;
3  output := UID(enable, IRQType);
```

(* Équivalence ST : *)

TESTOUTPUT := UID(TESTENABLE, 2) ;

Résultats



UIE (activer l'interruption)

Active une interruption par l'utilisateur spécifique.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

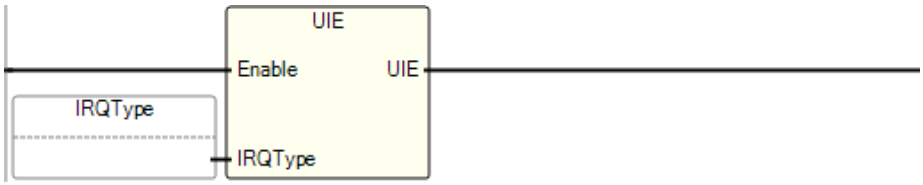
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description	
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : démarre l'opération d'activation. FAUX : n'exécute pas la fonction.	
IRQType	Entrée	UDINT	Utilise les mots STI définis. - IRQ_EII0 - IRQ_EII1	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5

Paramètre	Type de paramètre	Type de données	Description	
			- IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIE	Sortie	BOOL	Statut de la ligne (identique au paramètre Enable).	

Exemple de Diagramme de Bloc Fonction UIE

Exemple de logique à relais UIE



Exemple de texte structuré UIE

```
UIE(
```

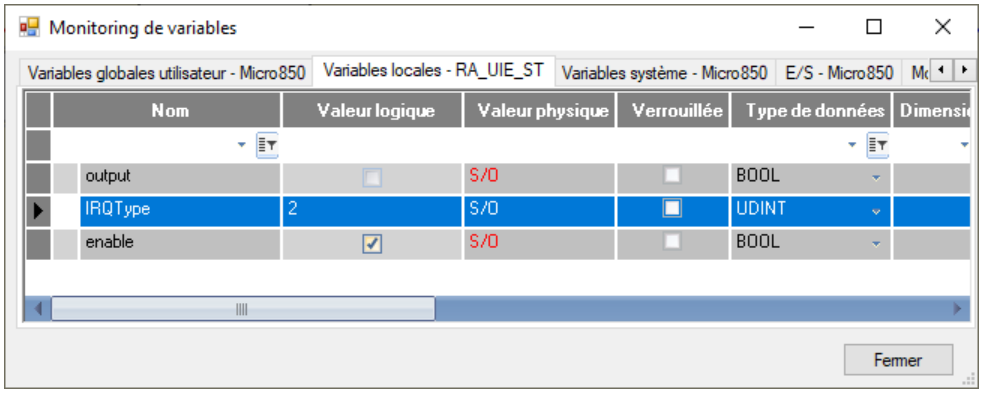
BOOL **UIE**(BOOL Enable, UDINT IRQType)
Active une interruption utilisateur spécifique.

```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIE(enable, IRQType);
```

(* Équivalence ST : *)

TESTOUTPUT := UIE(TESTENABLE, 2) ;

Résultats



UIF (vider une interruption imminente)

Vide ou supprime une interruption par l'utilisateur en attente.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

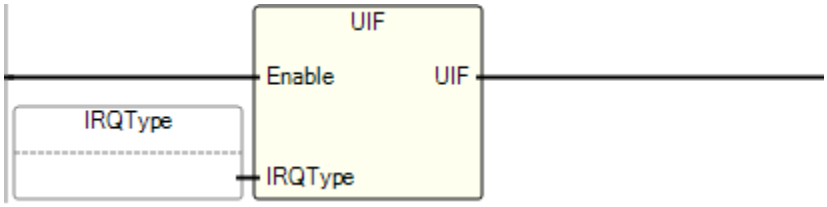
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

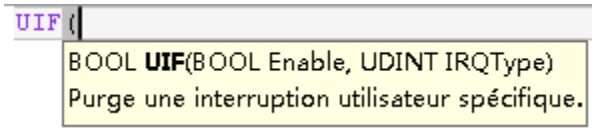
Paramètre	Type de paramètre	Type de données	Description	
Enable	Entrée	BOOL	Activation de l'instruction. VRAI : démarre l'opération UIF. FAUX : n'exécute pas la fonction.	
IRQType	Entrée	UDINT	Utilise les mots STI définis. - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIF	Sortie	BOOL	Statut de la ligne (identique au paramètre Enable).	

Exemple de Diagramme de Bloc Fonction UIF

Exemple de logique à relais UIF



Exemple de texte structuré UIF

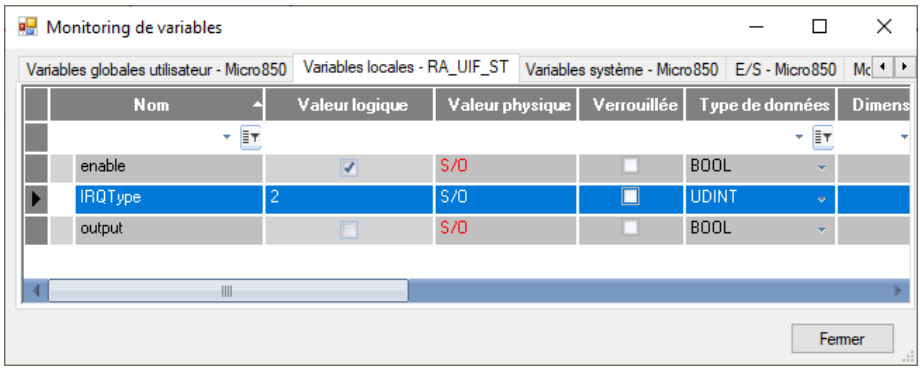



```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIF(enable, IRQType);
```

(* Équivalence ST : *)

TESTOUTPUT := UIF(TESTENABLE, 2) ;

Résultats



Instructions de contrôle de mouvement

Utilisez les instructions du contrôle des mouvements pour programmer et concevoir le mouvement d'un axe spécifique. Le contrôle des mouvements nécessite Connected Components Workbench 2.0 ou version ultérieure.

Conseil :

- Les blocs de fonction administratifs prennent en charge le mouvement PTO et le mouvement de retour codeur d'axe.
- Les instructions du contrôle des mouvements qui prennent en charge un axe FBAxis sont : MC_ReadActualPosition et MC_ReadActualVelocity.

Instruction	Description
Administratif	
MC_AbortTrigger sur la page 413	Annule les blocs de fonctions de mouvement connectés à des événements de déclenchement.
MC_Power sur la page 435	Contrôle la phase d'alimentation ON ou OFF.
MC_ReadAxisError sur la page 445	Lit les erreurs de l'axe non liées aux blocs d'instruction de contrôle des mouvements.
MC_ReadBoolParameter sur la page 449	Renvoie la valeur d'un paramètre de type BOOL spécifique à un fournisseur.
MC_ReadParameter sur la page 451	Renvoie la valeur d'un paramètre de type REAL spécifique à un fournisseur.
MC_ReadStatus sur la page 454	Renvoie l'état de l'axe en fonction du mouvement actuellement en cours.
MC_Reset sur la page 458	Fait passer l'axe de l'état Arrêt suite à une erreur à l'état Immobilisation en réinitialisant toutes les erreurs propres à l'axe.
MC_SetPosition sur la page 461	Bascule le système de coordonnées d'un axe par la manipulation de la position réelle.
MC_TouchProbe sur la page 466	Enregistre la position de l'axe lors du déclenchement d'un événement.
MC_WriteBoolParameter sur la page 470	Modifie la valeur d'un paramètre de type BOOL spécifique à un fournisseur.
MC_WriteParameter sur la page 473	Modifie la valeur d'un paramètre de type REAL spécifique à un fournisseur.
Mouvement	
MC_Halt sur la page 416	Commande un arrêt contrôlé du mouvement dans des conditions de fonctionnement normales.
MC_Home sur la page 419	Commande l'axe pour effectuer une séquence <*rechercher origine*>.
MC_MoveAbsolute sur la page 422	Commande un mouvement contrôlé vers une position absolue spécifiée.
MC_MoveRelative sur la page 426	Commande un mouvement contrôlé d'une distance spécifique relative à la position actuelle au moment de l'exécution.
MC_MoveVelocity sur la page 430	Commande un mouvement contrôlé sans fin à une vélocité spécifiée.
MC_ReadActualPosition sur la page 438	Renvoie la position réelle de l'axe retour codeur. Nécessite Connected Components Workbench 8 ou version ultérieure.
MC_ReadActualVelocity sur la page 442	Renvoie la vitesse réelle de l'axe retour codeur. Nécessite Connected Components Workbench 8 ou version ultérieure.
MC_Stop sur la page 463	Commande un arrêt contrôlé du mouvement et fait passer l'axe à l'état Arrêt.

Règles générales propres aux blocs fonctionnels de contrôle du mouvement

Les règles générales propres aux blocs de fonctions de contrôle des mouvements sur la page 403 du contrôleur programmable Micro800 reposent sur les spécifications de contrôle des mouvements PLCopen. Le tableau suivant décrit les règles générales sur l'interface des blocs fonctionnels de contrôle du mouvement.

Règle applicable à	Règle
Paramètres d'entrée	**Avec Execute** : les paramètres sont utilisés avec le front montant de l'entrée d'exécution. Pour modifier un paramètre, changez le ou les paramètres d'entrée et déclenchez de nouveau le mouvement. si l'instance d'un bloc fonctionnel reçoit une nouvelle commande Execute avant qu'elle ne se termine (dans le cadre d'une série de commandes sur la même instance), la nouvelle commande Execute n'est pas prise en compte, et l'exécution de l'instruction précédemment émise se poursuit. **Avec l'instruction Enable** : les paramètres sont utilisés avec le front montant de l'entrée d'activation et peuvent être constamment modifiés.
Paramètres d'entrée manquante	Les entrées manquantes sont capturées lors de la compilation de l'application utilisateur. Les erreurs propres aux entrées manquantes ne sont pas gérées au niveau du contrôleur.
Entrées dépassant les limites de l'application	Si un bloc de fonctions est commandé par des paramètres dont le résultat dépasse les limites de l'application, l'instance du bloc de fonctions génère une erreur. Dans ce cas, la sortie Error est activée et les informations propres à l'erreur sont indiquées par la sortie ErrorID. Dans la plupart des cas, le contrôleur reste en mode Exécuter et aucune erreur de mouvement n'est signalée en tant que défaut majeur du contrôleur.
Règles de signe pour les entrées	Les valeurs des entrées d'accélération, de décélération et de jerk sont toujours positives. Les entrées propres à la vitesse, à la position et à la distance, en revanche, peuvent correspondre à des valeurs positives et négatives.
Position et distance	La position est une valeur définie au sein d'un système de coordonnées. La distance, quant à elle, est une mesure relative exprimée en unités techniques. La distance correspond à la différence entre deux positions.
Entrée relative à la position/distance	Seuls les mouvements linéaires sont pris en charge sur les contrôleurs Micro800. Pour le bloc de fonctions MC_MoveAbsolute, l'entrée relative à la position correspond à l'emplacement absolu à transférer à l'axe. Pour le bloc de fonctions MC_MoveRelative, l'entrée relative à la distance correspond à l'emplacement relatif (en considérant que la position actuelle de l'axe est 0) par rapport à la position actuelle.
Entrée propre à la vitesse	La vitesse peut avoir une valeur signée mais vous pouvez aussi utiliser une entrée de direction pour définir le signe de la vitesse (vitesse négative x direction négative = vitesse positive). Pour des raisons de compatibilité, le paramètre « Direction » E s'applique à l'entrée et à la sortie propres à la vitesse.
Entrée propre à la direction	Pour les mouvements de distance (position) dont la position cible (absolue ou relative) est définie, la direction du mouvement est unique. L'entrée propre à la direction dans le cadre d'un mouvement de distance n'est pas prise en compte. Pour les mouvements de vitesse, la valeur correspondant à l'entrée Direction peut être 1 (direction positive), 0 (direction actuelle) ou -1 (direction négative). Pour toute autre valeur, seul le signe est pris en compte. Par exemple, -3 représente une direction négative, +2 une direction positive, etc. Pour des déplacements de vitesse (MC_MoveVelocity), le signe (vitesse x direction) détermine la direction réelle du mouvement si la valeur est différente de 0. Par exemple, si vélocité x direction = +300, la direction est positive.
Entrées propres à l'accélération, la décélération et le jerk	• Il est recommandé d'attribuer une valeur positive aux entrées propres à la décélération ou à l'accélération. Si les valeurs de décélération et d'accélération ne sont pas positives, le bloc fonctionnel signale une erreur (ID de l'erreur: MC_FB_ERR_RANGE). • L'entrée Jerk doit avoir une valeur non-négative. Si la valeur de Jerk est négative, le bloc de fonctions signale une erreur (ID de l'erreur: MC_FB_ERR_RANGE). • Si la valeur maximale de jerk est définie sur zéro, tous les paramètres de jerk pour le bloc fonctionnel de mouvement doivent être définis sur zéro, y compris le paramètre de jerk MC_Stop. Dans le cas contraire, le bloc fonctionnel signale une erreur (ID de l'erreur : MC_FB_ERR_RANGE). • Si le paramètre de jerk est défini à une valeur non nulle, un profil Courbe en S est généré, alors que si ce paramètre est défini à 0, le profil généré est de type trapézoïdal. • La configuration du jerk d'origine ne se limite pas à la configuration du jerk maximal. • Si le moteur de mouvement ne parvient pas à générer le profil de mouvement prescrit par les paramètres d'entrée dynamiques, le bloc fonctionnel signale une erreur (ID de l'erreur : MC_FB_ERR_PROFILE).

Règle applicable à	Règle
Exclusivité de la sortie	**Avec Execute** : Lorsque Execute vaut Vrai, l'une des sorties Busy, Done, Error ou CommandAborted doit être également à TRUE. Les sorties sont mutuellement exclusives : seule l'une d'entre elles peut être TRUE dans un bloc fonctionnel. Seule l'une des sorties Active, Error, Done et CommandAborted est définie à la fois. **Avec l'instruction Enable** : Les sorties Valid et Error sont mutuellement exclusives : seule l'une d'entre elles peut être TRUE dans un bloc fonctionnel.
État de la sortie	**Avec Execute** : Les sorties Done, Error, ErrorID et CommandAborted sont réinitialisées avec le front descendant de l'instruction Execute. Néanmoins, le front descendant de l'instruction Execute ne permet pas d'arrêter ni d'influencer l'exécution du bloc de fonctions réel. Même si l'instruction Execute est réinitialisée avant la fin du bloc de fonctions, les sorties correspondantes sont définies pour au moins un cycle. Si l'instance d'un bloc de fonctions reçoit une nouvelle commande Execute avant qu'elle ne se termine (dans le cadre d'une série de commandes sur la même instance par exemple), la nouvelle commande Execute n'est pas prise en compte et l'exécution de l'instruction précédemment émise se poursuit. **Avec l'instruction Enable** : les sorties Valid, Enabled, Busy, Error et ErrorID sont réinitialisées dès que possible avec le front descendant de l'instruction Enable.
Comportement de la sortie Done	La sortie Done est définie lorsque l'action commandée s'est terminée avec succès. Lorsque plusieurs blocs de fonctions sont actifs séquentiellement sur le même axe, la condition suivante s'applique : • Lorsqu'un mouvement sur un axe est interrompu par un autre mouvement sur le même axe avant qu'il ait atteint son objectif final, la sortie Done n'est pas définie sur le premier bloc fonctionnel.
Comportement de la sortie Busy	Chaque bloc de fonctions peut disposer d'une sortie Busy pour indiquer que son exécution n'est pas terminée (pour les blocs de fonctions disposant d'une entrée Execute) ou qu'elle ne fonctionne pas et que de nouvelles valeurs de sortie peuvent être générées (dans le cas d'une entrée Enable). L'entrée Busy est définie au niveau du front montant de l'instruction Execute et réinitialisée lorsque l'une des sorties Done, Aborted ou Error est définie. Le bloc de fonctions doit demeurer dans la boucle active du programme d'application au moins aussi longtemps que l'entrée Busy est VRAI, car il est possible que les sorties subissent des modifications. Les blocs de fonctions d'une même instance et qui sont Busy ne peuvent pas s'exécuter tant qu'elles sont Busy. Les blocs de fonctions d'instances différentes peuvent remplacer le bloc de fonction en cours d'exécution.
Comportement de la sortie CommandAborted	Cette sortie est définie lorsqu'un mouvement commandé est interrompu par une autre commande de mouvement. Le comportement de réinitialisation de la sortie CommandAborted est identique à celui de la sortie Done. Lors de la génération de la sortie CommandAborted, d'autres signaux de sortie, comme InVelocity, sont réinitialisés.
Sortie Active	Cette sortie est requise pour les blocs de fonctions en mémoire tampon. Elle est définie au moment où le bloc de fonctions prend le contrôle du mouvement de l'axe correspondant. Pour le mode hors mémoire tampon, les sorties Active et Busy peuvent avoir la même valeur.
Etats Enable et Valid	L'entrée Enable est couplée à une sortie Valid. L'entrée Enable dépend du niveau et la sortie Valid indique qu'un ensemble valide de sorties est disponible dans le bloc de fonctions. La sortie Valid est VRAI tant qu'une valeur de sortie valide est disponible et que l'entrée Enable est VRAI. Il est possible d'actualiser la valeur de sortie correspondante tant que l'entrée Enable est VRAI. Si une erreur survient au niveau du bloc de fonctions, la sortie n'est pas valide (la sortie Valid est définie sur FAUX). Lorsque la condition d'erreur disparaît, les valeurs réapparaissent et la sortie Valid est définie de nouveau.
Traitement des erreurs de sortie	**Sorties utilisées pour définir les erreurs** Tous les blocs disposent des deux sorties suivantes, qui sont utilisées pour les erreurs survenant lors de l'exécution : • Error : le front montant « Error » indique qu'une erreur est survenue lors de l'exécution du bloc fonctionnel. • ErrorID : numéro de l'erreur. les sorties Done et InVelocity sont utilisées pour une exécution réussie, elles excluent donc logiquement la sortie Error. Les erreurs d'instance ne génèrent pas systématiquement une erreur au niveau de l'axe (elles attribuent l'état ErrorStop à l'axe).

Règle applicable à	Règle
	Réinitialisation des sorties d'erreur • Les sorties d'erreur des blocs de fonctions correspondants sont réinitialisées avec le front descendant des commandes Exécuter et Activer. • Les sorties d'erreur d'un bloc fonctionnel avec une commande Enable peuvent être réinitialisées lors de l'exécution, sans qu'une réinitialisation de la commande Enable ne soit nécessaire.
	Types d'erreurs • Logique des blocs fonctionnels (par exemple, paramètres hors limite, tentative d'infraction des machines à état, etc.) • Limites matérielles ou logicielles • Mécanisme/moteur • Lecteur
Conventions de dénomination pour les types ENUM	En raison des contraintes de dénomination de la norme IEC au niveau de l'unicité des noms de variables, nous utiliserons la référence « mc » pour désigner l'espace de nom du contrôle de mouvement PLCopen pour les types ENUM. De cette façon, nous sommes en mesure d'éviter un conflit au niveau de l'utilisation des types ENUM « positif » et « négatif » pour les instances disposant de variables portant ces noms dans le reste du projet puisque nous les appellerons mcPositive et mcNegative respectivement.

Détails des paramètres des blocs fonctionnels de contrôle du mouvement

Les rubriques suivantes fournissent des détails sur les paramètres de contrôle du mouvement utiles à tous les blocs fonctionnels de contrôle du mouvement.

États des axes de contrôle du mouvement sur la page 406

Numéros des paramètres des blocs fonctionnels de contrôle du mouvement sur la page 408

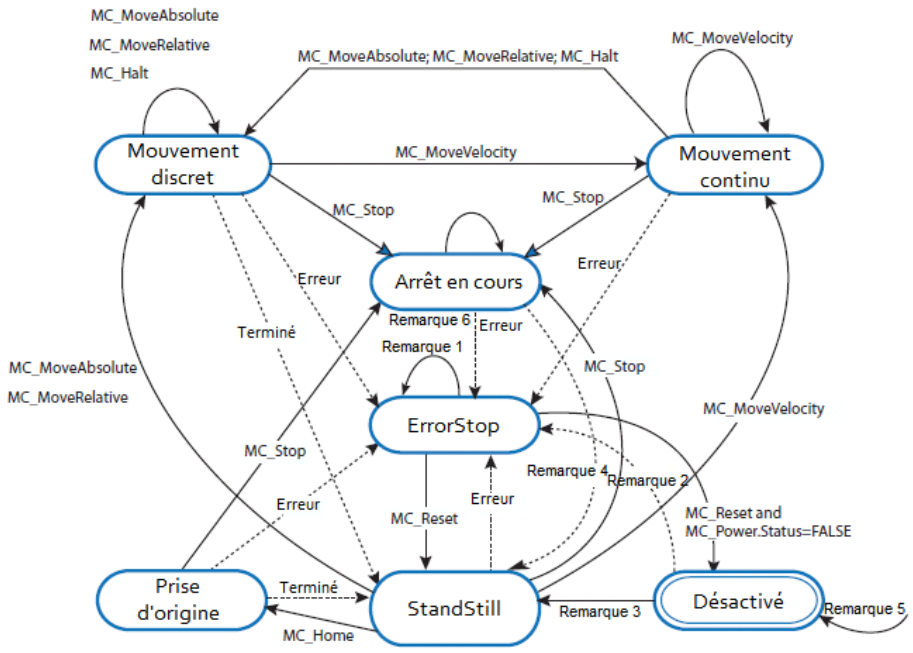
Identifiants des erreurs propres aux blocs fonctionnels de contrôle du mouvement sur la page 409

États des axes de contrôle du mouvement

Pour le comportement de l'axe à haut niveau lorsque plusieurs blocs fonctionnels de contrôle du mouvement sont activés, la règle de base est que les commandes de mouvement sont toujours prises de manière séquentielle, même si le contrôleur a la capacité de les traiter réellement en parallèle. Toute commande de mouvement est une transition qui modifie l'état de l'axe et, par conséquent, la manière dont le mouvement en cours est calculé.

Diagramme des états des axes de contrôle du mouvement

L'axe se trouve toujours dans l'un des états définis, comme l'indique le diagramme suivant.



Comportement des états des axes de contrôle du mouvement

Non	Remarque
1	Dans les états ErrorStop et Stopping, tous les blocs de fonctions (sauf MC_Reset), peuvent être appelés mais ils ne seront pas exécutés. MC_Reset génère une transition vers l'état Immobilisation. Si une erreur survient alors que l'état de la machine est Stopping, une transition est générée vers l'état ErrorStop.
2	Power.Enable = TRUE et une erreur existe sur l'axe.
3	Power.Enable = TRUE et il n'y pas d'erreur sur l'axe.
4	MC_Stop.Done AND NOT MC_Stop.Execute.
5	Lorsque le bloc de fonctions MC_Power est appelé alors que Enable vaut Faux, l'état de l'axe passe à Désactivé quel que soit l'état, même ErrorStop.
6	Si une erreur survient alors que l'état de la machine est Stopping, une transition est générée vers l'état ErrorStop.

Valeurs de code de l'état des axes de contrôle du mouvement

Vous pouvez surveiller l'état des axes grâce à la fonctionnalité Surveillance de l'axe. Le tableau suivant identifie les valeurs utilisées pour définir chacun des états d'axes prédéfinis.

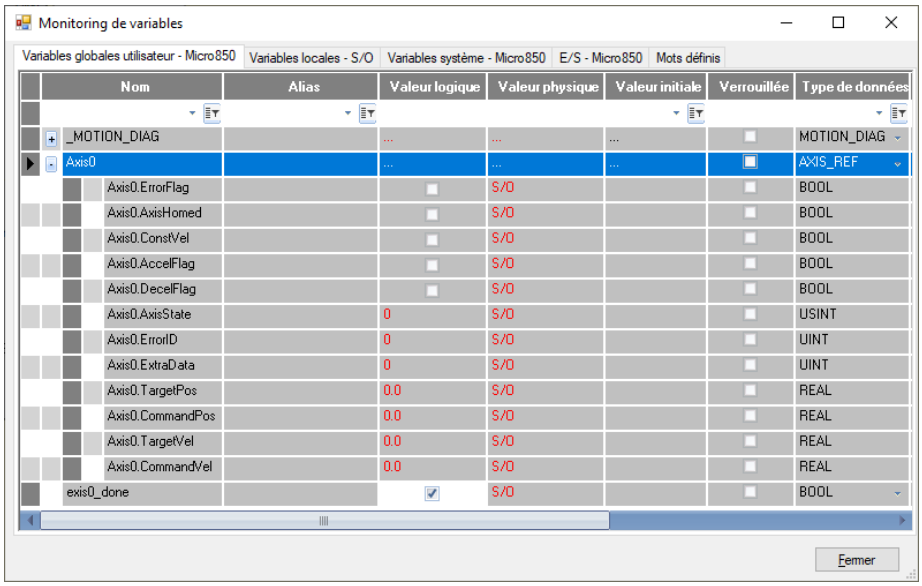
Valeur de l'état	Nom de l'état
0x00	Désactivé
0x01	Immobilisation

Valeur de l'état	Nom de l'état
0x02	Mouvement discret
0x03	Mouvement continu
0x04	Prise d'origine
0x06	Arrêt en cours
0x07	Arrêt suite à une erreur

Mises à jour de l'état des axes

Lors de l'exécution du mouvement, la mise à jour d'état de l'axe dépend du moment où le bloc fonctionnel de mouvement correspondant est appelé par la scrutation POU. Ceci est toujours le cas, même lorsque le profil de mouvement est contrôlé par le moteur de mouvement dans le cadre d'une tâche d'arrière-plan, et ne dépend pas de la scrutation POU.

Par exemple, sur l'axe en mouvement d'une unité POU de logique à relais (état de ligne = Vrai), un bloc fonctionnel MC_MoveRelative de la ligne est scruté et l'axe commence sont déplacement. Avant la fin de MC_MoveRelative, l'état de la ligne passe à Faux et MC_MoveRelative n'est plus scruté. Dans ce cas, l'état de l'axe ne peut passer de Mouvement discret à Immobilisation, même lorsque l'axe cesse complètement de bouger, et la vitesse passe à 0.



Numéros des paramètres des blocs fonctionnels de contrôle du mouvement

Les blocs fonctionnels suivants utilisent des numéros de paramètres spécifiques lorsque les blocs fonctionnels sont programmés.

- MC_ReadParameter
- MC_ReadBoolParameter
- MC_WriteParameter
- MC_WriteBoolParameter

Identification du numéro de paramètre

Les numéros de paramètre compris entre 0 et 999 sont réservés aux paramètres standard. Les extensions d'un fournisseur ou d'un utilisateur sont également autorisées bien que leur utilisation risque d'affecter la portabilité entre les différentes plateformes. Si le numéro d'un paramètre est supérieur à 999, ce paramètre est propre au fournisseur.

Numéro du paramètre	Nom du paramètre	Type de données	L/E	Description
1	Commanded Position	REAL	R	Position commandée.
2	SWLimitPos	REAL	L/E	Position du commutateur limite logiciel positif.
3	SWLimitNeg	REAL	L/E	Position du commutateur limite logiciel négatif.
4	EnableLimitPos	BOOL	L/E	Active le commutateur limite logiciel positif.
5	EnableLimitNeg	BOOL	L/E	Active le commutateur limite logiciel négatif.
8	MaxVelocitySystem	REAL	R	Vitesse maximale autorisée de l'axe dans le système de mouvement.
9	MaxVelocityAppl	REAL	L/E	Vitesse maximale autorisée de l'axe dans l'application.
11	CommandedVelocity	REAL	R	Vitesse commandée.
12	MaxAccelerationSystem	REAL	R	Accélération maximale autorisée de l'axe dans le système de mouvement.
13	MaxAccelerationAppl	REAL	L/E	Accélération maximale autorisée de l'axe dans l'application.
14	MaxDecelerationSystem	REAL	R	Décélération maximale autorisée de l'axe dans le système.
15	MaxDecelerationAppl	REAL	L/E	Décélération maximale autorisée de l'axe dans l'application.
16	MaxJerk	REAL	L/E	Jerk maximal autorisé de l'axe.
1001	TargetPosition	REAL	R	La position cible finale du bloc fonctionnel de déplacement actif.
1002	TargetVelocity1	REAL	R	La vitesse cible finale du bloc fonctionnel de déplacement actif.
1005	Duty Cycle	REAL	L/E	Cycle d'utilisation pour une impulsion. La valeur correcte est comprise entre 0 et 100 (pour 0 % - 100 %). (La fonction PWM peut être appliquée en ajustant cette valeur). Ce paramètre ne peut être configuré qu'à l'aide de ce bloc fonctionnel. La valeur par défaut est définie sur 50.0 par le contrôleur. **Remarque** : Pour le cycle d'utilisation, la valeur est remplacée par le paramètre par défaut, 50.0 dès que le contrôleur passe du mode Exécuter à PRG et RUN, ou lorsqu'il change de cycle.
1006	PulsePerRevolution	REAL	R	Le paramètre Impulsions par révolution a été saisi par l'utilisateur dans l'interface utilisateur de CCW.
1007	TravelPerRevolution	REAL	R	Le paramètre Distance parcourue par révolution a été saisi par l'utilisateur dans l'interface utilisateur de CCW.

Identifiants des erreurs propres aux blocs fonctionnels de contrôle du mouvement

Lorsqu'un bloc de fonctions de contrôle des mouvements se termine par une erreur et que l'état de l'axe correspond à Arrêt suite à une erreur, utilisez le bloc de fonctions MC_Reset, ou désactivez/activez MC_Power et utilisez MC_Reset, pour récupérer l'axe. Le mouvement normal de l'axe peut de nouveau être exécuté sans qu'il y ait besoin d'arrêter le fonctionnement du contrôleur.

Ce tableau vous permet de déterminer les erreurs pour les blocs de fonctions du contrôle des mouvements.

Valeur	ID DE MACRO	Description

Valeur	ID DE MACRO	Description
00	MC_FB_ERR_ NO	Le bloc de fonctions a été exécuté avec succès.
01	MC_FB_ERR_ WRONG_STATE	Le bloc de fonctions ne peut pas être exécuté car l'état de l'axe est incorrect. Vérifiez l'état de l'axe sur la page 406.
02	MC_FB_ERR_ RANGE	Le bloc de fonctions ne peut pas être exécuté en raison d'un ou de plusieurs paramètres dynamiques non valides (vitesse, accélération, décélération ou jerk) définis pour l'axe dans le bloc de fonctions. Corrigez la définition des paramètres dynamiques dans le bloc de fonctions par rapport à la page de configuration relative à la dynamique de l'axe.
03	MC_FB_ERR_ PARAM	Le bloc de fonctions ne peut pas être exécuté en raison d'un paramètre non valide autre que la vitesse, l'accélération, la décélération ou le jerk défini dans le bloc de fonctions. Corrigez la définition des paramètres (par exemple le mode ou la position) pour le bloc de fonctions.
04	MC_FB_ERR_ AXISNUM	Le bloc de fonctions ne peut pas être exécuté car l'axe n'existe pas, les données de configuration de l'axe sont endommagées ou l'axe n'est pas configuré correctement.
05	MC_FB_ERR_ MECHAN	Le bloc de fonctions ne peut pas être exécuté car une panne est survenue sur l'axe en raison de problèmes mécaniques ou au niveau du variateur. Vérifiez la connexion entre le variateur et le contrôleur (signaux relatifs à la disponibilité du variateur et à la position) et assurez-vous que le variateur fonctionne normalement.
06	MC_FB_ERR_ NOPOWER	Le bloc de fonctions ne peut pas être exécuté car l'axe n'est pas sous tension. Mettez l'axe sous tension à l'aide du bloc de fonctions MC_Power.
07	MC_FB_ERR_ RESOURCE	Le bloc de fonctions ne peut pas être exécuté car la ressource requise par le bloc de fonctions est contrôlée par un autre bloc de fonctions ou n'est pas disponible. Vérifiez que la ressource requise par le bloc de fonctions est disponible et qu'elle peut être utilisée. Exemples : • Des blocs fonctionnels MC_Power essaient de contrôler le même axe. • Des blocs fonctionnels MC_Stop sont exécutés sur le même axe au même moment. • Des blocs fonctionnels MC_TouchProbe sont exécutés sur le même axe au même moment. • Un bloc fonctionnel MC_TouchProbe est exécuté alors que l'entrée du palpeur n'est pas activée dans la configuration du mouvement.
08	MC_FB_ERR_ PROFILE	Le bloc de fonctions ne peut pas être exécuté car le profil de mouvement défini dans le bloc de fonctions ne peut pas être atteint. Corrigez le profil dans le bloc de fonctions.
09	MC_FB_ERR_ VELOCITY	Le bloc de fonctions ne peut pas être exécuté car le profil de mouvement demandé dans le bloc de fonctions ne peut pas être atteint en raison de la vitesse actuelle de l'axe. Exemples : • Le bloc fonctionnel demande à l'axe de changer de direction alors que l'axe est en mouvement. • Le profil de mouvement requis ne peut pas être atteint en raison d'une vitesse actuelle trop faible ou trop élevée. Vérifiez le paramétrage du profil de mouvement dans le bloc de fonctions, puis corrigez le profil ou exécutez de nouveau le bloc de fonctions lorsque la vitesse de l'axe est compatible avec le profil de mouvement demandé.
0A	MC_FB_ERR_ SOFT_LIMIT	Ce bloc de fonctions ne peut pas être exécuté car la limite logicielle sera dépassée ou il est annulé car cette limite a déjà été atteinte. Vérifiez le paramétrage de la vitesse ou de la position cible dans le bloc de fonctions ou réglez le paramètre relatif à la limite logicielle.
0B	MC_FB_ERR_ HARD_LIMIT	Ce bloc de fonctions est annulé en raison de la détection de l'état actif du commutateur de limite matérielle au cours du mouvement de l'axe ou avant son démarrage. Éloignez l'axe du commutateur de limite matérielle dans la direction opposée.

Valeur	ID DE MACRO	Description
0C	MC_FB_ERR_ LOG_LIMIT	Ce bloc de fonctions ne peut pas être exécuté car la limite logique de l'accumulateur PTO sera dépassée ou il est annulé car cette limite a déjà été atteinte. Vérifiez le paramétrage de la vitesse ou de la position cible pour le bloc de fonctions. Utilisez également le bloc de fonctions MC_SetPosition pour régler le système de coordonnées de l'axe.
0D	MC_FB_ERR_ ERR_ENGINE	Une erreur relative à l'exécution du moteur de mouvement est détectée lors de l'exécution de ce bloc de fonctions. Arrêtez, puis remettez sous tension l'intégralité de la configuration de mouvement, y compris le contrôleur, les variateurs et les actionneurs, puis téléchargez de nouveau en aval l'application utilisateur. Si la panne persiste, contactez le support technique.
10	MC_FB_ERR_ NOT_HOMED	Le bloc de fonctions ne peut pas être exécuté car il est nécessaire de procéder au préalable à la prise d'origine de l'axe. Exécutez la prise d'origine de l'axe à l'aide du bloc de fonctions MC_Home.
80	MC_FB_ERR_ PARAM_MODIFIED	Avertissement : la vitesse demandée pour l'axe a été réglée sur une valeur inférieure. Le bloc de fonctions s'exécute sans erreur à une vitesse plus faible.

Scénarios des erreurs au niveau de l'axe

Dans la plupart des cas, lorsqu'une instruction de bloc de fonctions de mouvement envoyée à un axe provoque une erreur de bloc de fonctions, l'axe passe dans un état Erreur et l'élément ErrorID correspondant est défini sur les données AXIS_REF pour l'axe.

Dans les situations suivantes, il se peut que l'axe ne passe pas dans un état Erreur et il est possible pour l'application utilisateur d'émettre avec succès un bloc de fonctions de mouvement pour l'axe après la modification de l'état de l'axe.

Scénario	Exemple
Un bloc fonctionnel de mouvement donne des instructions à un axe, mais l'axe est dans un état ne permettant pas d'exécuter correctement le bloc fonctionnel.	L'axe n'a pas d'alimentation ou l'axe est dans une séquence de prise d'origine ou il est arrêté suite à une erreur.
Un bloc fonctionnel de mouvement donne des instructions à un axe, mais l'axe est encore contrôlé par un autre bloc fonctionnel de mouvement. L'axe ne peut pas autoriser le contrôle du mouvement par le nouveau bloc fonctionnel sans passer par un arrêt complet.	Le nouveau bloc fonctionnel ordonne à l'axe de changer de direction de mouvement.
Lorsqu'un bloc fonctionnel de mouvement tente de contrôler un axe, mais que l'axe est encore contrôlé par un autre bloc fonctionnel de mouvement, le profil de mouvement défini dernièrement ne peut pas être réalisé par le contrôleur.	L'application utilisateur émet un bloc fonctionnel MC_MoveAbsolute Courbe en S à un axe avec une distance trop courte étant donné le moment où l'axe se déplace.

Type de données AXIS_REF

Le type de données AXIS_REF est une structure de données qui contient des informations pour un axe de mouvement, et qui est utilisée comme une variable d'entrée et de sortie dans tous les blocs de fonctions de contrôle des mouvements. Une instance d'un type de données AXIS_REF est automatiquement créée lorsque vous ajoutez un axe de mouvement à la configuration.

Paramètre	Type de données	Description
Axis_ID	AXIS_REF	L'identifiant de l'axe logique est automatiquement attribué par Connected Components Workbench. Il ne peut être ni modifié ni consulté par l'utilisateur.
Error Flag	BOOL	Indique si une erreur existe sur l'axe. Une fois qu'une erreur est indiquée sur un axe, et que l'identifiant de l'erreur n'est pas zéro, l'axe doit être réinitialisé à l'aide de MC_Reset avant d'émettre tout autre bloc de fonctions de mouvement.

Paramètre	Type de données	Description
AxisHomed	BOOL	Indique si une opération de prise d'origine a été exécutée avec succès ou non pour l'axe. Lorsque l'utilisateur tente d'effectuer une nouvelle prise d'origine pour un axe dont l'état AxisHomed a déjà été défini (prise d'origine concluante), mais que l'opération échoue, l'état AxisHomed est effacé.
ConstVel	BOOL	Indique si le mouvement de l'axe est en vitesse constante ou non. Un axe stationnaire n'est pas considéré comme étant en vitesse constante.
AccFlag	BOOL	Indique si le mouvement de l'axe est en phase d'accélération ou non.
DecelFlag	BOOL	Indique si le mouvement de l'axe est en phase de décélération ou non.
AxisState	USINT	Indique l'état actuel de l'axe.
ErrorID	UINT	Indique la cause de l'erreur survenue sur l'axe lorsqu'une erreur est signalée par ErrorFlag. Cette erreur est généralement générée par l'échec de l'exécution d'un bloc de fonctions de contrôle du mouvement.
ExtraData	UINT	Réservé.
TargetPos	REAL	Indique la position cible finale de l'axe pour les blocs de fonctions MoveAbsolute et MoveRelative. Pour les blocs de fonctions MoveVelocity, Stop et Halt, TargetPos correspond à 0 sauf lorsque la valeur TargetPos définie par les blocs de fonctions de position précédents n'est pas effacée.
CommandPos	REAL	Lors du mouvement, il s'agit de la position actuelle que doit adopter l'axe suite à la commande du contrôleur. Il est possible qu'il y ait un léger décalage entre la position réelle de l'axe et ce paramètre CommandPos.
TargetVel	REAL	Vitesse cible maximale demandée à l'axe pour un bloc de fonctions de mouvement. La valeur de TargetVel dans le bloc fonctionnel actuel ou dans un bloc plus petit dépend des autres paramètres du même bloc fonctionnel.
CommandVel	REAL	Lors du mouvement, il s'agit de la vitesse actuelle que doit adopter l'axe suite à la commande du contrôleur. Notez qu'il peut y avoir une légère différence entre la vitesse réelle de l'axe et le paramètre CommandVel en raison du dépassement du décalage ou de l'ajustement du variateur.

Type de données FB_AXIS_REF

Le type de données FB_AXIS_REF est une structure de données contenant des informations relatives à un axe de retour codeur de mouvement. Elle est utilisée sous la forme d'une variable d'entrée et de sortie dans tous les blocs de fonctions de contrôle des mouvements. Une instance de type de données FB_AXIS_REF est créée automatiquement lorsque vous ajoutez un module HSC et que le mode est configuré comme mode d'axe de retour codeur.

IMPORTANT une fois qu'une erreur est indiquée sur un axe de retour codeur et que l'identifiant de l'erreur n'est pas zéro, le FBAxis doit être réinitialisé à l'aide de MC_Reset avant d'émettre tout autre bloc de fonctions de mouvement.

Paramètre	Type de données	Description
Axis_ID	FB_AXIS_REF	L'identifiant de l'axe logique est automatiquement attribué par Connected Components Workbench. Il ne peut être ni modifié ni consulté par l'utilisateur.
ErrorFlag	BOOL	Indique si une erreur existe sur l'axe de retour codeur.
ConstVel	BOOL	Indique si le mouvement de l'axe de retour codeur est en vitesse constante ou non. Un axe stationnaire n'est pas considéré comme étant en vitesse constante.
AccelFlag	BOOL	Indique si le mouvement de l'axe de retour codeur est en phase d'accélération ou non.
DecelFlag	BOOL	Indique si le mouvement de l'axe de retour codeur est en phase de décélération ou non.

Paramètre	Type de données	Description
AxisState	USINT	Indique l'état actuel de l'axe de retour codeur.
ErrorID	UINT	Indique la cause de l'erreur survenue sur l'axe lorsqu'une erreur est signalée par ErrorFlag. Cette erreur est généralement générée par l'échec de l'exécution d'un bloc de fonctions de contrôle du mouvement.
ExtraData	UINT	Réservé.
ActualPos	REAL	Position mécanique réelle lue en provenance du canal de retour codeur de mouvement (HSC).
ActualVel	REAL	Vitesse mécanique réelle lue en provenance du canal de retour codeur de mouvement (HSC).

Variables d'axe

Les variables d'axe sont utilisées pour contrôler la position, la vitesse, l'accélération et les erreurs d'un axe de contrôle du mouvement donné.

Pour affecter une variable à un paramètre de sortie d'axe :

- Dans un diagramme de bloc fonctionnel

  Vous pouvez graphiquement connecter le paramètre de sortie de l'axe d'un bloc de fonctions de contrôle des mouvements afin de bloquer à votre convenance le paramètre d'entrée AxisIn d'un autre bloc de fonctions de contrôle des mouvements. Par exemple, connectez le paramètre de sortie MC_POWER au paramètre d'entrée AxisIn de MC_HOME.
- Dans une logique à relais

  Une variable ne peut pas être assignée au paramètre de sortie d'axe d'un bloc de fonctions de contrôle des mouvements, car ce paramètre est en lecture seule.

Surveillance d'une variable AXIS_REF

Vous pouvez surveiller une variable AXIS_REF ou FBAXIS_REF via le logiciel, lorsque vous êtes connecté au contrôleur, lorsque le moteur de mouvement est actif ou dans l'application utilisateur dans le cadre de la logique utilisateur. Vous pouvez également surveiller à distance la variable AXIS_REF ou FBAXIS_REF par le biais de divers canaux de communication.

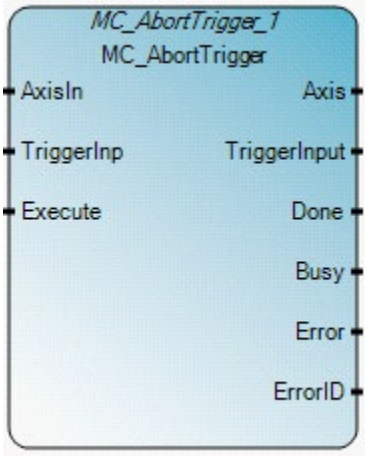
MC_AbortTrigger (déclencheur d'abandon du contrôle des mouvements)

Annule les blocs de fonctions de mouvement connectés à des événements de déclenchement.

MC_AbortTrigger ne s'exécute que lorsqu'il est affecté à un axe contrôlé par MC_TouchProbe.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

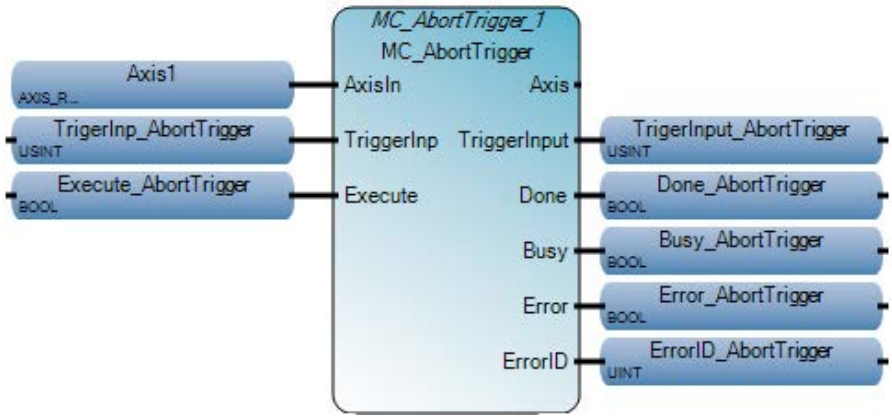
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



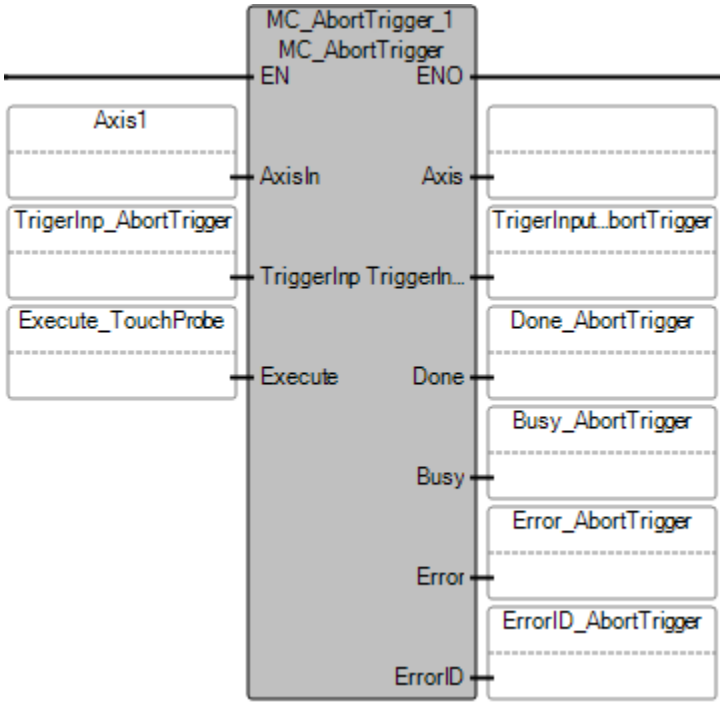
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_AbortTrigger actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF sur la page 411 FB_AXIS_REF sur la page 412	Utilisez le type de données AXIS_REF pour définir le paramètre AxisIn. Si l'axe est un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF pour définir les paramètres AxisIn.
TriggerInp	Entrée	USINT	Ce paramètre n'est pas pris en compte.
Execute	Entrée	BOOL	Lorsque ce paramètre est TRUE, l'événement de déclenchement est annulé lors du front montant.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
TriggerInput	Sortie	USINT	Ce paramètre n'est pas pris en compte.
Done	Sortie	BOOL	TRUE : l'événement de déclenchement est annulé.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé.
Error	Sortie	BOOL	VRAI : erreur détectée.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements.

Exemple de Diagramme de Bloc Fonction MC_AbortTrigger



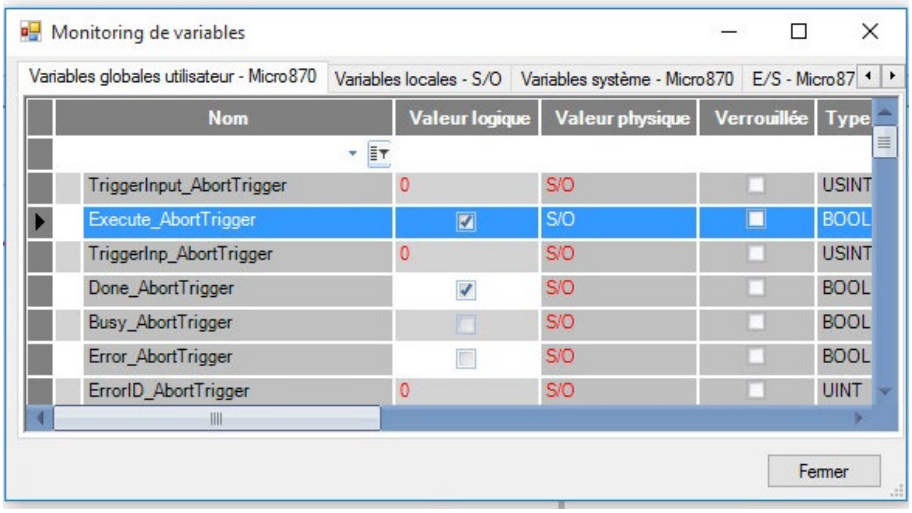
Exemple de langage à relais MC_AbortTrigger



Exemple de texte structuré MC_AbortTrigger

```
MC_AbortTrigger_1(Axis1,TrigerInp_AbortTrigger,Execute_AbortTrigger);
Done_AbortTrigger := MC_AbortTrigger_1.Done;
Busy_AbortTrigger := MC_AbortTrigger_1.Busy;
Error_AbortTrigger := MC_AbortTrigger_1.Error;
ErrorID_AbortTrigger := MC_AbortTrigger_1.ErrorID;
```

Résultats



MC_Halt (arrêt du contrôle des mouvements)

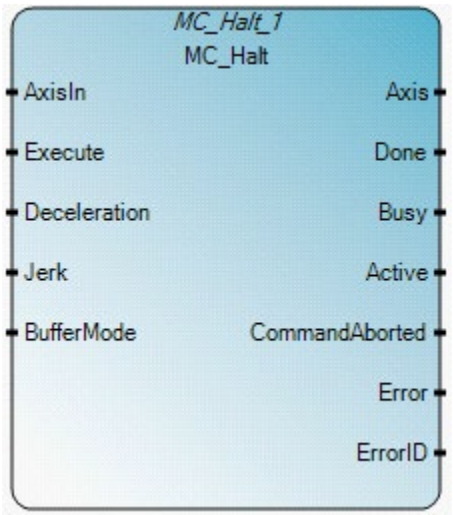
Commande un arrêt contrôlé du mouvement dans des conditions de fonctionnement normales.

Détails de l'opération :

- L'axe passe à l'état Mouvement discret jusqu'à ce que la vitesse soit égale à zéro. Lorsque la vitesse atteint zéro, le paramètre Done est défini sur VRAI et l'axe passe à l'état Immobilisation.
  - Il est possible d'exécuter une autre commande de mouvement pendant la décélération de l'axe, ce qui remplace MC_Halt.
  - Si MC_Halt est émis alors que l'état de l'axe est en prise d'origine, le bloc d'instruction signale une erreur et le processus de prise d'origine n'est pas interrompu.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

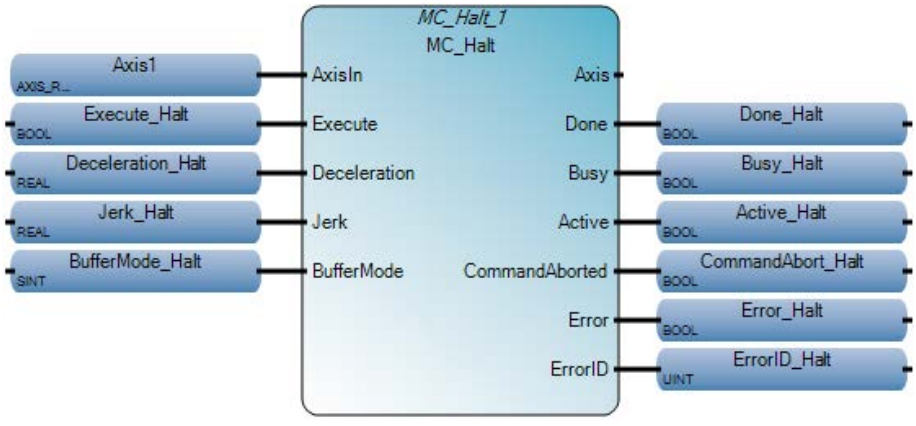
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



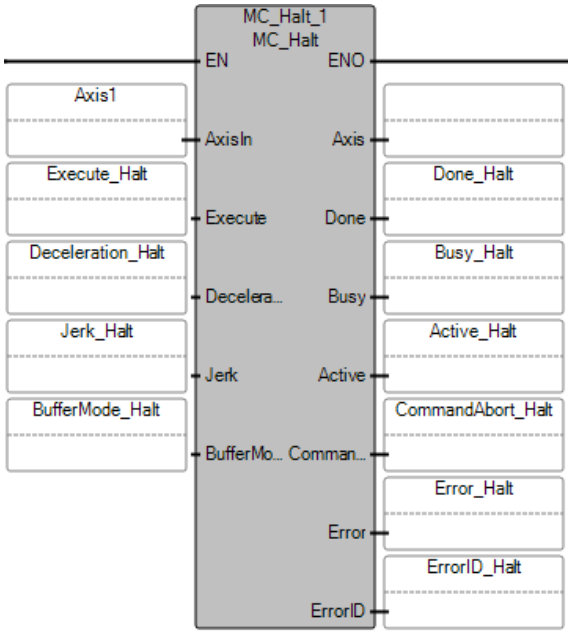
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_Halt actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
AxisIn	Entrée	AXIS_REF	Utilisez les paramètres de type de données AXIS_REF pour définir AxisIn.
Execute	Entrée	BOOL	Indique quand démarrer le mouvement. VRAI : démarrage du mouvement au moment du front montant. FAUX : le mouvement ne démarre pas. Si vous exécutez MC_Halt lors de la prise d'origine, le paramètre MC_Halt est défini sur MC_FB_ERR_STATE et le processus de prise d'origine se poursuit.
Deceleration	Entrée	REAL	Valeur correspondant à la décélération (toujours positive, réduction de la quantité d'énergie envoyée au moteur.) Si Deceleration <= 0 et que l'axe n'est pas en mode Immobilisation, MC_Halt est défini sur MC_FB_ERR_RANGE.
Jerk	Entrée	REAL	Valeur correspondant au jerk (toujours positive). Si Jerk <= 0 et que l'axe est en mode Immobilisation, MC_Halt est défini à MC_FB_ERR_RANGE.
BufferMode	Entrée	SINT	Non utilisé. Le mode correspond toujours à MC_Aborting.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF sur la page 411	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	Vitesse zéro atteinte.
Busy	Sortie	BOOL	Le bloc d'instruction n'est pas terminé.
Active	Sortie	BOOL	Indique que le bloc d'instruction contrôle l'axe.
CommandAborted	Sortie	BOOL	La commande est remplacée par une autre commande ou arrêtée suite à une erreur.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

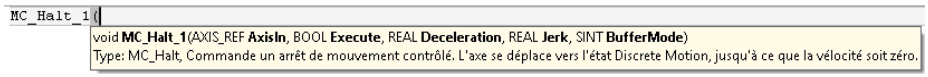
Exemple de Diagramme de Bloc Fonction MC_Halt



Exemple de langage à relais MC_Halt

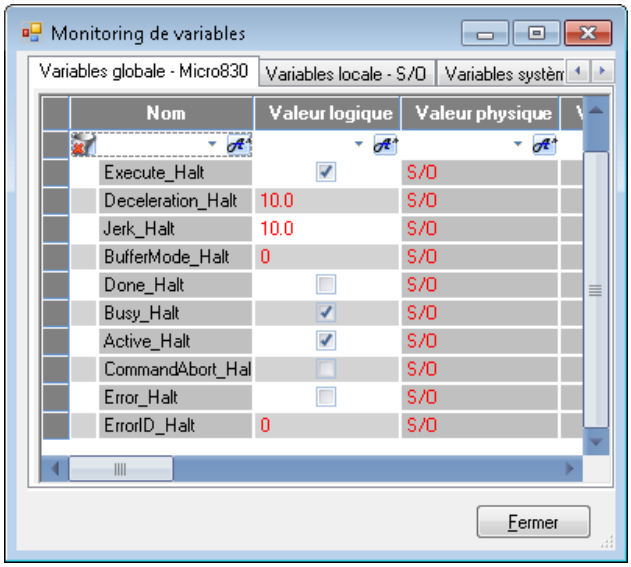


Exemple de texte structuré MC_Halt




```
Deceleration_Halt := 10.0;
Jerk_Halt := 10.0;
MC_Halt_1(Axis1,Execute_Halt,Deceleration_Halt,Jerk_Halt,BufferMode_Halt);
Done_Halt := MC_Halt_1.Done;
Busy_Halt := MC_Halt_1.Busy;
Active_Halt := MC_Halt_1.Active;
CommandAbort_Halt := MC_Halt_1.CommandAborted;
Error_Halt := MC_Halt_1.Error;
ErrorID_Halt := MC_Halt_1.ErrorID;
```

Résultats



MC_Home (origine du contrôle des mouvements)

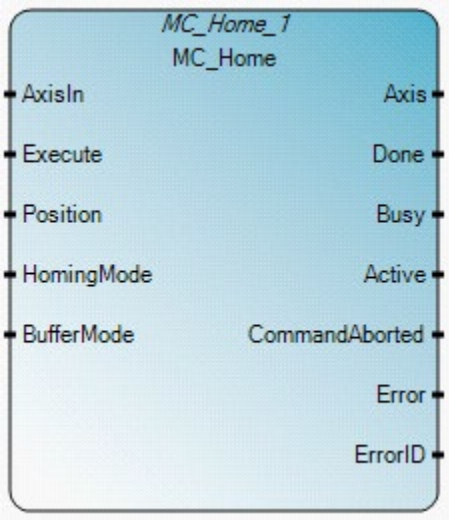
Commande l'axe pour effectuer une séquence <*rechercher origine*>. Les détails de cette séquence dépendent du fabricant et peuvent être définis par les paramètres de l'axe. L'entrée Position est utilisée pour définir la position absolue lorsqu'un signal de référence est détecté et que le décalage par rapport à l'origine configuré est atteint.

Détails de l'opération :

- Après émission de MC_Power, l'état Origine de l'axe est remis à zéro (hors position d'origine). Dans la plupart des cas, après mise sous tension de l'axe, le bloc de fonctions MC_Home doit être exécuté to pour calibrer la position de l'axe et la position de référence de l'origine.
- Le bloc de fonctions MC_Home ne peut être annulé qu'à l'aide des blocs de fonctions MC_Stop ou MC_Power. Si MC_Home est annulé avant la fin de son cycle, la position d'origine recherchée précédemment est considérée comme non valide et l'état Origine de l'axe est effacé.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_Home actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF	Utilisez les paramètres de type de données AXIS_REF pour définir AxisIn.
Execute	Entrée	BOOL	Indique quand démarrer le mouvement. VRAI : démarrage du mouvement au moment du front montant. FAUX : le mouvement ne démarre pas.

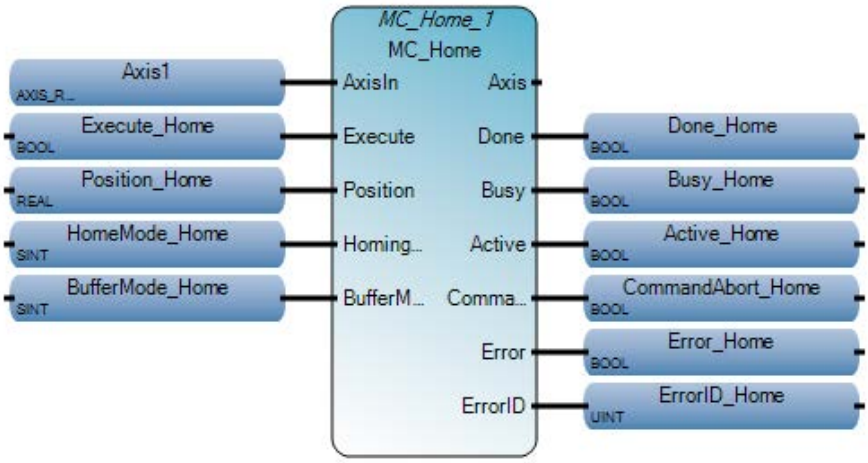
Paramètre	Type de paramètre	Type de données	Description
Position	Entrée	REAL	La position absolue est définie lorsque le signal de référence est détecté et que le décalage configuré par rapport à l'origine est atteint. La plage de valeurs pour cette entrée correspond à -0x40000000 – 0x40000000 pulsations physiques après la conversion de la position exprimée en unité de position utilisateur en pulsations PTO. Définissez la valeur Position dans la limite logicielle. Une valeur d'entrée non valide génère une erreur. Identifiant de l'erreur = **MC_FB_ERR_PARAM.**
HomingMode	Entrée	SINT	Entrée Enum pour le mode Prise d'origine.
BufferMode	Entrée	SINT	Non utilisé. Le mode correspond toujours à mcAborting.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF sur la page 411	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	VRAI : l'opération de prise d'origine se termine de façon concluante et l'état de l'axe est défini sur Immobilisation. FAUX : l'opération de prise d'origine est en cours ou incomplète.
Busy	Sortie	BOOL	VRAI : le bloc d'instruction n'est pas terminé. FAUX : le bloc d'instruction est terminé.
Active	Sortie	BOOL	VRAI : indique que le bloc d'instruction contrôle l'axe.
CommandAborted	Sortie	BOOL	TRUE : la commande a été remplacée par une autre commande ou arrêtée suite à une erreur.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UNIT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

HomingModes

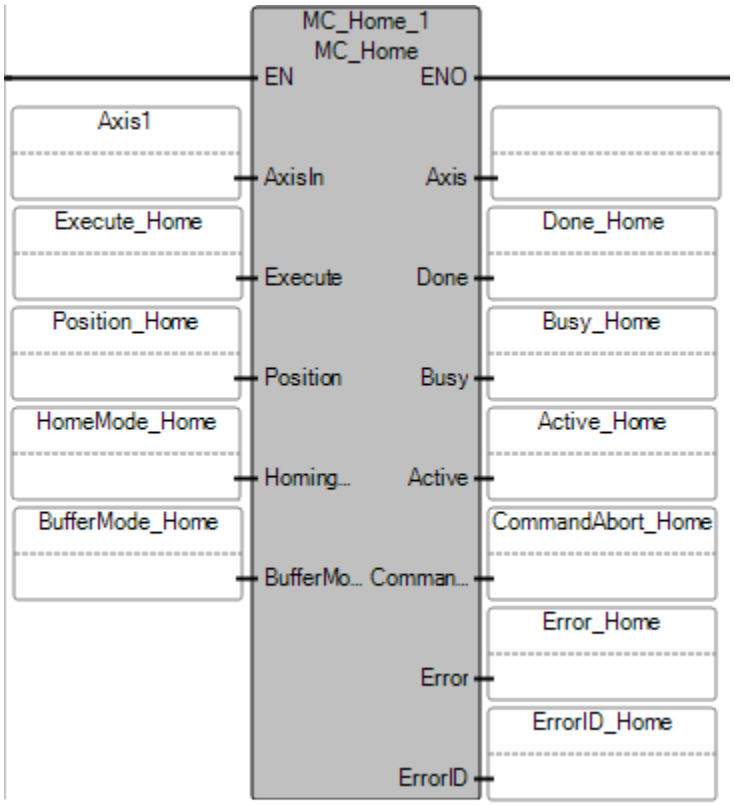
Ce tableau vous aide à déterminer les valeurs pour le paramètre HomingMode dans l'instruction de contrôle des mouvements sur la page 403 MC_Home.

Valeur	Nom	Description
0x00	MC_HOME_ABS_SWITCH	Processus de prise d'origine consistant à parcourir le commutateur relatif à l'origine absolue
0x01	MC_HOME_LIMIT_SWITCH	Processus de prise d'origine consistant à parcourir le commutateur relatif aux limites
0x02	MC_HOME_REF_WITH_ABS	Processus de prise d'origine consistant à parcourir le commutateur relatif à l'origine absolue et à utiliser les pulsations de référence du codeur
0x03	MC_HOME_REF_PULSE	Processus de prise d'origine consistant à parcourir le commutateur relatif aux limites et à utiliser les pulsations de référence du codeur
0x04	MC_HOME_DIRECT	Processus de prise d'origine statique avec définition directe forcée de la position d'origine à partir des références utilisateur. Le bloc fonctionnel définit la position actuelle du mécanisme comme position d'origine, sa position étant déterminée par le paramètre d'entrée Position.

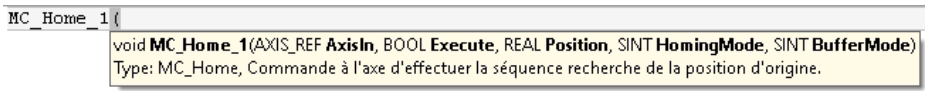
Exemple de Diagramme de Bloc Fonction MC_Home



Exemple de langage à relais MC_Home

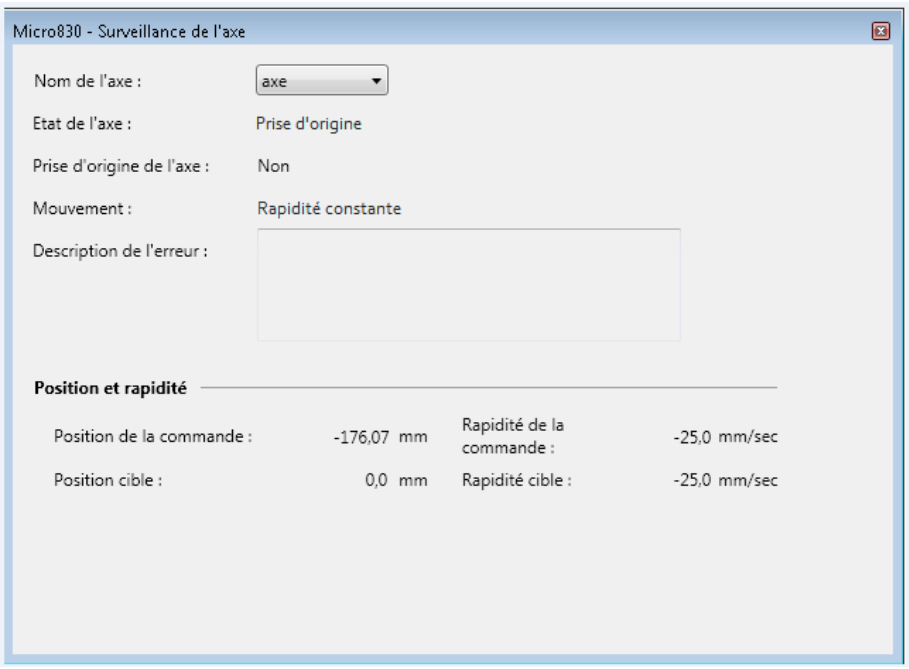


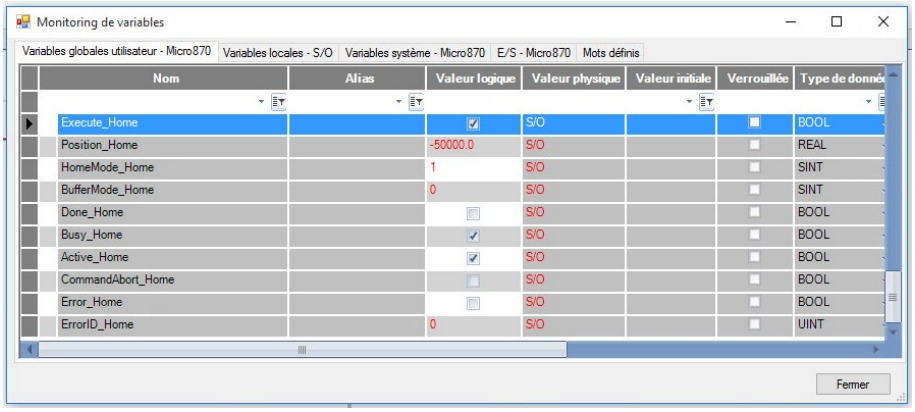
Exemple de texte structuré MC_Home



```
Position_Home := -50000.0;
HomeMode_Home := 4;  (*1*)
MC_Home_1(Axis1,Execute_Home,Position_Home,HomeMode_Home,BufferMode_Home);
Done_Home := MC_Home_1.Done;
Busy_Home := MC_Home_1.Busy;
Active_Home := MC_Home_1.Active;
CommandAbort_Home := MC_Home_1.CommandAborted;
Error_Home := MC_Home_1.Error;
ErrorID_Home := MC_Home_1.ErrorID;
```

Résultats





MC_MoveAbsolute (déplacement absolu du contrôle des mouvements)

Commande un mouvement contrôlé vers une position absolue spécifiée.

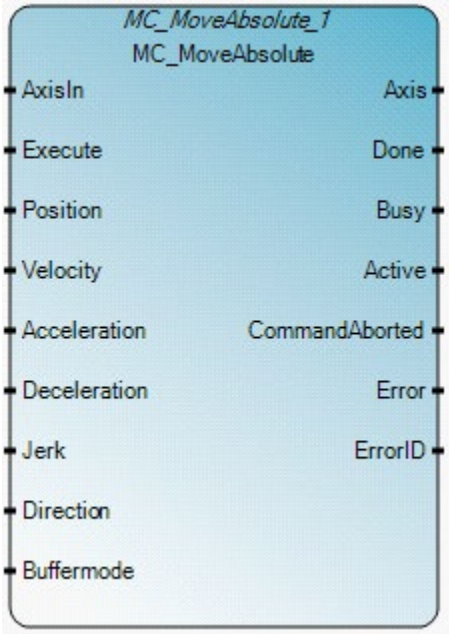
Détails de l'opération :

- Pour le contrôleur Micro800,
  - Le signe de l'entrée Vitesse d'un bloc de fonctions MC_MoveAbsolute est ignoré car la direction des mouvements est déterminée par la position actuelle et la position cible.
  - L'entrée Direction d'un bloc de fonctions MC_MoveAbsolute est ignorée car il n'existe qu'une seule solution mathématique pour atteindre la position cible.

- Si le bloc de fonctions MC_MoveAbsolute est démarré alors que l'état de l'axe du contrôleur Micro800 est Immobilisation et que la distance relative du déplacement est nulle, l'exécution du bloc de fonctions est immédiatement signalée comme Done.
- If un bloc de fonctions MC_MoveAbsolute est envoyé à un axe qui n'est pas dans la position Origine, le bloc de fonctions signale une erreur.
- Le bloc de fonctions MoveAbsolute se termine avec Velocity égale à zéro s'il n'est pas remplacé par un autre bloc de fonctions.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

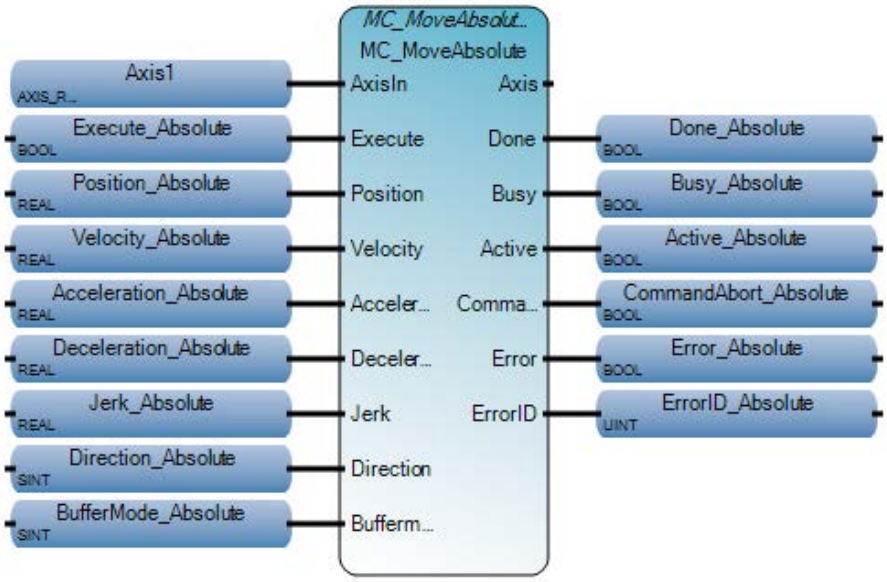


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

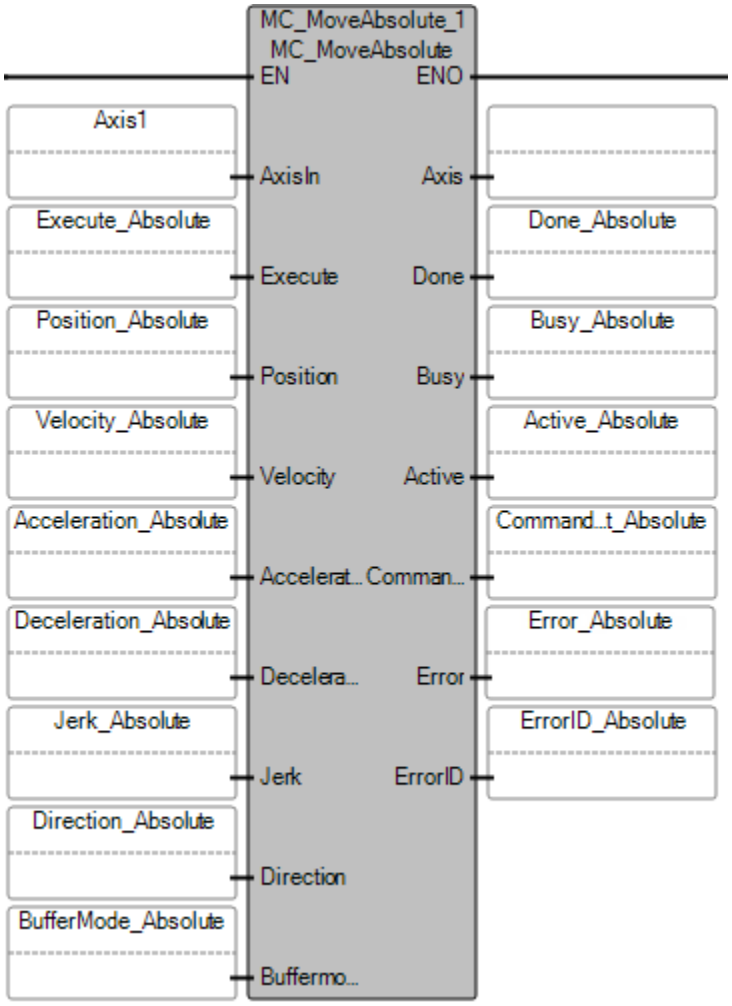
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_MoveAbsolute actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF	Utilisez les paramètres de type de données AXIS_REF pour définir AxisIn.
Execute	Entrée	BOOL	Indique quand démarrer le mouvement. VRAI : démarrage du mouvement au moment du front montant. FAUX : le mouvement ne démarre pas. Il est recommandé de placer l'axe dans sa position d'origine lorsque cette commande d'exécution est générée ou qu'une erreur survient, MC_FB_ERR_NOT_HOMED.
Position	Entrée	REAL	Position cible pour le mouvement exprimée en unité technique (négative ou positive). L'unité technique est définie sur la page de configuration **Mouvement - Général** d'un axe.

Paramètre	Type de paramètre	Type de données	Description
Velocity	Entrée	REAL	Valeur correspondant à la vitesse maximale. Il est possible que la vitesse maximale ne soit pas atteinte lorsque la valeur Jerk = 0. Le signe de la valeur Velocity n'est pas pris en compte et la direction du mouvement est déterminée par l'entrée Position.
Acceleration	Entrée	REAL	Valeur correspondant à l'accélération (toujours positive, augmentation de la quantité d'énergie envoyée au moteur.) unité utilisateur/seconde^2
Deceleration	Entrée	REAL	Valeur correspondant à la décélération (toujours positive, réduction de la quantité d'énergie envoyée au moteur.) u/sec^2
Jerk	Entrée	REAL	Valeur correspondant au jerk (toujours positive). u/sec^3 Lorsque la valeur de l'entrée Jerk = 0, le moteur de mouvement génère un profil trapézoïdal. Lorsque la valeur Jerk est supérieure à 0, le profil calculé est de type Courbe en S.
Direction	Entrée	SINT	Ce paramètre n'est pas utilisé.
BufferMode	Entrée	SINT	Ce paramètre n'est pas utilisé.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF sur la page 411	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	Lorsque ce paramètre est VRAI, la position de la commande est atteinte. Lorsque l'entrée In-Position est définie sur Enabled pour cet axe, le variateur doit définir le signal d'entrée In-Position comme actif avant que ce bit Done ne bascule à l'état VRAI. Cette action se termine à vitesse nulle à moins qu'elle ne soit annulée.
Busy	Sortie	BOOL	Lorsque True, l'exécution du bloc fonctionnel est en progrès.
Active	Sortie	BOOL	Lorsque ce paramètre est VRAI, il indique que le bloc de fonctions contrôle l'axe.
CommandAborted	Sortie	BOOL	Lorsque ce paramètre est VRAI, la commande a été remplacée par une autre commande ou arrêtée suite à une erreur.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Exemple de Diagramme de Bloc Fonction MC_MoveAbsolute



Exemple de langage à relais MC_MoveAbsolute



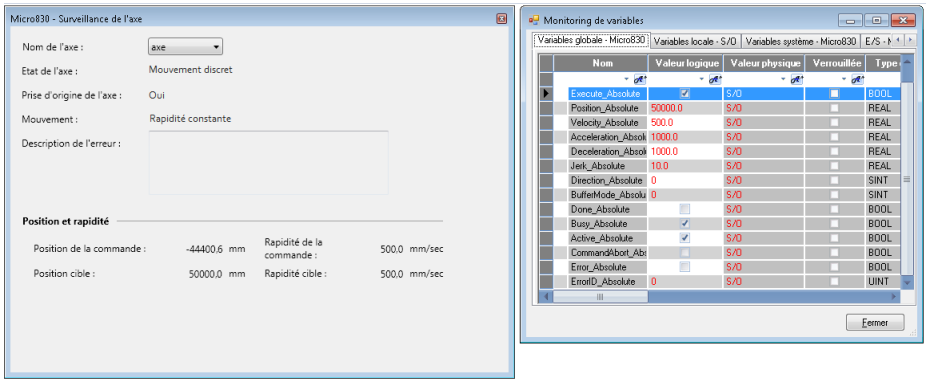
Exemple de texte structuré MC_MoveAbsolute

```
MC_MoveAbsolute_1(
```

void **MC_MoveAbsolute_1**(AXIS_REF **AxisIn**, BOOL **Execute**, REAL **Position**, REAL **Velocity**, REAL **Acceleration**, REAL **Deceleration**, REAL **Jerk**, SINT **Direction**, SINT **BufferMode**)
Type: MC_MoveAbsolute, Commande un mouvement contrôlé vers une position absolue spécifiée.

```
Position_Absolute := 50000.0;
Velocity_Absolute := 500.0;
Acceleration_Absolute := 1000.0;
Deceleration_Absolute := 1000.0;
Jerk_Absolute := 10.0;
MC_MoveAbsolute_1(Axis1,Execute_Absolute,Position_Absolute,Velocity_Absolute,
Acceleration_Absolute,Deceleration_Absolute,Jerk_Absolute,Direction_Absolute,BufferMode_Absolute);
Done_Absolute := MC_MoveAbsolute_1.Done;
Busy_Absolute := MC_MoveAbsolute_1.Busy;
Active_Absolute := MC_MoveAbsolute_1.Active;
CommandAbort_Absolute := MC_MoveAbsolute_1.CommandAborted;
Error_Absolute := MC_MoveAbsolute_1.Error;
ErrorID_Absolute := MC_MoveAbsolute_1.ErrorID;
```

Résultats



MC_MoveRelative (déplacement relatif du contrôle des mouvements)

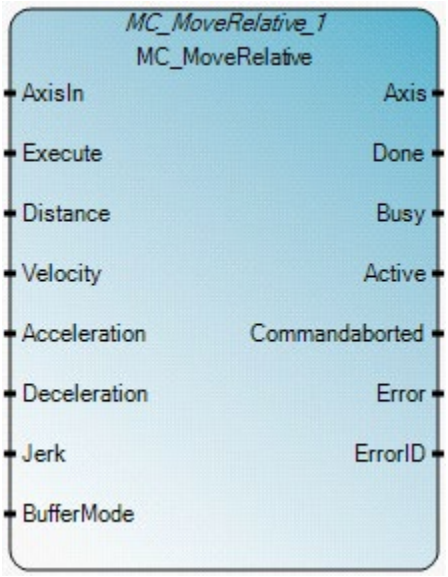
Commande un mouvement contrôlé d'une distance spécifique relative à la position actuelle au moment de l'exécution.

Détails de l'opération :

- La direction des mouvements de MC_MoveRelative étant déterminée par la position actuelle et la position cible, le signe de vitesse n'est pas pris en compte.
- MoveRelative se termine avec une vitesse égale à zéro s'il n'est pas remplacé par un autre bloc de fonctions.
- Si MC_MoveRelative est émis alors que l'état de l'axe du contrôleur Micro800 est Immobilisation et que la distance relative du mouvement est nulle, l'exécution du bloc de fonctions est immédiatement signalée comme Done.
- Pour un contrôleur Micro800, le signe de l'entrée Vitesse pour MC_MoveRelative est ignoré, car la direction des mouvements est déterminée par la position actuelle et la position cible.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



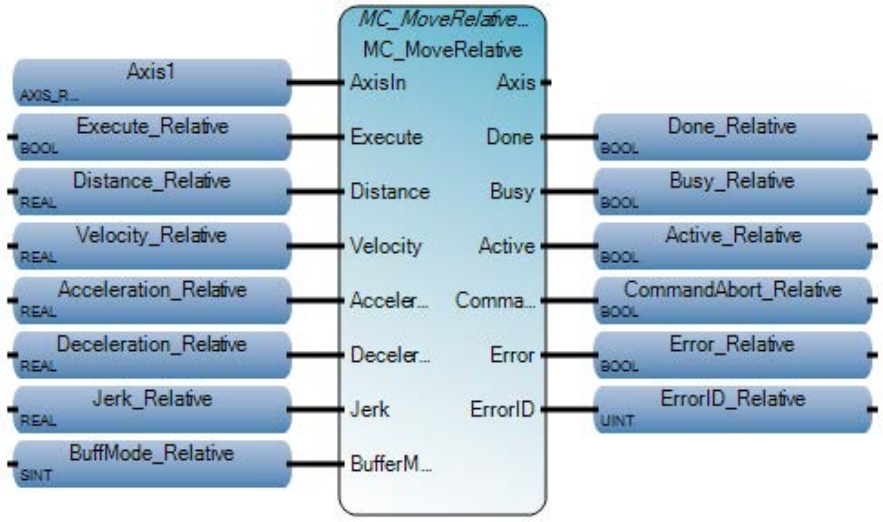
Exécution de MC_MoveRelative

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

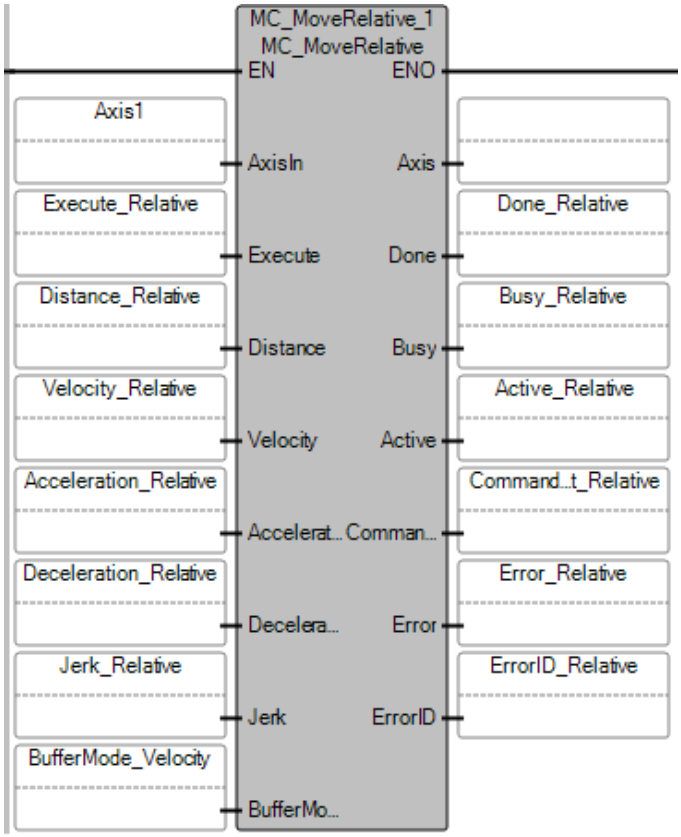
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_MoveRelative actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF sur la page 411	Utilisez les paramètres de type de données AXIS_REF pour définir AxisIn.
Execute	Entrée	BOOL	Indique quand démarrer le mouvement. VRAI : démarrage du mouvement au moment du front montant. FAUX : le mouvement ne démarre pas.
Distance	Entrée	REAL	Distance relative pour le mouvement (en unité technique [u]).
Velocity	Entrée	REAL	Valeur correspondant à la vitesse maximale (pas nécessairement atteinte) [u/s]. Etant donné que la direction du mouvement est déterminée par l'entrée Position, le signe de la valeur Velocity n'est pas pris en compte par le bloc de fonctions. Il est possible que la vitesse maximale ne soit pas atteinte lorsque la valeur Jerk = 0.
Acceleration	Entrée	REAL	Valeur correspondant à l'accélération (augmentation de la quantité d'énergie fournie au moteur) [u/s2]
Deceleration	Entrée	REAL	Valeur correspondant à la décélération (réduction de la quantité d'énergie fournie au moteur) [u/s2]
Jerk	Entrée	REAL	Valeur correspondant au jerk [u/s^3]
BufferMode	Entrée	SINT	Ce paramètre n'est pas utilisé.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.

Paramètre	Type de paramètre	Type de données	Description
Done	Sortie	BOOL	VRAI : distance commandée atteinte. Lorsque l'entrée In-Position est activée pour un axe, le signal In-Position doit être défini comme actif avant que Done = Vrai.
Busy	Sortie	BOOL	VRAI : le bloc d'instruction n'est pas terminé. FAUX : le bloc d'instruction est terminé.
Active	Sortie	BOOL	VRAI : indique que le bloc d'instruction contrôle l'axe.
CommandAborted	Sortie	BOOL	TRUE : la commande a été remplacée par une autre commande ou arrêtée suite à une erreur.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

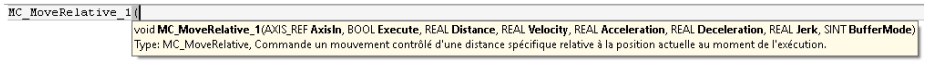
Exemple de Diagramme de Bloc Fonction MC_MoveRelative



Exemple de langage à relais MC_MoveRelative

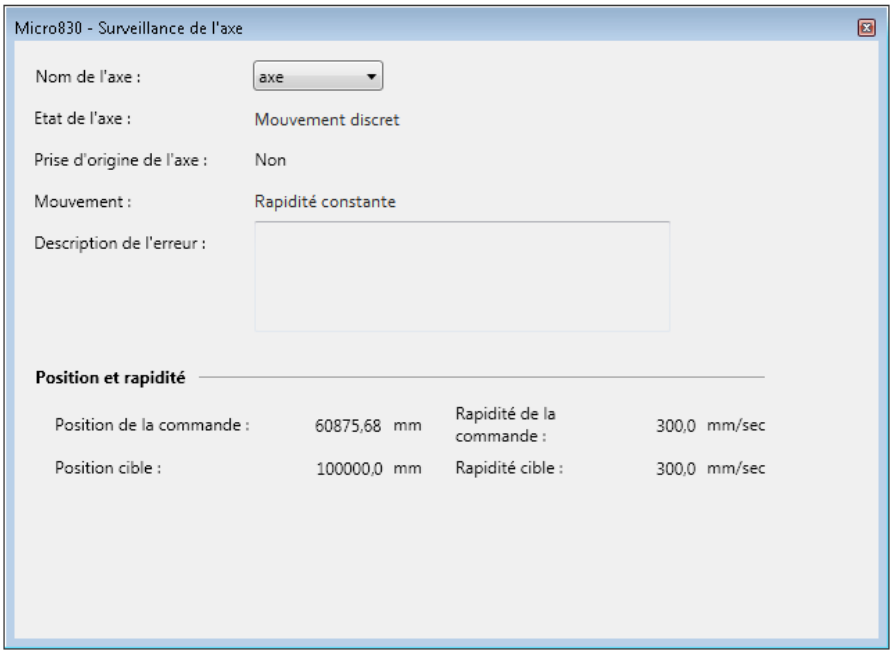


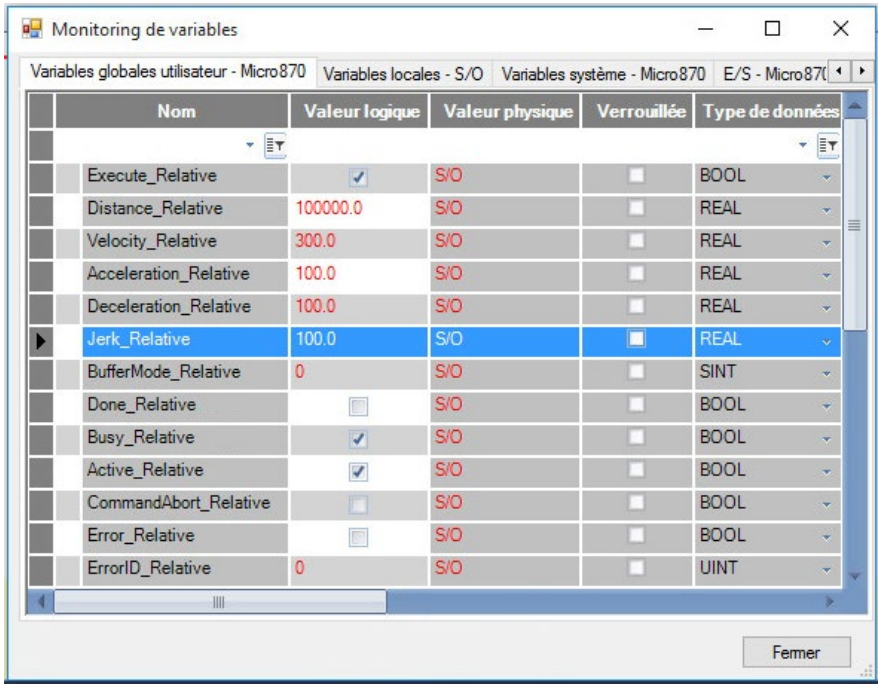
Exemple de texte structuré MC_MoveRelative




```
Distance_Relative := 100000.0;
Velocity_Relative := 300.0;
Acceleration_Relative := 100.0;
Deceleration_Relative := 100.0;
Jerk_Relative := 100.0;
MC_MoveRelative_1(Axis1,Execute_Relative,Distance_Relative,
Velocity_Relative,Acceleration_Relative,Deceleration_Relative,Jerk_Relative,BuffMode_Relative);
Done_Relative := MC_MoveRelative_1.Done;
Busy_Relative := MC_MoveRelative_1.Busy;
Active_Relative := MC_MoveRelative_1.Active;
CommandAbort_Relative := MC_MoveRelative_1.Commandaborted;
Error_Relative := MC_MoveRelative_1.Error;
ErrorID_Relative := MC_MoveRelative_1.ErrorID;
```

Résultats





MC_MoveVelocity (vitesse de déplacement du contrôle des mouvements)

Commande un mouvement contrôlé sans fin à une vélocité spécifiée.

Détails de l'opération :

- Si l'entrée DirectionIn de MC_MoveVelocity est égale à 0 alors que :
  - l'axe est en mouvement, le signe de l'entrée Vitesse est ignoré, l'axe poursuit son mouvement dans la direction en cours et de nouveaux paramètres dynamiques sont appliqués ;
  - l'axe n'est pas en mouvement, MC_MoveVolecity signale une erreur.
- Si la limite de pulsation PTO est atteinte pendant l'exécution de MC_MoveVelocity, la valeur de l'accumulateur PTO est ramenée à 0 (où

à la valeur opposée de la limite souple si elle est activée) et l'exécution se poursuit.

- Si l'axe est en mouvement alors que MC_MoveVelocity demande un mouvement dont la direction (signe du produit Vitesse * Direction) est opposée à celle du mouvement en cours, MC_MoveVelocity signale une erreur.
- Lorsque le signal « InVelocity » est activé, il indique que MC_MoveVelocity est terminé. Les éventuels événements de mouvement suivants n'ont aucun effet sur les sorties de MC_MoveVelocity, à l'exception du signal « InVelocity ».
- La sortie InVelocity de MC_MoveVelocity conserve la valeur Vrai dès que la valeur Vitesse de l'axe atteint la valeur Vitesse demandée ou jusqu'à ce que MC_MoveVelocity soit arrêté.
- Le signe du produit Vitesse * Direction détermine la direction des mouvements pour MC_MoveVelocity. Si Velocity et Direction sont de même signe, le mouvement commandé est positif. Si Velocity et Direction sont de signes contraires, le mouvement commandé est négatif.
- Le signal « InVelocity » est réinitialisé lorsque MC_MoveVelocity est remplacé par un autre bloc de fonctions ou événement d'axe, ou au moment du front descendant de « Execute ».
- Pour arrêter ou modifier le mouvement initié par MC_MoveVelocity, le bloc d'instruction doit être remplacé par un autre bloc d'instruction, notamment par une nouvelle exécution de MC_MoveVelocity avec des paramètres différents.
- Si MC_MoveVelocity est démarré alors que l'axe est à l'état Immobilisation (non commandé par un autre bloc de fonctions), toute erreur du bloc de fonctions attribue la valeur Arrêt suite à une erreur à l'état de l'axe.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

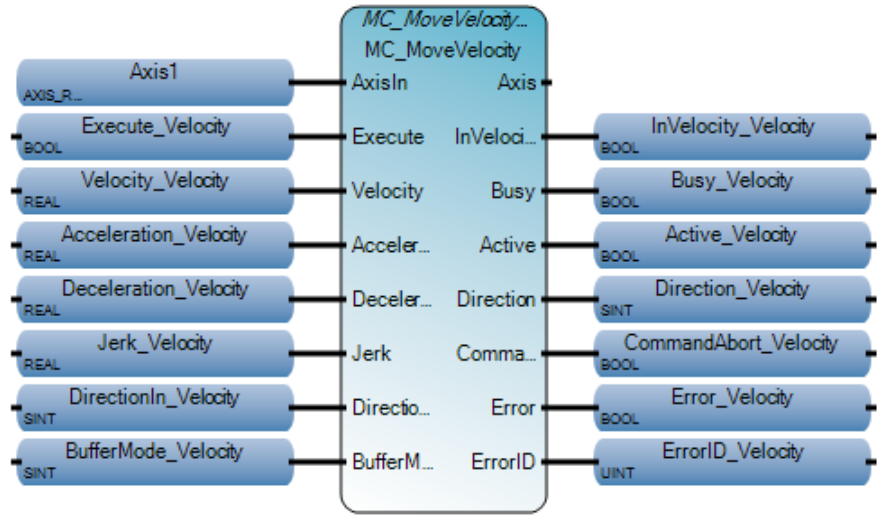
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

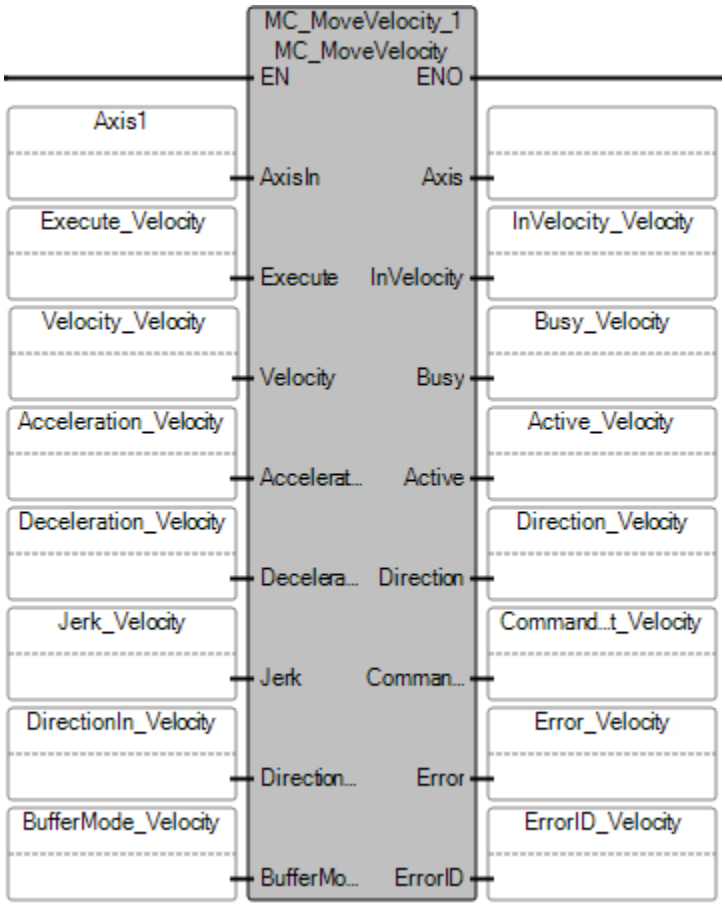
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_MoveVelocity actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF	Utilisez les paramètres de type de données AXIS_REF pour définir AxisIn.
Execute	Entrée	BOOL	Indique quand démarrer le mouvement. VRAI : démarrage du mouvement au moment du front montant. FAUX : le mouvement ne démarre pas.
Velocity	Entrée	REAL	Valeur correspondant à la vitesse maximale [u/s].
Acceleration	Entrée	REAL	Valeur correspondant à l'accélération (augmentation de la quantité d'énergie fournie au moteur) [u/s^2]
Deceleration	Entrée	REAL	Valeur correspondant à la décélération (réduction de la quantité d'énergie fournie au moteur) [u/s^2]
Jerk	Entrée	REAL	Valeur correspondant au jerk [u/s^3]
DirectionIn	Entrée	SINT	Les valeurs valides sont les suivantes : -1, 0, 1.
BufferMode	Entrée	SINT	Ce paramètre n'est pas utilisé.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF sur la page 411	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
InVelocity	Sortie	BOOL	VRAI : la vitesse commandée a été atteinte (pour la première fois).
Busy	Sortie	BOOL	VRAI : le bloc d'instruction n'est pas terminé. FAUX : le bloc d'instruction est terminé.
Active	Sortie	BOOL	VRAI : indique que le bloc de fonctions contrôle l'axe.
Direction	Sortie	SINT	Les valeurs valides sont les suivantes : -1, 0, 1.
CommandAborted	Sortie	BOOL	TRUE : la commande a été remplacée par une autre commande ou arrêtée suite à une erreur.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont

Paramètre	Type de paramètre	Type de données	Description
			définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Exemple de Diagramme de Bloc Fonction MC_MoveVelocity



Exemple de langage à relais MC_MoveVelocity



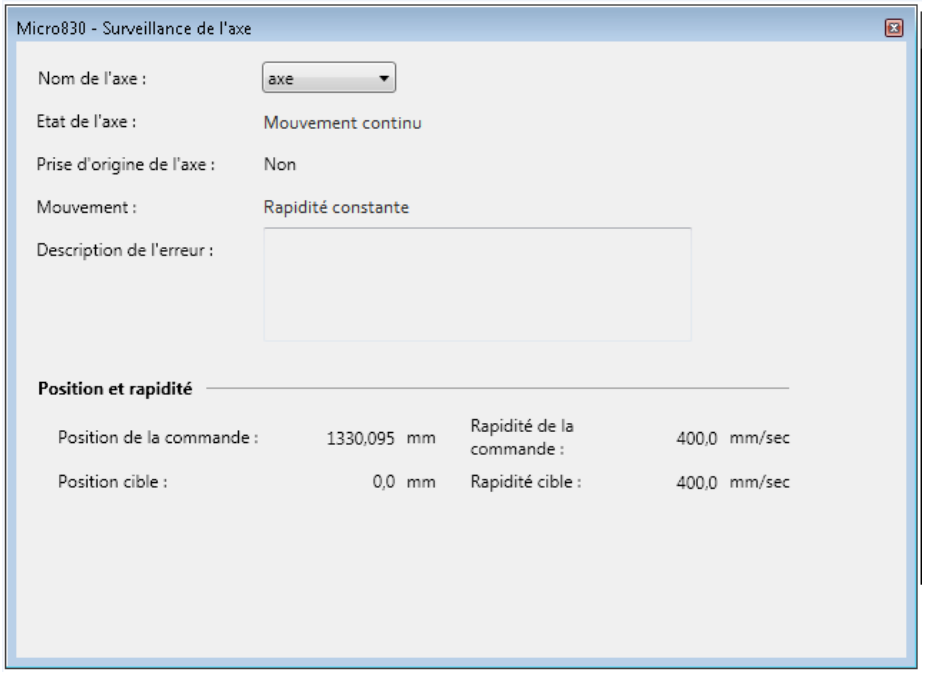
Exemple de texte structuré MC_MoveVelocity

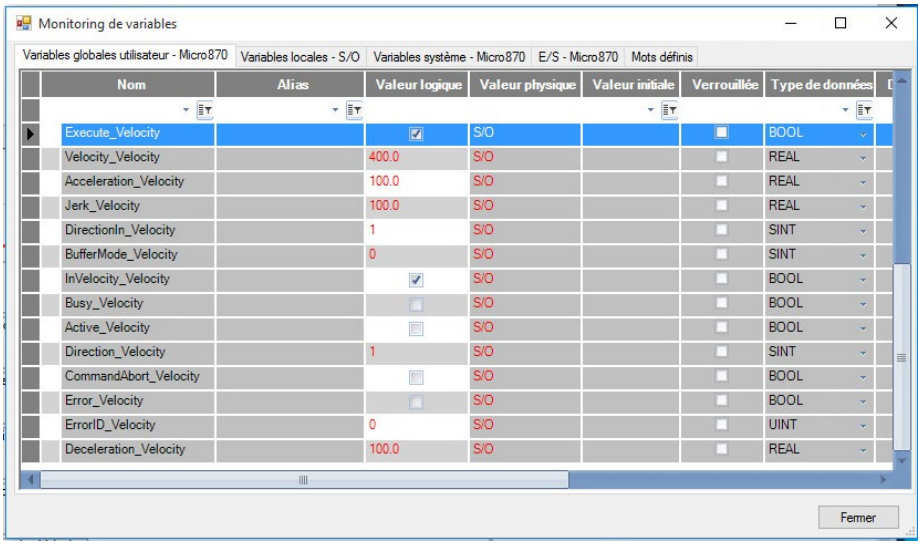
```
MC_MoveVelocity_1(
```

void **MC_MoveVelocity_1**(AXIS_REF AxisIn, BOOL Execute, REAL Velocity, REAL Acceleration, REAL Deceleration, REAL Jerk, SINT DirectionIn, SINT BufferMode)
Type : MC_MoveVelocity, Commands a never ending controlled motion at a specified velocity.

```
Velocity_Velocity := 400.0;
Acceleration_Velocity := 100.0;
Deceleration_Velocity := 100.0;
Jerk_Velocity := 100.0;
DirectionIn_Velocity := 1;
MC_MoveVelocity_1(Axis1,Execute_Velocity,Velocity_Velocity,
Acceleration_Velocity,Deceleration_Velocity,Jerk_Velocity,DirectionIn_Velocity,BufferMode_Velocity);
InVelocity_Velocity := MC_MoveVelocity_1.InVelocity;
Busy_Velocity := MC_MoveVelocity_1.Busy;
Active_Velocity := MC_MoveVelocity_1.Active;
Direction_Velocity := MC_MoveVelocity_1.Direction;
CommandAbort_Velocity := MC_MoveVelocity_1.CommandAborted;
Error_Velocity := MC_MoveVelocity_1.Error;
ErrorID_Velocity := MC_MoveVelocity_1.ErrorID;
```

Résultats





MC_Power (puissance du contrôle des mouvements)

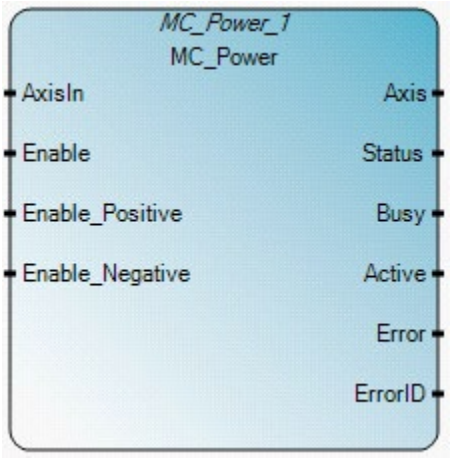
Contrôle la phase d'alimentation (ON ou OFF).

Détails de l'opération :

- Si vous importez dans CCW 8 un projet créé dans CCW 7, le nouveau paramètre d'entrée Mc_Power, __DTI_AxisIn, s'affiche. Si une erreur d'assemblage se produit, resélectionnez l'instruction et assemblez de nouveau.
- Après finalisation de la mise sous tension de l'axe, l'état Origine de l'axe est remis à zéro (hors position d'origine).
- L'entrée Enable_Positive et l'entrée Enable_Negative pour MC_Power sont toutes deux activées en fonction du niveau : elles sont vérifiées lorsque l'entrée Enable passe de OFF à ON. Les changements à la volée de l'entrée Enable_Positive et de l'entrée Enable_Negative sans bascule de l'entrée Enable ne sont pas vérifiés.
- En cas de panne d'alimentation pendant le fonctionnement (lorsque l'asservissement a été détecté), l'état de l'axe passe à ErrorStop.
- L'instruction MC_Power a une valeur de délai d'expiration de 2 minutes. MC_Power renvoie une erreur au moment de l'expiration du délai lorsque l'option Entrée servo variateur prêt est définie sur FALSE.
- Si un bloc de fonctions MC_Power avec le paramètre Activé défini sur Vrai est appelé alors que l'état de l'axe est Désactivé, l'état de l'axe passe à Immobilisation en l'absence d'erreur ou à Arrêt suite à une erreur dans le cas contraire.
- Un seul bloc de fonctions MC_Power doit être émis par axe. Si un bloc de fonctions MC_Power différent est utilisé pour commander simultanément le même axe, il sera rejeté par le moteur de mouvement.
- Lorsqu'un commutateur d'état Marche / Arrêt est associé à un axe, la position absolue de l'axe n'est pas réinitialisée.
- Si un bloc de fonctions MC_Power est appelé alors que le paramètre Activé est défini sur Faux, l'état de l'axe passe à Désactivé quel que soit l'état, même Arrêt suite à une erreur.
- Le bloc de fonctions MC_Power peut mettre l'axe sous tension si le paramètre Activé est défini sur Vrai et le mettre hors tension si ce paramètre est défini sur Faux.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

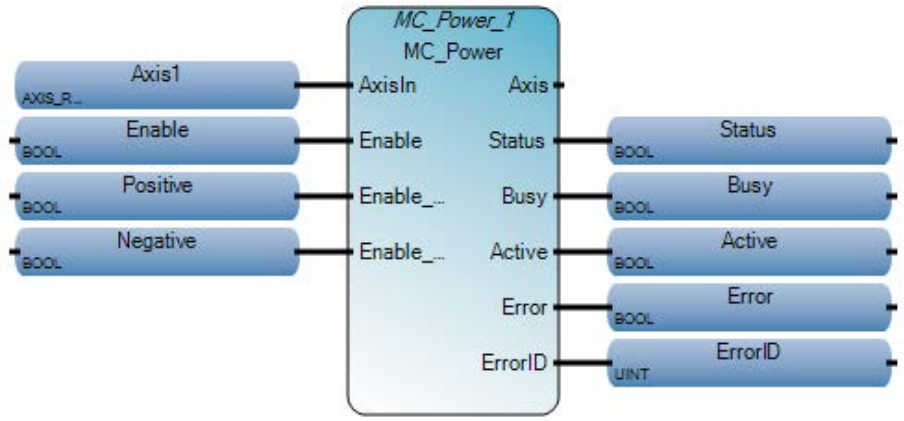
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



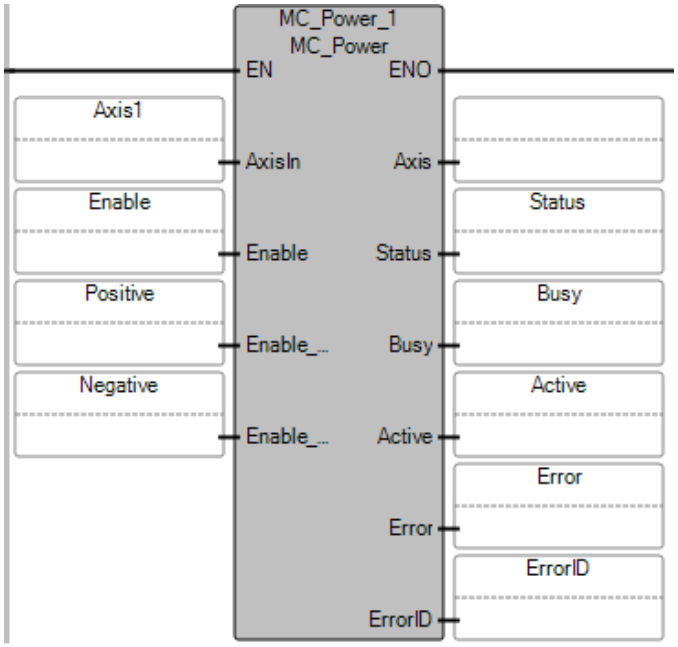
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_Power actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez les paramètres de type de données AXIS_REF sur la page 411 pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.
Enable	Entrée	BOOL	VRAI : mise sous tension activée. FAUX : mise sous tension désactivée.
Enable_Positive	Entrée	BOOL	TRUE : le mouvement est autorisé dans la direction positive.
Enable_Negative	Entrée	BOOL	TRUE : le mouvement est autorisé dans la direction négative.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule. Type de données AXIS_REF.
Status	Sortie	BOOL	Etat de la phase d'alimentation : • VRAI : le variateur est sous tension. • FAUX : le variateur n'est pas sous tension.
Busy	Sortie	BOOL	VRAI : le bloc d'instruction n'est pas terminé. FAUX : le bloc d'instruction est terminé.
Active	Sortie	BOOL	VRAI : indique que le bloc de fonctions contrôle l'axe.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Exemple de Diagramme de Bloc Fonction MC_Power



Exemple de langage à relais MC_Power



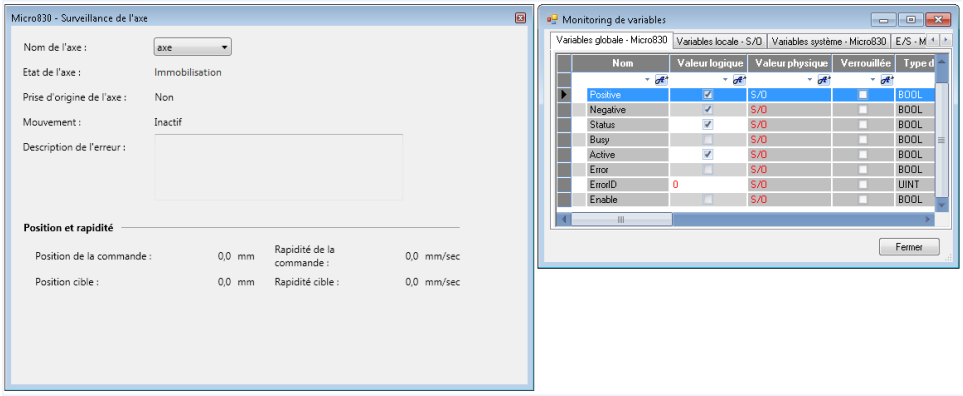
Exemple de texte structuré MC_Power

```
MC_Power_1(
```

void **MC_Power_1**([AXIS_REF|FB_AXIS_REF] **AxisIn**, BOOL **Enable**, BOOL **Enable_Positive**, BOOL **Enable_Negative**)
Type: MC_Power, Contrôle le statut du pouvoir (activé ou désactivé).

```
Positive := True;
Negative := True;
MC_Power_1(Axis1,Enable,Positive,Negative);
Status := MC_Power_1.Status;
Busy := MC_Power_1.Busy;
Active := MC_Power_1.Active;
Error := MC_Power_1.Error;
ErrorID := MC_Power_1.ErrorID;
```

Résultats



MC_ReadActualPosition (contrôle des mouvements, lecture de la position réelle)

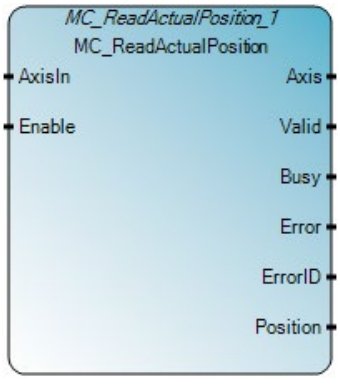
Renvoie la position réelle de l'axe retour codeur. MC_ReadActualPosition s'applique uniquement au mouvement de retour.

Détails de l'opération :

- Avant d'exécuter MC_ReadActualPosition, vérifiez que l'axe se trouve dans l'un des états suivants :
  - Désactivé
  - Immobilisation
  - Mouvement discret
  - Arrêt suite à une erreur
- La position réelle d'un axe retour codeur n'est pas réinitialisée sur 0 après un chargement. Pour réinitialiser ou effacer la position d'un axe retour codeur, utilisez l'instruction MC_Home ou MC_SetPosition.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

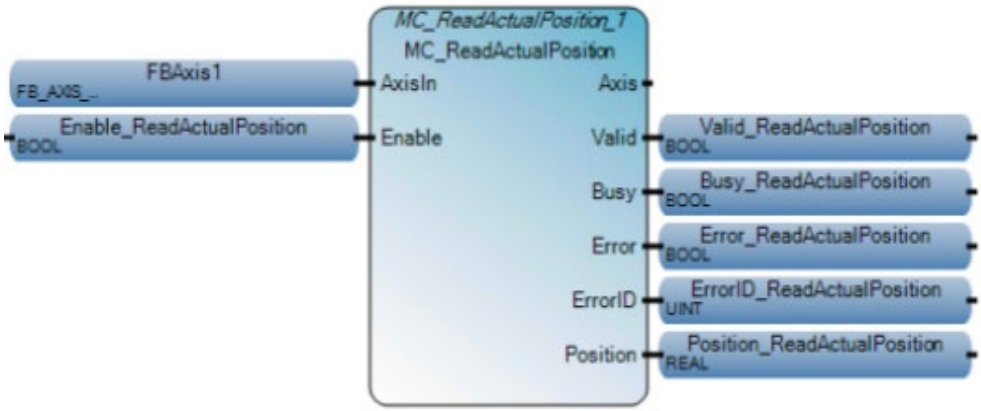


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

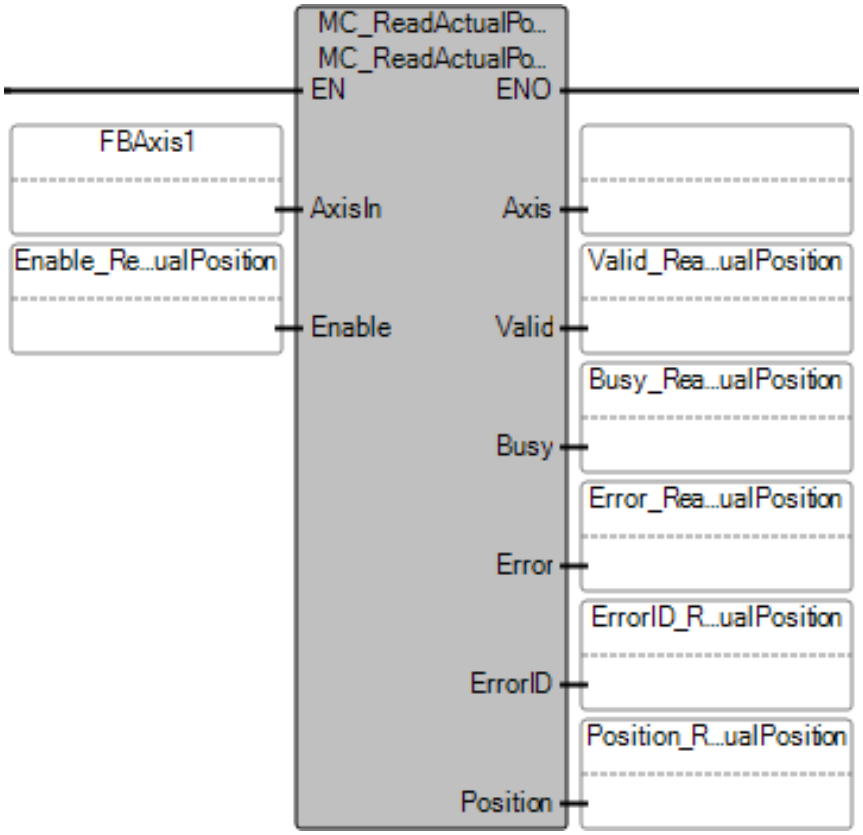
Paramètre	Type de paramètre	Type de données	Description
AxisIn	Entrée	FB_AXIS_REF	Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	VRAI : obtention constante de la valeur du paramètre si activé. FAUX : en attente.
Axis	Sortie	FB_AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule. Les paramètres de sortie de l'axe sont définis dans le type de données FB_AXIS_REF.
Valid	Sortie	BOOL	VRAI : le bloc d'instruction est actif et de nouvelles valeurs de sortie sont attendues. FAUX : le bloc d'instruction est inactif.
Busy	Sortie	BOOL	VRAI : le bloc d'instruction n'est pas terminé. FAUX : le bloc d'instruction est terminé.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UNIT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.
Position	Sortie	REAL	Valeur de la position absolue réelle pour l'axe de mouvement de retour codeur. (en unité d'axe [u])

Exemple de diagramme de bloc de fonctions MC_ReadActualPosition



Exemple de langage à relais MC_ReadActualPosition



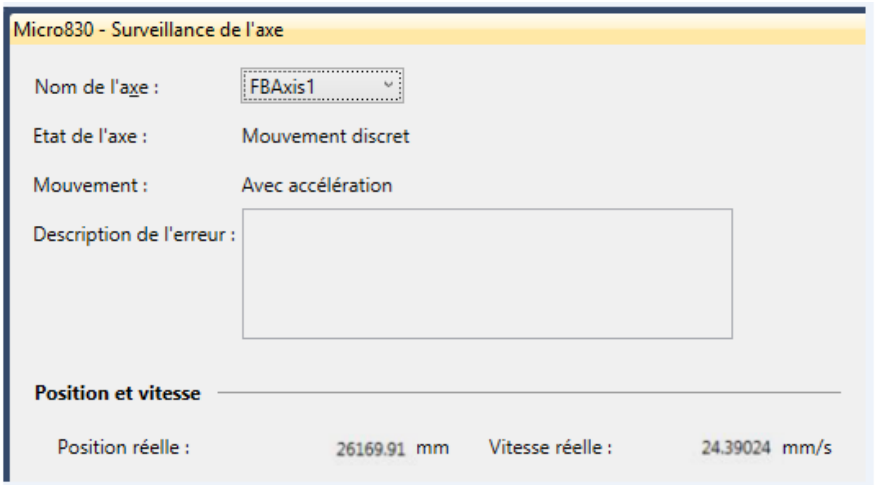
Exemple de texte structuré MC_ReadActualPosition

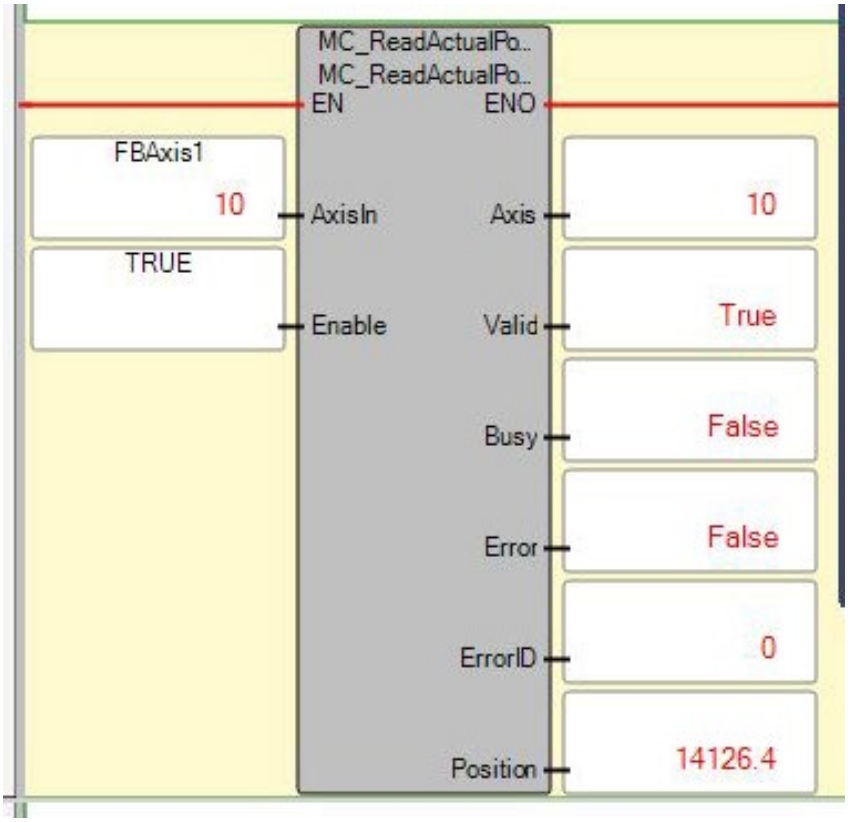
```
void MC_ReadActualPosition_1(
```

void **MC_ReadActualPosition_1**(FB_AXIS_REF **AxisIn**, BOOL **Enable**)
Type: MC_ReadActualPosition, Obtient la position actuelle pour un axe.

```
MC_ReadActualPosition_1(FBAxis, Enable_ReadActualPosition)
Valid_ReadActuralPosition := MC_ReadActualPosition_1.Valid
Busy_ReadActuralPosition := MC_ReadActualPosition_1.Busy
Error_Valid_ReadActuralPosition := MC_ReadActualPosition_1.Error
ErrorID_Valid_ReadActuralPosition := MC_ReadActualPosition_1.ErrorID
Position_Valid_ReadActuralPosition := MC_ReadActualPosition_1.Position
```

Résultats





	Nom	Alias	Valeur logique
-	MC_ReadActualPosition_2		...
+	MC_ReadActualPosition_2.AxisIn		...
	MC_ReadActualPosition_2.Enable		☑
+	MC_ReadActualPosition_2.Axis		...
	MC_ReadActualPosition_2.Valid		☑
	MC_ReadActualPosition_2.Busy		☐
	MC_ReadActualPosition_2.Error		☐
	MC_ReadActualPosition_2.ErrorID		0
	MC_ReadActualPosition_2.Position		14126.39

MC_ReadActualVelocity (contrôle des mouvements, lecture de la vitesse réelle)

Renvoie la valeur de la vitesse réelle de l'axe retour codeur.

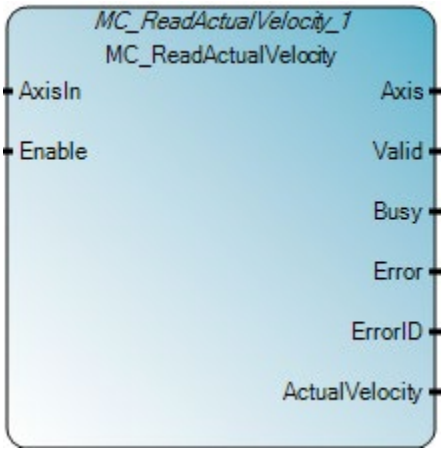
MC_ReadActualVelocity s'applique uniquement au mouvement de retour.

Avant d'exécuter MC_ReadActualVelocity, vérifiez que l'axe se trouve dans l'un des états suivants :

- Immobilisation
- Mouvement discret
- Arrêt suite à une erreur

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

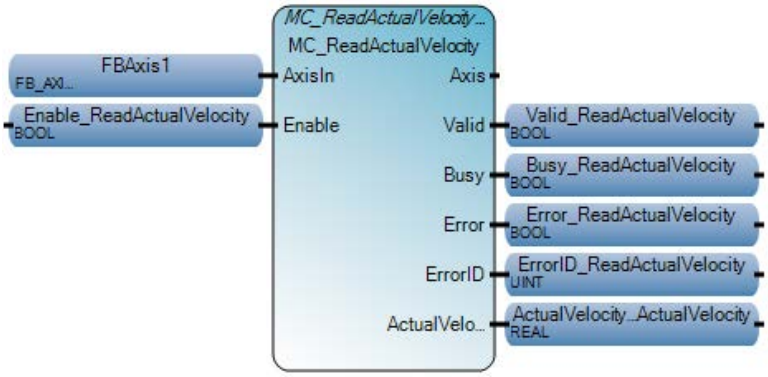


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

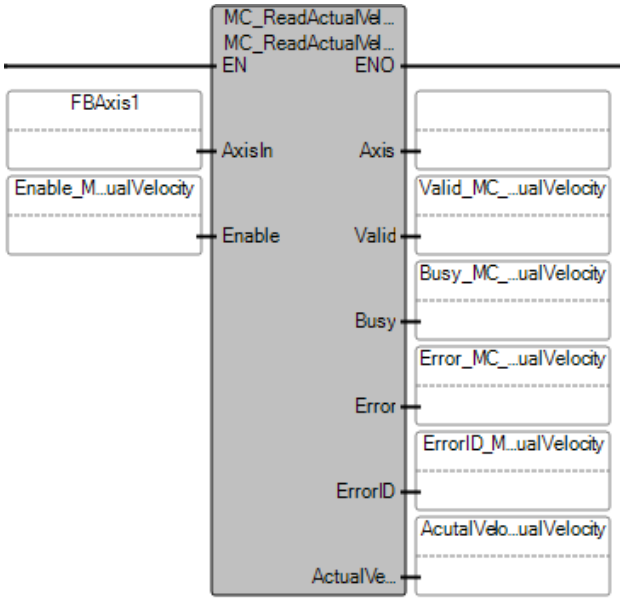
Paramètre	Type de paramètre	Type de données	Description
AxisIn	Entrée	FB_AXIS_REF	Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF pour définir AxisIn.
Enable	Entrée	BOOL	VRAI : obtention constante de la valeur de vitesse réelle. FAUX : les données ne sont plus valides, toutes les sorties sont réinitialisées sur 0, le paramètre Valide est défini sur Faux.
Axis	Sortie	FB_AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule. Les paramètres de sortie de l'axe sont définis dans le type de données FB_AXIS_REF sur la page 412.
Valid	Sortie	BOOL	VRAI : le bloc de fonctions est actif et de nouvelles valeurs de sortie sont attendues. FAUX : la fonction n'attend pas de nouvelles valeurs de sortie.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé. FAUX : la fonction est en attente.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UNIT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Paramètre	Type de paramètre	Type de données	Description
ActualVelocity	Sortie	REAL	Valeur de la vitesse réelle de l'axe retour codeur (en unité d'axe [u/s]). ActualVelocity est une valeur signée qui comprend les informations de direction.

Exemple de diagramme de bloc de fonctions MC_ReadActualVelocity



Exemple de langage à relais MC_ReadActualVelocity



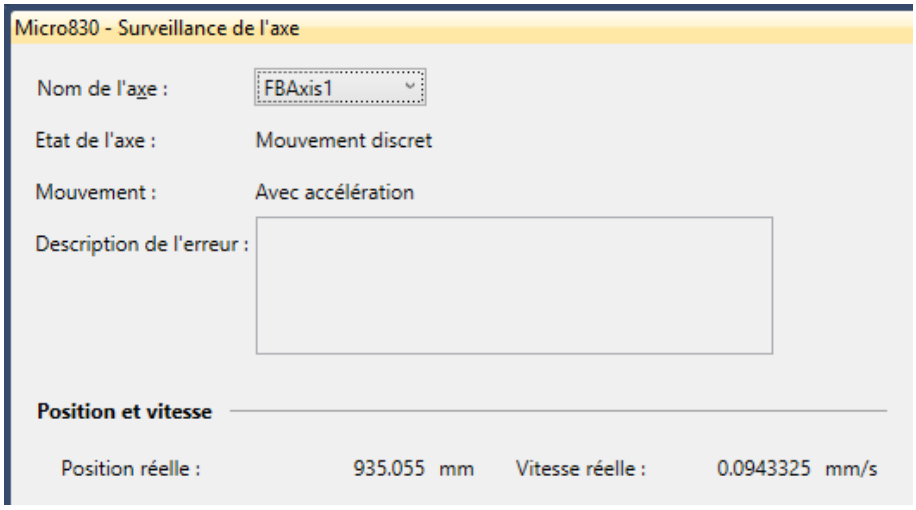
Exemple de texte structuré MC_ReadActualVelocity

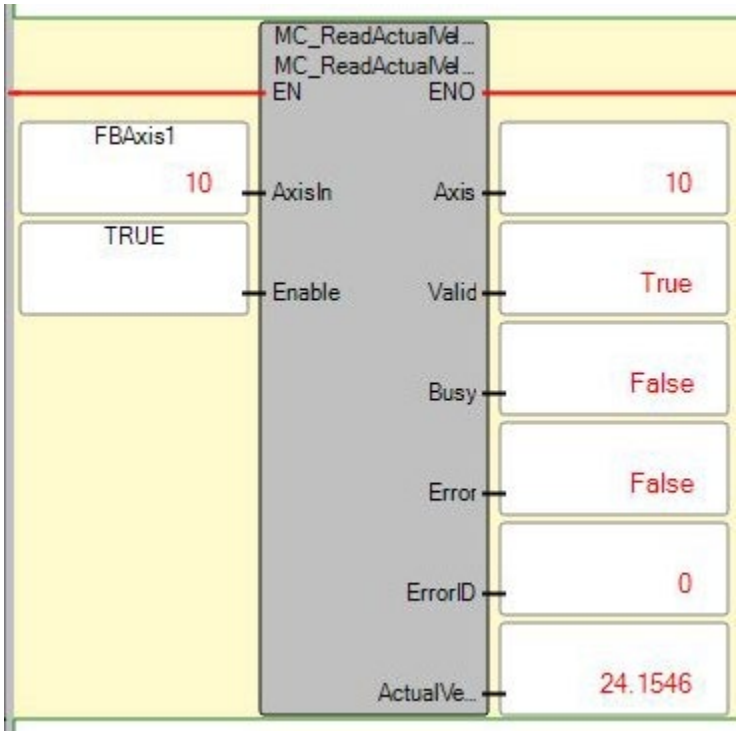
```
void MC_ReadActualVelocity_1(
```

void **MC_ReadActualVelocity_1**(FB_AXIS_REF **AxisIn**, BOOL **Enable**)
Type: MC_ReadActualVelocity, Obtient la vélocité actuelle pour un axe.

```
MC_ReadActualVelocity_1(FBAxis1,Enable_ReadActualVelocity)
Valid_ReadActualVelocity := MC_ReadActualVelocity_1.Valid
Busy_ReadActualVelocity := MC_ReadActualVelocity_1.Busy
Error_ReadActualVelocity := MC_ReadActualVelocity_1.Error
ErrorID_ReadActualVelocity := MC_ReadActualVelocity_1.ErrorID
Position_ReadActualVelocity := MC_ReadActualVelocity_1.ErrorID
```

Résultats





	Nom	Alias	Valeur logique
▶ -	MC_ReadActualPosition_1		...
+	MC_ReadActualPosition_1.AxisIn	Axis	...
	MC_ReadActualPosition_1.Enable	En	☐
+	MC_ReadActualPosition_1.Axis	Axis	...
	MC_ReadActualPosition_1.Valid	Val	☐
	MC_ReadActualPosition_1.Busy	Busy	☐
	MC_ReadActualPosition_1.Error	Err	☑
	MC_ReadActualPosition_1.ErrorID	ErID	4
	MC_ReadActualPosition_1.Position	Pos	0.0

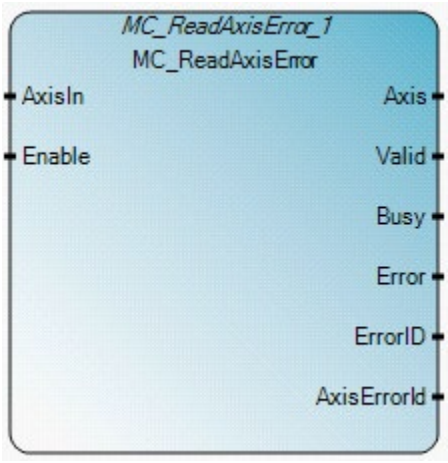
MC_ReadAxisError (contrôle des mouvements, lecture de l'erreur de l'axe)

Lit les erreurs de l'axe non liées aux blocs d'instruction de contrôle des mouvements.

Lorsqu'un axe est à l'état Désactivé, MC_ReadAxisError peut, ou non, obtenir un identifiant d'erreur non-nul pour l'axe, car un axe désactivé peut contenir des erreurs, ou non.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

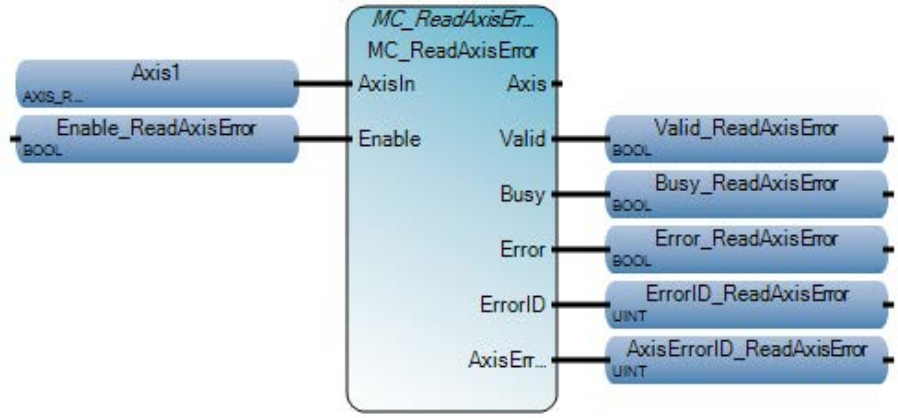


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

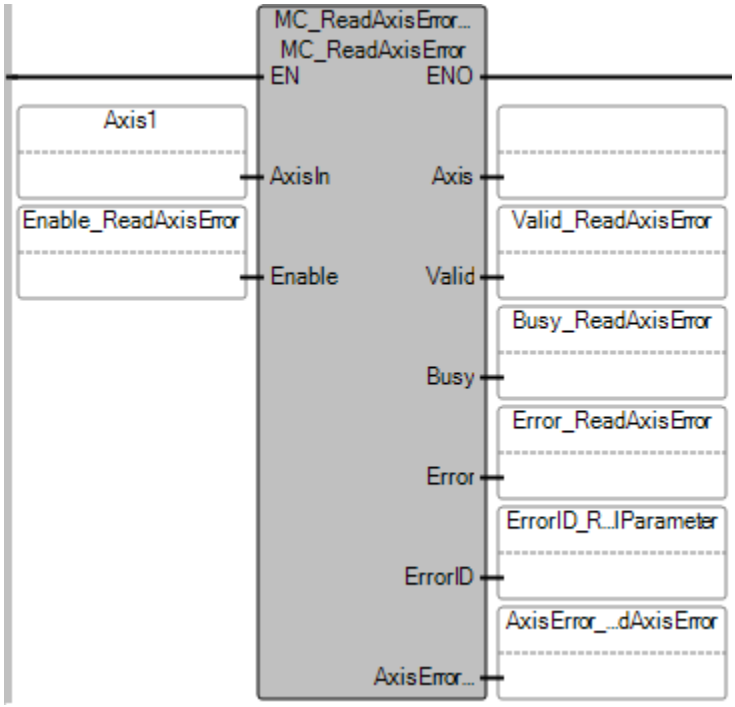
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_ReadAxisError actuel. FAUX : Error, ErrorID et AxisErrorID sont redéfinis sur Faux (ou 0). Ne s'applique qu'aux programmes LD.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez les paramètres de type de données AXIS_REF sur la page 411 pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF pour définir AxisIn.
Enable	Entrée	BOOL	VRAI : obtention constante de la valeur du paramètre si activé. FAUX : réinitialisation des sorties Error, ErrorID et AxisErrorID sur 0.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule. Les paramètres de sortie de l'axe sont définis dans le type de données FB_AXIS_REF sur la page 412.
Valid	Sortie	BOOL	VRAI : le bloc d'instruction est actif et de nouvelles valeurs de sortie sont attendues. FAUX : le bloc d'instruction est inactif.
Busy	Sortie	BOOL	VRAI : le bloc d'instruction n'est pas terminé. FAUX : le bloc d'instruction est terminé.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.

Paramètre	Type de paramètre	Type de données	Description
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements.
AxisErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur de l'axe. Les erreurs pour cette instruction sont définies dans les codes d'erreur AxisErrorID sur la page 447.

Exemple de diagramme de bloc de fonctions MC_ReadAxisError



Exemple de langage à relais MC_ReadAxisError



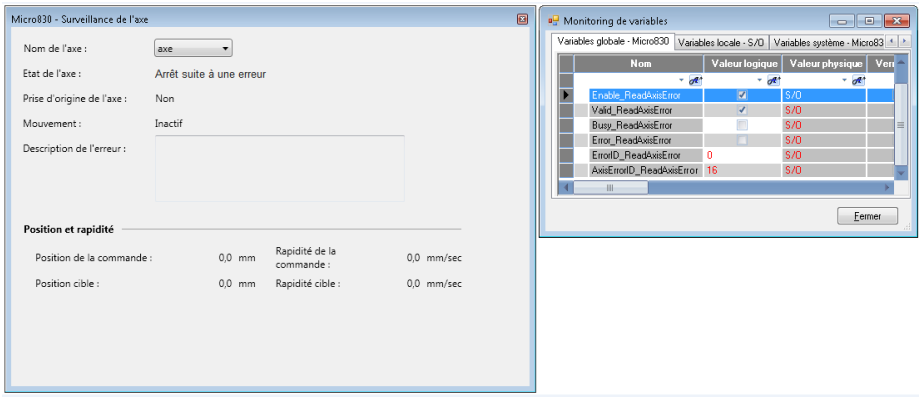
Exemple de texte structuré MC_ReadAxisError

```
void MC_ReadAxisError_1(
```

void **MC_ReadAxisError_1**([AXIS_REF|FB_AXIS_REF] **AxisIn**, BOOL **Enable**)
Type: MC_ReadAxisError, Lecture de l'information d'erreur pour un axe.

```
MC_ReadAxisError_1(Axis1,Enable_ReadAxisError);
Valid_ReadAxisError := MC_ReadAxisError_1.Valid;
Busy_ReadAxisError := MC_ReadAxisError_1.Busy;
Error_ReadAxisError := MC_ReadAxisError_1.Error;
ErrorID_ReadAxisError := MC_ReadAxisError_1.ErrorID;
AxisErrorID_ReadAxisError := MC_ReadAxisError_1.AxisErrorId;
```

Résultats



Codes d'erreur AxisErrorID

Le tableau suivant décrit les codes d'erreur de l'axe identifiés dans AxisErrorID pour MC_ReadAxisError sur la page 445.

Valeur	ID DE MACRO	Description
00	MC_FB_ERR_NO	L'axe est dans un état opérationnel (rien à afficher).
01	MC_FB_ERR_WRONG_STATE	L'axe n'est pas opérationnel car un état d'axe incorrect a été détecté au cours de l'exécution d'un bloc de fonctions. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset.
02	MC_FB_ERR_RANGE	L'axe n'est pas opérationnel car un paramètre dynamique non valide (vitesse, accélération, décélération ou jerk) a été défini au niveau de l'axe dans un bloc de fonctions. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Dans le bloc de fonctions, corrigez le ou les paramètres dynamiques en conflit avec ceux indiqués sur la page de configuration relative à la dynamique de l'axe.
03	MC_FB_ERR_PARAM	L'axe n'est pas opérationnel car un paramètre non valide (autre que la vitesse, l'accélération, la décélération ou le jerk) a été défini dans un bloc fonctionnel. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Dans le bloc fonctionnel, corrigez la configuration des paramètres, comme le mode ou la position.
04	MC_FB_ERR_AXISNUM	Défaut interne du mouvement, identifiant de l'erreur = 0x04. Contactez le représentant local du support technique Rockwell Automation. Pour obtenir des informations sur les coordonnées, consultez le site suivant : http://www.rockwellautomation.com/support
05	MC_FB_ERR_MECHAN	L'axe n'est pas opérationnel car un problème mécanique ou au niveau du variateur a été détecté. Vérifiez la connexion entre le variateur et le contrôleur (signaux relatifs à la disponibilité du variateur et à la position) et assurez-vous que le variateur fonctionne normalement. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset.

Valeur	ID DE MACRO	Description
06	MC_FB_ERR_ NOPOWER	L'axe n'est pas sous tension. Mettez l'axe sous tension à l'aide du bloc de fonctions MC_Power. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Reset.
07	MC_FB_ERR_ RESOURCE	L'axe n'est pas opérationnel parce que ses ressources associées, requises par un autre bloc de fonctions, ou lui-même sont contrôlés par un autre bloc de fonctions ou ne sont pas disponibles. Vérifiez que l'axe ou ses ressources associées requises par le bloc de fonctions sont disponibles et qu'ils peuvent être utilisés. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Passez en revue l'application et corrigez-la si plusieurs instances du même bloc de fonctions tentent de contrôler l'axe au même moment.
08	MC_FB_ERR_ PROFILE	L'axe n'est pas opérationnel car le profil de mouvement défini dans un bloc de fonctions n'est pas valide. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Corrigez le profil dans le bloc de fonctions.
09	MC_FB_ERR_ VELOCITY	L'axe n'est pas opérationnel car le profil de mouvement demandé par un bloc de fonctions est en conflit avec la vitesse actuelle de l'axe. Causes possibles : • Le bloc fonctionnel demande à l'axe de changer de direction alors que l'axe est en mouvement. • La vitesse actuelle est trop faible ou trop élevée pour le profil de mouvement requis. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Corrigez le profil de mouvement dans le bloc fonctionnel ou exécutez de nouveau le bloc fonctionnel lorsque la vitesse de l'axe est compatible avec le profil de mouvement demandé.
10	MC_FB_ERR_ SOFT_LIMIT	L'axe n'est pas opérationnel, car une erreur a été détectée au niveau de la limite logicielle ou que l'exécution du bloc de fonctions entraînerait une telle erreur. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Vérifiez le paramétrage de la vitesse ou de la position cible dans le bloc de fonctions ou réglez le paramètre relatif à la limite logicielle.
11	MC_FB_ERR_ HARD_LIMIT	L'axe n'est pas opérationnel car une erreur a été détectée au niveau de la limite matérielle. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Reset, puis éloignez l'axe du commutateur de limite matérielle dans la direction opposée.
12	MC_FB_ERR_ LOG_LIMIT	L'axe n'est pas opérationnel car une erreur a été détectée au niveau de la limite logique de l'accumulateur PTO ou l'exécution du bloc de fonctions entraînerait une erreur à ce niveau. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Vérifiez le paramétrage de la vitesse ou de la position cible pour le bloc de fonctions. Utilisez le bloc de fonctions MC_SetPosition pour régler le système de coordonnées de l'axe.
13	MC_FB_ERR_ ERR_ENGINE	L'axe n'est pas opérationnel car une erreur a été détectée au niveau de l'exécution du moteur de mouvement. Arrêtez, puis remettez sous tension l'ensemble de la machine et téléchargez de nouveau en aval l'application utilisateur vers le contrôleur. Si le défaut persiste, contactez votre représentant local du Support technique Rockwell Automation. Pour obtenir des informations sur les coordonnées, consultez le site suivant : http://www.rockwellautomation.com/support.
16	MC_FB_ERR_ NOT_HOMED	L'axe n'est pas opérationnel car il n'a pas repris sa position d'origine. Réinitialisez l'état de l'axe à l'aide du bloc de fonctions MC_Power et MC_Reset. Exécutez la prise d'origine de l'axe à l'aide du bloc de fonctions MC_Home.

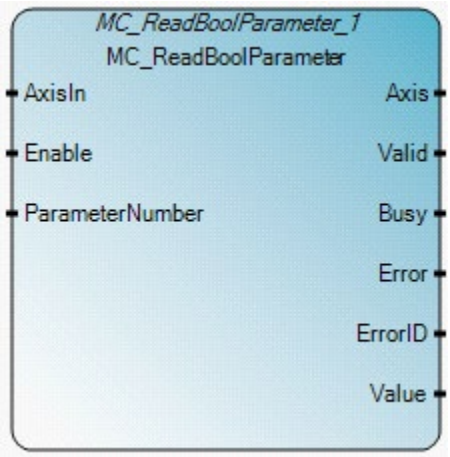
Valeur	ID DE MACRO	Description
128	MC_FB_ERR_ PARAM_MODIFIED	Avertissement interne relatif au mouvement, identifiant d'avertissement = 0x80. Contactez le représentant local du support technique Rockwell Automation. Pour obtenir des informations sur les coordonnées, consultez le site suivant : http://www.rockwellautomation.com/support.

MC_ReadBoolParameter (contrôle des mouvements, lecture du paramètre BOOL)

Renvoie la valeur d'un paramètre de type BOOL spécifique à un fournisseur.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

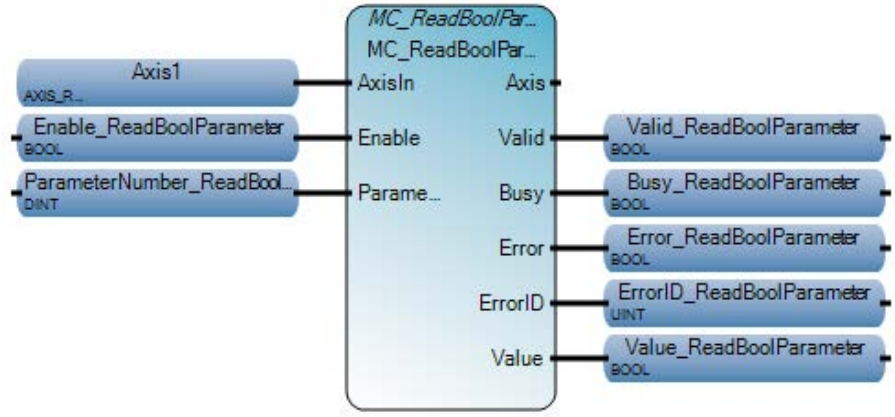


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

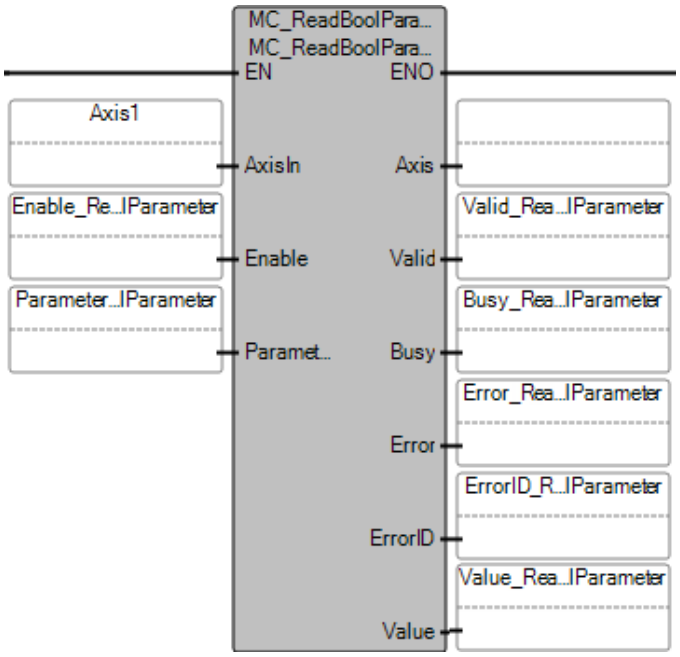
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_ReadBoolParameter actuel. FAUX : réinitialisation de la sortie Valeur sur 0. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF sur la page 411 pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.
Enable	Entrée	BOOL	VRAI : obtention constante de la valeur du paramètre si activé. FAUX : la sortie Valeur est réinitialisée sur 0.
ParameterNumber	Entrée	DINT	Identification du paramètre. Les définitions de numéros des paramètres sont déterminées dans les numéros des paramètres de bloc de fonctions du contrôle des mouvements.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Valid	Sortie	BOOL	VRAI : la valeur du paramètre est disponible. FAUX : la valeur du paramètre n'est pas disponible.

Paramètre	Type de paramètre	Type de données	Description
Busy	Sortie	BOOL	VRAI : le bloc de fonctions est actif et de nouvelles valeurs de sortie sont attendues. FAUX : la fonction est en attente.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.
Value	Sortie	BOOL	Valeur du paramètre spécifié dans le type de donnée, comme indiqué par le fournisseur.

Exemple de diagramme de bloc de fonctions MC_ReadBoolParameter



Exemple de langage à relais MC_ReadBoolParameter



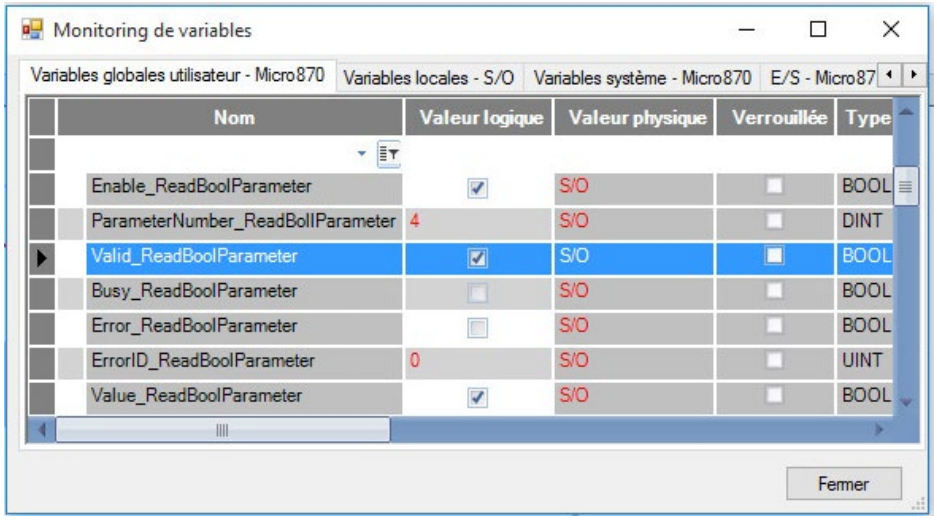
Exemple de texte structuré MC_ReadBoolParameter

```
MC_ReadBoolParameter_1(
```

void **MC_ReadBoolParameter_1**([AXIS_REF|FB_AXIS_REF] **AxisIn**, BOOL **Enable**, DINT **ParameterNumber**)
Type: MC_ReadBoolParameter, Obtient la valeur d'un paramètre BOOL de mouvement.

```
ParameterNumber_ReadBoolParameter := 4;
MC_ReadBoolParameter_1(Axis1,Enable_ReadBoolParameter,ParameterNumber_ReadBoolParameter);
Valid_ReadBoolParameter := MC_ReadBoolParameter_1.Valid;
Busy_ReadBoolParameter := MC_ReadBoolParameter_1.Busy;
Error_ReadBoolParameter := MC_ReadBoolParameter_1.Error;
ErrorID_ReadBoolParameter := MC_ReadBoolParameter_1.ErrorID;
Value_ReadBoolParameter := MC_ReadBoolParameter_1.Value;
```

Résultats

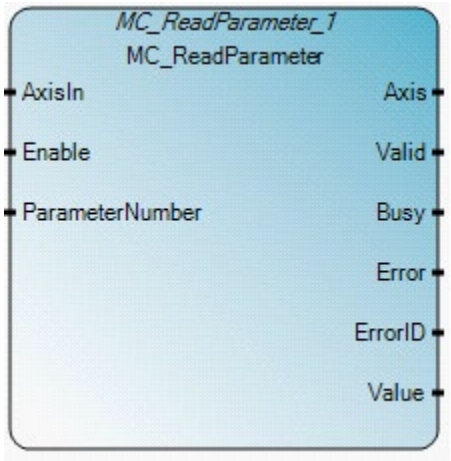


MC_ReadParameter (contrôle des mouvements, lecture d'un paramètre)

Renvoie la valeur d'un paramètre spécifique d'un fournisseur pour un type de données REAL.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

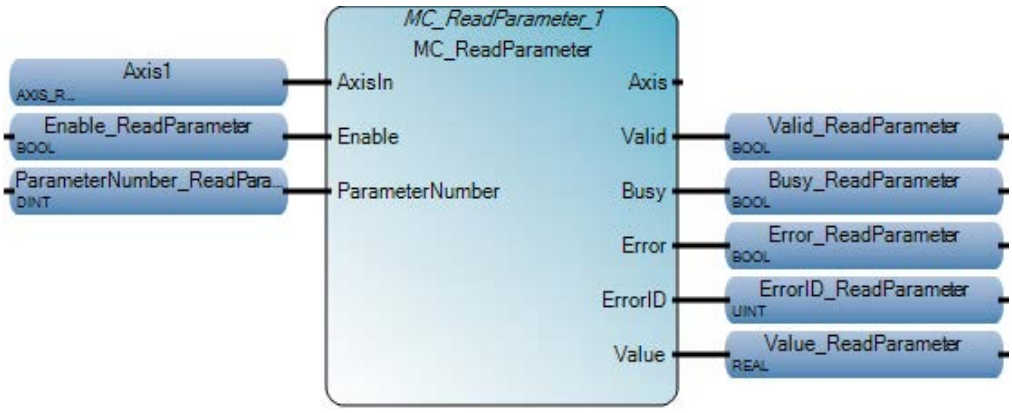
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



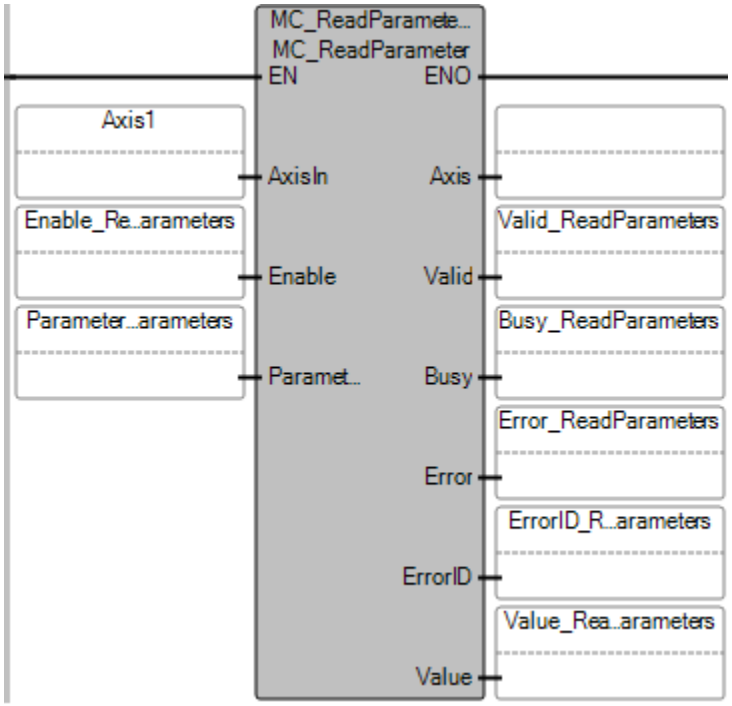
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_ReadParameter actuel. FAUX : la sortie Valeur est réinitialisée sur 0. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF sur la page 411 pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.
Enable	Entrée	BOOL	VRAI : obtention constante de la valeur du numéro de paramètre. FAUX : la sortie Valeur est réinitialisée sur 0.
ParameterNumber	Entrée	DINT	Identification du paramètre. Les définitions de numéros des paramètres sont déterminées dans les numéros des paramètres de bloc de fonctions du contrôle des mouvements.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Valid	Sortie	BOOL	VRAI : les sorties valides sont disponibles. FAUX : les sorties valides ne sont pas disponibles.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions est actif et de nouvelles valeurs de sortie sont attendues. FAUX : le bloc de fonctions est en attente.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.
Value	Sortie	REAL	Valeur du paramètre spécifié dans le type de donnée, comme indiqué par le fournisseur.

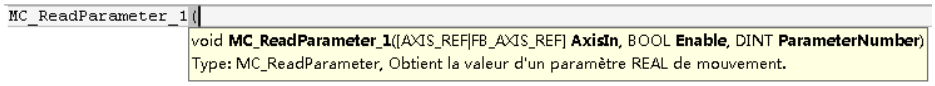
Exemple de diagramme de bloc de fonctions MC_ReadParameter



Exemple de langage à relais MC_ReadParameter

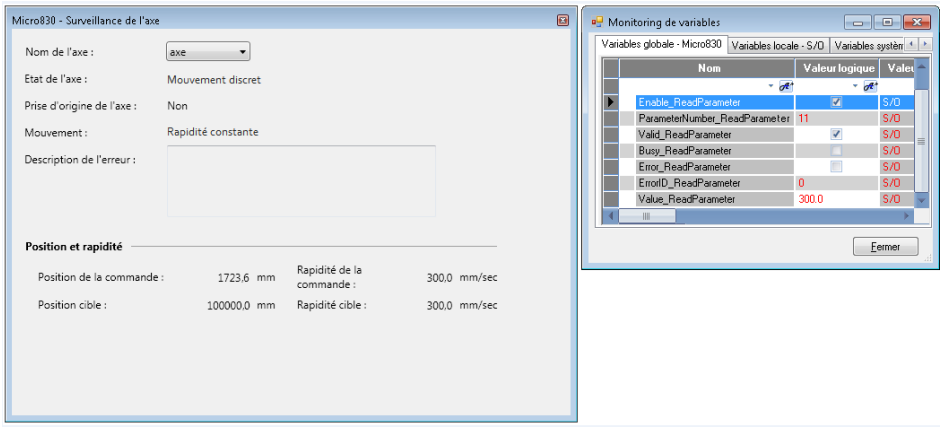


Exemple de t**exte structuré MC_ReadParameter**




```
ParameterNumber_ReadParameter := 11;
MC_ReadParameter_1(Axis1,Enable_ReadParameter,ParameterNumber_ReadParameter);
Valid_ReadParameter := MC_ReadParameter_1.Valid;
Busy_ReadParameter := MC_ReadParameter_1.Busy;
Error_ReadParameter := MC_ReadParameter_1.Error;
ErrorID_ReadParameter := MC_ReadParameter_1.ErrorID;
Value_ReadParameter := MC_ReadParameter_1.Value;
```

Résultats

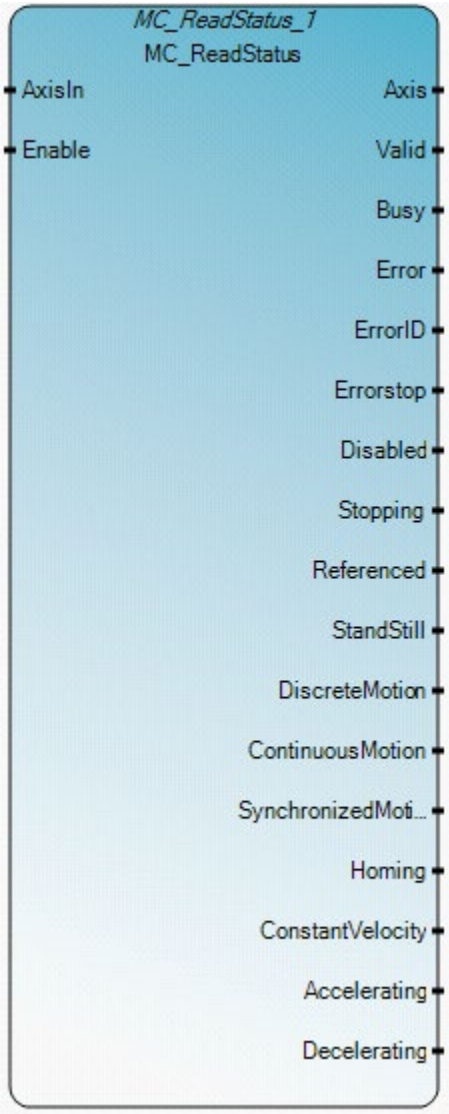


MC_ReadStatus (contrôle des mouvements, lecture d'un état)

Renvoie l'état de l'axe en fonction du mouvement actuellement en cours.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

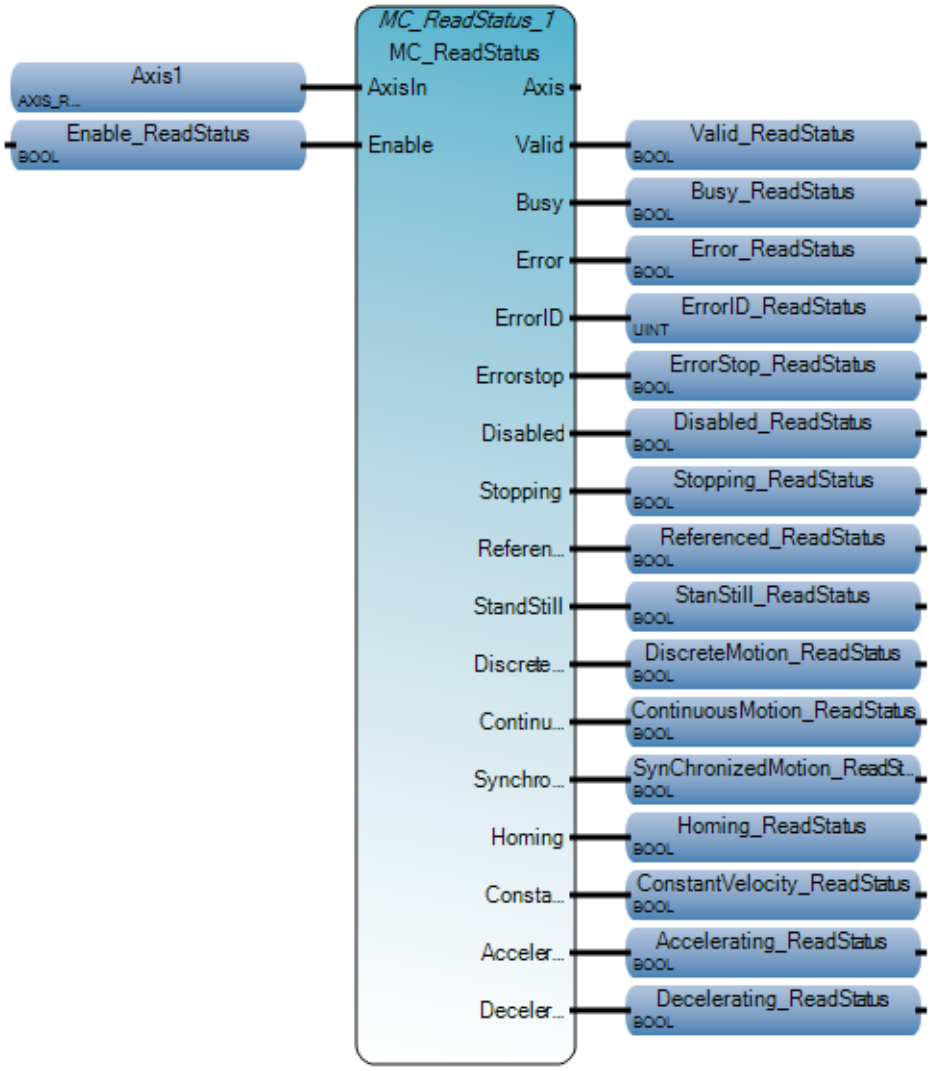


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

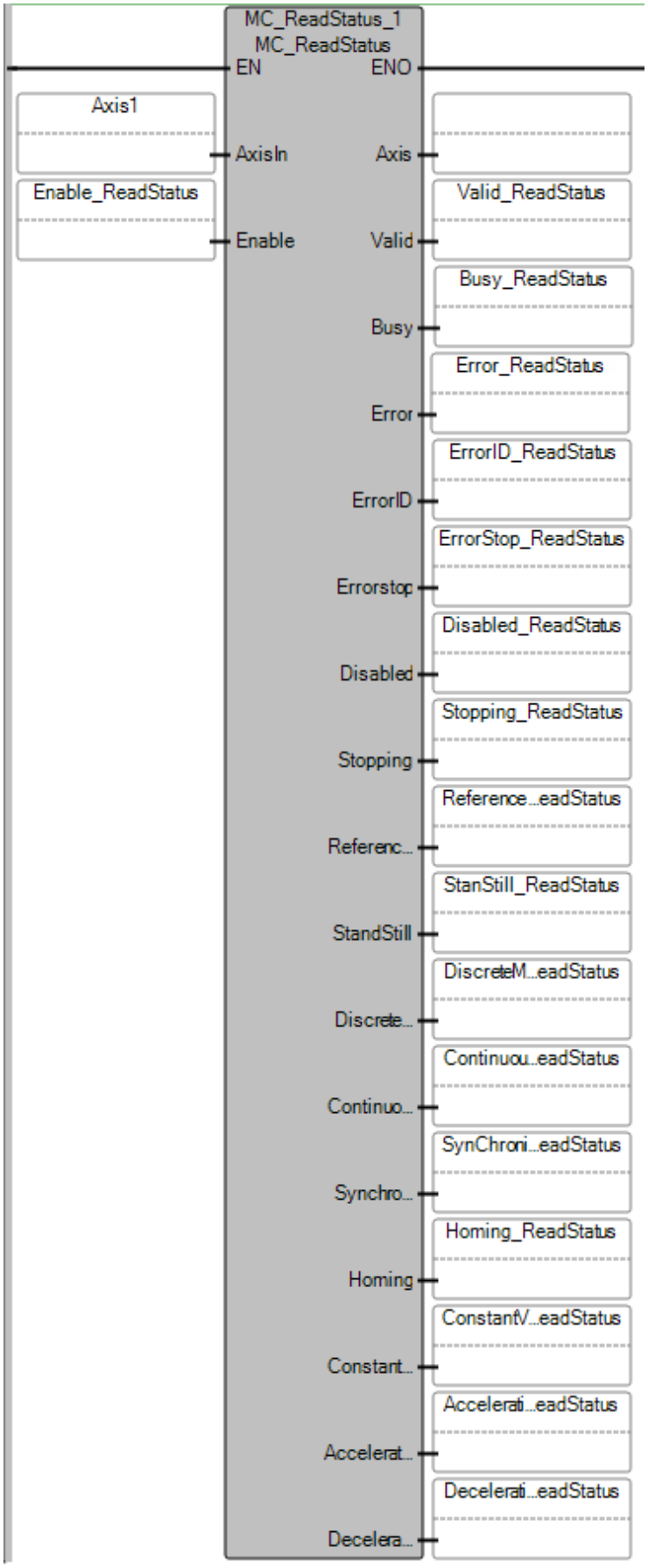
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_ReadStatus actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez les paramètres de type de données AXIS_REF sur la page 411 pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.
Enable	Entrée	BOOL	VRAI : obtention constante de l'état de l'axe. FAUX : toutes les sorties d'état sont réinitialisées sur 0.

Paramètre	Type de paramètre	Type de données	Description
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Valid	Sortie	BOOL	VRAI : les sorties valides sont disponibles. FAUX : sorties indisponibles.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions est actif et de nouvelles valeurs de sortie sont attendues. FAUX : le bloc de fonctions est en attente.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements.
ErrorStop	Sortie	BOOL	VRAI : l'état de l'axe correspond à Arrêt suite à une erreur. Les états de l'axe sont définis dans les noms et les valeurs des états de l'axe du contrôle des mouvements.
Disabled	Sortie	BOOL	VRAI : l'état de l'axe correspond à Désactivé.
Stopping	Sortie	BOOL	VRAI : l'état de l'axe correspond à Arrêt.
Referenced	Sortie	BOOL	VRAI : l'état de l'axe a repris sa position d'origine, la position de référence absolue est connue pour l'axe.
StandStill	Sortie	BOOL	VRAI : l'état de l'axe correspond à Immobilisation.
DiscreteMotion	Sortie	BOOL	VRAI : l'état de l'axe correspond à Mouvement discret.
ContinuousMotion	Sortie	BOOL	VRAI : l'état de l'axe correspond à Mouvement continu.
SynchronizedMotion	Sortie	BOOL	Les mouvements synchronisés ne sont pas pris en charge par les contrôleurs Micro800. VRAI : jamais vrai. FAUX : toujours faux.
Homing	Sortie	BOOL	VRAI : l'état de l'axe correspond à Prise d'origine.
ConstantVelocity	Sortie	BOOL	VRAI : la vitesse du moteur est constante.
Accelerating	Sortie	BOOL	VRAI : l'axe accélère, l'énergie fournie au moteur augmente.
Decelerating	Sortie	BOOL	VRAI : l'axe ralentit, l'énergie fournie au moteur diminue.

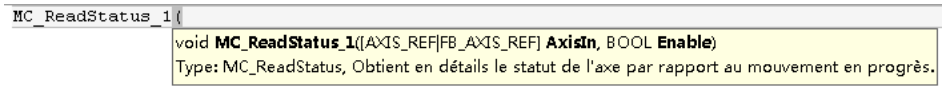
Exemple de diagramme de bloc de fonctions MC_ReadStatus



Exemple de langage à relais MC_ReadStatus

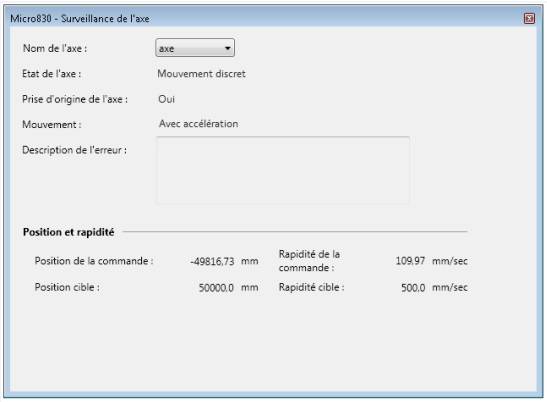


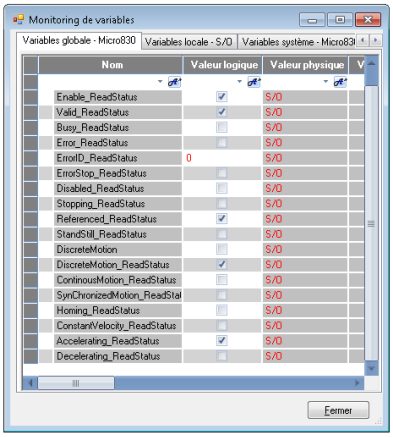
Exemple de texte structuré MC_ReadStatus




```
MC_ReadStatus_1(Axis1,Enable_ReadStatus);
Valid_ReadStatus := MC_ReadStatus_1.Valid;
Busy_ReadStatus := MC_ReadStatus_1.Busy;
Error_ReadStatus := MC_ReadStatus_1.Error;
ErrorID_ReadStatus := MC_ReadStatus_1.ErrorID;
ErrorStop_ReadStatus := MC_ReadStatus_1.Errorstop;
Disabled_ReadStatus := MC_ReadStatus_1.Disabled;
Stopping_ReadStatus := MC_ReadStatus_1.Stopping;
Referenced_ReadStatus := MC_ReadStatus_1.Referenced;
StanStill_ReadStatus := MC_ReadStatus_1.StandStill;
DiscreteMotion_ReadStatus := MC_ReadStatus_1.DiscreteMotion;
ContinuousMotion_ReadStatus := MC_ReadStatus_1.ContinuousMotion;
SynChronizedMotion_ReadStatus := MC_ReadStatus_1.SynchronizedMotion;
Homing_ReadStatus := MC_ReadStatus_1.Homing;
ConstantVelocity_ReadStatus := MC_ReadStatus_1.ConstantVelocity;
Accelerating_ReadStatus := MC_ReadStatus_1.Accelerating;
Decelerating_ReadStatus := MC_ReadStatus_1.Decelerating;
```

Résultats





MC_Reset (contrôle des mouvements, réinitialisation)

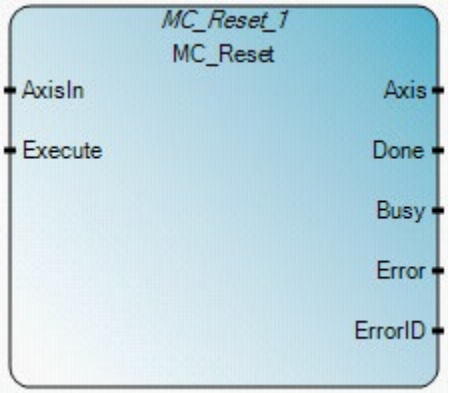
Fait passer l'axe de l'état Arrêt suite à une erreur à l'état Immobilisation en réinitialisant toutes les erreurs propres à l'axe. Les sorties des instances du bloc fonctionnel ne sont pas modifiées.

Détails de l'opération :

- Si l'état d'alarme de l'axe n'est pas changé après l'exécution de MC_Reset, exécutez MC_Power, puis MC_Reset.
- MC_Reset modifie uniquement l'état de l'axe depuis Arrêt suite à une erreur à Immobilisation. L'exécution de MC_Reset lorsque l'axe présente d'autres états, y compris Désactivé, entraîne une erreur et n'a aucune incidence sur le mouvement en cours ou l'état de l'axe.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

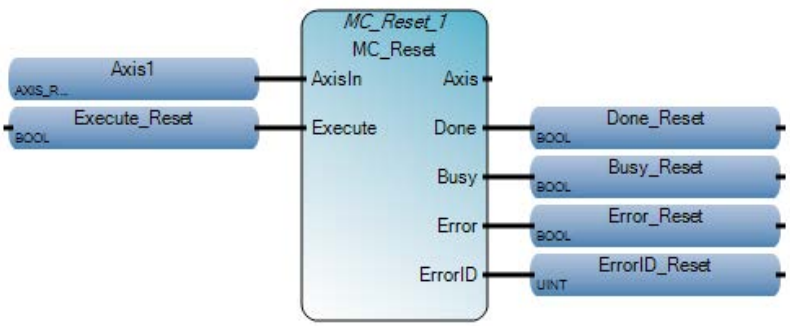
Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



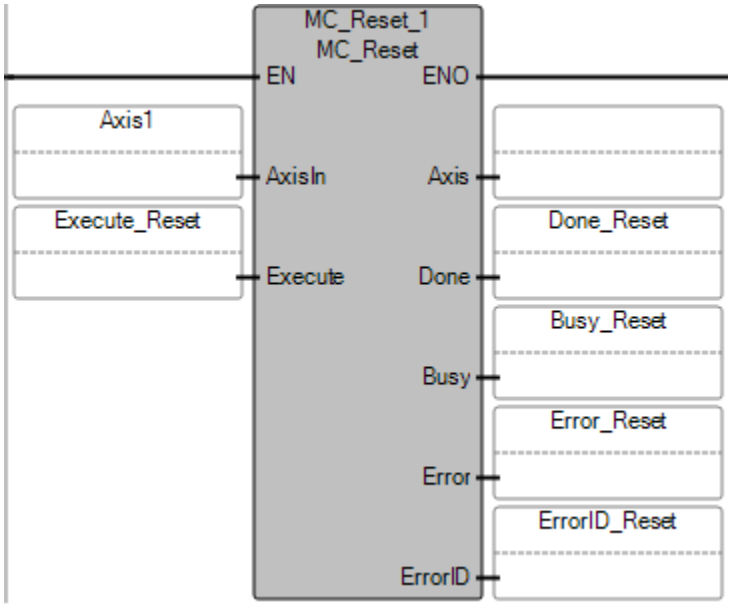
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_Reset actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.
Execute	Entrée	BOOL	VRAI : l'axe est réinitialisé au moment du front montant.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	VRAI : l'état de l'axe correspond à Immobilisation ou Désactivé.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Exemple de diagramme de bloc de fonctions MC_Reset



Exemple de langage à relais MC_Reset



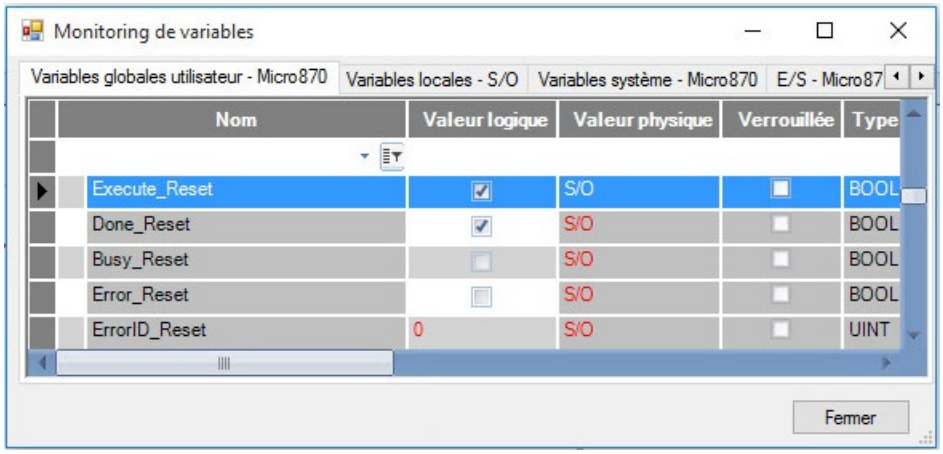
Exemple de texte structuré MC_Reset

```
MC_Reset_1(
```

void **MC_Reset_1**([AXIS_REF|FB_AXIS_REF] **AxisIn**, BOOL **Execute**)
Type: MC_Reset, Réinitialise toutes les erreurs internes relatives aux axes.

```
MC_Reset_1(Axis1,Execute_Reset);
Done_Reset := MC_Reset_1.Done;
Busy_Reset := MC_Reset_1.Busy;
Error_Reset := MC_Reset_1.Error;
ErrorID_Reset := MC_Reset_1.ErrorID;
```

Résultats



MC_SetPosition (contrôle des mouvements, définition de position)

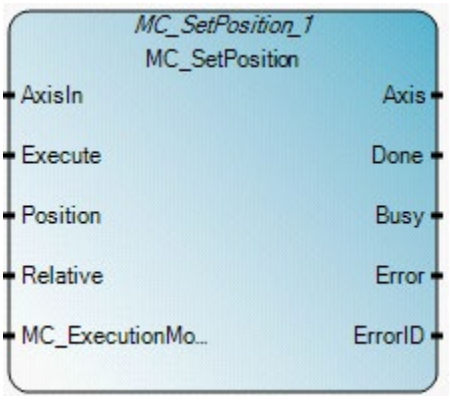
Remplace le système de coordonnées d'un axe en manipulant sa position réelle avec la même valeur sans générer de mouvement.

Détails de l'opération :

- MC_SetPosition peut se terminer de façon concluante uniquement lorsque l'axe est à l'état Immobilisation, mouvement continu (MC_ExecutionMode = 0) ou lorsque le mouvement en cours s'achève pour se terminer avec l'état Immobilisation (MC_ExecutionMode = 1).
- MC_SetPosition fonctionne de la même manière que MC_Home lorsque HomingMode = MC_HOME_DIRECT (0x04), sauf lorsque le bloc de fonctions MC_Home définit l'état d'origine de l'axe.
- Lorsque MC_ExecutionMode = 0 (mcImmediately), l'exécution du bloc de fonctions MC_SetPosition signale une erreur si l'axe est en mouvement discontinu.
- Lorsque MC_ExecutionMode = 1 (mcQueued), l'ajustement de la position en cours ne s'effectue que lorsque tous les autres mouvements précédents en cours sont terminés. Autrement dit, les blocs fonctionnels précédents doivent avoir au moins un des paramètres Done, Aborted ou Error est égal à Vrai.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

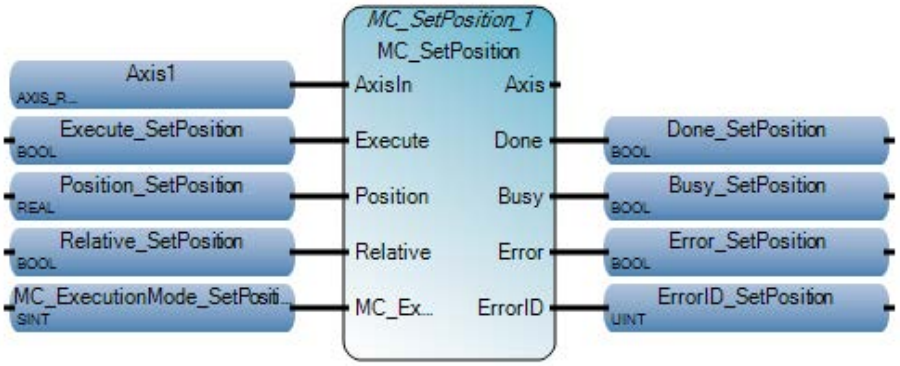


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

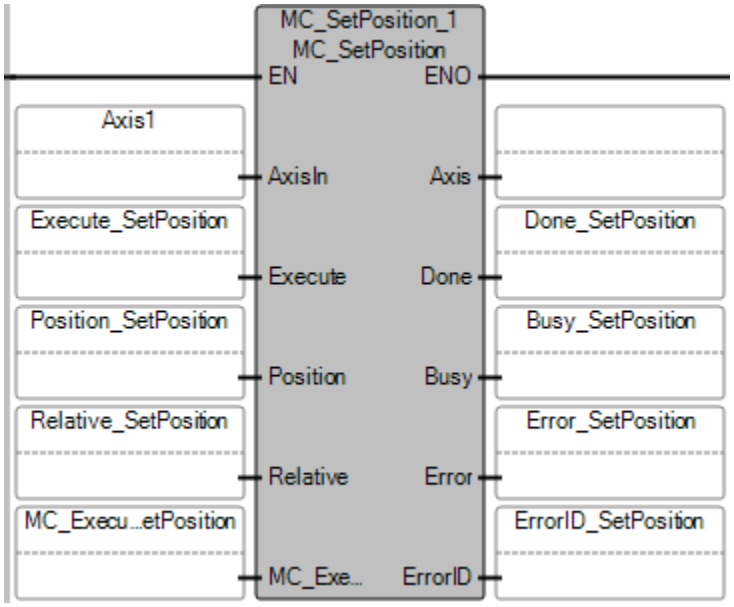
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_SetPosition actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF pour définir AxisIn. Pour un axe retour codeur (FB_Axis), utilisez le type de données FB_AXIS_REF sur la page 412 pour définir AxisIn.
Execute	Entrée	BOOL	VRAI : la position de l'axe commence à être définie.

Paramètre	Type de paramètre	Type de données	Description
Position	Entrée	REAL	La position absolue ou la distance relative à fixer pour l'axe.
Relative	Entrée	BOOL	VRAI : définition de la distance relative pour l'axe. FAUX : définition de la position absolue pour l'axe.
MC_ExecutionMode	Entrée	SINT	Les valeurs sont les suivantes : • 0 (*mcImmediately*), la fonctionnalité est valide immédiatement. • 1 (*mcQueued*), la nouvelle fonctionnalité devient valide lorsque : • toutes les commandes de mouvement précédentes définissent l'un des paramètres de sortie suivants : Done, Aborted ou Error • L'axe n'est pas en mouvement. Pour (MC_ExecutionMode = 0), ce bloc de fonctions se termine de façon concluante lorsque l'axe est à l'état Désactivé ou Immobilisation. L'exécution de ce bloc de fonctions signale une erreur si l'axe est en mouvement discontinu dans ce mode. Pour (MC_ExecutionMode = 1), ce bloc de fonctions se termine de façon concluante lorsque l'axe est à l'état Désactivé ou Immobilisation ou que le mouvement continu peut se terminer, avec l'état Immobilisation. D'autres valeurs d'entrée sont actuellement réservées et considérées comme des paramètres non valides.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	VRAI : le paramètre Position contient une nouvelle valeur.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

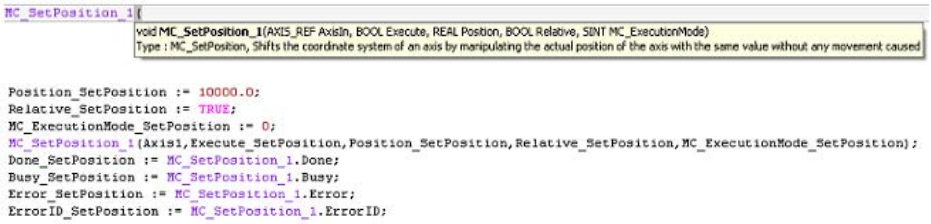
Exemple de diagramme de bloc de fonctions MC_SetPosition



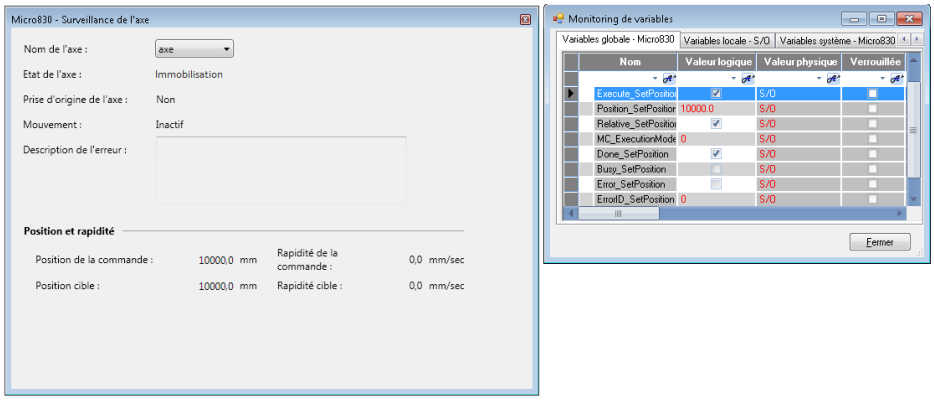
Exemple de langage à relais MC_SetPosition



Exemple de texte structuré MC_SetPosition



Résultats



MC_Stop (contrôle des mouvements, arrêt)

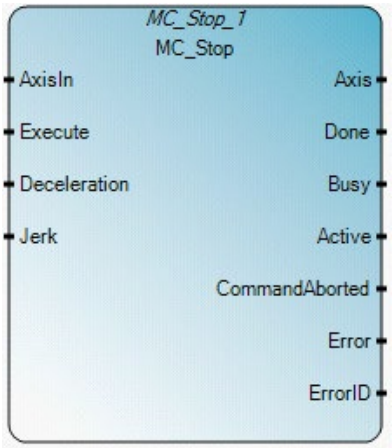
Commande un arrêt contrôlé du mouvement et fait passer l'axe à l'état Arrêt. Tout autre bloc fonctionnel en cours d'exécution est annulé. Toutes les commandes de déplacement des blocs fonctionnels sont ignorées tant que l'axe ne passe pas à l'état Immobilisation.

Détails de l'opération :

- Tant que l'entrée Execute input est élevée, l'axe demeure à l'état Arrêt. Lorsque l'axe est dans l'état Arrêt, les autres blocs de fonctions de mouvement ne peuvent pas effectuer de mouvement sur cet axe.
- Si la Décélération est nulle, les paramètres MC_Stop sont déterminés par le paramètre Arrêt d'urgence de la configuration de l'axe, notamment le type E-Stop, E-stop Deceleration et E-stop Jerk.
- Lorsqu'aucune erreur n'est détectée au cours de la séquence d'arrêt, l'état de l'axe passe à Immobilisation après que le bit Done est DÉFINI et que l'entrée Execute passe à l'état Faux.
- Utilisez MC_Stop pour la fonctionnalité d'arrêt d'urgence et les situations exceptionnelles. Utilisez MC_Halt pour un arrêt normal du mouvement.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

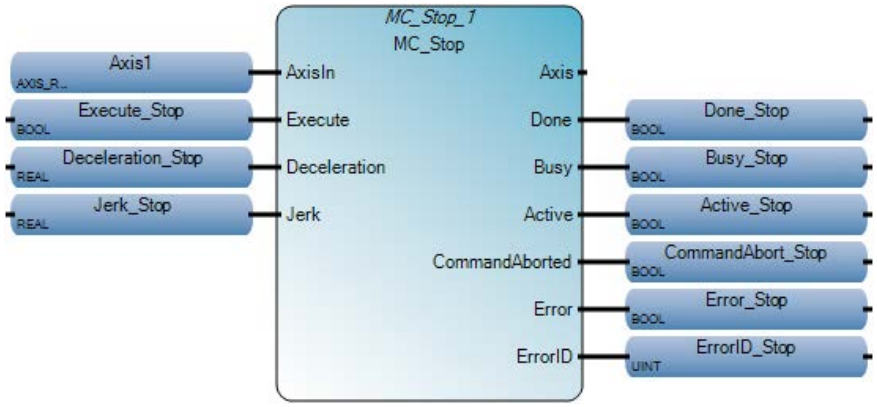


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

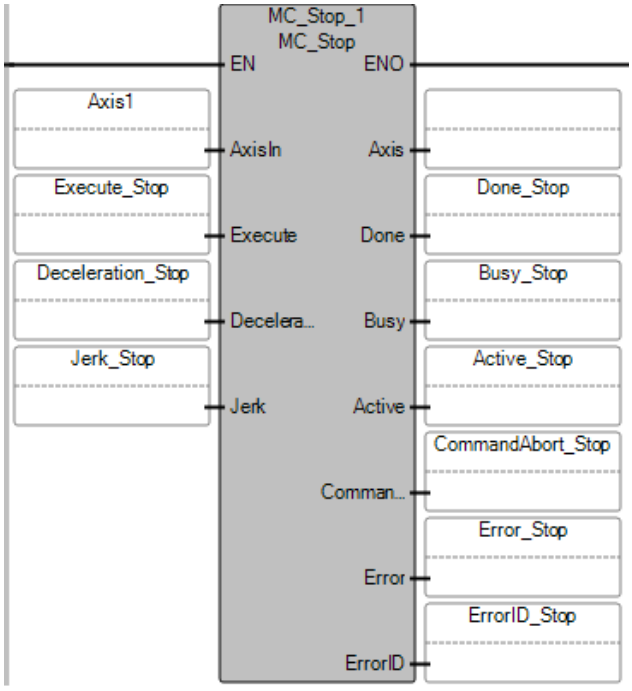
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_Stop actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF	Utilisez le type de données AXIS_REF pour définir les paramètres pour AxisIn.
Execute	Entrée	BOOL	VRAI : l'action est lancée au moment du front montant. FAUX : aucune exécution.
Deceleration	Entrée	REAL	Valeur correspondant à la décélération [u/s^2].
Jerk	Entrée	REAL	Valeur correspondant au jerk [u/s^3].
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF sur la page 411	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.

Paramètre	Type de paramètre	Type de données	Description
Done	Sortie	BOOL	VRAI : la vitesse zéro a été atteinte sans qu'aucune erreur n'ait été générée lors de la séquence d'arrêt.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé.
Active	Sortie	BOOL	VRAI : indique que le bloc de fonctions contrôle l'axe.
CommandAborted	Sortie	BOOL	TRUE : la commande a été remplacée par le bloc de fonctions MC_Power(OFF) ou ErrorStop.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Exemple de diagramme de bloc de fonctions MC_Stop



Exemple de langage à relais MC_Stop



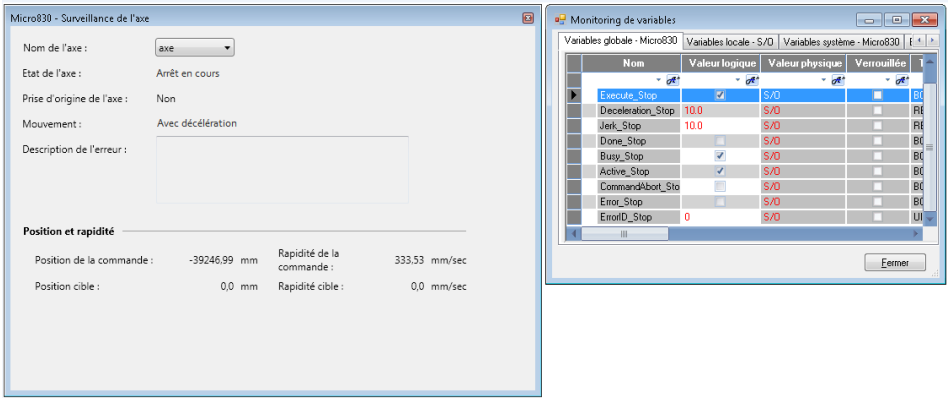
Exemple de texte structuré MC_Stop

```
MC_Stop_1(
```

void **MC_Stop_1**(AXIS_REF **AxisIn**, BOOL **Execute**, REAL **Deceleration**, REAL **Jerk**)
Type: MC_Stop, Commande un arrêt de mouvement contrôlé et transfère l'axe à l'état Stopping.

```
Deceleration_Stop := 10.0;
Jerk_Stop := 10.0;
MC_Stop_1(Axis1,Execute_Stop,Deceleration_Stop,Jerk_Stop);
Done_Stop := MC_Stop_1.Done;
Busy_Stop := MC_Stop_1.Busy;
Active_Stop := MC_Stop_1.Active;
CommandAbort_Stop := MC_Stop_1.CommandAborted;
Error_Stop := MC_Stop_1.Error;
ErrorID_Stop := MC_Stop_1.ErrorID;
```

Résultats



MC_TouchProbe (contrôle des mouvements, sonde tactile)

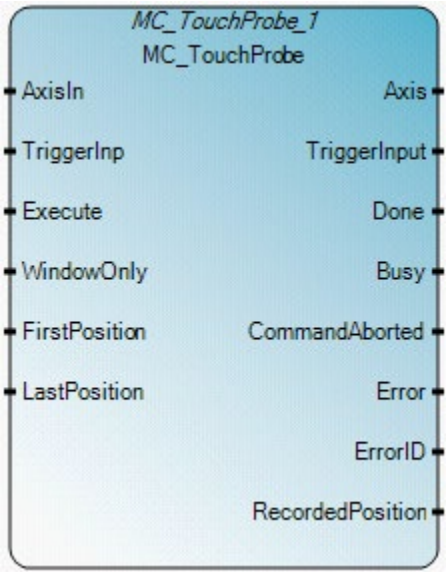
Enregistre la position de l'axe lors du déclenchement d'un événement.

Détails de l'opération :

- Si la direction de la fenêtre (première position --> dernière position) est opposée à la direction des mouvements, la fenêtre du palpeur ne s'active pas.
- Si le paramètre de fenêtre (première ou dernière position) est incorrect, le bloc de fonctions MC_TouchProbe signale une erreur.
- Si une deuxième instance du bloc de fonctions MC_TouchProbe est envoyée au même axe alors que le premier bloc de fonctions est Busy, le deuième bloc de fonctions signale une erreur.
- Une seule instance du bloc de fonctions MC_TouchProbe doit être émise par axe.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

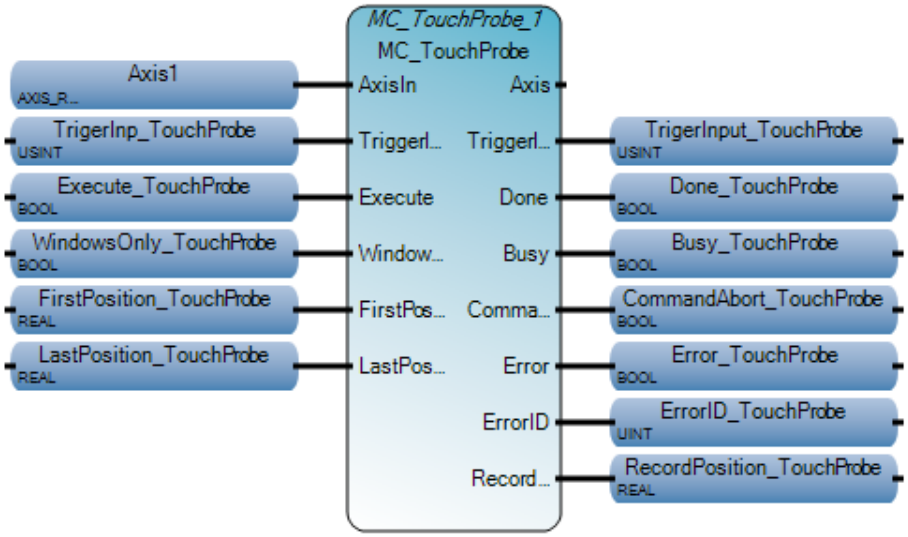
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_TouchProbe actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF pour définir les paramètres pour AxisIn. Pour un axe retour codeur, utilisez le type de données FB_AXIS_REF sur la page 412 pour définir les paramètres pour AxisIn.
TriggerInp	Entrée	USINT	Actuellement non pris en charge. Configure le déclenchement d'entrée dans la configuration de l'axe.
Execute	Entrée	BOOL	VRAI : début de l'enregistrement de la sonde tactile au moment du front montant. FAUX : aucune exécution.
WindowOnly	Entrée	BOOL	VRAI : seule la fenêtre est utilisée pour accepter des événements de déclenchement. La résolution du mouvement est limitée à l'intervalle du moteur de mouvement que l'utilisateur a configuré. Pour la fonctionnalité WindowOnly TouchProbe, il existe un délai de réponse maximal correspondant à l'intervalle du moteur de mouvement pour les activations FirstPosition et LastPosition. Le plus long décalage possible dans la position de déclenchement (aussi bien pour FirstPosition que pour LastPosition) peut être calculé par (intervalle du moteur de mouvement * vitesse de déplacement).
FirstPosition	Entrée	REAL	Position de départ de la fenêtre à partir de laquelle les événements de déclenchement sont acceptés (en unités techniques [u]). Valeur incluse dans la fenêtre.
LastPosition	Entrée	REAL	Position de fin de la fenêtre à partir de laquelle les événements de déclenchement ne sont pas acceptés (en unités techniques [u]). Valeur incluse dans la fenêtre.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF FB_AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.

Paramètre	Type de paramètre	Type de données	Description
TriggerInput	Sortie	USINT	Actuellement non pris en charge.
Done	Sortie	BOOL	VRAI : l'événement de déclenchement a été enregistré.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé.
CommandAborted	Sortie	BOOL	TRUE : la commande a été remplacée par le bloc de fonctions MC_Power(OFF) ou arrêtée suite à une erreur.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.
RecordedPosition	Sortie	REAL	Position à laquelle l'événement de déclenchement est survenu (en unités techniques [u]) Le mouvement est de type boucle ouverte. Position de l'axe au moment de l'exécution de l'événement de déclenchement. Si le mouvement de l'axe est de type boucle ouverte, il s'agit de la position commandée (qui n'est pas une position réelle) au moment de l'exécution de l'événement de déclenchement, en supposant qu'il n'y ait pas de décalage de mouvement entre le moteur et le variateur.

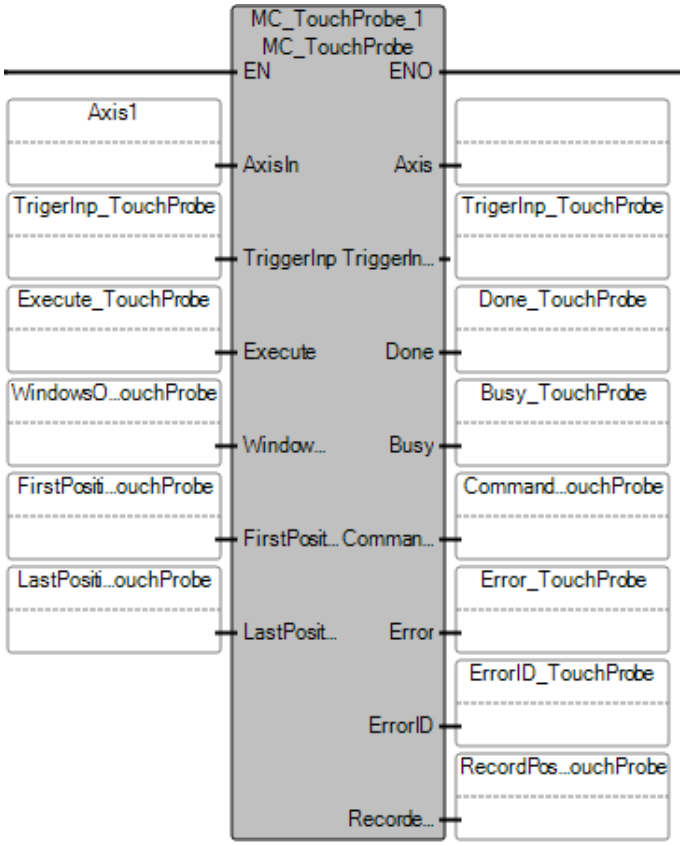
Entrée/sortie fixe de mouvement

Signaux de mouvement	PT00	PT01	PT02
Pulsation PTO	Output_0	Output_0	Output2
Direction PTO	Output_3	Output_4	Output_5
Commutateur de limite inférieure (négative)	Input_0	Input_4	Input_8
Commutateur de limite supérieure (positive)	Input_1	Input_5	Input_9
Commutateur d'origine absolue	Input_2	Input_6	Input_10
Commutateur d'entrée du palpeur	Input_3	Input_7	Input_11

Exemple de diagramme de bloc de fonctions MC_TouchProbe



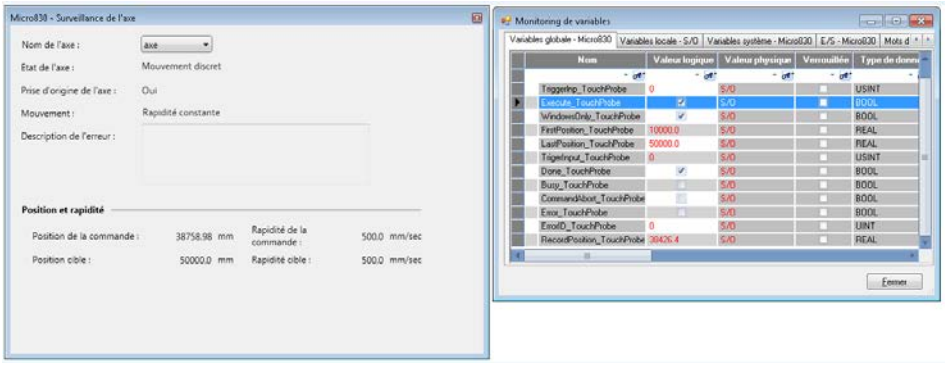
Exemple de langage à relais MC_TouchProbe



Exemple de texte structuré MC_TouchProbe

```
FirstPosition_TouchProbe := 10000.0;
LastPosition_TouchProbe := 50000.0;
MC_TouchProbe_1(Axis1,TrigerInp_TouchProbe,Execute_TouchProbe,WindowsOnly_TouchProbe,
FirstPosition_TouchProbe,LastPosition_TouchProbe);
Done_TouchProbe := MC_TouchProbe_1.Done;
Busy_TouchProbe := MC_TouchProbe_1.Busy;
Error_TouchProbe := MC_TouchProbe_1.Error;
ErrorID_TouchProbe := MC_TouchProbe_1.ErrorID;
RecordPosition_TouchProbe := MC_TouchProbe_1.RecordedPosition;
```

Résultats



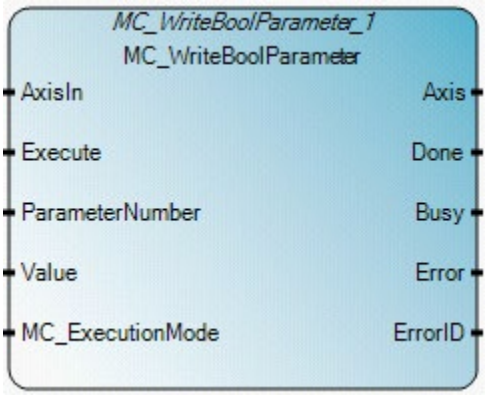
MC_WriteBoolParameter (contrôle des mouvements, écriture du paramètre BOOL)

Modifie la valeur d'un paramètre de type BOOL spécifique à un fournisseur.

Les paramètres définis par le bloc de fonctions MC_WriteBoolParameter ne sont appliqués que provisoirement à l'application. Ils sont écrasés par les paramètres permanents configurés par l'utilisateur dans Connected Components Workbench Motion Configuration, lorsque le contrôleur passe de PRG au mode Exécuter ou qu'il est ARRÊTÉ, puis REMIS SOUS TENSION.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

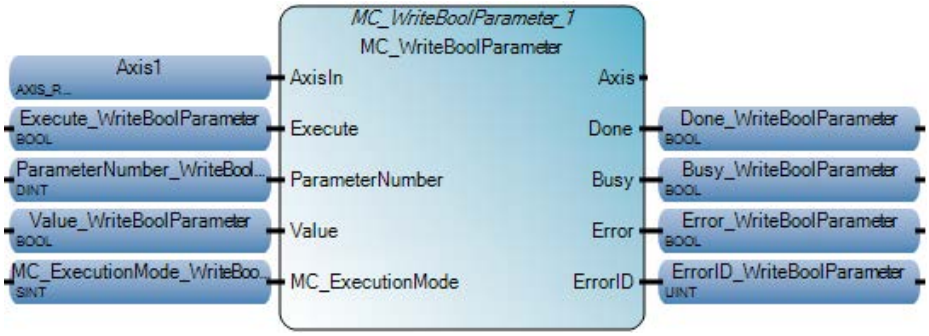


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

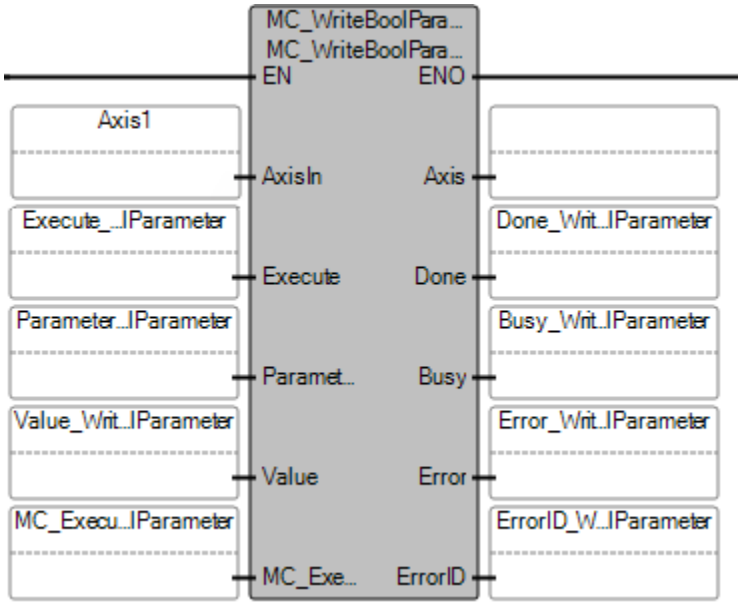
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_WriteBoolParameter actuel. FAUX : la sortie Valeur est réinitialisée sur 0. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF pour définir les paramètres pour AxisIn. Pour un axe retour codeur, utilisez le type de données FB_AXIS_REF pour définir les paramètres pour AxisIn.
Execute	Entrée	BOOL	VRAI : la valeur du paramètre est écrite au moment du front montant.
ParameterNumber	Entrée	DINT	Identification du paramètre. Le numéro du paramètre est défini dans les détails des paramètres du bloc de fonctions de contrôle des mouvements sur la page 406.
Value	Entrée	BOOL	VRAI : le paramètre spécifique contient une nouvelle valeur.

Paramètre	Type de paramètre	Type de données	Description
MC_ExecutionMode	Entrée	SINT	Les valeurs sont les suivantes : • 0 (*mcImmediately*), la fonctionnalité est valide immédiatement. • 1 (*mcQueued*), la nouvelle fonctionnalité devient valide lorsque : • Toutes les commandes de mouvement précédentes définissent l'un des paramètres de sortie suivants : Done, Aborted ou Error • L'axe n'est pas en mouvement. Si (MC_ExecutionMode = 0), pour tous les paramètres à l'exception de Duty Cycle (1005), ce bloc de fonctions peut être terminé de façon concluante uniquement lorsque l'axe est à l'état Désactivé ou Immobilisation. Si (MC_ExecutionMode = 0), pour Parameter Duty Cycle (1005), ce bloc de fonctions peut être terminé de façon concluante, sauf si l'axis est à l'état Homing ou ErrorStop. Pour (MC_ExecutionMode = 1), ce bloc de fonctions peut être terminé de façon concluante uniquement lorsque l'axe est à l'état Désactivé ou Immobilisation ou que le mouvement continu peut se terminer, avec l'état Immobilisation. D'autres valeurs d'entrée sont actuellement réservées et considérées comme des paramètres non valides.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	VRAI : le paramètre a été écrit correctement.
Busy	Sortie	BOOL	VRAI : le bloc de fonctions n'est pas terminé.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

Exemple de diagramme de bloc de fonctions MC_WriteBoolParameter



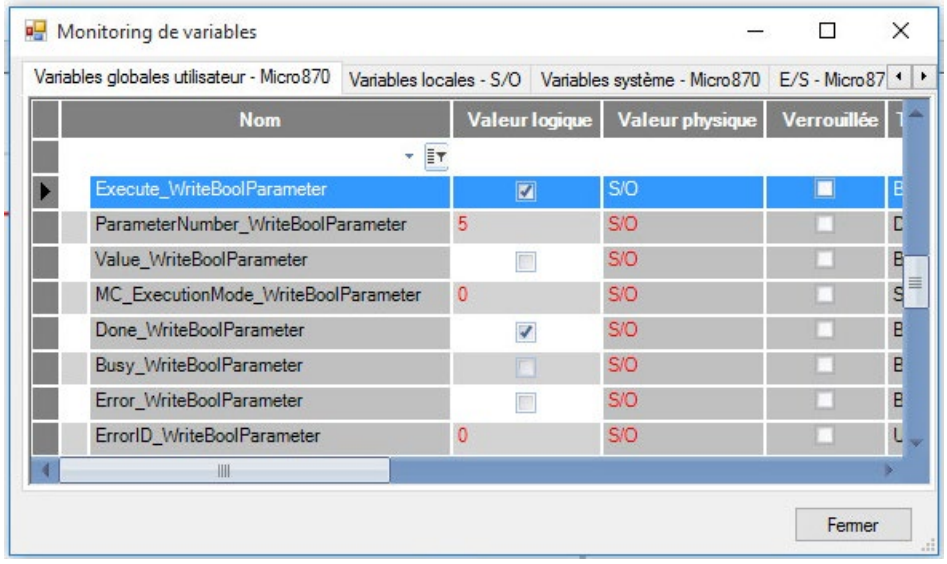
Exemple de langage à relais MC_WriteBoolParameter



Exemple de texte structuré MC_WriteBoolParameter

```
ParameterNumber_WriteBoolParameter := 5;
MC_WriteBoolParameter_1(Axis1,Execute_WriteBoolParameter,ParameterNumber_WriteBoolParameter,
Value_WriteBoolParameter,MC_ExecutionMode_WriteBoolParameter);
Done_WriteBoolParameter := MC_WriteBoolParameter_1.Done;
Busy_WriteBoolParameter := MC_WriteBoolParameter_1.Busy;
Error_WriteBoolParameter := MC_WriteBoolParameter_1.Error;
ErrorID_WriteBoolParameter := MC_WriteBoolParameter_1.ErrorID;
```

Résultats



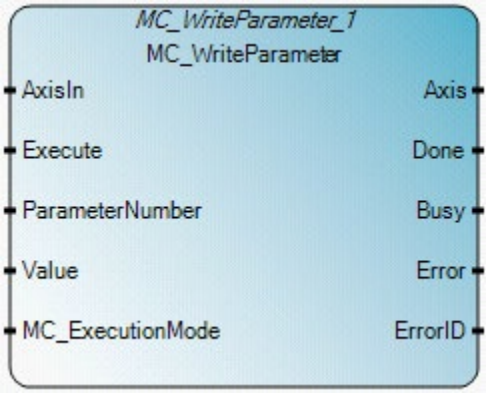
MC_WriteParameter (contrôle des mouvements, écriture du paramètre)

Modifie la valeur d'un paramètre de type REAL spécifique à un fournisseur.

Les paramètres définis par le bloc de fonctions MC_WriteParameter ne sont appliqués que provisoirement à l'application. Ils sont écrasés par les paramètres permanents configurés par l'utilisateur dans Connected Components Workbench Motion Configuration, lorsque le contrôleur passe de PRG au mode Exécuter ou qu'il est ARRÊTÉ, puis REMIS SOUS TENSION.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro830, Micro850 et Micro870 qui prennent en charge le contrôle des mouvements.

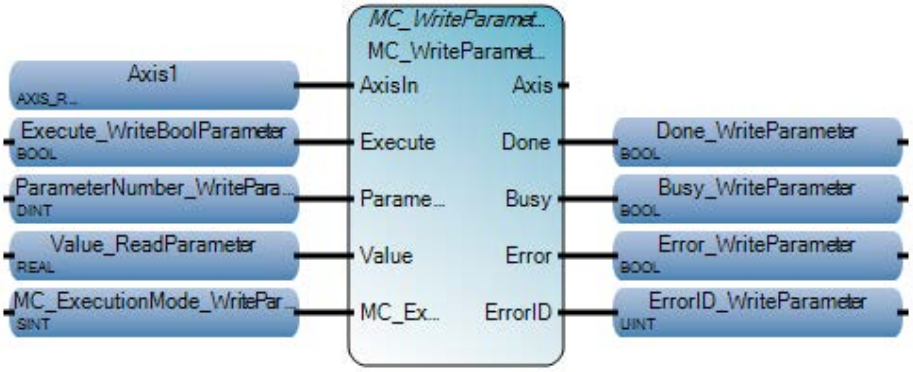


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

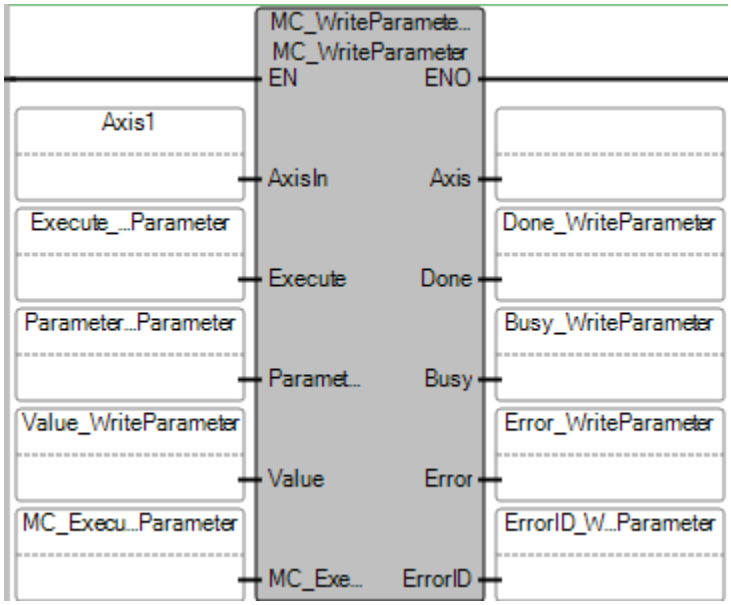
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du calcul MC_WriteParameter actuel. FAUX : aucun calcul. Ne s'applique qu'aux programmes de langage à relais.
AxisIn	Entrée	AXIS_REF FB_AXIS_REF	Utilisez le type de données AXIS_REF sur la page 411 pour définir les paramètres pour AxisIn. Pour un axe retour codeur, utilisez le type de données FB_AXIS_REF sur la page 412 pour définir les paramètres pour AxisIn.
Execute	Entrée	BOOL	VRAI : la valeur du paramètre est écrite au moment du front montant.
ParameterNumber	Entrée	DINT	Identification du paramètre. Le numéro du paramètre est défini dans les détails des paramètres du bloc de fonctions de contrôle des mouvements sur la page 406.
Value	Entrée	REAL	Nouvelle valeur du paramètre spécifié.

Paramètre	Type de paramètre	Type de données	Description
MC_ExecutionMode	Entrée	SINT	Les valeurs sont les suivantes : • 0 (*mcImmediately*), la fonctionnalité est valide immédiatement. • 1 (*mcQueued*), la nouvelle fonctionnalité devient valide lorsque : • Toutes les commandes de mouvement précédentes définissent l'un des paramètres de sortie suivants : Done, Aborted ou Error • L'axe n'est pas en mouvement. • Le paramètre de sortie Busy est défini sur FALSE. Si (MC_ExecutionMode = 0), pour tous les paramètres à l'exception de Duty Cycle (1005), ce bloc de fonctions peut être terminé de façon concluante uniquement lorsque l'axe est à l'état Désactivé ou Immobilisation. Si (MC_ExecutionMode = 0), pour Parameter Duty Cycle (1005), ce bloc de fonctions peut être terminé de façon concluante, sauf si l'axis est à l'état Homing ou ErrorStop. Pour (MC_ExecutionMode = 1), ce bloc de fonctions peut être terminé de façon concluante uniquement lorsque l'axe est à l'état Désactivé ou Immobilisation ou que le mouvement continu peut se terminer, avec l'état Immobilisation. D'autres valeurs d'entrée sont actuellement réservées et considérées comme des paramètres non valides.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Axis	Sortie	AXIS_REF	Dans les programmes de schémas à relais, la sortie de l'axe est en lecture seule.
Done	Sortie	BOOL	VRAI : le paramètre a été écrit correctement.
Busy	Sortie	BOOL	VRAI : indique que le bloc de fonctions contrôle l'axe.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.
ErrorID	Sortie	UINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs de cette instruction sont définies dans les ID des erreurs de bloc de fonctions du contrôle des mouvements sur la page 409.

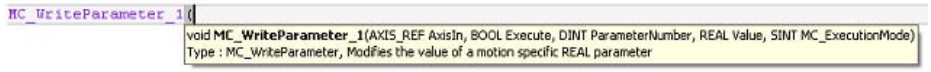
Exemple de diagramme de bloc de fonctions MC_WriteParameter



Exemple de langage à relais MC_WriteParameter

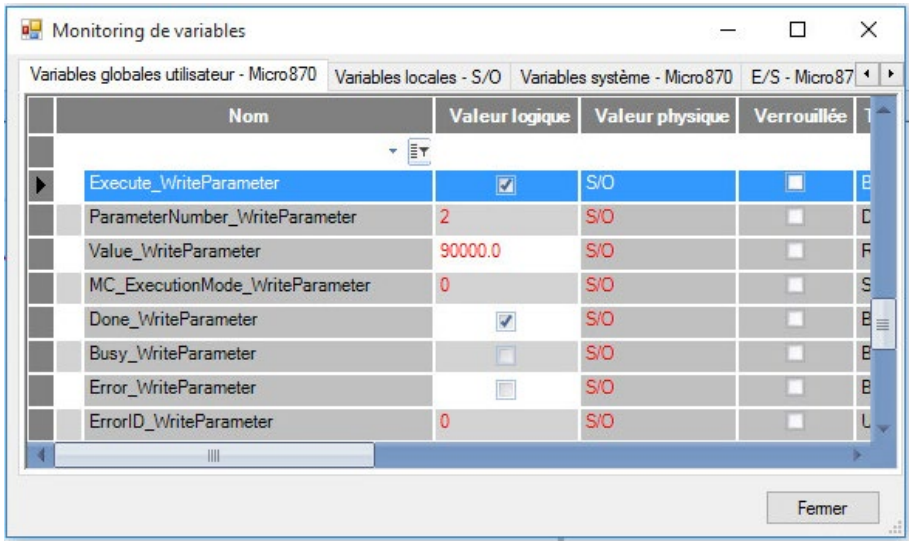


Exemple de texte structuré MC_WriteParameter




```
ParameterNumber_WriteParameter := 2;
Value_WriteParameter := 90000.0;
MC_WriteParameter_1(Axis1,Execute_WriteParameter,ParameterNumber_WriteParameter,
Value_WriteParameter,MC_ExecutionMode_WriteParameter);
Done_WriteParameter := MC_WriteParameter_1.Done;
Busy_WriteParameter := MC_WriteParameter_1.Busy;
Error_WriteParameter := MC_WriteParameter_1.Error;
ErrorID_WriteParameter := MC_WriteParameter_1.ErrorID;
```

Résultats



Instructions de contrôle de traitement

Utilisez les instructions de contrôle de procédés pour surveiller et maintenir les boucles de traitement, pour des valeurs de quantité telles que la pression, la température, le débit et le niveau de liquide. Les contrôles de processus régulent le flux en envoyant un signal de sortie à la vanne de régulation.

Instruction	Description
DERIVATE sur la page 477	Différencie une valeur réelle pendant un temps de cycle défini.
FFL sur la page 479	Charge des données de 8 bits, 16 bits, 32 bits ou 64 bits dans un tableau créé par l'utilisateur appelé une pile FIFO.
FFU sur la page 487	Décharge des données de 8 bits, 16 bits, 32 bits ou 64 bits depuis un tableau créé par l'utilisateur appelé une pile FIFO. Les données sont déchargées dans le même ordre que lors du chargement grâce à une instruction FFL.
HYSTER sur la page 492	Effectue unehystérésis booléenne sur différence entre réels.
INTEGRAL sur la page 494	Intègre une valeur réelle pendant le temps de cycle défini.
LFL (chargement LIFO) sur la page 500	Charge des données de 8 bits, 16 bits, 32 bits ou 64 bits dans un tableau créé par l'utilisateur appelé une pile LIFO.
LFU (déchargement LIFO) sur la page 502	Décharge des données de 8 bits, 16 bits, 32 bits ou 64 bits depuis un tableau créé par l'utilisateur appelé une pile LIFO. Les données sont déchargées dans le même ordre que lors du chargement grâce à une instruction LFL.
PWM sur la page 505	Active ou désactive la sortie de la modulation d'impulsions en durée (PWM) pour un canal PWM configuré.
SCALER sur la page 507	Met la valeur d'entrée à l'échelle en fonction de la plage de sortie.
STACKINT sur la page 509	Gère une pile de valeurs entières.
LIMIT sur la page 519	Restreint des valeurs entières à un intervalle donné.
TND sur la page 518	Arrête le cycle actuel de la scrutation du programme utilisateur.

DERIVATE

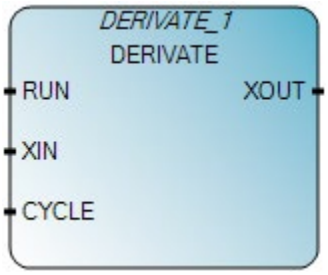
Différencie une valeur réelle pendant un temps de cycle défini.

Détails de l'opération :

- Si la valeur du paramètre CYCLE est inférieure à la durée du cycle d'exécution de l'appareil, la période d'échantillonnage est alignée de force sur la durée de ce cycle.
- La dérivation est effectuée avec une base de temps en millisecondes.Par exemple, la dérivation d'une entrée égale à 1 000 qui passe progressivement à 2 000 sur une période de 1 seconde donne la valeur de 1. Pour convertir la sortie de l'instruction en unités de secondes, multipliez-la par 1 000.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

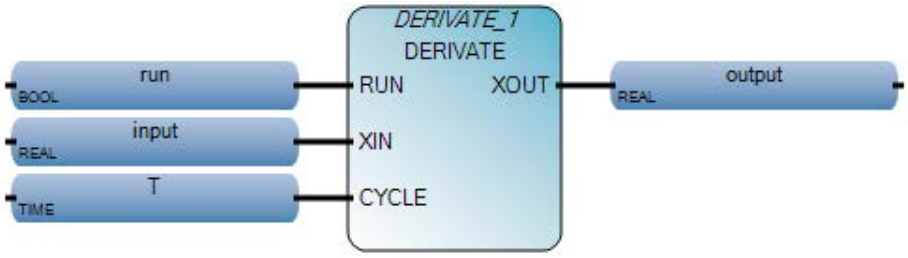
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



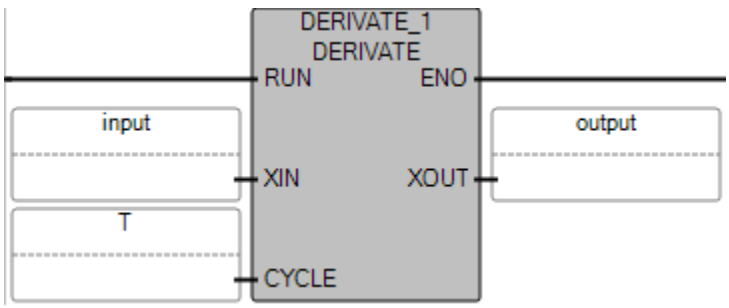
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
RUN	Entrée	BOOL	Indique le mode opérationnel de l'instruction. VRAI : normal (effecteur le calcul) FAUX : réinitialisation
XIN	Entrée	REAL	Définit la valeur à laquelle le calcul de dérivation est effectué. La valeur doit être une valeur RÉELLE.
CYCLE	Entrée	TIME	Définit la période d'échantillonnage pendant laquelle collecter les valeurs. Les valeurs possibles de la période sont comprises entre 0 ms et 49j 17 h 2 m 47 s 294 ms.
XOUT	Sortie	REAL	Sortie différenciée.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

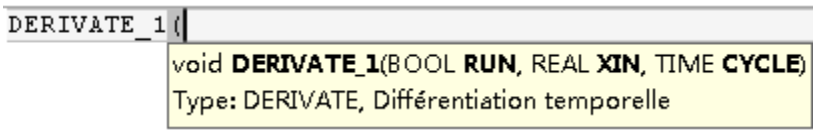
Exemple de Diagramme de Bloc Fonction DERIVATE



Exemple de langage à relais DERIVATE



Exemple de texte structuré DERIVATE




```
1  DERIVATE_1(run, input, T);
2  output := DERIVATE_1.XOUT;
```

(* Équivalence ST : DERIVATE1 est une instance d'un bloc DERIVATE *)

```
DERIVATE1(manual_mode, sensor_value, t#100ms) ;
derivated_value := DERIVATE1.XOUT ;
```

FFL (chargement FIFO)

Charge des données de 8 bits, 16 bits, 32 bits ou 64 bits dans un tableau créé par l'utilisateur appelé une pile FIFO.

Détails de l'opération :

- Instruction FFL - Mode sans exécution à mode d'exécution
  - Lorsque l'exécution passe de FAUX à VRAI :
    - Les conditions d'erreur sont vérifiées.
    - Les contenus de Src sont chargés dans la pile FIFO dans la position disponible et la Position augmente par incréments de 1 si la Position est inférieure ou égale à zéro, et inférieure à la Longueur.
    - Complet est défini sur VRAI si la Longueur est égale à la Position. Complet est FAUX si la Position est inférieure ou égale à zéro, et inférieure à la Longueur.
    - Done est défini lorsque l'instruction s'exécute correctement.
  - Lorsque l'exécution passe de VRAI à FAUX :
    - Error, Done et ErrorID sont définis sur FALSE.
    - Vide est défini sur VRAI si la Position est égale à zéro.
    - Complet est défini sur VRAI si la Longueur est égale à la Position. Complet est FAUX si la Position est inférieure ou égale à zéro, et inférieure à la Longueur.
    - Les conditions d'erreur FFL ne sont pas vérifiées.
  - Lorsque l'exécution passe de VRAI à VRAI :
    - Aucune opération de chargement n'est effectuée.
    - Vide est défini sur VRAI si la Position est égale à zéro.
    - Complet est défini sur FAUX si la Position est inférieure ou égale à zéro, et inférieure à la Longueur. Complet est défini sur VRAI si la Longueur est égale à la Position.
    - Les conditions d'erreur FFL ne sont pas vérifiées.
  - Lorsque l'exécution passe de FAUX à FAUX :
    - Les bits Error, Done et ErrorID sont définis sur FALSE.
    - Les bits Complet et Vide conservent les valeurs du précédent état d'exécution.

- Vide est défini sur VRAI si la Position est égale à zéro.
  - Complet est défini sur VRAI si la Longueur est égale à la Position. Complet est FAUX si la Position est inférieure ou égale à zéro, et inférieure à la Longueur.
  - Les conditions d'erreur FLL ne sont pas vérifiées.
- Instruction FFL - Mode d'exécution à mode sans exécution :
  - Error, Done et ErrorID, Empty et Full conservent l'état du mode d'exécution.

- Pour créer un seul élément pour le paramètre FIFO :
  - Non-tableau :
    - Une adresse basée sur la variable (comme Fifo1) est autorisée pour FIFO.
    - La longueur doit être définie sur 1.
  - Tableau :
    - Une adresse basée sur la variable (comme Fifo1 ou Fifo1[0]) est autorisée pour FIFO.
    - La longueur doit être définie sur 1.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : si le front montant est détecté, l'opération FFL commence. FAUX : front montant non détecté.
Src	Entrée	ANY_ELEMENTARY	L'opérande Src est l'adresse de la valeur utilisée pour renseigner la position actuellement disponible dans la pile FIFO. Les types de données élémentaires pris en charge pour Scr sont les suivants : • DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • La chaîne n'est pas prise en charge. • Les éléments de tableau comme Array[1] ou Array[Index] sont pris en charge.

Paramètre	Type de paramètre	Type de données	Description
FIFO	Entrée	ANY_ELEMENTARY	L'adresse de début de la pile. FIFO doit être configuré de la même façon pour les instructions FFL et FFU. Les types de données élémentaires pris en charge pour FIFO sont les suivants : DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • La chaîne n'est pas prise en charge. • Seuls les tableaux à une dimension sont pris en charge pour FIFO.
FIFOCon	Entrée	FF_LF_CON	Configuration et contrôle FIFO. La même configuration doit être définie pour les instructions FFL et FFU. Utilisez le type de données FF_LF_CON pour configurer la Position et la Longueur.
Done	Sortie	BOOL	Indique quand l'opération FFU est terminée. VRAI : opération terminée avec succès. FAUX : l'opération a rencontré une erreur, ou l'instruction FFU ne s'exécute pas.
Empty	Sortie	BOOL	Indique quand la pile FIFO est vide. VRAI : lorsque la Position est égale à 0. FAUX : lorsque la Position n'est pas égale à 0.
Full	Sortie	BOOL	Indique quand la pile FIFO est complète. VRAI : lorsque la Longueur est égale à la Position. FAUX : lorsque la Position est supérieure ou égale à zéro, et inférieure à la Longueur.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : l'opération a rencontré une erreur. FAUX : l'opération s'est terminée avec succès ou l'instruction ne s'exécute pas.
ErrorID	Sortie	USINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur FFL.

Type de données FF_LF_CON

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données FF_LF_CON.

Paramètre	Type de données	Description
Length	UINT	Nombre d'éléments utilisés pour l'opération FIFO. La limite maximale est 1024.
Position	USINT	Détermine le prochain emplacement disponible dans le FIFO pour l'entrée ou la suppression Src. La position est le décalage du tableau. Exemple 1 : • Tableau configuré par l'utilisateur, arr[0..5]. La position initiale est définie sur 1. Les données sont poussées dans le tableau arr[1] et la position est incrémentée de +1. Exemple 2 : • Tableau configuré par l'utilisateur, arr[1..5]. La position initiale est définie sur 1. Les données sont poussées dans le tableau arr[2] et la position est incrémentée de +1.

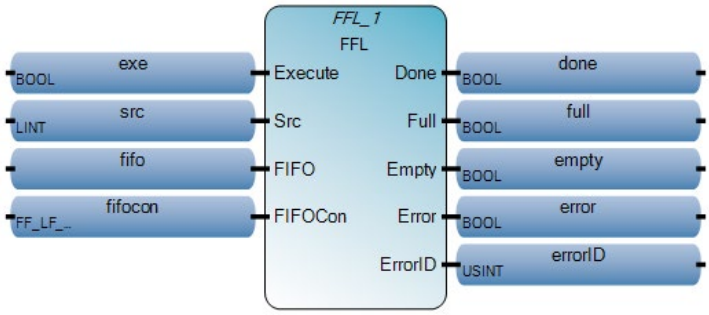
Codes d'erreur FFL

Utilisez ce tableau pour déterminer les codes d'erreur FFL et FFU et les descriptions.

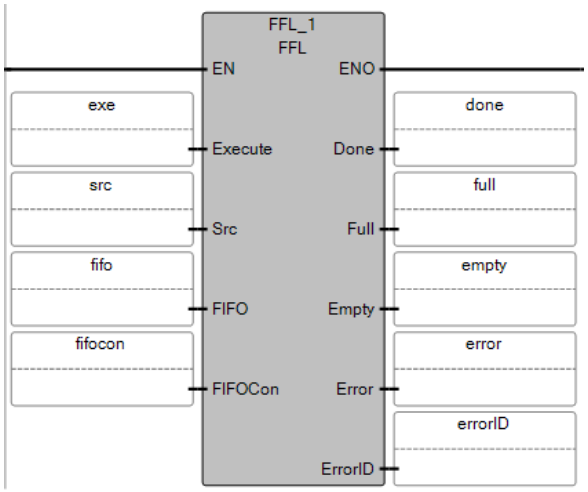
Code d'erreur	Description de l'erreur
0	Aucune erreur.
1	Le type de données Src FFL n'est pas pris en charge.

Code d'erreur	Description de l'erreur
2	Le type de données Dest FFU n'est pas pris en charge.
3	Le type de donnée FIFO n'est pas pris en charge.
4	Les types de données Src et Dest ne correspondent pas au type de données FIFO. Action corrective : Le type de données des paramètres FFL Src et FFU Dest doivent correspondre au type de données du tableau FIFO.
5	FIFO - La dimension du tableau n'est pas prise en charge. Action corrective : FIFO prend uniquement en charge les tableaux d'une dimension unique.
6	La longueur de contrôle FIFOCon ne peut pas dépasser la taille du tableau FIFO. Action corrective : La longueur de contrôle FIFOCon ne peut dépasser la taille du tableau FIFO.
7	La longueur FIFOCon dépasse la longueur maximale.
8	La longueur FIFOCon est zéro.
9	La position FIFOCon dépasse la longueur FIFOCon.
10	La longueur de contrôle FFL et la position sont égales.
11	La position de contrôle FFU est zéro.
12	La dimension du tableau FFL ou FFU n'est pas prise en charge. Action corrective : FFL et FFU prennent uniquement en charge les tableaux à une dimension.
13	FFL ou FFU DestOffset dépasse la taille du tableau Dest.

Exemple de diagramme de bloc de fonctions FFL



Exemple de langage à relais FFL



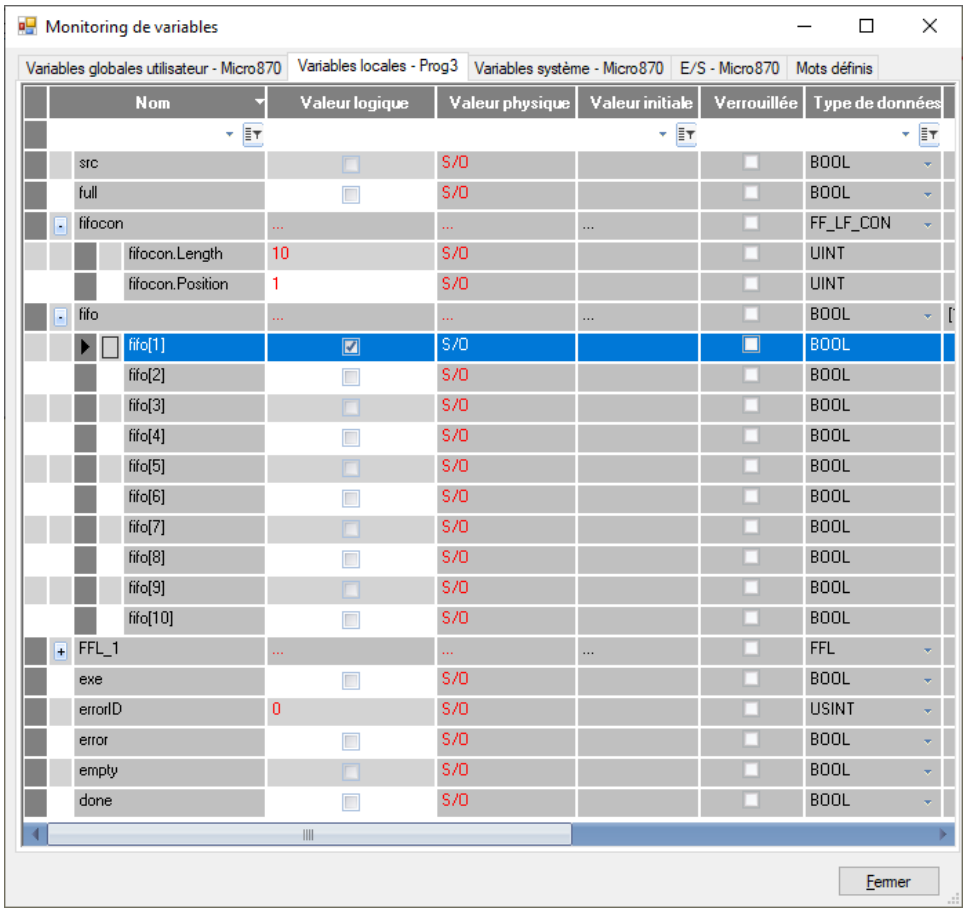
Exemple de texte structuré FFL

```
1  FFL_1(exe, src, fifo, fifocon);
2  done := FFL_1.Done;
3  full := FFL_1.Full;
4  empty := FFL_1.Empty;
5  error := FFL_1.Error;
6  errorID := FFL_1.ErrorID;
```

```
FFL_1(
```

void **FFL_1**(BOOL **Execute**, ANY_ELEMENTARY **Src**, ANY_ELEMENTARY[1..1] **FIFO**, FF_LF_CON **FIFOCon**)
Type: FFL, FIFO charger

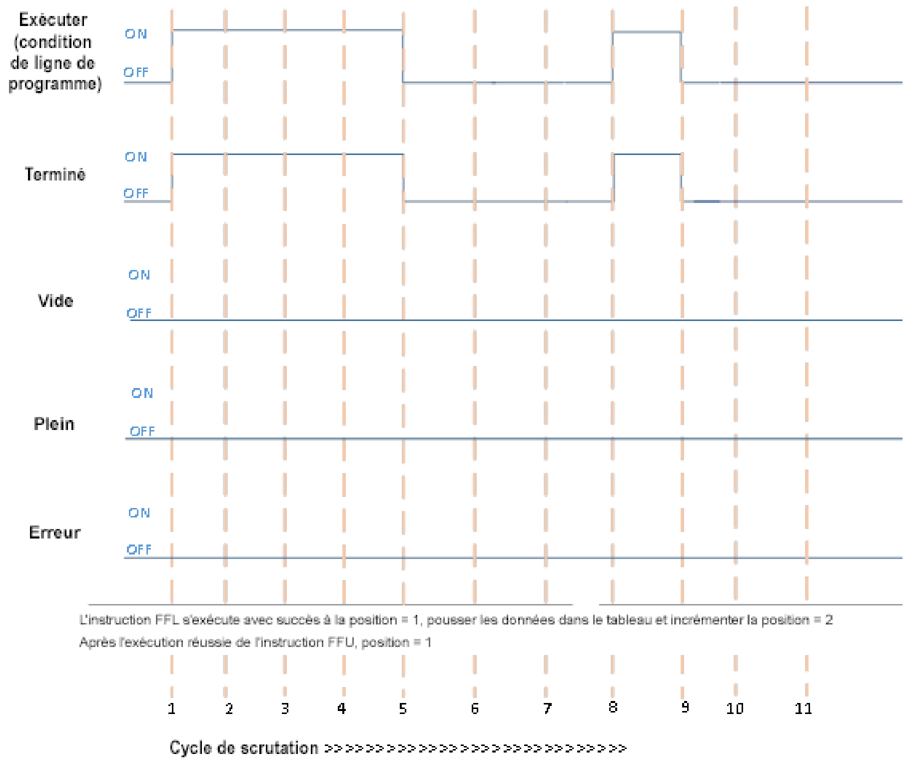
Résultats



Diagrammes de temporisation d'instruction FFL et FFU

Les exemples de diagrammes de temporisation d'instruction décrivent des scénarios d'exécution pour les instructions FFL sur la page 479 (chargement FIFO) et FFU sur la page 487 (déchargement FIFO).

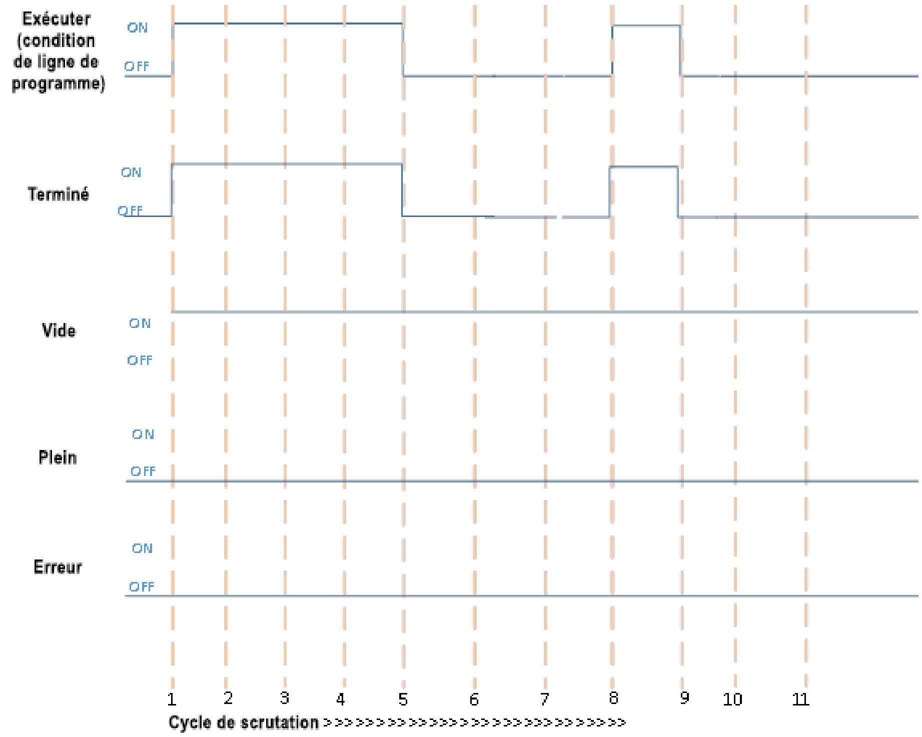
Exécution FFL réussie suivie d'une exécution FFU réussie



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. • Chargez les données dans la pile FIFO. • Le bit de sortie Terminé est VRAI.
2,3,4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Le bit de sortie Terminé est FAUX.
6, 7	Aucun changement de la condition de ligne. • Le bit Exécuter est FAUX. • Le bit de sortie Terminé est FAUX.
8	La ligne de programme devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. • Déchargez les données de la pile FIFO. • Le bit de sortie Terminé est VRAI.
9	Aucun changement de la condition de ligne. • Le bit Exécuter est FAUX. • Le bit de sortie Terminé est FAUX.
10, 11	Aucun changement de la condition de ligne.

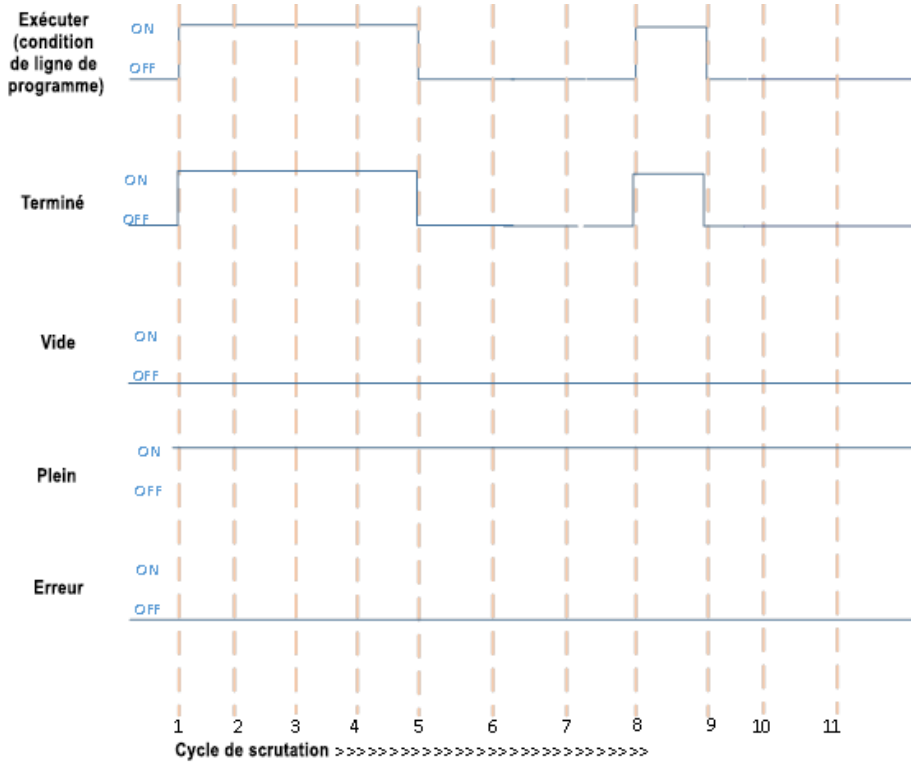
Exécution réussie lorsque le bit Vide est VRAI



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. Début de l'exécution. • La position est zéro. Le bit Vide est VRAI. • Le bit de sortie Terminé est VRAI.
2,3,4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Le bit Vide est VRAI. • Le bit de sortie Terminé est FAUX.
6, 7	Aucun changement de la condition de ligne.
8	La ligne de programme devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. Début de l'exécution. • Le bit Vide est VRAI. • Le bit de sortie Terminé est VRAI.
9	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Le bit Vide est VRAI. • Le bit de sortie Terminé est FAUX.
10, 11	Aucun changement de la condition de ligne.

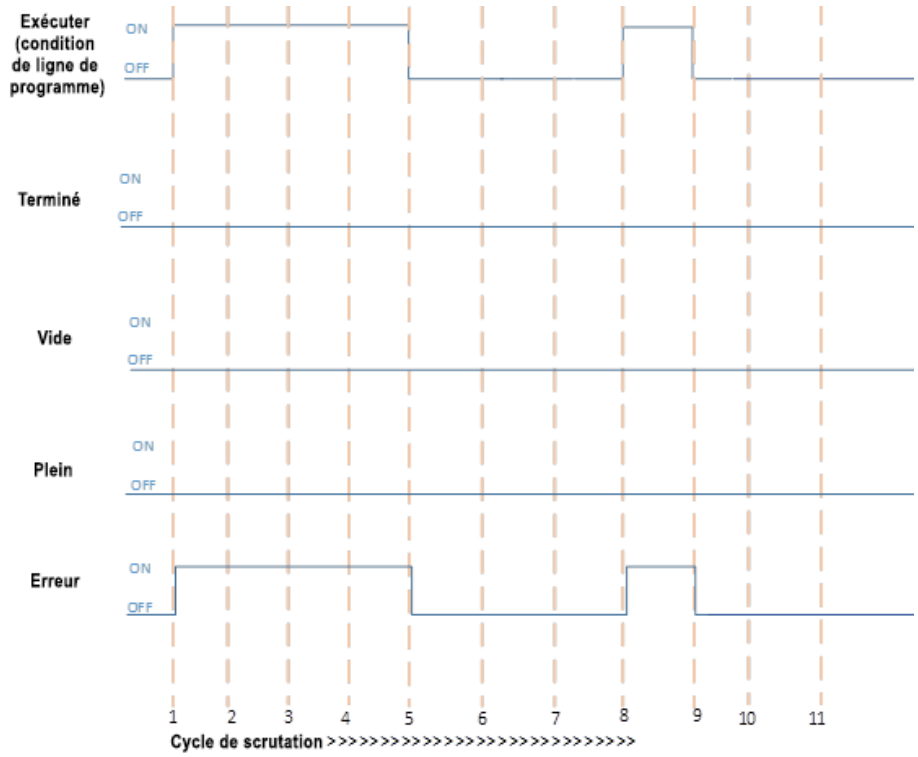
Exécution réussie lorsque le bit Vide est VRAI



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. Début de l'exécution. • La position est égale à la Longueur, le bit Complet est VRAI. • Le bit de sortie Terminé est VRAI.
2,3,4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Le bit Complet est VRAI. • Le bit de sortie Terminé est FAUX.
6, 7	Aucun changement de la condition de ligne.
8	La ligne de programme devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. Début de l'exécution. • Le bit Complet est VRAI. • Le bit de sortie Terminé est VRAI.
9	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Le bit Complet est VRAI. • Le bit de sortie Terminé est FAUX.
10, 11	Aucun changement de la condition de ligne.

Une erreur a été rencontrée durant l'exécution FFL et FFU



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. Début de l'exécution. • Le bit Erreur est VRAI.
2,3,4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Les bits Erreur et ID d'erreur sont FAUX.
6, 7	Aucun changement de la condition de ligne.
8	La ligne de programme devient VRAI lorsque : • Le bit d'entrée Exécuter est VRAI. Début de l'exécution. • Le bit Erreur est VRAI.
9	La condition de ligne devient FAUX lorsque : • Le bit Exécuter est FAUX. • Les bits Erreur et ID d'erreur sont FAUX.
10, 11	Aucun changement de la condition de ligne.

FFU (déchargement FIFO)

Décharge des données 8 bits, 16 bits, 32 bits ou 64 bits d'un tableau créé par l'utilisateur et appelé pile FIFO (First In First Out, premier entré, premier

sorti) dans le même ordre que les données ont été chargées avec l'instruction FFL.

Détails de l'opération :

- Instruction FFU - Mode sans exécution à mode d'exécution :
  - Lorsque l'exécution passe de FAUX à VRAI :
    - Les conditions d'erreur FFU sont vérifiées.
    - Décharge le contenu de la pile FIFO en position zéro si la Position est supérieure à zéro et inférieure ou égale à la Longueur.
    - Les éléments restants sont décalés d'une position vers zéro, les éléments les plus élevés de la pile FIFO sont définis sur zéro, et la Position est diminuée de 1.
    - Vide est défini sur VRAI si la Position est égale à zéro.
    - Done est défini lorsque l'instruction s'exécute correctement.
  - Lorsque l'exécution passe de VRAI à FAUX :
    - Error, Done et ErrorID sont définis sur FALSE.
    - Vide est défini sur VRAI si la Position est égale à zéro.
    - Complet est défini sur VRAI si la Longueur est égale à la Position. Complet est défini sur FAUX si la Position est inférieure ou égale à zéro, et inférieure à la Longueur.
    - Les conditions d'erreur FFU ne sont pas vérifiées.
  - Lorsque l'exécution passe de VRAI à VRAI :
    - Aucune opération de déchargement n'est effectuée.
    - Vide est défini sur VRAI si la Position est égale à zéro.
    - Complet est défini sur VRAI si la Longueur est égale à la Position.
    - Les conditions d'erreur FFU ne sont pas vérifiées.
  - Lorsque l'exécution passe de FAUX à FAUX :
    - Error, Done et ErrorID sont définis sur FALSE.
    - Vide est défini sur VRAI si la Position est égale à zéro.
    - Complet est défini sur VRAI si la Longueur est égale à la Position. Complet est défini sur FAUX si la Position est inférieure ou égale à zéro, et inférieure à la Longueur.
    - Les conditions d'erreur FLU ne sont pas vérifiées.
- Instruction FFU - Mode d'exécution à mode sans exécution :
  - Error, Done et ErrorID, Empty et Full conservent l'état du mode d'exécution.
- Pour créer un seul élément pour le paramètre FIFO :
  - Non-tableau :
    - Une adresse basée sur la variable (comme Fifo1) est autorisée pour FIFO.
    - La longueur doit être définie sur 1.
  - Tableau :

- Une adresse basée sur la variable (comme Fifo1 ou Fifo1[0]) est autorisée pour FIFO.
- La longueur doit être définie sur 1.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : si le front montant est détecté, l'opération FFU commence. FAUX : front montant non détecté.
Dest	Entrée	ANY_ELEMENTARY	Conserve la valeur qui existe dans la pile FIFO. Les types de données élémentaires pris en charge pour Dest sont les suivants : • DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • La chaîne n'est pas prise en charge. • Prend uniquement en charge les tableaux à une dimension, comme Array[1] ou Array[Index].
DestOffset	Entrée	UINT	Décalage de l'élément de destination. Décaler l'élément si la destination est un type de données Tableau. Sinon, le décalage doit être défini sur 0. Pour le type de données Tableau, le décalage doit être défini sur 0 afin de décharger le premier élément.
FIFO	Entrée	ANY_ELEMENTARY	L'adresse de début de la pile. FIFO doit être configuré de la même façon dans les instructions FFL et FFU. • Les types de données élémentaires pris en charge pour FIFO sont les suivants : DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • La chaîne n'est pas prise en charge. • Une seule dimension est prise en charge pour FIFO.
FIFOCon	Entrée	FF_LF_CON	Configuration et contrôle FIFO. La même configuration doit être définie pour les instructions FFL et FFU. Utilisez le type de données FF_LF_CON pour configurer la Position et la Longueur.
Done	Sortie	BOOL	Indique quand l'opération FFU est terminée. VRAI : opération terminée avec succès. FAUX : l'opération a rencontré une erreur, ou l'instruction FFU ne s'exécute pas.

Paramètre	Type de paramètre	Type de données	Description
Full	Sortie	BOOL	Indique quand la pile FIFO est complète. VRAI : lorsque la Longueur est égale à la Position. FAUX : lorsque la Position est supérieure à zéro, et inférieure à la Longueur.
Empty	Sortie	BOOL	Indique quand la pile FIFO est vide. VRAI : lorsque la Position est égale à 0. FAUX : lorsque la Position n'est pas égale à 0.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : l'opération a rencontré une erreur. FAUX : l'opération s'est terminée avec succès ou l'instruction ne s'exécute pas.
ErrorID	Sortie	USINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur FFU.

Type de données FF_LF_CON

Ce tableau permet de déterminer les valeurs de paramètre pour le type de données FF_LF_CON.

Paramètre	Type de données	Description
Length	UINT	Nombre d'éléments utilisés pour l'opération FIFO. La limite maximale est 1024.
Position	USINT	Détermine le prochain emplacement disponible dans le FIFO pour l'entrée ou la suppression Src. La position est le décalage du tableau. Exemple 1 : • Tableau configuré par l'utilisateur, arr[0..5]. La position initiale est définie sur 1. Les données sont poussées dans le tableau arr[1] et la position est incrémentée de +1. Exemple 2 : • Tableau configuré par l'utilisateur, arr[1..5]. La position initiale est définie sur 1. Les données sont poussées dans le tableau arr[2] et la position est incrémentée de +1.

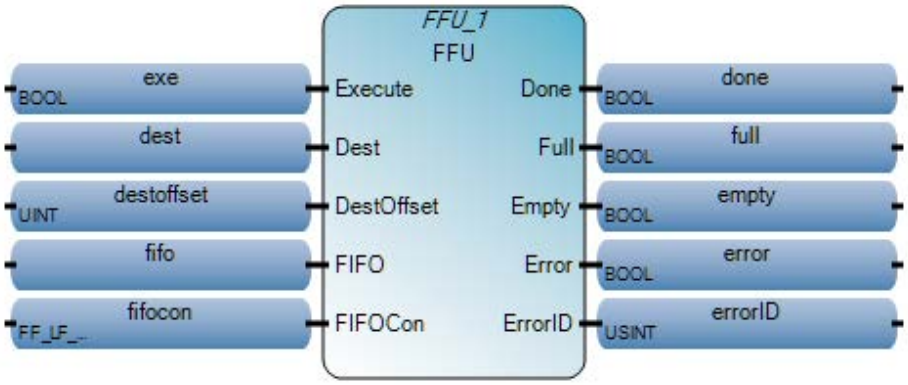
Codes d'erreur FFU

Utilisez ce tableau pour déterminer les codes d'erreur FFL et FFU et les descriptions.

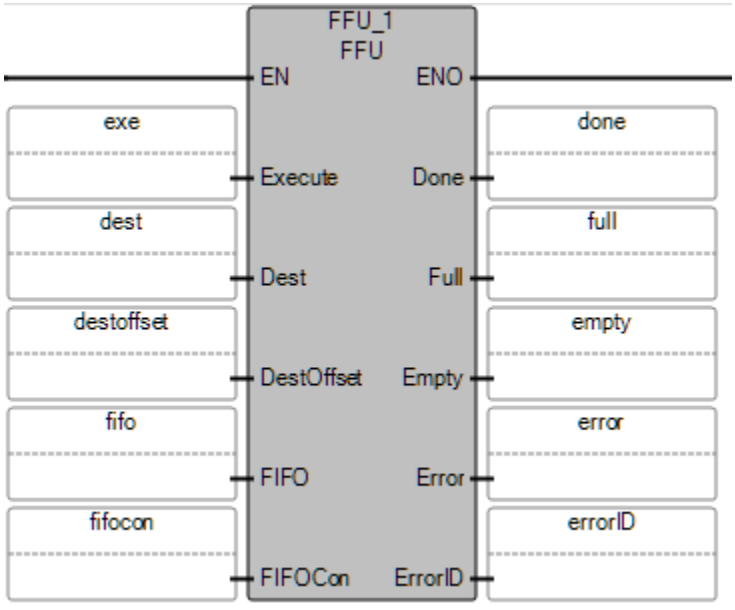
Code d'erreur	Description de l'erreur
0	Aucune erreur.
1	Le type de données Src FFL n'est pas pris en charge.
2	Le type de données Dest FFU n'est pas pris en charge.
3	Le type de donnée FIFO n'est pas pris en charge.
4	Les types de données Src et Dest ne correspondent pas au type de données FIFO. Action corrective : Le type de données des paramètres FFL Src et FFU Dest doivent correspondre au type de données du tableau FIFO.
5	FIFO - La dimension du tableau n'est pas prise en charge. Action corrective : FIFO prend uniquement en charge les tableaux d'une dimension unique.
6	La longueur de contrôle FIFOCon ne peut pas dépasser la taille du tableau FIFO. Action corrective : La longueur de contrôle FIFOCon ne peut dépasser la taille du tableau FIFO.
7	La longueur FIFOCon dépasse la longueur maximale.

Code d'erreur	Description de l'erreur
8	La longueur FIFOCon est zéro.
9	La position FIFOCon dépasse la longueur FIFOCon.
10	La longueur de contrôle FFL et la position sont égales.
11	La position de contrôle FFU est zéro.
12	La dimension du tableau FFL ou FFU n'est pas prise en charge. Action corrective : FFL et FFU prennent uniquement en charge les tableaux à une dimension.
13	FFL ou FFU DestOffset dépasse la taille du tableau Dest.

Exemple de diagramme de bloc de fonctions FFU



Exemple de langage à relais FFU

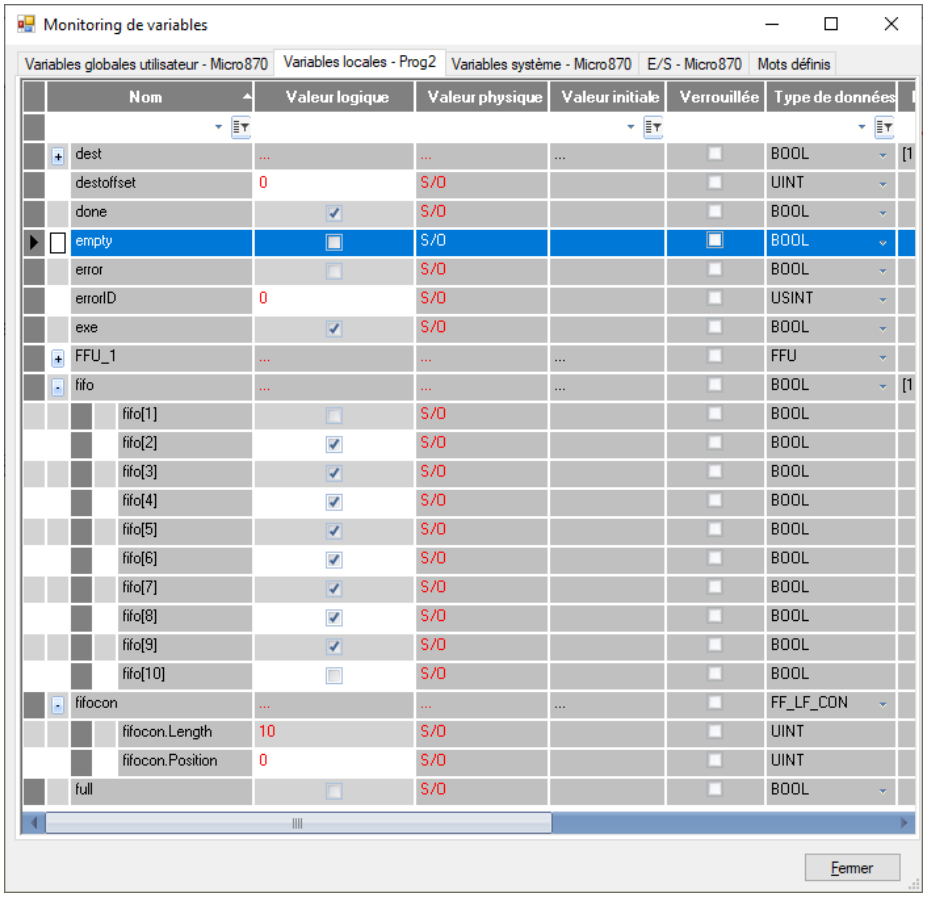


Exemple de texte structuré FFU

```
1  FFU_1(exe, dest, destoffset, fifo, fifocon);
2  done := FFU_1.Done;
3  full := FFU_1.Full;
4  empty := FFU_1.Empty;
5  error := FFU_1.Error;
6  errorID := FFU_1.ErrorID;
```

```
FFU_1(
void FFU_1(BOOL Execute, ANY_ELEMENTARY[1..1] Dest, UINT DestOffset, ANY_ELEMENTARY[1..1] FIFO, FF_LF_CON FIFOCon)
Type: FFU, FIFO décharger
```

Résultats

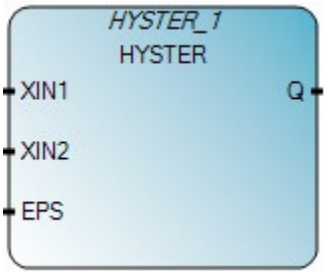


HYSTER (hystérésis)

Hystérésis de type booléen sur la différence entre des réels. Compare la valeur actuelle d'une entrée avec la limite haute établie en ajoutant la quantité historique de décalage mesurée par une hystérésis à la valeur attendue d'une entrée et en déterminant si la valeur actuelle dépasse cette limite.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

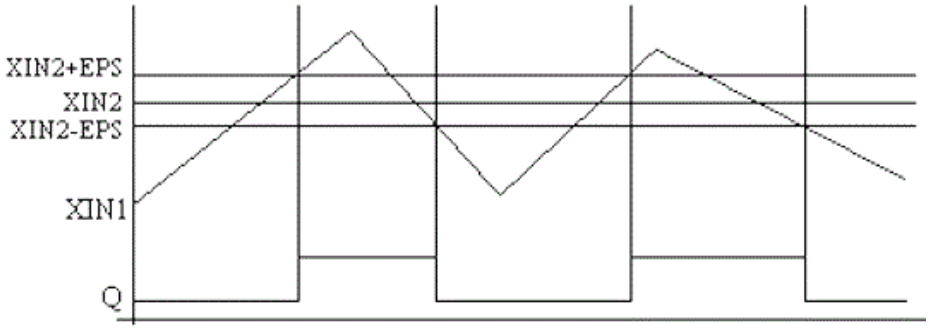


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

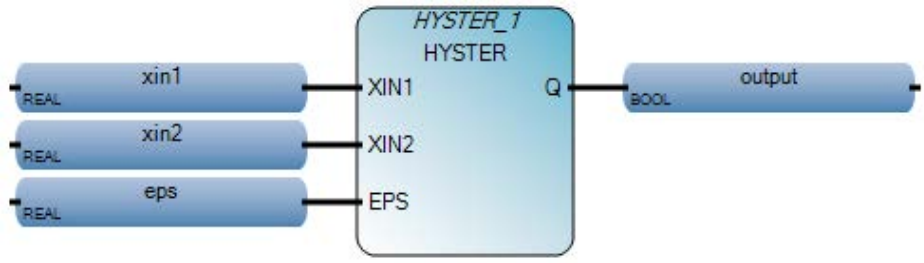
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution du bloc d'instruction. FAUX : aucune exécution du bloc d'instruction. Ne s'applique qu'aux programmes de langage à relais.
XIN1	Entrée	REAL	Valeur réelle, quelle qu'elle soit.
XIN2	Entrée	REAL	Pour tester si la valeur d'entrée XIN1 a dépassé la limite haute définie pour cette entrée XIN2 + EPS.
EPS	Entrée	REAL	Valeur d'hystérésis (doit être supérieure à zéro).
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Q	Sortie	BOOL	Résultat de l'instruction HYSTER. VRAI : l'entrée a dépassé la limite supérieure, mais n'est pas en dessous de la limite inférieure. FAUX : l'entrée n'a pas dépassé la limite supérieure.

Exemple de diagramme de temporisation HYSTER

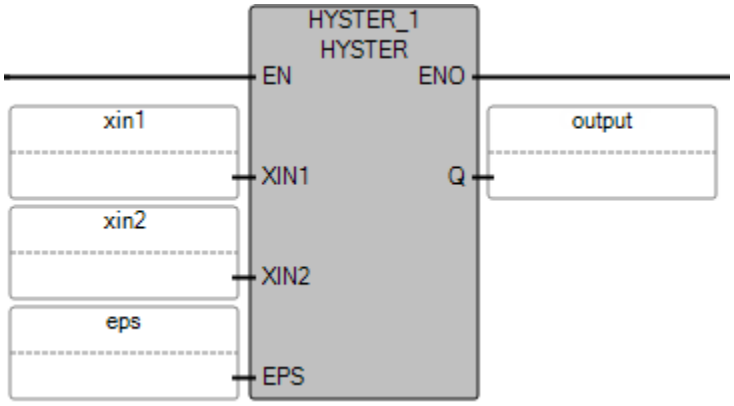
Dans le diagramme suivant, HYSTER est utilisé pour évaluer le décalage de performance dû à la friction du moteur pendant 5 secondes. L'instruction est exécutée toutes les 10 millisecondes. Lors de la phase de démarrage, le moteur fonctionnait moins efficacement.



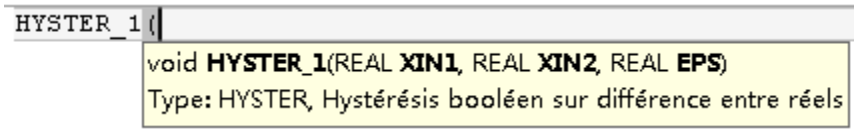
Exemple de Diagramme de Bloc Fonction HYSTER



Exemple de langage à relais HYSTER



Exemple de texte structuré HYSTER




```
1  xin1 := 10.0;
2  xin2 := 1.0;
3  eps := 1.0;
4  HYSTER_1(xin1, xin2, eps);
5  output := HYSTER_1.Q;
```

INTEGRAL

Intègre une valeur réelle pendant le temps de cycle défini.

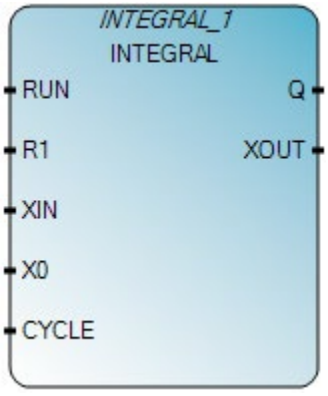
Détails de l'opération :

- lorsque le bloc fonctionnel INTEGRAL est initialisé pour la première fois, ses valeurs initiales ne sont pas prises en compte. Utilisez le paramètre R1 pour définir les valeurs initiales d'un calcul.
- Pour éviter la perte de la valeur intégrée, la valeur d'intégration n'est pas effacée automatiquement lorsque le contrôleur passe de PROGRAM à RUN ou lorsque le paramètre RUN passe de FAUX à VRAI. Utilisez le paramètre R1 pour effacer la valeur intégrale lors du premier passage du contrôleur de PROGRAM à RUN et lors du démarrage d'une nouvelle intégration.

- Il est recommandé de ne pas utiliser les paramètres EN ou ENO facultatifs avec ce bloc de fonctions, car le calcul du temps de cycle sera faussé si EN équivaut à « FAUX », ce qui se traduira par une intégration incorrecte. Si vous utilisez les paramètres EN ou ENO, basculez le paramètre R1 en définissant EN sur « TRUE » pour effacer le résultat actuel et assurer une intégration correcte.
- L'intégration est calculée en millisecondes (l'intégration de l'entrée 1 ayant la valeur initiale 0 sur une période d'une seconde donne la valeur 1 000). Pour convertir la sortie de l'instruction en unités de secondes, divisez-la par 1 000.
- Si la valeur du paramètre CYCLE est inférieure à la durée du cycle d'exécution de l'appareil, la période d'échantillonnage est alignée de force sur la durée de ce cycle.
- L'échantillonnage XIN et les exécutions de blocs de fonction interviennent à chaque temps de cycle + gigue du temps de scrutation.
- Pour un programme utilisateur donné, la gigue de temps de scrutation varie d'un contrôleur à l'autre.
- Le temps de cycle détermine la sensibilité du bloc fonctionnel Integral. Les modifications intervenues dans XIN entre deux échantillonnages (ou pendant le temps de cycle) ne sont pas prises en compte dans le calcul de la valeur de l'intégrale XOUT.
- Le temps du cycle et la gigue du temps de scrutation contribuent à l'incertitude globale de la sortie Integral comme indiqué dans le XIN synchronisé avec l'exemple d'exécution d'un bloc de fonctions et dans le XIN non synchronisé avec l'exemple d'exécution d'un bloc de fonctions.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

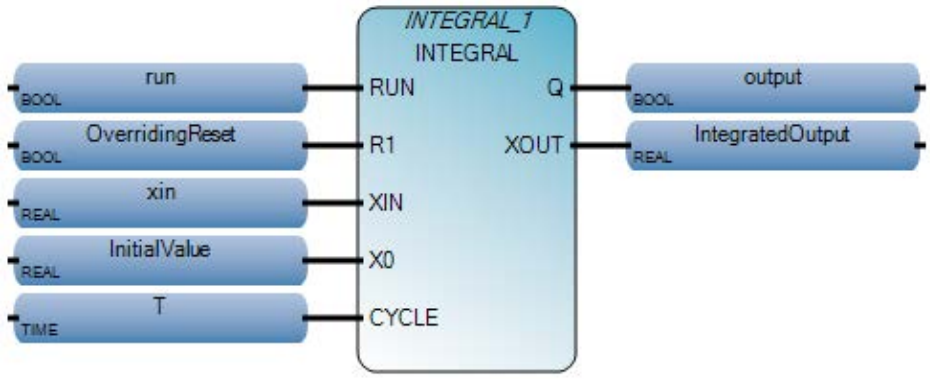


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

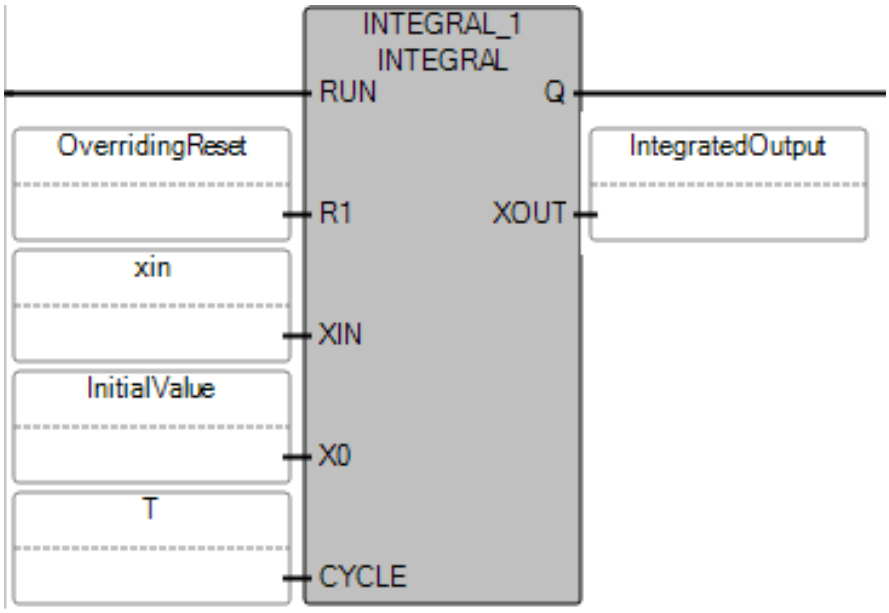
Paramètre	Type de paramètre	Type de données	Description
RUN	Entrée	BOOL	Mode : VRAI = intégration / FAUX = conservation.

Paramètre	Type de paramètre	Type de données	Description
R1	Entrée	BOOL	Annulation de la réinitialisation.
XIN	Entrée	REAL	Entrée : valeur réelle, quelle qu'elle soit.
X0	Entrée	REAL	Valeur initiale.
CYCLE	Entrée	TIME	Période d'échantillonnage. Les valeurs possibles sont comprises entre 0 ms et 49 j 17 h 2 m 47 s 294 ms.
Q	Sortie	BOOL	Différent de R1.
XOUT	Sortie	REAL	Sortie intégrée.

Exemple de Diagramme de Bloc Fonction INTEGRAL



Exemple de logique à relais INTEGRAL



Exemple de texte structuré INTEGRAL

```
INTEGRAL_1(
```

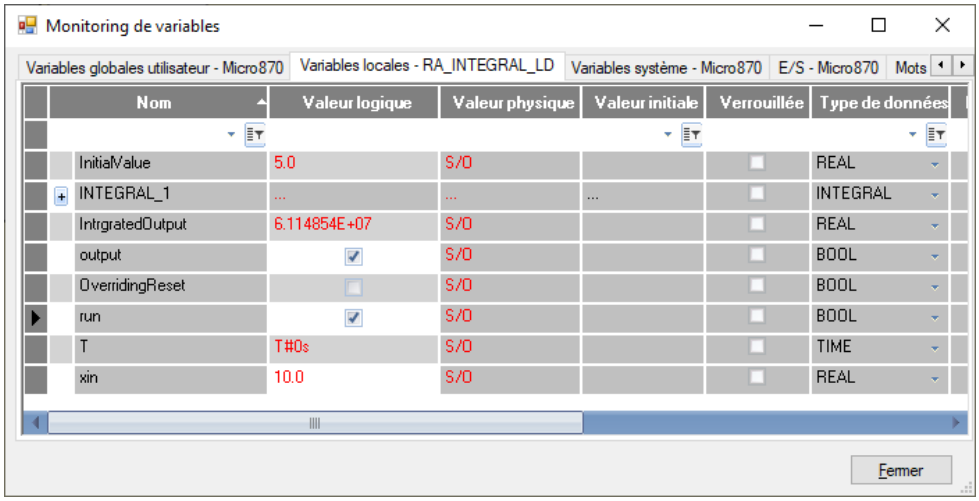
void **INTEGRAL_1**(BOOL **RUN**, BOOL **R1**, REAL **XIN**, REAL **X0**, TIME **CYCLE**)
Type: INTEGRAL, Différentiation par rapport au temps

```
1  run := TRUE;
2  xin := 10.0;
3  InitialValue := 5.0;
4  T := T#10s;
5  INTEGRAL_1(run, OverridingReset, xin, InitialValue, T);
6  output := NOT OverridingReset;
7  IntegratedOutput := INTEGRAL_1.XOUT;
```

(* Équivalence ST : INTEGRAL1 est une instance d'un bloc INTEGRAL *)

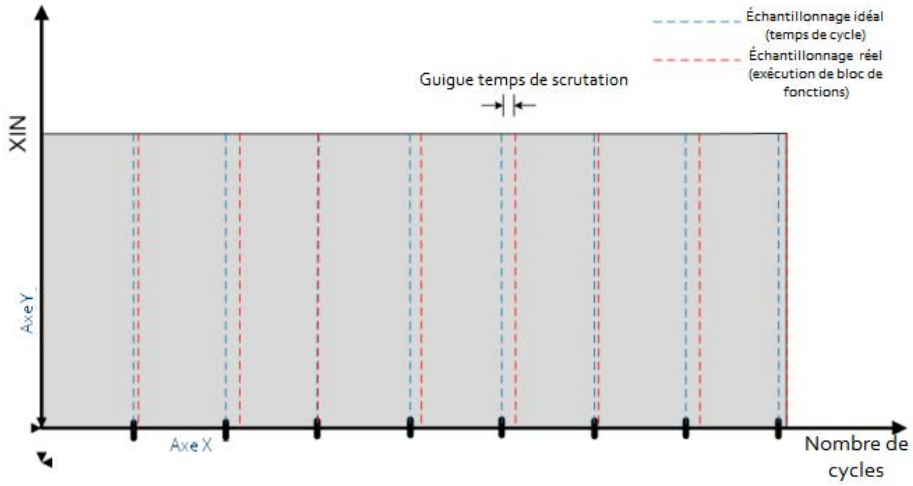
```
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value,
init_value, t#100ms) ;
controlled_value := INTEGRAL1.XOUT ;
```

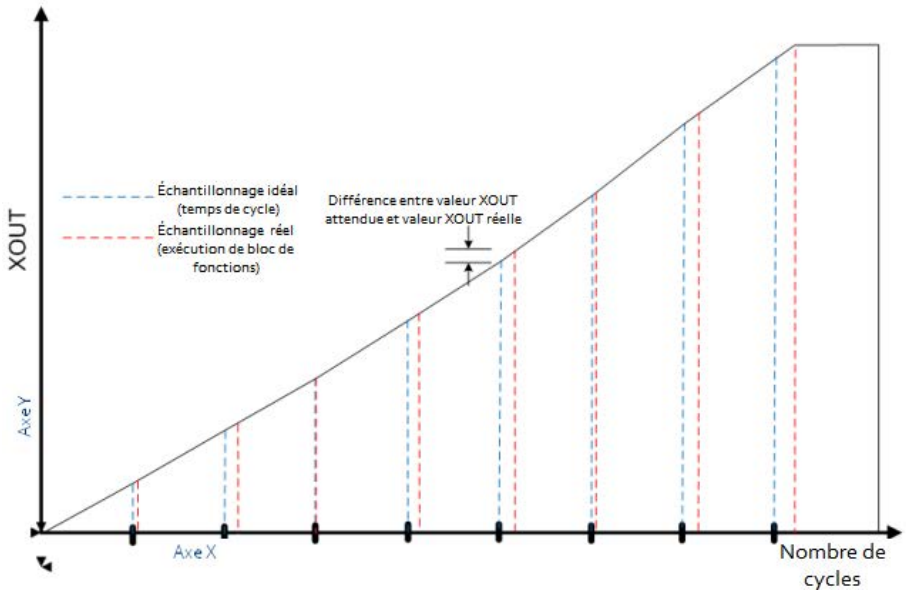
Résultats



XIN synchronisé avec l'exécution d'un bloc fonctionnel

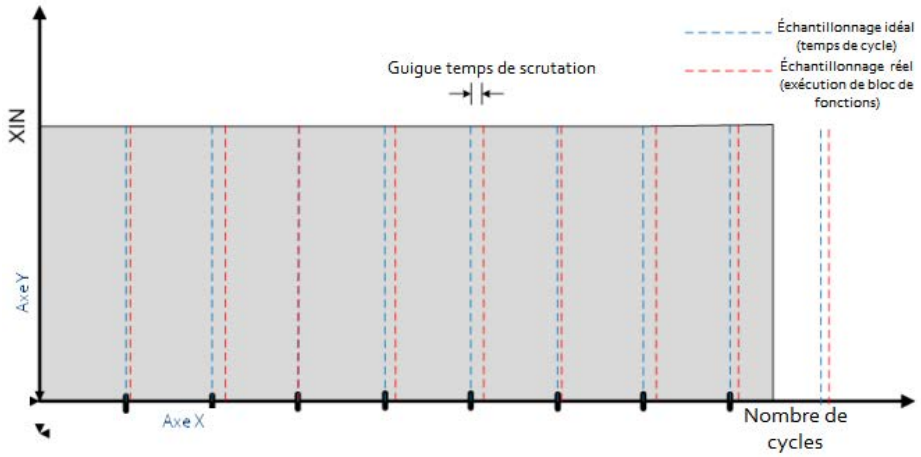
Les images suivantes montrent l'effet de la gigue du temps de scrutation sur la valeur XOUT :

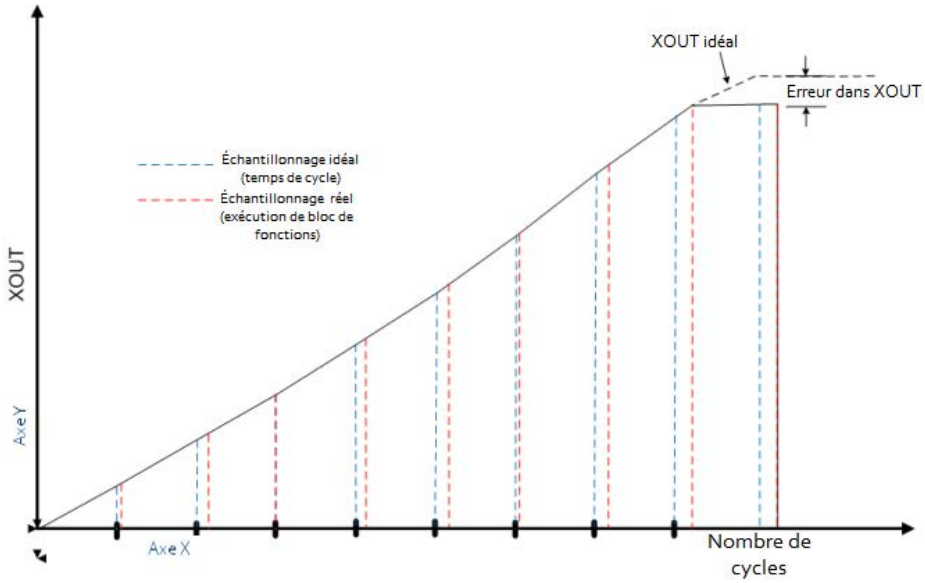




Exemple de XIN non synchronisé avec l'exécution d'un bloc fonctionnel

Les images suivantes montrent un exemple dans lequel une erreur est introduite dans la valeur XOUT du bloc de fonctions Integral :



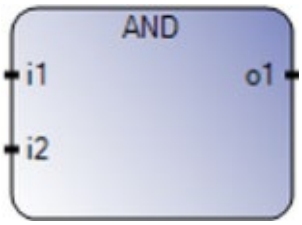


AND

Effectue une opération booléenne AND (ET) entre deux ou plusieurs valeurs.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
i1	Entrée	BOOL	Valeur de type de données booléen.
i2	Entrée	BOOL	Valeur de type de données booléen.
o1	Sortie	BOOL	Résultat de l'opération booléenne AND (ET) sur les valeurs d'entrée.

Exemple de texte structuré AND

(* Équivalence ST : *)

```
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

LFL (chargement LIFO)

L'instruction LFL est utilisée pour charger les données (8 bits, 16 bits, 32 bits ou 64 bits) dans un tableau créé par l'utilisateur appelé une pile LIFO. Les instructions LFL et LFU sur la page 502 sont utilisées par paires.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

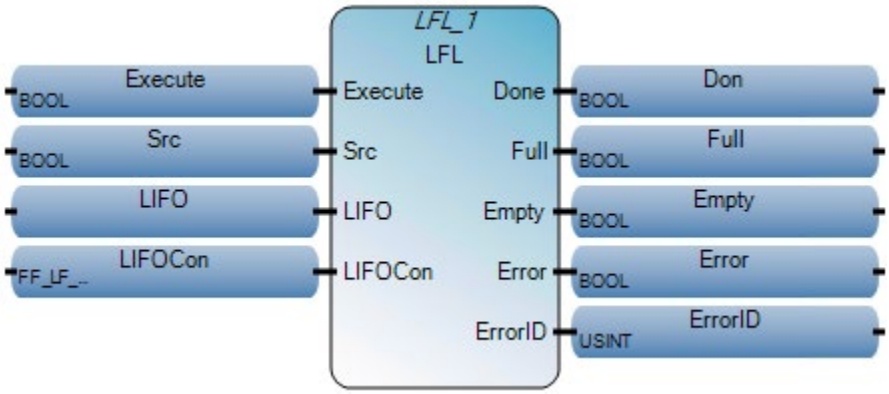
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour ces instructions.

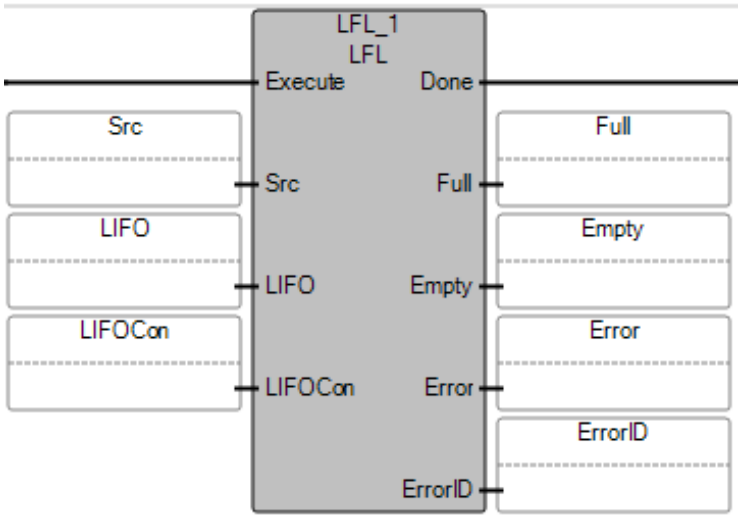
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : si le front montant est détecté, l'opération LFL commence. FAUX : front montant non détecté.
Src	Entrée	ANY_ELEMENTARY	L'opérande Src est l'adresse de la valeur pour renseigner la position actuellement disponible dans la pile LIFO. Types de données d'élément pris en charge : • DWORD, REAL, TIME, DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT.\ • Les éléments de tableau comme Array[1] ou Array[Index]. • La chaîne n'est pas prise en charge.

Paramètre	Type de paramètre	Type de données	Description
LIFO	Entrée	ANY_ELEMENTARY	L'adresse de début de la pile. LIFO doit être configuré de la même façon pour les instructions LFL et LFU. Types de données d'élément pris en charge : • DWORD, REAL, TIME, DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • Une seule dimension est prise en charge. • La chaîne n'est pas prise en charge.
LIFOCon	Entrée	FF_LF_CON	Configuration et contrôle LIFO. La même configuration doit être définie pour les instructions LFL et LFU. Utilisez le type de données FF_LF_CON pour configurer la Position et la Longueur.
Full	Sortie	BOOL	Indique quand la pile LIFO est complète. VRAI : lorsque la Longueur est égale à la Position. FAUX : lorsque la Position est supérieure ou égale à zéro, et inférieure à la Longueur.
Empty	Sortie	BOOL	Indique quand la pile LIFO est vide. VRAI : lorsque la Position est égale à 0. FAUX : lorsque la Position n'est pas égale à 0.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : l'opération a rencontré une erreur. FAUX : l'opération s'est terminée avec succès ou l'instruction ne s'exécute pas.
ErrorID	Sortie	USINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur LFL.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : opération terminée avec succès. FAUX : l'opération a rencontré une erreur, ou l'instruction LFL ne s'exécute pas.

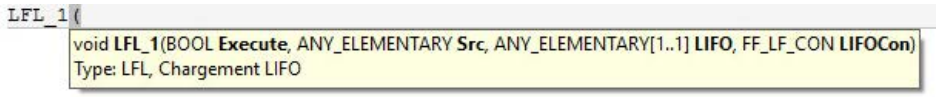
Exemple de Diagramme de Bloc Fonction LFL



Exemple de langage à relais LFL



Exemple de texte structuré LFL




```
1  LFL_1(exe, Src, fifo, fifocon);
2  Done:=LFL_1.Done;
3  Full:=LFL_1.Full;
4  Empty:=LFL_1.Empty;
5  Error:=LFL_1.Error;
6  ErrorID:=LFL_1.ErrorID;
```

LFU (déchargement LIFO)

L'instruction LFU décharge les données (8 bits, 18 bits, 32 bits ou 64 bits) depuis un tableau créé par l'utilisateur appelé une pile LIFO. Les instructions LFU et LFL sur la page 500 sont utilisées par paires.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

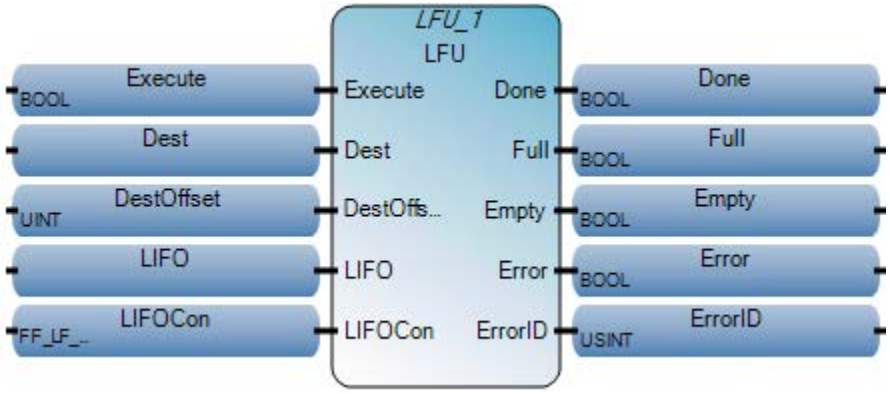
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour ces instructions.

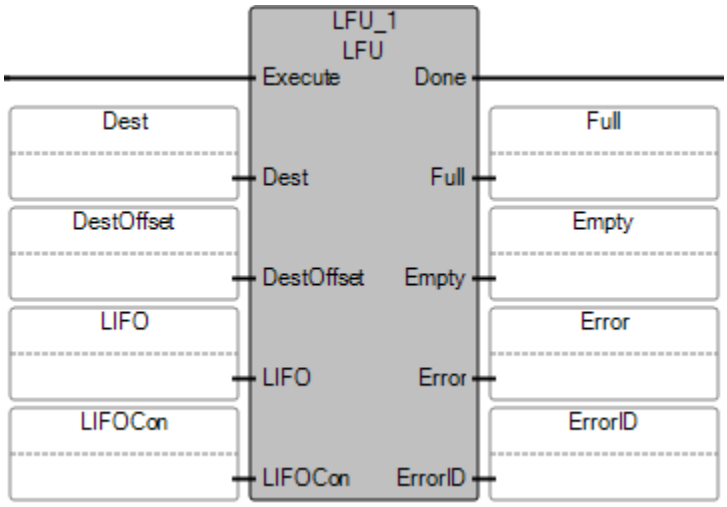
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : si le front montant est détecté, l'opération LFU commence. FAUX : front montant non détecté.
Dest	Entrée	ANY_ELEMENTARY	Conserve la valeur qui existe dans la pile LIFO. Les types de données élémentaires pris en charge pour Dest sont les suivants : • DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • La chaîne n'est pas prise en charge. • Prend uniquement en charge les tableaux à une dimension, comme Array[1] ou Array[Index].
DestOffset	Entrée	UINT	Décalage de l'élément de destination. Décaler l'élément si la destination est un type de données Tableau. Sinon, le décalage doit être défini sur 0. Pour le type de données Tableau, le décalage doit être défini sur 0 afin de décharger le premier élément.
LIFO	Entrée	ANY_ELEMENTARY	L'adresse de début de la pile. LIFO doit être configuré de la même façon pour les instructions LFL et LFU. Types de données d'élément pris en charge : • DWORD, REAL, TIME, DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • Une seule dimension est prise en charge. • La chaîne n'est pas prise en charge.
LIFOCon	Entrée	FF_LF_CON	Configuration et contrôle LIFO. La même configuration doit être définie pour les instructions LFL et LFU. Utilisez le type de données FF_LF_CON pour configurer la Position et la Longueur.
Full	Sortie	BOOL	Indique quand la pile LIFO est complète. VRAI : lorsque la Longueur est égale à la Position. FAUX : lorsque la Position est supérieure ou égale à zéro, et inférieure à la Longueur.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : l'opération a rencontré une erreur. FAUX : l'opération s'est terminée avec succès ou l'instruction ne s'exécute pas.
ErrorID	Sortie	USINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur LFU.

Paramètre	Type de paramètre	Type de données	Description
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : opération terminée avec succès. FAUX : l'opération a rencontré une erreur, ou l'instruction LFU ne s'exécute pas.

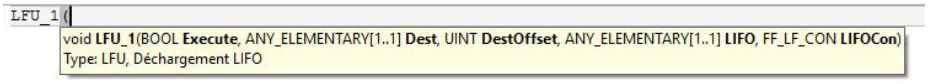
Exemple de Diagramme de Bloc Fonction LFU



Exemple de langage à relais LFU



Exemple de texte structuré LFU




```
1  LFU_1(exe, Dest, destoffset, lifo, lifocon);
2  Done:=LFU_1.Done;
3  Full:=LFU_1.Full;
4  Empty:=LFU_1.Empty;
5  Error:=LFU_1.Error;
6  ErrorID:=LFU_1.ErrorID;
```

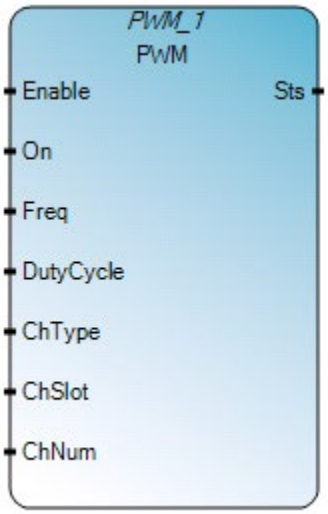
PWM (Pulse Width Modulation)

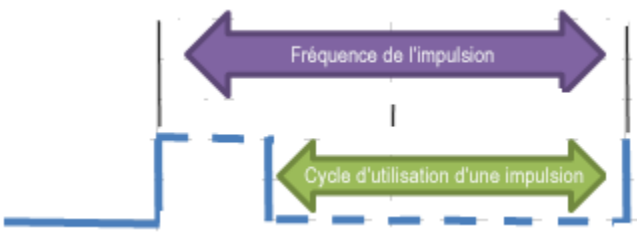
Active ou désactive la sortie PWM (Pulse Width Modulation) pour un canal PWM configuré.

Ce bloc d'instruction est utilisé avec les contrôleurs Micro820 2080-LC20-20QBB et prend en charge un canal PWM en utilisant le canal de sortie intégrée 6.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique uniquement au contrôleur Micro820.

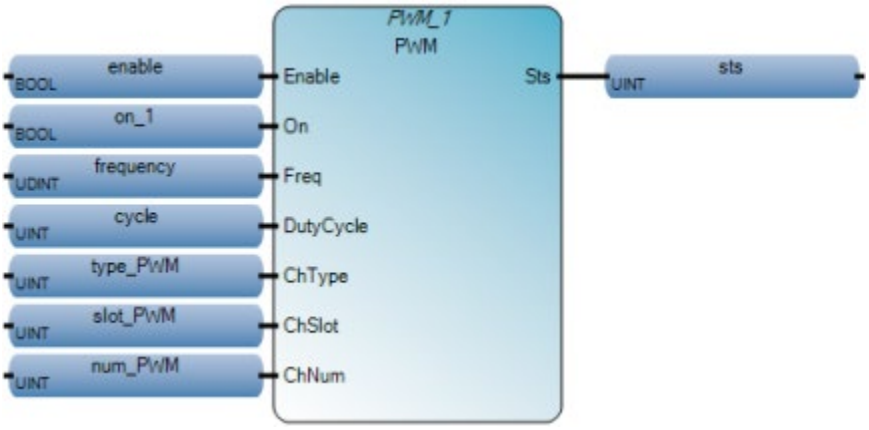




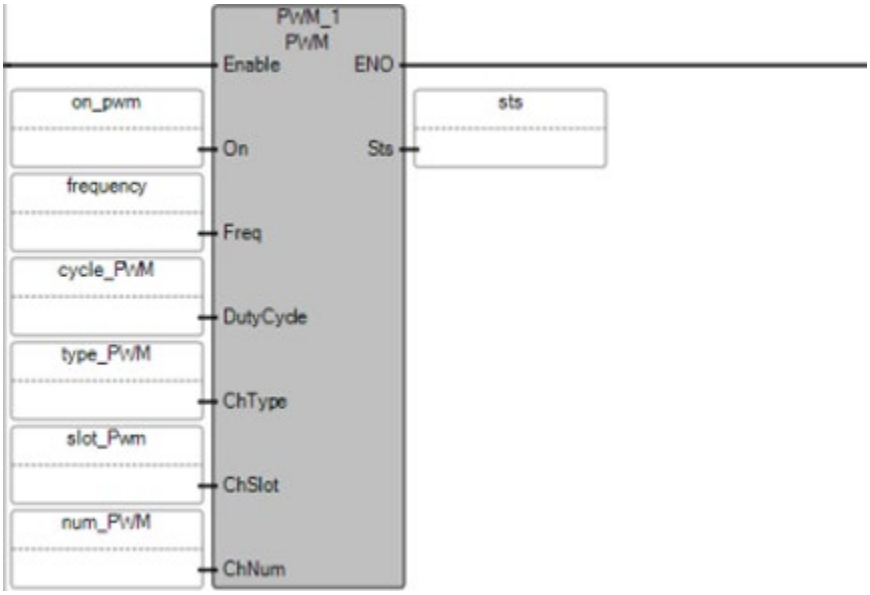
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. Ce niveau est activé en fonction du bloc d'instruction. VRAI : mise à jour de Sts. PWM est rendu actif ou inactif selon le paramètre d'entrée Actif et selon que la configuration est valide ou non. FAUX : Sts est uniquement mis à jour. L'état PWM (actif ou inactif) n'est pas affecté.
On	Entrée	BOOL	Active ou désactive la sortie PWM. VRAI : la sortie PWM est active ou reste active avec la dernière configuration valide. La LED de sortie est allumée lorsque PWM est active, même si le cycle d'utilisation est défini à 0 %. FAUX : la sortie PWM est inactive si la configuration est également valide.
Freq	Entrée	UDINT	Fréquence de l'impulsion. • 1 à 100 000 Hz
DutyCycle	Entrée	UINT	Cycle d'utilisation d'une impulsion. • 0 – 1 000 (0 % - 100 %)
ChType	Entrée	UINT	Type de voie • 0 – intégré • 1 – enfichable • 2 – expansion
ChSlot	Entrée	UINT	Emplacement pour voie • 0 – intégré
ChNum	Entrée	UINT	Numéro pour voie • 0 – PWM CH0
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.
Sts	Sortie	UINT	Codes d'état PWM : • 00 - Bloc fonctionnel non activé (aucune opération). • 01 - Configuration PWM réussie. • 02 - Cycle d'utilisation non valide. • 03 - Fréquence non valide. • 04 - Type de canal non valide. • 05 - Emplacement du canal non valide. • 06 - Numéro du canal non valide. • 07 - Catalogue non valide. La fonctionnalité PWM n'est pas prise en charge dans le catalogue en cours d'utilisation.

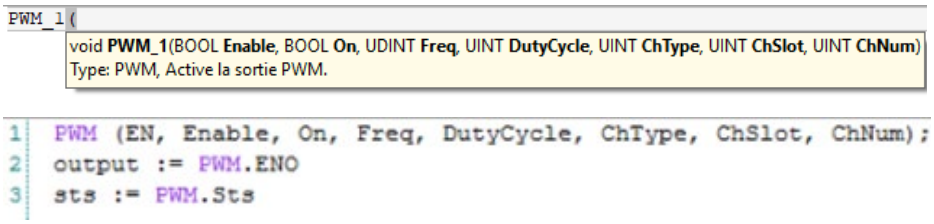
Exemple de diagramme de bloc de fonctions PWM



Exemple de langage à relais PWM



Exemple de texte structuré PWM



SCALER (échelle)

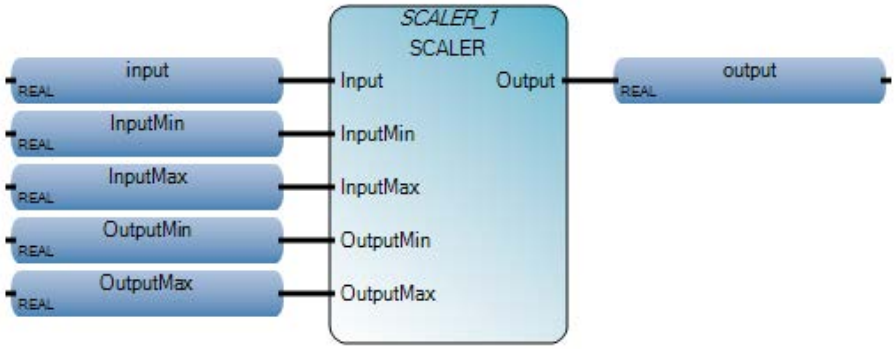
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

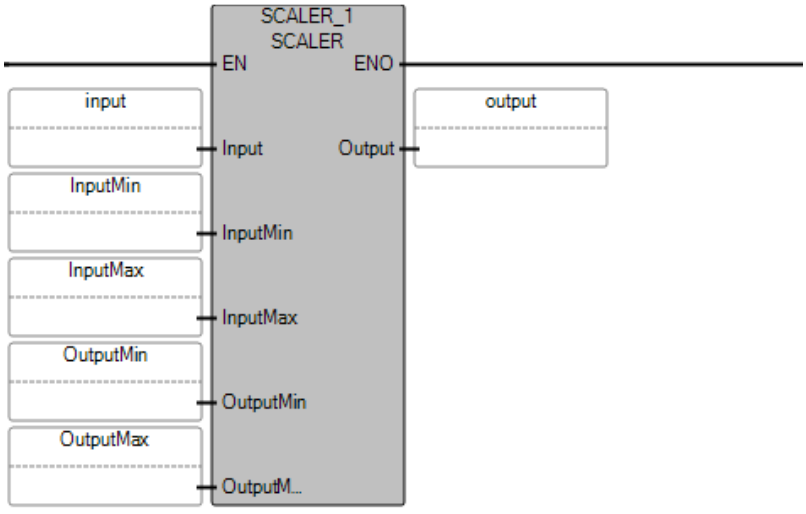
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de l'équation de mise à l'échelle. FAUX : aucune équation de mise à l'échelle. Ne s'applique qu'aux programmes de langage à relais.
Input	Entrée	REAL	Signal d'entrée. L'entrée n'est pas limitée par InputMin et InputMax. Pour limiter l'entrée, une instruction LIM_ALRM est nécessaire afin de conditionner l'entrée avant qu'elle ne soit saisie dans l'instruction SCALER.
InputMin	Entrée	REAL	Détermine la valeur de l'inclinaison et du décalage.
InputMax	Entrée	REAL	Détermine la valeur de l'inclinaison et du décalage.
OutputMin	Entrée	REAL	Détermine la valeur de l'inclinaison et du décalage.
OutputMax	Entrée	REAL	Détermine la valeur de l'inclinaison et du décalage.
Output	Sortie	REAL	Sortie mise à l'échelle. La sortie n'est pas limitée entre OutputMin et OutputMax.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions SCALER



Exemple de langage à relais SCALER



Exemple de texte structuré SCALER

```
SCALER_1(
```

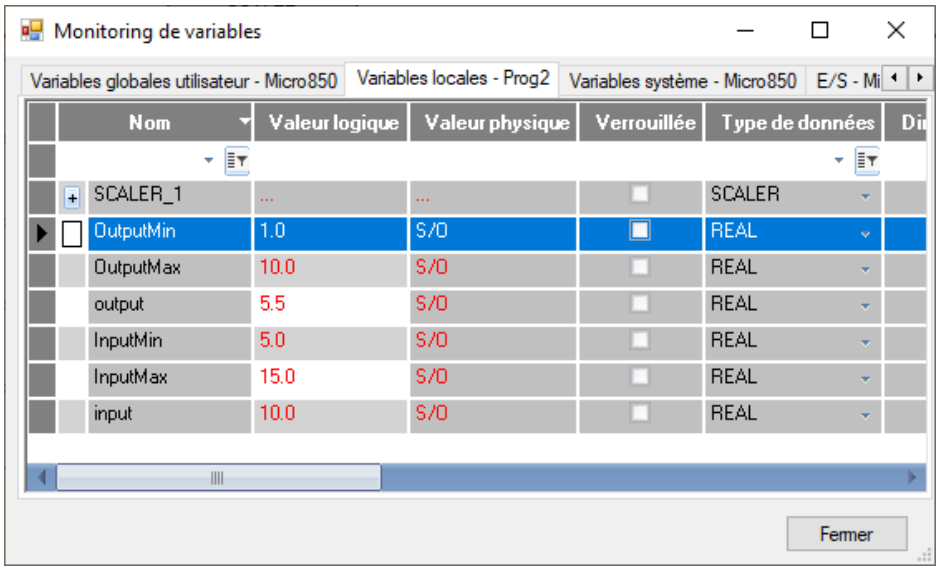
void **SCALER_1**(REAL **Input**, REAL **InputMin**, REAL **InputMax**, REAL **OutputMin**, REAL **OutputMax**)
Type: SCALER, Mise en échelle de la valeur d'entrée selon la plage de la sortie.

```
1  input := 10.0;
2  InputMin := 5.0;
3  InputMax := 15.0;
4  OutputMin := 1.0;
5  OutputMax := 10.0;
6  SCALER_1(input, InputMin, InputMax, OutputMin, OutputMax);
7  output := SCALER_1.Output;
```

(* Équivalence ST : SCALER1 est une instance du bloc SCALER *)

```
SCALER1(Signal_In, 4.0, 20.0 , 0.0 , 150.0 ) ;
Out_Temp := SCALER1.Output ;
```

Résultats



STACKINT (pile d'entiers)

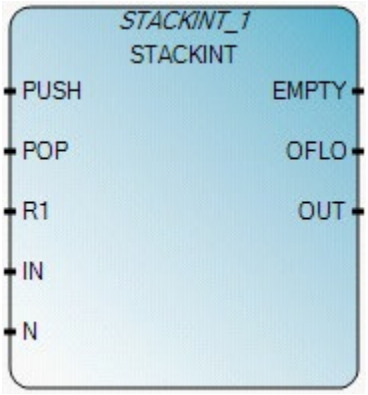
Gère une pile de valeurs entières.

Détails de l'opération :

- STACKINT inclut la détection d'un front montant pour les commandes PUSH et POP. La taille maximale de la pile est de 128. La valeur OFLO n'est valide qu'après une réinitialisation (le paramètre R1 a été défini sur TRUE au moins une fois, puis de nouveau sur FALSE).
- La taille de la pile définie par l'application (N) ne peut être inférieure à 1 ni supérieure à 128.
  - Si N < 1, STACKINT adopte une taille de 1.
  - Si N > 128, STACKINT adopte une taille de 128.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

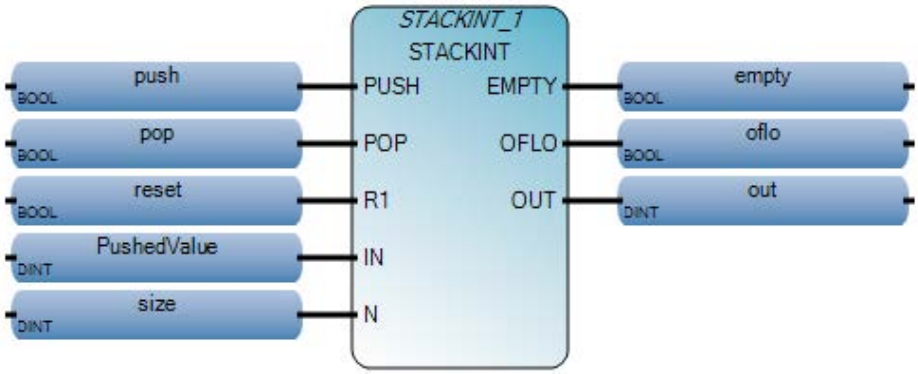
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



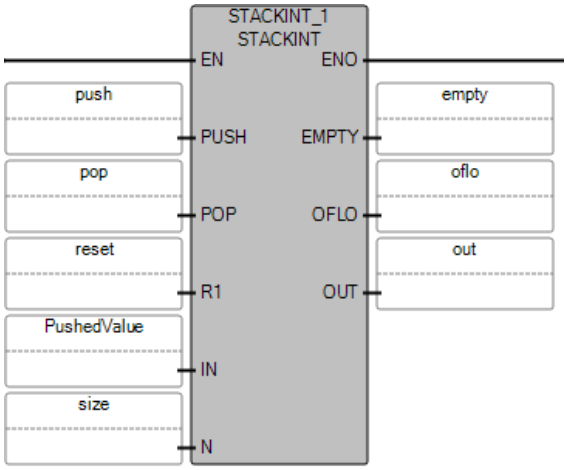
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
PUSH	Entrée	BOOL	VRAI : front montant détecté sur la commande PUSH. Ajoute la valeur IN en haut de la pile. FAUX : front montant non détecté sur la commande PUSH.
POP	Entrée	BOOL	VRAI : front montant détecté sur la commande PUSH. Supprime la dernière valeur poussée en haut de la pile. FAUX : front montant non détecté sur la commande POP.
R1	Entrée	BOOL	VRAI : réinitialise la pile en la vidant. FAUX : aucune réinitialisation.
IN	Entrée	DINT	Valeur poussée.
N	Entrée	DINT	Taille de la pile définie par l'application. La taille maximale de la pile est de 128.
EMPTY	Sortie	BOOL	VRAI : la pile est vide. FAUX : la pile contient des valeurs.
OFLO	Sortie	BOOL	VRAI : débordement, la pile est pleine et le paramètre R1 a été défini sur VRAI au moins une fois, puis de nouveau sur FAUX. FAUX : la taille de la pile est égale ou inférieure à 128. Pas de débordement.
OUT	Sortie	DINT	Valeur située en haut de la pile. OUT est égal à 0 lorsque OFLO est défini sur VRAI.

Exemple de diagramme de bloc de fonctions STACKINT



Exemple de langage à relais STACKINT



Exemple de texte structuré STACKINT

```
STACKINT_1(
```

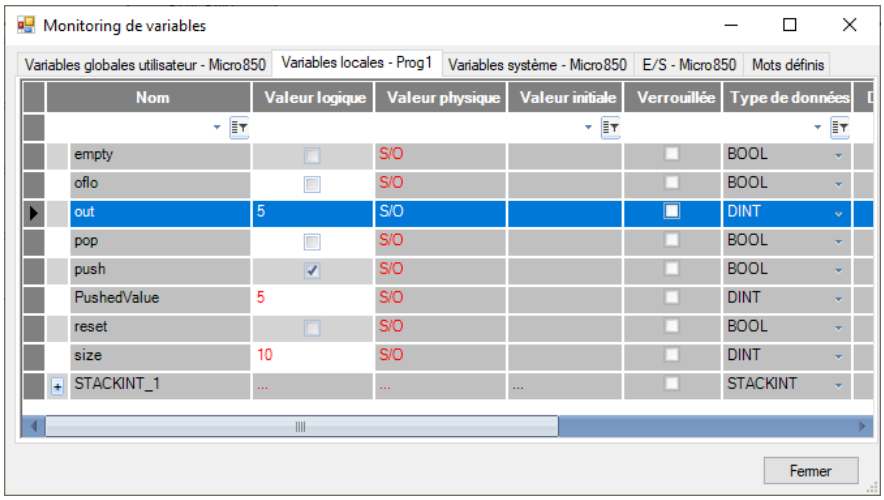
void **STACKINT_1**(BOOL **PUSH**, BOOL **POP**, BOOL **R1**, DINT **IN**, DINT **N**)
Type: STACKINT, Pile d'analogues entiers

```
1  PushedValue := 5;
2  size := 10;
3  STACKINT_1(push, pop, reset, PushedValue, size);
4  empty := STACKINT_1.EMPTY;
5  oflo := STACKINT_1.OFLO;
6  out := STACKINT_1.OUT;
```

(* Équivalence ST : STACKINT1 est une instance d'un bloc STACKINT *)

```
STACKINT1(err_detect, accusé réception, manual_mode,
err_code, max_err) ;
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY) ;
err_alarm := STACKINT1.OFLO ;
last_error := STACKINT1.OUT ;
```

Résultats

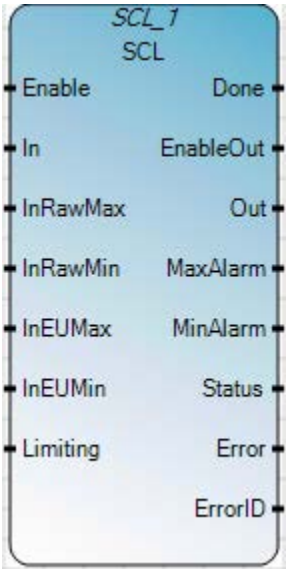


SCL

Convertit une valeur d'entrée REAL sans échelle en une valeur de virgule flottante REAL dans les unités Ingénieur et inclut les alarmes et limites de la sortie.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

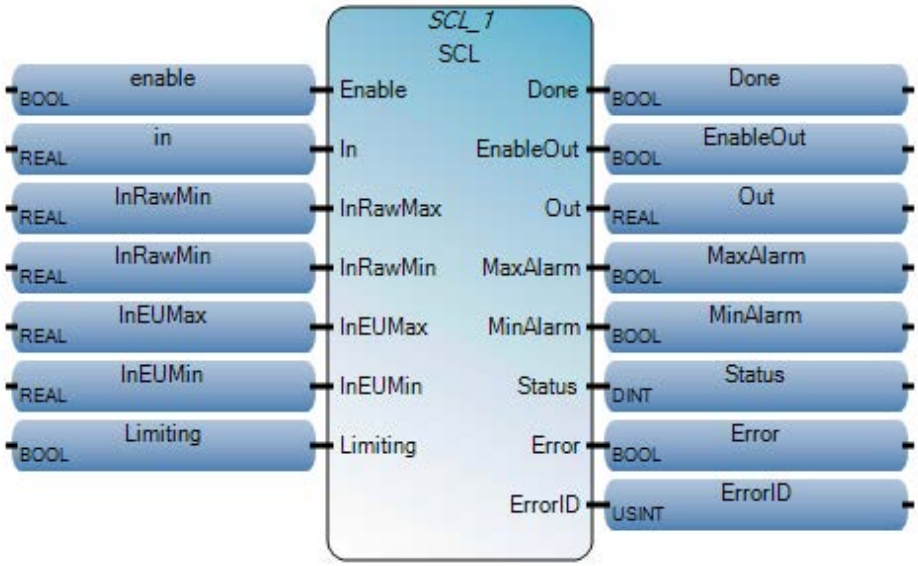
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	VRAI : front montant détecté. • Si InRawMin >= InRawMax, les bits Status.0 et Status.1 sont définis sur 1. Done est effacé, MaxAlarm et MinAlarm sont effacés. Error est défini sur VRAI et ErrorID est défini sur 1. • Sinon, Out est calculé d'abord. Ensuite, les conditions d'alarme sont vérifiées. Si MaxAlarm est défini, MinAlarm est effacé et vice versa. Ensuite, si Limiting est défini, Out sera compris dans la plage entre InEUMin et InEUMax. Ensuite, le bit Done est défini sur VRAI. Les bits Status sont définis sur 0. Error et ErrorID sont définis sur 0. • La valeur Out calculée est comparée avec NAN (Not a Number). Si NAN est la valeur Out, EnableOut est effacée. Done est effacé si la condition d'erreur est définie. FAUX : front montant non détecté. • L'instruction ne s'exécute pas. • Les sorties ne sont pas mises à jour, sauf Error, ErrorID, EnableOut et Done qui sont définis sur 0.
In	Entrée	REAL	L'entrée de signal analogique. Valide = n'importe quelle valeur flottante Par défaut = 0,0
InRawMax	Entrée	REAL	La valeur maximale atteignable par l'entrée dans l'instruction. Si InRawMax <= InRawMin, l'instruction définit le bit approprié dans Status, et Error ainsi que ErrorID sont mis à jour. La mise à jour des sorties s'arrête. Valide = InRawMax > InRawMin Par défaut = 0,0

Paramètre	Type de paramètre	Type de données	Description
InRawMin	Entrée	REAL	La valeur minimale atteignable par l'entrée dans l'instruction. Si InRawMin >= InRawMax, l'instruction définit le bit approprié dans Status, et Error ainsi que ErrorID sont mis à jour. La mise à jour des sorties s'arrête. Valide = InRawMin < InRawMax Par défaut = 0,0
InEUMax	Entrée	REAL	La valeur mise à l'échelle de l'entrée correspondant à InRawMax. Valide = valeur réelle, quelle qu'elle soit Par défaut = 0,0
InEUMin	Entrée	REAL	La valeur mise à l'échelle de l'entrée correspondant à InRawMin. Valide = valeur réelle, quelle qu'elle soit Par défaut = 0,0
Limiting	Entrée	BOOL	Sélecteur de limitation. VRAI : la sortie est limitée entre InEUMin et InEUMax.
Done	Sortie	BOOL	Indique quand l'opération est terminée. VRAI : l'opération est terminée avec succès. FAUX : l'opération a rencontré une erreur, ou Enable est défini sur FAUX.
EnableOut	Sortie	BOOL	Indique si l'instruction est activée. Définit sur Faux si Out déborde.
Out	Sortie	REAL	Représente la valeur mise à l'échelle de l'entrée analogique.
MaxAlarm	Sortie	BOOL	L'indicateur d'alarme d'entrée maximal. Cette valeur est définie sur VRAI lorsque Input > InRawMax.
MinAlarm	Sortie	BOOL	L'indicateur d'alarme d'entrée minimal. Cette valeur est définie sur VRAI lorsque Input < InRawMin.
Status	Sortie	DINT	État du bloc de fonctions. InstructFault (Status.0) Il sera défini lors de la détection d'erreurs d'exécution de l'instruction. Il ne s'agit pas de l'erreur de contrôleur mineure ou majeure. Vérifiez les bits d'état restants pour déterminer ce qui s'est produit. InRawRangeInv (Status.1) InRawMin >= InRawMax Les bits Status.3 à Status.31 sont réservés pour une utilisation future, et leur valeur est 0.
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : l'opération a rencontré une erreur. FAUX : l'opération s'est terminée avec succès ou l'instruction ne s'exécute pas.
ErrorID	Sortie	BOOL	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur.

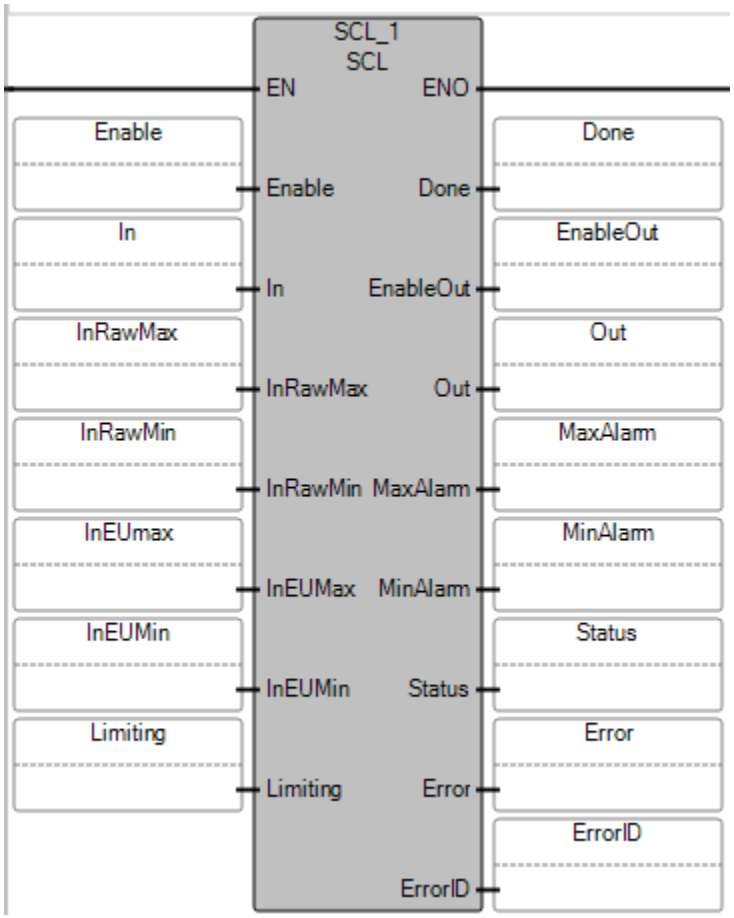
Code d'erreur

Code ErrorID	Description de l'erreur
1	InRawMax <= InRawMin

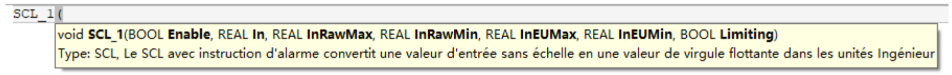
Exemple de Diagramme de Bloc Fonction SCL



Exemple de langage à relais SCL



Exemple de texte structuré SCL

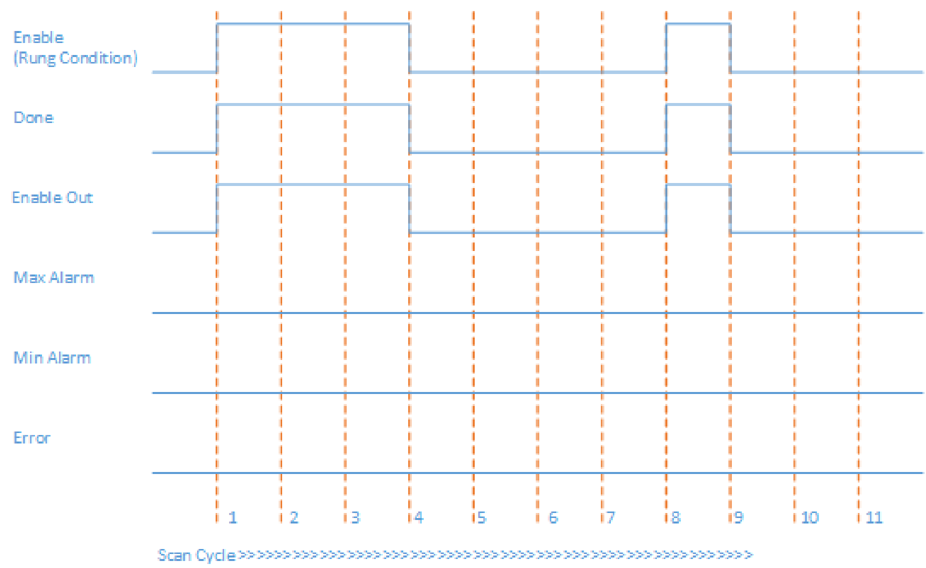



```
SCL_1(Enable, In, InRawMax, InRawMin, InEUMax, InEUMin, Limiting);
Done :=SCL_1.Done;
EnableOut :=SCL_1.EnableOut;
Out :=SCL_1.Out;
MaxAlarm :=SCL_1.MaxAlarm;
MinAlarm :=SCL_1.MinAlarm;
Status :=SCL_1.Status;
Error :=SCL_1.Error;
ErrorID :=SCL_1.ErrorID;
```

Exemples de diagrammes de temporisation d'instructions SCL

Les exemples de diagrammes de temporisation suivants décrivent les scénarios d'exécution pour l'instruction SCL sur la page 512.

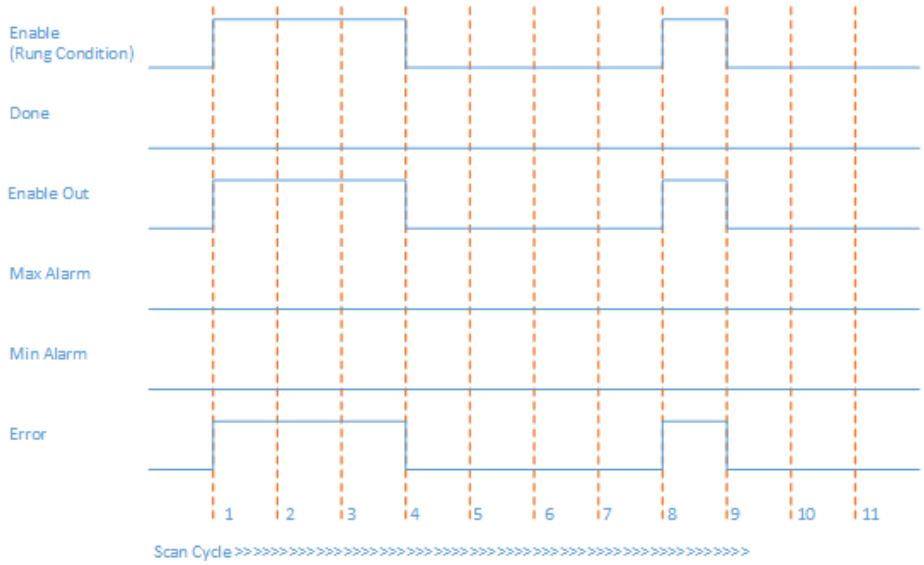
Exécution réussie de SCL



Cycle de scrutation	Description
1	Lorsque Enable est défini sur VRAI et que les paramètres d'entrée sont valides et dans la plage, l'exécution du bloc de fonctions démarre. • Done et EnableOut sont définis sur VRAI. • Out est calculé selon les entrées données. • MaxAlarm, MinAlarm et Error sont définis sur FAUX. • Status et ErrorID sont définis sur 0, car aucune erreur n'est générée.
2, 3	Aucun changement de la condition de ligne.
4	Lorsque Enable est défini sur FAUX, l'exécution du bloc de fonctions s'arrête. • Done, EnableOut et Error sont définis sur FAUX. • ErrorID est défini sur 0. • Out, MaxAlarm, MinAlarm et Status conservent leur dernière valeur.
5, 6, 7	Aucun changement de la condition de ligne.

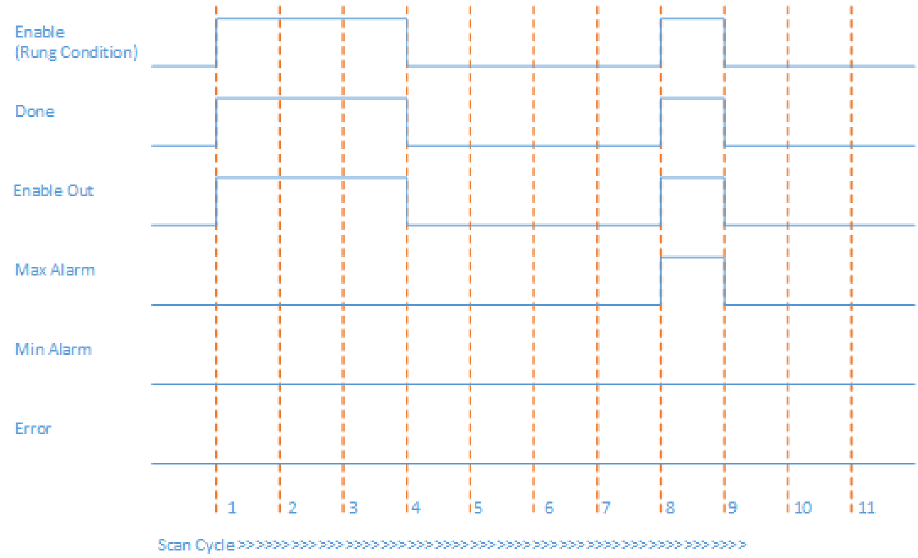
Cycle de scrutation	Description
8	• Lorsque Enable est défini sur VRAI et que les paramètres d'entrée sont valides et dans la plage, l'exécution du bloc de fonctions démarre. • Done et EnableOut sont définis sur VRAI. • Out est calculé selon les entrées données. • MaxAlarm, MinAlarm et Error sont définis sur FAUX. • ErrorID et Status sont définis sur 0, car aucune erreur n'est générée.
9	Lorsque Enable est défini sur FAUX, l'exécution du bloc de fonctions s'arrête. • Done, EnableOut et Error sont définis sur FAUX. • ErrorID est défini sur 0. • Out, MaxAlarm, MinAlarm et Status conservent leur dernière valeur.
10, 11	Aucun changement de la condition de ligne.

Exécution échouée de SCL



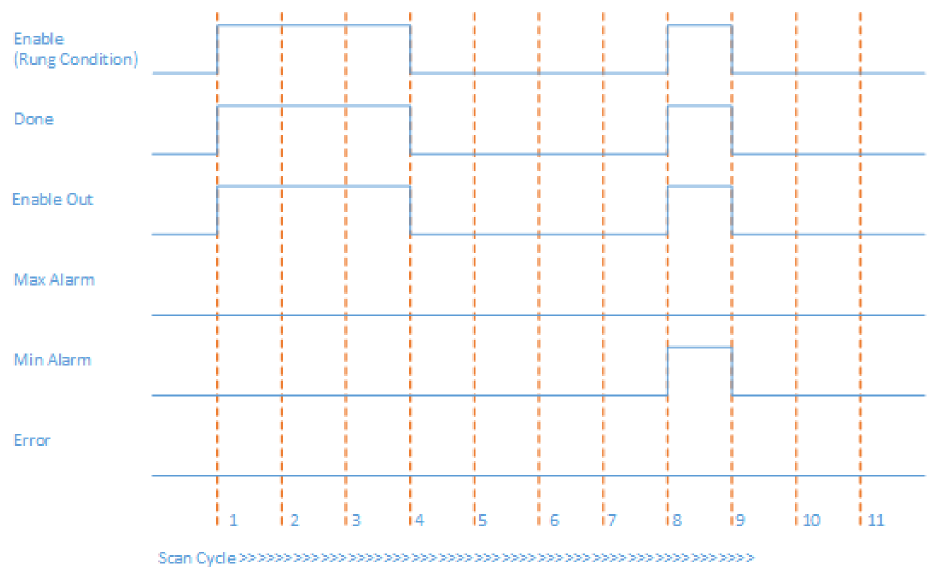
Dans cet exemple, tous les paramètres d'entrée sont valides et dans la plage, mais InRawMin >= InRawMax. Dans les cycles de scrutation 1 et 8, lorsque Enable est défini sur VRAI et que l'exécution du bloc de fonctions démarre, Error est défini sur VRAI et ErrorID, sur 1.

Génération de MaxAlarm



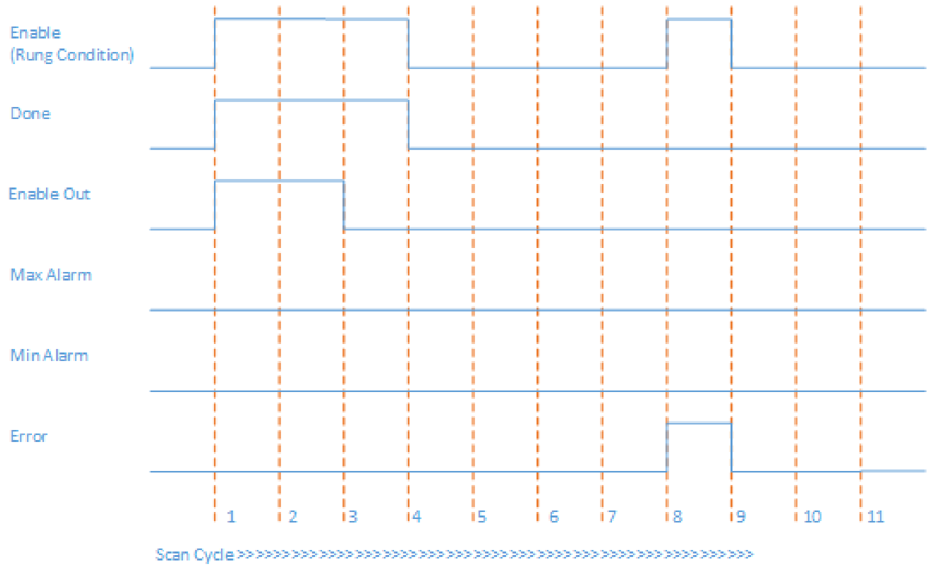
Dans le cycle de scrutation 8 de cet exemple, tous les paramètres d'entrée sont valides et dans la plage, mais In > InRawMax. Lorsque Enable est défini sur VRAI et que l'exécution du bloc de fonctions démarre, MaxAlarm est défini sur VRAI.

Génération de MinAlarm



Dans le cycle de scrutation 8 de cet exemple, tous les paramètres d'entrée sont valides et dans la plage, mais In < InRawMin. Lorsque Enable est défini sur VRAI et que l'exécution du bloc de fonctions démarre, MinAlarm est défini sur VRAI.

Condition de débordement de sortie et erreur de configuration de l'entrée



Dans le cycle de scrutation 3 de cet exemple, les paramètres d'entrée du bloc de fonctions sont valides et dans la plage, mais Out est débordé en raison des paramètres d'entrée. EnableOut est défini sur FAUX. La valeur Out est non valide.

Dans le cycle de scrutation 8 de cet exemple, les paramètres d'entrée du bloc de fonctions sont valides et dans la plage, mais Out est débordé en raison des paramètres d'entrée et InRawMin >= InRawMax ; Error est défini sur VRAI. ErrorID est défini sur 1 et Status, sur 3.

TND (arrêt du programme en cours)

Arrête le cycle actuel de la scrutation du programme utilisateur. Le programme utilisateur est réexécuté depuis le début de la première routine après la scrutation de la sortie et de l'entrée et les opérations de maintenance.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

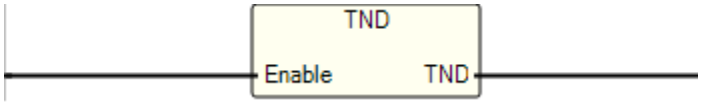
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation de la fonction Lorsque Enable = VRAI, exécution de la fonction. Lorsque Enable = FAUX, la fonction n'est pas exécutée.

Paramètre	Type de paramètre	Type de données	Description
TND	Sortie	BOOL	Si VRAI, la fonction est exécutée. Lorsque la surveillance des variables est activée, la valeur des variables de surveillance est attribuée à la sortie de l'instruction. Lorsque la surveillance des variables est désactivée, la valeur des variables de sortie est attribuée à la sortie de l'instruction.

Exemple de Diagramme de Bloc Fonction TND

Exemple de logique à relais TND



Exemple de texte structuré TND

```
TND(
```

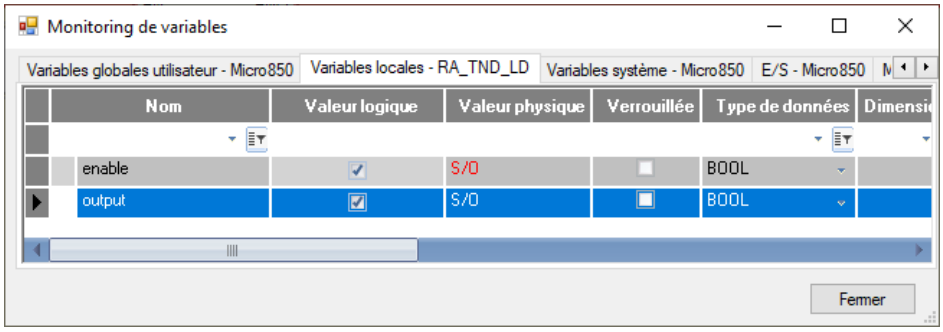
BOOL **TND**(BOOL Enable)
Abandonne le balayage du programme utilisateur courant.

```
1  enable := TRUE;
2  output := TND(enable);
```

(* Équivalence ST : *)

TESTOUTPUT := TND(TESTENABLE) ;

Résultats



LIMITE (test limite)

Restreint des valeurs entières à un intervalle donné. Aucun changement n'est apporté aux entiers situés entre les limites minimale et maximale. Les entiers

supérieurs à la limite maximale sont remplacés par la valeur maximale. Les entiers inférieurs à la limite minimale sont remplacés par la valeur minimale.

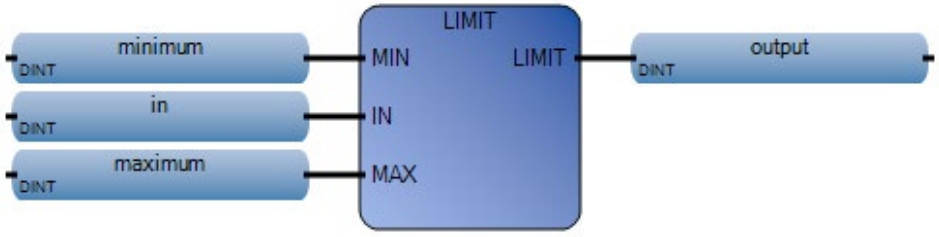
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

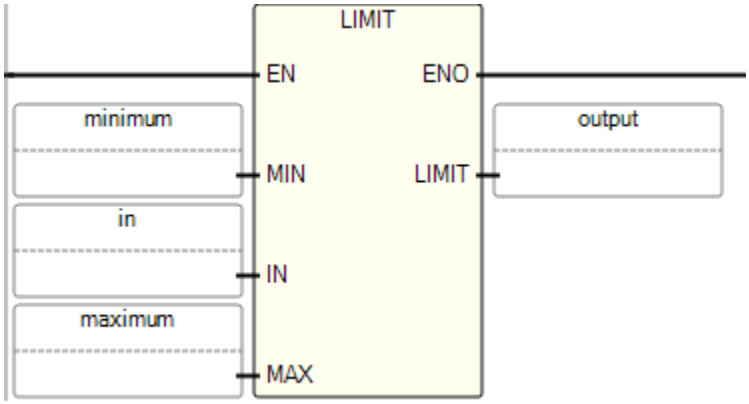
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de la fonction VRAI : exécution du calcul LIMIT actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
MIN	Entrée	DINT	Valeur minimale prise en charge.
IN	Entrée	DINT	Valeur entière signée, quelle qu'elle soit.
MAX	Entrée	DINT	Valeur maximale prise en charge.
LIMIT	Sortie	DINT	Valeur d'entrée limitée à la plage prise en charge.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

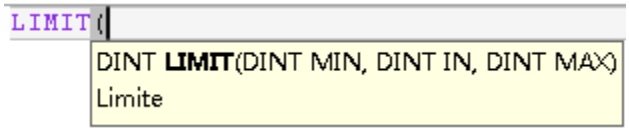
Exemple de Diagramme de Bloc Fonction LIMIT



Exemple de logique à relais LIMIT



Exemple de texte structuré LIMIT



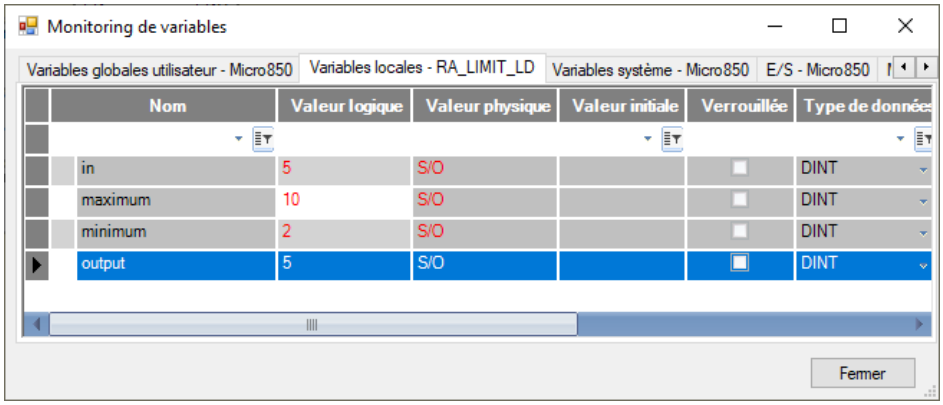

```
1  minimum := 2;
2  in := 5;
3  maximum := 10;
4  output := LIMIT(minimum, in, maximum);
```

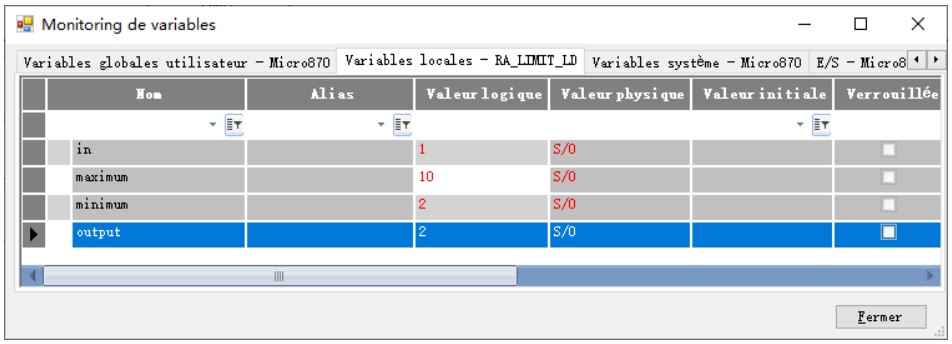
(* Équivalence ST : *)

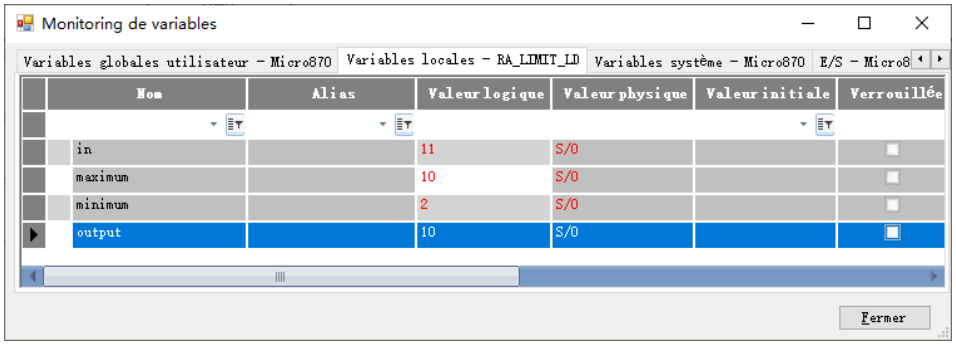
new_value := LIMIT (min_value, valeur, max_value) ;

(* limite la valeur à l'ensemble [min_value..max_value] *)

Résultats


Monitoring de variables
Variables globales utilisateur - Micro850
Variables locales - RA_LIMIT_LD
Variables système - Micro850
E/S - Micro850
Nom
Valeur logique
Valeur physique
Valeur initiale
Verrouillée
Type de donnée
in 5 S/O DINT
maximum 10 S/O DINT
minimum 2 S/O DINT
output 5 S/O DINT
Fermer


Monitoring de variables
Variables globales utilisateur - Micro870
Variables locales - RA_LIMIT_LD
Variables système - Micro870
E/S - Micro8
Nom
Alias
Valeur logique
Valeur physique
Valeur initiale
Verrouillée
in 1 S/O
maximum 10 S/O
minimum 2 S/O
output 2 S/O
Fermer


Monitoring de variables
Variables globales utilisateur - Micro870
Variables locales - RA_LIMIT_LD
Variables système - Micro870
E/S - Micro8
Nom
Alias
Valeur logique
Valeur physique
Valeur initiale
Verrouillée
in 11 S/O
maximum 10 S/O
minimum 2 S/O
output 10 S/O
Fermer

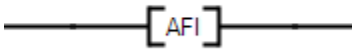
Instructions de contrôle de programme

Utilisez l'instruction de contrôle de programme pour contrôler simultanément les instructions d'un programme utilisateur et d'un appareil d'interface d'opérateur.

Instruction	Description
AFI sur la page 523	Désactive une ligne de programme.
NOP sur la page 523	Fonctionne comme un paramètre fictif.
SUS sur la page 523	Suspend l'exécution du contrôleur <800>.

AFI (toujours faux)

Utilisez l'instruction AFI au début d'une ligne de programme pour désactiver temporairement cette ligne lors du débogage sans devoir la supprimer du programme. La sortie de cette instruction est toujours définie sur FAUX.

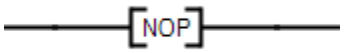


Langage pris en charge : Logique à relais.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

NOP (aucune opération)

L'instruction NOP fonctionne comme un paramètre fictif. Vous pouvez la placer n'importe où dans une ligne de programme.



Langage pris en charge : Logique à relais.

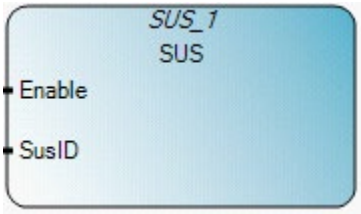
Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

SUS (attente)

Suspend l'exécution du contrôleur <800>. Le contrôleur reste en mode EXECUTER mais son exécution est suspendue indéfiniment. La mise en suspens permet de récupérer les erreurs propres au programme utilisateur et de simplifier la surveillance de ce dernier. Placez l'instruction SUS dans les sections du programme utilisateur dans lesquelles vous souhaitez repérer des conditions inhabituelles. En mode suspension, la commande RUN LED est définie sur OFF pour indiquer que la scrutation du programme est inactive.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

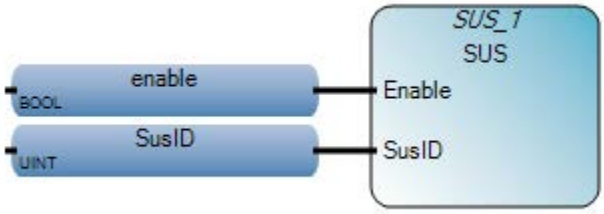
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



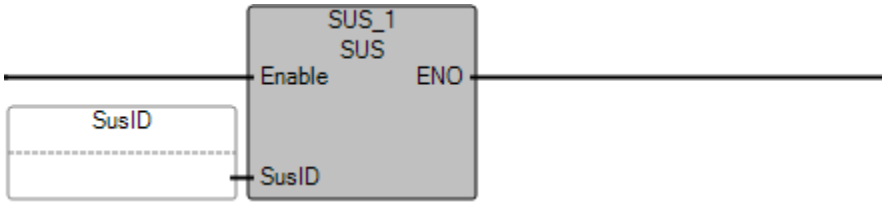
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécute la fonction. FAUX : n'exécute pas la fonction.
SusID	Entrée	UINT	ID de suspension
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Exemple de diagramme de bloc de fonctions SUS



Exemple de langage à relais SUS



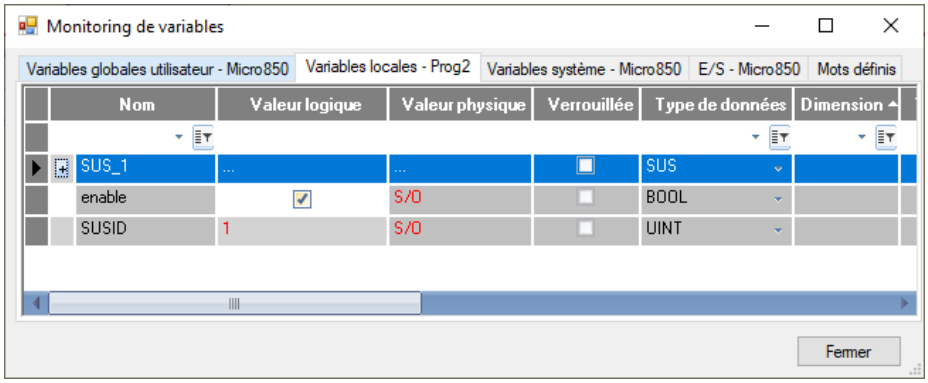
Exemple de texte structuré SUS

```
SUS_1(
```

void **SUS_1**(BOOL **Enable**, UINT **SusID**)
Type: SUS, Suspension de l'exécution de l'application.

```
1  SusID := 1;
2  SUS_1(enable, SusID);
```

Résultats



Instructions de dérivé intégral proportionnel (PID)

Utilisez les instructions de dérivé intégral proportionnel (PID) pour contrôler avec plus de précision le traitement, grâce à la fonctionnalité PID.

Instruction	Description
IPIDCONTROLLER sur la page 527	Configure et contrôle les entrées et sorties utilisées pour la logique de dérivé intégral proportionnel (PID).
PID sur la page 552	Configure et contrôle les sorties qui contrôlent les propriétés physiques, telles que la température, la pression, le niveau de liquide ou le débit utilisant les boucles de processus.

Qu'est-ce que le contrôle de dérivé intégral proportionnel (PID) ?

Le contrôle de dérivé intégral proportionnel (PID) permet au contrôle de traitement de maintenir avec précision le point de consigne, en ajustant les sorties de contrôle. Un bloc fonctionnel PID combine tous les éléments logiques nécessaires pour effectuer un contrôle PID (proportionnel/intégral/dérivé).

IPIDCONTROLLER (contrôleur de dérivé intégral proportionnel)

Configure et contrôle les entrées et sorties utilisées pour la logique de dérivé intégral proportionnel (PID). La logique PID est utilisée pour contrôler les propriétés physiques, telles que la température, la pression, le niveau, le liquide ou le débit, en utilisant les boucles de processus qui permettent de calculer une valeur d'erreur sous forme de différence entre un point de consigne souhaité et une variable de processus mesurée. Le contrôleur tente de réduire l'erreur au fil du temps en ajustant une variable de contrôle. Le calcul inclut les termes proportionnel (P), intégral (I) et dérivé (D), utilisés comme suit :

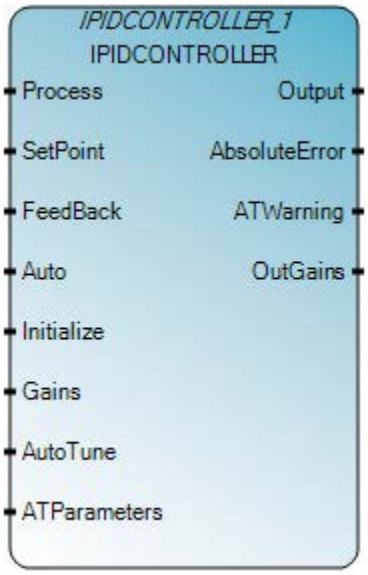
- P : valeurs actuelles de l'erreur.

- I : valeurs passées de l'erreur.

- D : possibles valeurs futures de l'erreur (en fonction de son taux de changement actuel) qui contrôle les propriétés physiques, telles que la température, la pression, le niveau de liquide ou le débit, en utilisant les boucles de processus.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

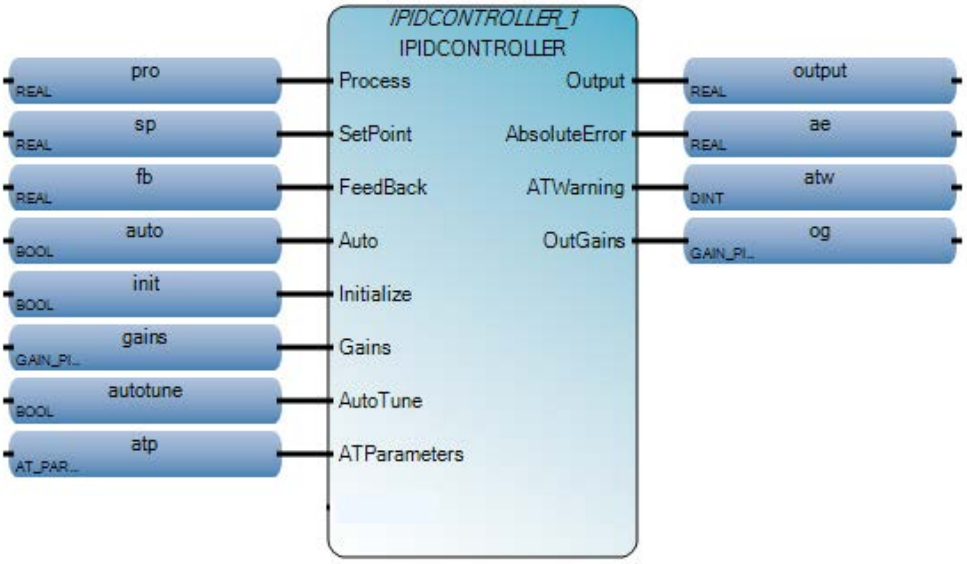


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

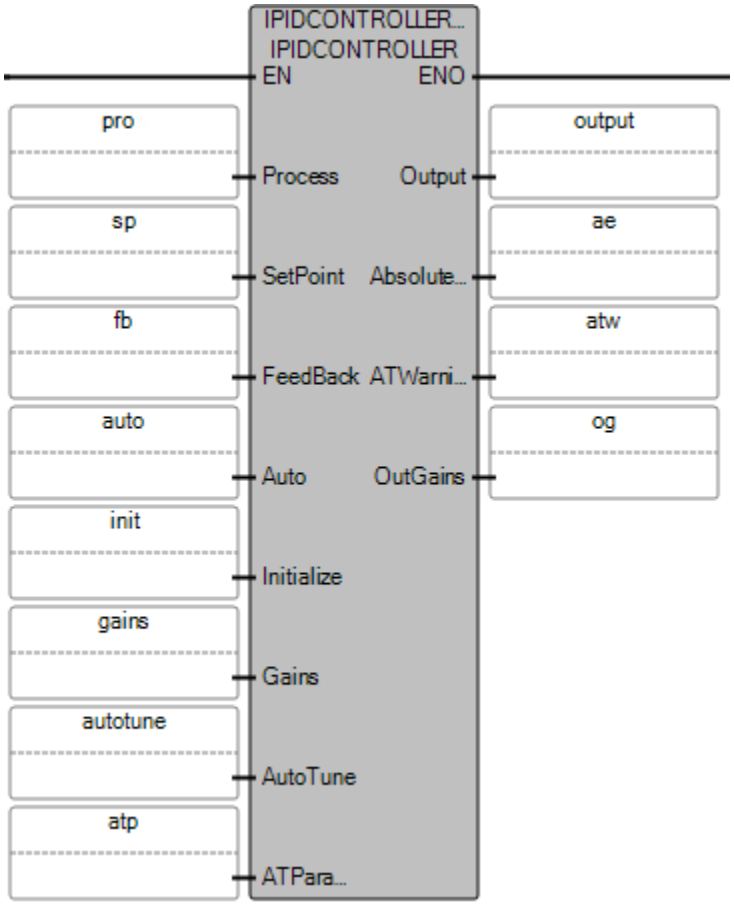
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Lorsque ce paramètre est VRAI, le bloc d'instruction est activé. VRAI : exécution du calcul PID. FAUX : le bloc d'instruction est en attente. S'applique aux programmes de schémas à relais.
Process	Entrée	REAL	Valeur de traitement, c'est-à-dire la valeur mesurée à partir de la sortie du processus.
SetPoint	Entrée	REAL	Point de consigne.
FeedBack	Entrée	REAL	Signal de retour, c'est-à-dire la valeur de la variable de contrôle appliquée au processus. Par exemple, le retour peut correspondre à la sortie IPIDCONTROLLER.
Auto	Entrée	BOOL	Mode de fonctionnement du contrôleur PID : • TRUE - le contrôleur fonctionne en mode normal. • FALSE : le contrôleur déclenche la réinitialisation R pour assurer le suivi de (F-GE).
Initialize	Entrée	BOOL	Changement apporté à la valeur (de TRUE à FAUX ou de FAUX à TRUE) entraînant la suppression des gains proportionnels au cours du cycle. Initialise également les séquences de réglage automatique.
Gains	Entrée	GAIN_PID	Valeur PID des gains pour IPIDController Utilisez le type de données GAIN_PID sur la page 531 pour définir les paramètres de l'entrée Gains.
AutoTune	Entrée	BOOL	VRAI : lorsque le paramètre de réglage automatique est défini sur VRAI et que les paramètres Auto et Initialize sont définis sur FAUX, la séquence de réglage automatique démarre. FAUX : le réglage automatique ne démarre pas.
ATParameters	Entrée	AT_Param	Paramètres de réglage automatique Utilisez le type de données AT_Param sur la page 532 pour définir les paramètres de l'entrée ATParameters.
Output	Sortie	REAL	Valeur de sortie issue du contrôleur.
AbsoluteError	Sortie	REAL	Erreur absolue (processus – point de consigne) issue du contrôleur.

Paramètre	Type de paramètre	Type de données	Description
ATWarnings	Sortie	DINT	(ATWarning) Avertissement pour la séquence de réglage automatique. Les valeurs possibles sont les suivantes : • 0 - pas de réglage automatique. • 1 : en mode Réglage automatique. • 2 : réglage automatique terminé. • -1 : entrée ERREUR 1 automatiquement définie sur TRUE, aucun réglage automatique possible. • -2 : erreur de réglage automatique ERREUR 2, expiration de ATDynaSet.
OutGains	Sortie	GAIN_PID	Gains calculés suite aux séquences de réglage automatique Utilisez le type de données GAIN_PID pour définir la sortie OutGains.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

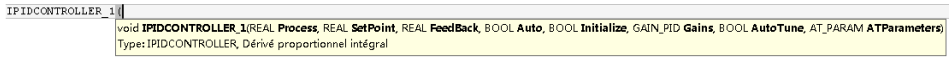
Exemple de Diagramme de Bloc Fonction IPIDCONTROLLER



Exemple de logique à relais IPIDCONTROLLER



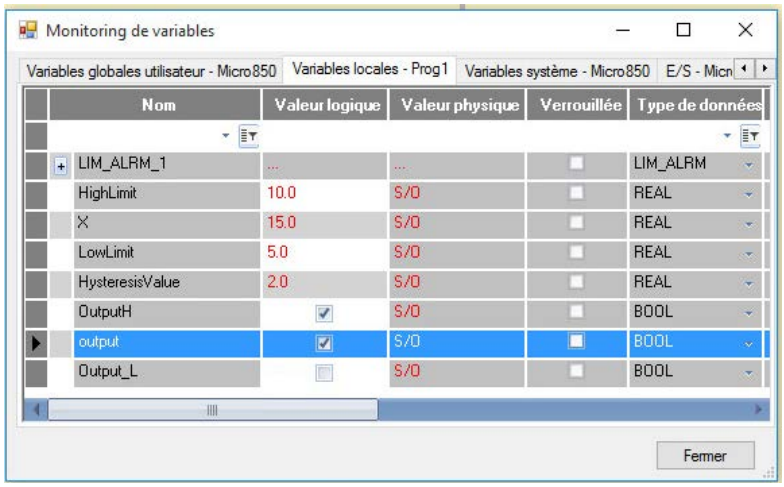
Exemple de texte structuré IPIDCONTROLLER




```
1 IPIDCONTROLLER_1(pro, sp, fb, auto, init, gains, autotune, atp, em);
2 output := IPIDCONTROLLER_1.Output;
3 ae := IPIDCONTROLLER_1.AbsoluteError;
4 atw := IPIDCONTROLLER_1.ATWarning;
5 og := IPIDCONTROLLER_1.OutGains;
```

```
(* Équivalence ST : IPIDController1 est une instance du
bloc IPIDController *)

IPIDController1(Proc,
SP,
 FBK,
 Auto,
 Init,
 G_In,
 A_Tune,
 A_TunePar,
 Err );
Out_process := IPIDController1.Output ;
A_Tune_Warn := IPIDController1.ATWarning ;
Gain_Out := IPIDController1.OutGains ;
```

Résultats



Type de données GAIN_PID

Le tableau suivant décrit le type de données GAIN_PID de l'instruction IPIDCONTROLLER sur la page 527.

Paramètre	Type de données	Description
DirectActing	BOOL	Type d'action : • VRAI **:** action directe (la sortie va dans la même direction que l'erreur). Autrement dit, la valeur de traitement réelle est supérieure au point de consigne et la mesure de contrôle adéquate consiste à augmenter la sortie (par exemple : refroidir). • FAUX **:** action inverse (la sortie va dans la direction opposée à l'erreur). Autrement dit, la valeur de traitement réelle est supérieure au point de consigne et la mesure de contrôle adéquate consiste à diminuer la sortie (par exemple : réchauffer).
ProportionalGain	REAL	Gain proportionnel pour PID (>= 0,0001). Lorsque ProportionalGain est (< 0,0001), alors ProportionalGain = 0,0001 **Gain proportionnel pour PID (P_Gain)** Un gain proportionnel plus élevé provoque un plus grand changement dans la sortie en fonction de la différence entre la valeur de traitement mesurée (PV) et la valeur du point de consigne (SV). Plus le gain est élevé, plus l'erreur est réduite rapidement, mais cela peut entraîner de l'instabilité (notamment, des oscillations). Plus le gain est bas, plus l'erreur est réduite lentement, mais le système est plus stable et moins sensible aux grandes erreurs. P_Gain est souvent le gain le plus important à définir et le premier gain à définir pendant le réglage.
TimeIntegral	REAL	Valeur temporelle intégrale pour PID en secondes (>= 0,0001). Lorsque TimeIntegral est (< 0,0001), alors TimeIntegral = 0,0001 **Valeur temporelle intégrale pour PID** Une constante temporelle intégrale plus petite provoque un plus grand changement dans la sortie en fonction de la différence entre la valeur de traitement mesurée (PV) et la valeur du point de consigne (SV) intégrée au fil du temps. Une constante temporelle intégrale plus petite réduit l'erreur d'état stable (erreur survenant lorsque la SV n'est pas modifiée), mais elle réduit les probabilités d'instabilité (notamment, les oscillations). Une constante temporelle intégrale plus grande réduit le temps de réponse du système et le stabilise, mais la PV se rapproche plus lentement de la SV.
TimeDerivative	REAL	Valeur temporelle dérivative pour PID en secondes (>= 0,0). Lorsque TimeDerivative est (< = 0,0), alors TimeDerivative = 0,0 Lorsque TimeDerivative = 0, IPID agit en tant que PI. **Valeur temporelle dérivative pour PID (Td)** Une constante temporelle dérivative plus petite entraîne un changement plus rapide dans la sortie en fonction du taux de changement de la différence entre la valeur de traitement mesurée (PV) et la valeur du point de consigne (SV). Une constante temporelle dérivative plus petite améliore la réactivité du système face aux changements soudains dans l'erreur (la SV est modifiée), mais augmente les risques d'instabilité (notamment, les oscillations). Une constante temporelle plus grande réduit la réactivité du système face aux changements soudains dans l'erreur, et le système est moins sujet au bruit et aux changements d'étape dans la PV. La valeur TimeDerivative (Td) est liée au gain dérivatif, mais

Paramètre	Type de données	Description
		il permet de régler la contribution dérivative au PID avec le temps pour que l'échantillon temporel soit pris en compte.
DerivativeGain	REAL	Gain dérivé pour PID (> 0,0). Lorsque DerivativeGain est (< 0,0), alors DerivativeGain = 0,1 **Gain dérivé pour PID (D_Gain)** Un gain dérivatif plus élevé entraîne un changement plus important dans la sortie en fonction du taux de changement de la différence entre la valeur de traitement mesurée (PV) et la valeur du point de consigne (SV). Un gain plus élevé améliore la réactivité du système face aux changements soudains dans l'erreur (la SV est modifiée), mais augmente les risques d'instabilité (notamment, les oscillations). Un gain plus petit réduit la réactivité du système face aux changements soudains dans l'erreur, et le système est moins sujet au bruit et aux changements d'étape dans la PV.

Type de données AT_Param

Le tableau suivant décrit les paramètres du type de données AT_Param.

Paramètre	Type de données	Description
Load	REAL	Chargement du paramètre pour le réglage automatique. Il s'agit de la valeur de sortie lors du démarrage du réglage automatique.
Deviation	REAL	Écart pour le réglage automatique. Il s'agit de l'écart type utilisé pour évaluer la bande de bruit nécessaire au réglage automatique.
Step	REAL	Valeur d'étape pour le réglage automatique. Doit être supérieure à la bande de bruit et inférieure à la moitié du chargement.
ATDynamSet	REAL	Temps d'attente en secondes avant l'abandon du réglage automatique.
ATReset	BOOL	Indique si la valeur de sortie est réinitialisée sur zéro après une séquence de réglage automatique : • TRUE : réinitialise la sortie sur zéro. • FAUX : conserve la valeur du chargement pour la sortie.

Mise en œuvre du contrôle PID par le bloc fonctionnel IPIDController

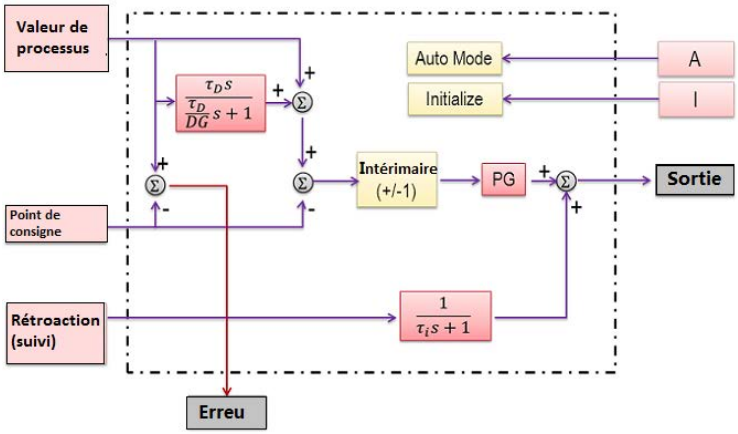
Le bloc de fonctions IPIDController, disponible dans le jeu d'instructions Connected Components Workbench, est basé sur la théorie de contrôle PID et combine tous les éléments logiques nécessaires pour effectuer un traitement de canal d'entrée analogique et un contrôle de dérivé intégral proportionnel (PID). Dans l'interface homme-machine, la façade avant IPID peut être utilisée avec le bloc de fonctions IPIDController.

Description du bloc de fonctions IPIDController

Le bloc de fonctions IPIDController utilise les composants de bloc de fonctions suivants :

- R : action (+/- 1)
- PG : gain proportionnel
- DG : gain de filtre dérivé
- td : ãD

- ti : ãI



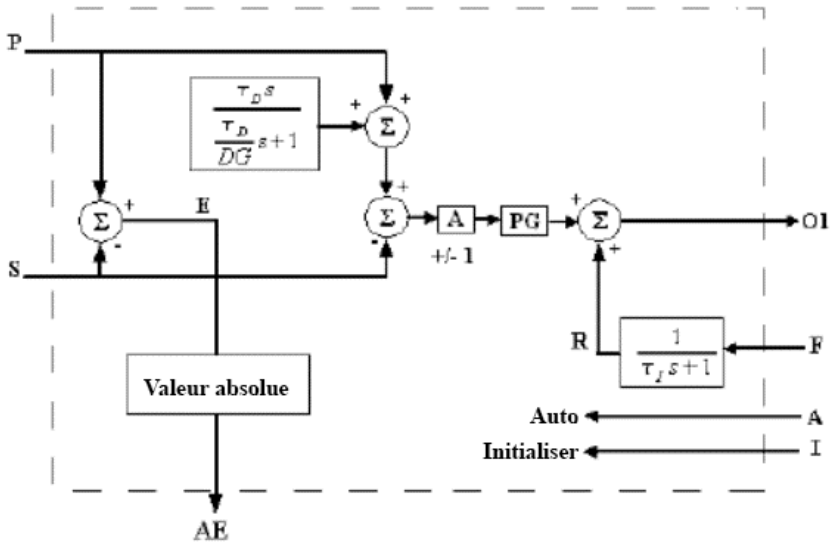
Prévention d'enroulement intégral

Si la différence entre la valeur de point de consigne et la valeur de traitement est importante, la valeur de sortie augmente de manière significative, et le traitement ne contrôle plus rien pendant le temps nécessaire pour que cette valeur diminue. Le bloc de fonctions IPIDController assure un suivi interactif du retour codeur et empêche un enroulement intégral. Lorsque la sortie est saturée, le terme intégral du contrôleur est recalculé afin que sa nouvelle valeur offre une sortie dans les limites de saturation autorisées.

Opération du bloc fonctionnel IPIDController

Lorsque la **Valeur d'entrée automatique** est définie sur **VRAI**, IPIDController est exécuté en mode automatique normal.

Lorsque la **Valeur d'entrée automatique** est définie sur **FAUX**, la réinitialisation R est déclenchée pour assurer le suivi de (F-GE). Cette configuration oblige la sortie IPIDController à suivre le retour codeur dans les limites IPIDController et le contrôleur revient en mode automatique sans incrémenter la sortie.



Dans le cas de l'initialisation de l'entrée, si vous passez de FALSE à TRUE ou de TRUE à FALSE lorsque le réglage automatique est défini sur FALSE, IPIDController supprime toutes les actions de gain proportionnel lors de ce cycle (par exemple, en utilisant le paramètre Initialize). Utilisez ce processus pour empêcher de secouer la sortie lorsque des changements sont apportés au point de consigne à l'aide d'un bloc de fonctions de commutateur d'activation.

Pour exécuter une séquence de réglage automatique :

Pour exécuter une séquence de réglage automatique, la valeur d'entrée ATParameters doit être précisée. Les paramètres de gain d'entrée et d'action directe doivent être définis en fonction du traitement et de l'ensemble DerivativeGain (en général 0,1). Démarrez la séquence de réglage automatique de la façon suivante :

1. Définissez la valeur d'entrée Initialize sur TRUE.
2. Définissez la valeur d'entrée Autotune sur TRUE.
3. Changez la valeur d'entrée Initialize en FALSE.
4. Attendez que la valeur de sortie ATWarning passe à 2.
5. Transférez les valeurs de sortie OutGains vers le paramètre d'entrée Gains.

Pour finaliser le réglage, il peut être nécessaire de procéder à des ajustements en fonction des processus et des besoins. Lorsque vous définissez la valeur TimeDerivative sur 0,0, IPIDController impose la valeur 1,0 au paramètre DerivativeGain, puis fonctionne en tant que contrôleur PI.

Utilisation de l'instruction de dérivé intégral proportionnel

Cette section fournit des détails précis et des exemples sur l'utilisation de l'instruction de dérivé intégral proportionnel, notamment :

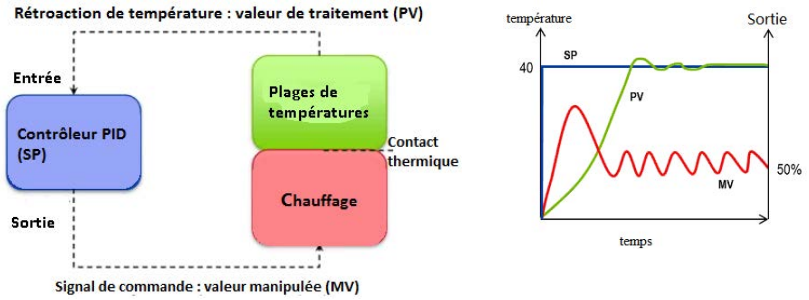
Exemple : Création d'une boucle de rétroaction pour la valeur manipulée

L'ajout d'une boucle de rétroaction pour la valeur manipulée empêche un dépassement excessif, en fournissant des valeurs minimale et maximale pour MV.

Exemple de boucle de rétroaction de température

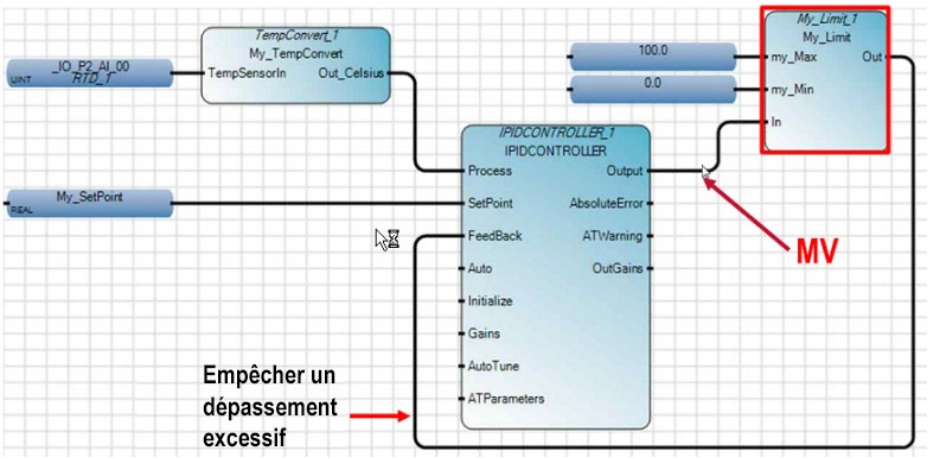
Au début du traitement de contrôle de la température, la différence entre la valeur de traitement (PV) et la valeur du point de consigne (SP) est importante, comme illustré dans le graphique suivant. Dans cet exemple de boucle de rétroaction de température, la valeur PV commence à 0 degré Celsius et continue à monter vers la valeur SP de 40 °C. Notez également que la fluctuation entre la valeur manipulée (MV) haute et basse se stabilise avec le

temps. Le comportement de MV dépend des valeurs utilisées dans chaque paramètre : P, I et D.



IPIDController avec boucle de rétroaction

Le diagramme de bloc fonctionnel suivant comprend une boucle de rétroaction pour la valeur manipulée, qui empêche un dépassement excessif en fournissant des valeurs minimale et maximale pour MV.



Exemple : Mise en œuvre du réglage automatique dans un bloc fonctionnel IPIDController

Vous pouvez utiliser le paramètre AutoTune du bloc de fonctions IPIDController pour mettre en œuvre la fonction de réglage automatique dans le programme de contrôle.

Exigences et recommandations en matière de réglage automatique

Vous trouverez ci-après une synthèse des exigences et des recommandations pour la mise en œuvre réussie du réglage automatique.

- Le réglage auto doit entraîner l'oscillation de la boucle de contrôle de la sortie, ce qui signifie qu'il est nécessaire d'appeler IPIDController

suffisamment souvent pour échantillonner de manière adéquate l'oscillation.

- Le bloc d'instructions IPIDController doit ainsi être exécuté à un intervalle assez constant.
- Configurez le temps de scrutation du programme afin qu'il soit équivalent à la moitié du temps d'oscillation.
- Vous pouvez envisager l'utilisation d'une instruction STI (interruption de texte structuré) pour contrôler le bloc d'instructions IPIDController.

Exemple : Ajout d'un UDFB à un programme PID

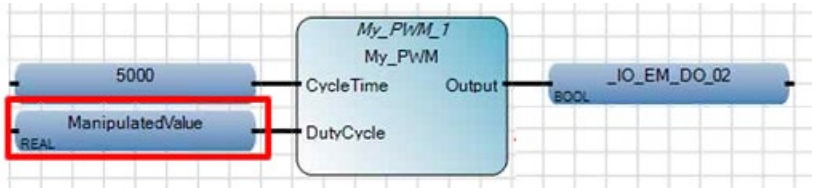
Vous pouvez ajouter des UDFB en dehors du programme principal, afin d'exécuter des fonctions spécialisées, telles que la conversion d'unités ou le transfert de valeurs.

Transfert de la valeur de gain du réglage automatique

Ce bloc UDFB transfère la valeur de gain Autotune à My_GainTransfer, afin que le contrôleur puisse l'utiliser.

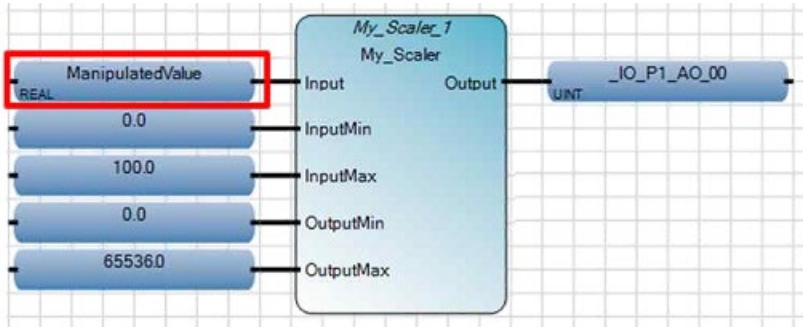
Conversion d'une valeur manipulée en sortie numérique

Cet UDFB convertit une valeur manipulée (MV) en une sortie numérique (DO), afin de pouvoir l'utiliser pour contrôler une entrée numérique (DI).



Conversion d'une valeur manipulée en sortie analogique

Cet UDFB convertit une valeur manipulée (MV) en une sortie analogique (AO), afin de pouvoir l'utiliser pour contrôler une entrée analogique (AI).



Utilisation du réglage automatique avec le bloc de fonctions IPIDController

Vous pouvez utiliser le paramètre AutoTune du bloc de fonctions IPIDController pour mettre en œuvre la fonction de réglage automatique dans le programme de contrôle.

Exigences et recommandations en matière de réglage automatique

Vous trouverez ci-après une synthèse des exigences et des recommandations pour la mise en œuvre réussie du réglage automatique.

- Le réglage auto doit entraîner l'oscillation de la boucle de contrôle de la sortie, ce qui signifie qu'il est nécessaire d'appeler IPIDController suffisamment souvent pour échantillonner de manière adéquate l'oscillation.
- Le bloc d'instructions IPIDController doit ainsi être exécuté à un intervalle assez constant.
- Configurez le temps de scrutation du programme afin qu'il soit équivalent à la moitié du temps d'oscillation.
- Vous pouvez envisager l'utilisation d'un bloc d'instruction programmable en Interruption temporisée (STI) pour contrôler le bloc d'instruction IPIDController.

Réglage automatique de systèmes primaires et secondaires

Utilisez la fonction de réglage automatique sur un système primaire, qui utilise un élément simple, ou sur un système secondaire, qui utilise deux éléments indépendants.

Un système primaire utilise un seul élément de stockage d'énergie indépendant. En voici quelques exemples :

- Refroidissement d'un réservoir de liquide, avec énergie de chauffage comme unité de stockage.
- Débit de liquide depuis un réservoir, avec énergie potentielle comme unité de stockage.
- Moteur à couple constant entraînant un volant à disque, avec énergie cinétique de rotation comme unité de stockage.

- Réseau d'avance électrique RC, avec énergie de stockage capacitif comme unité de stockage.

Dans un système primaire, la fonction peut être écrite dans un format standard, par exemple $f(t) = \tau\ dy/dt + y(t)$

Où :

Variable	Description	Exemple : Refroidissement d'un réservoir de liquide, avec énergie de chauffage comme élément de stockage.
t	Constante temporelle du système	Égal à RC Où R = Résistance thermique des parois du réservoir C = Capacité thermique du liquide
f	Fonction de forçage	Égal à la température ambiante
y	Variable d'état du système	Égal à la température du liquide

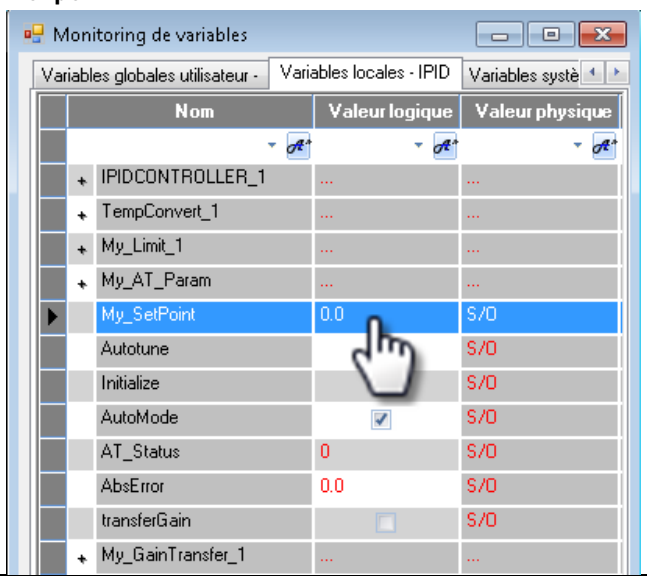
Un système secondaire utilise deux éléments de stockage d'énergie indépendants, qui s'échangent l'énergie stockée. En voici quelques exemples :

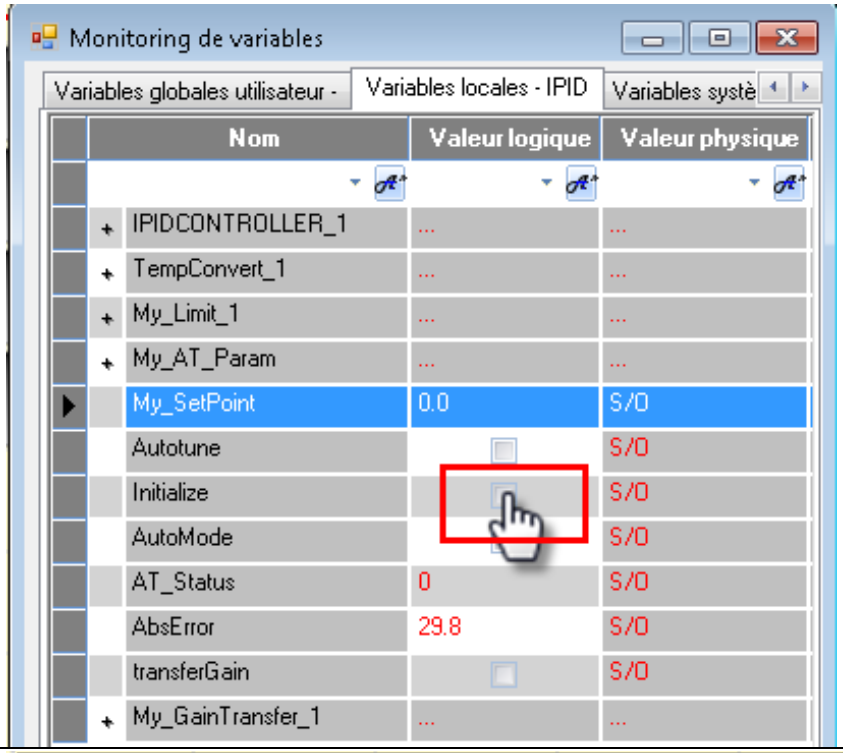
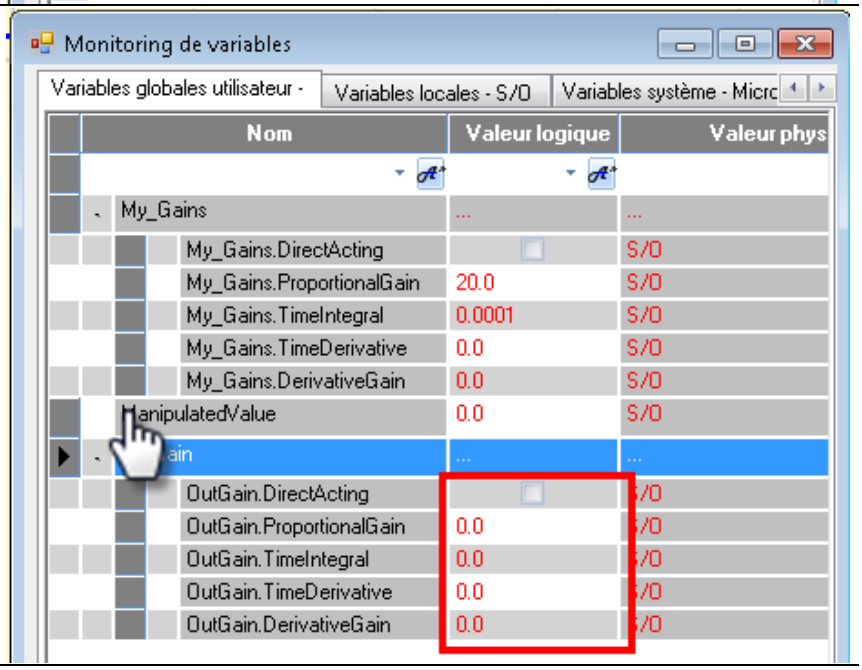
- Moteur entraînant un volant à disque, avec le moteur couplé au volant via un arbre rigide en torsion ; énergie cinétique de rotation et énergie de torsion comme unités de stockage.
- Circuit électrique composé d'une source de courant entraînant une série LR (bobine d'induction et résistance) avec un shunt C (condensateur) ; énergie inductive et énergie capacitive comme unités de stockage.

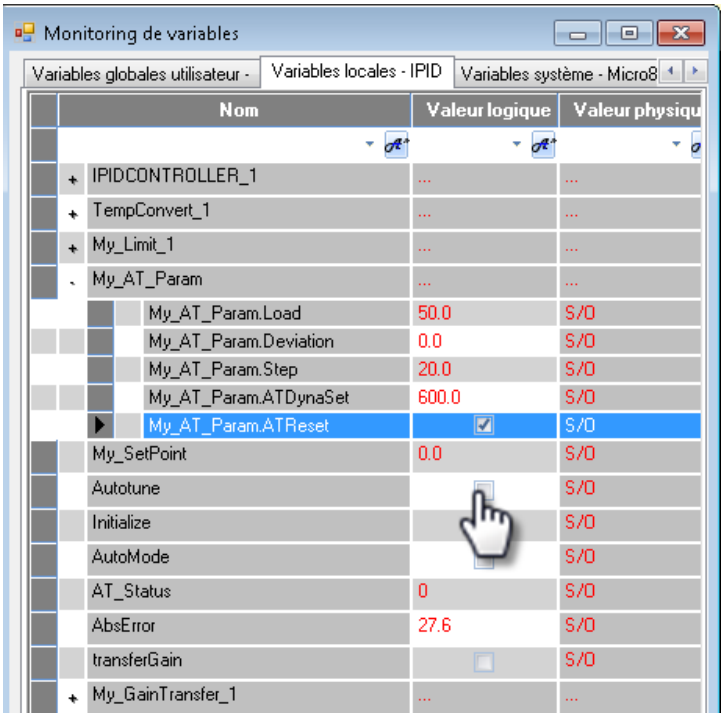
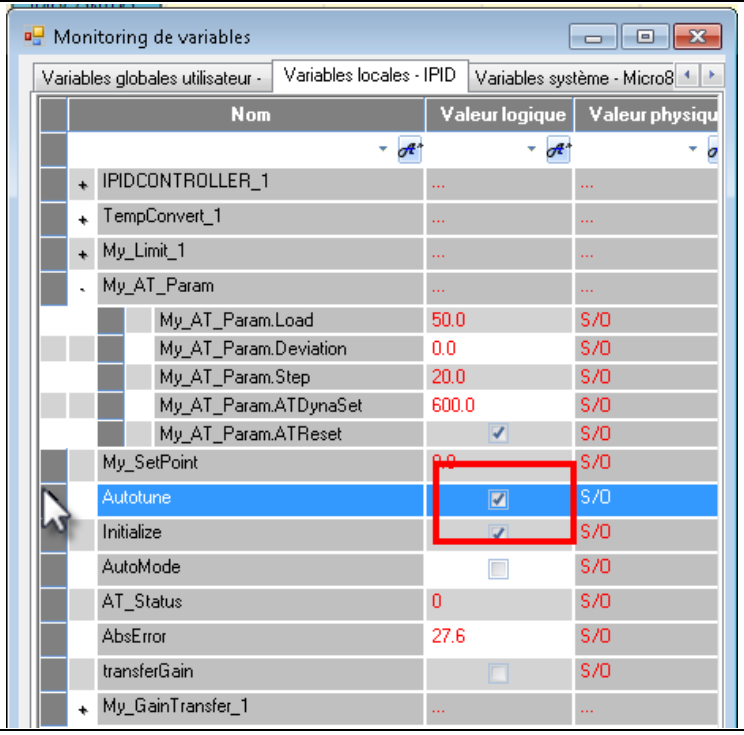
Les systèmes entraînés par moteur et les systèmes de chauffage peuvent généralement être modélisés par le système LR et le circuit électrique C.

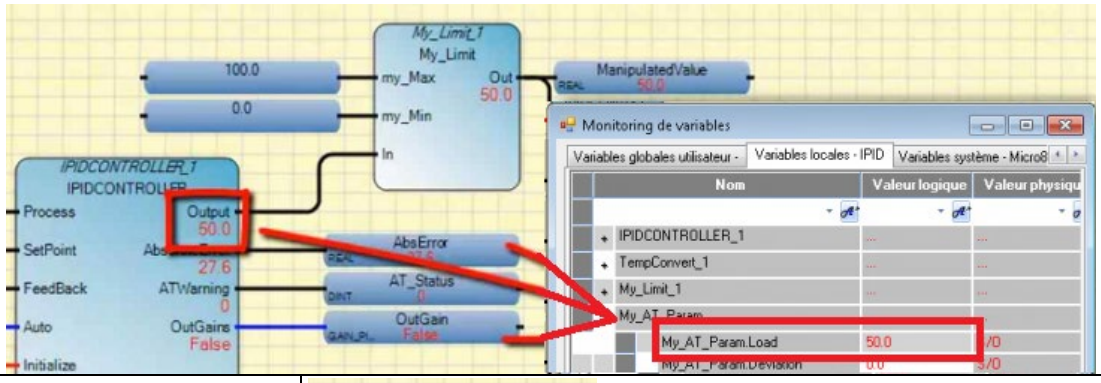
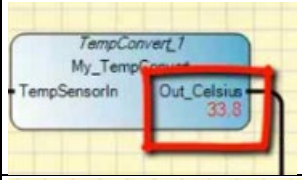
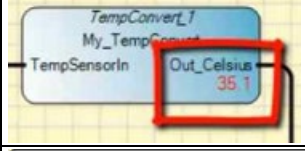
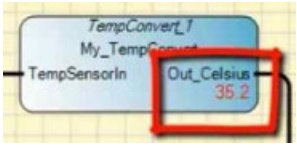
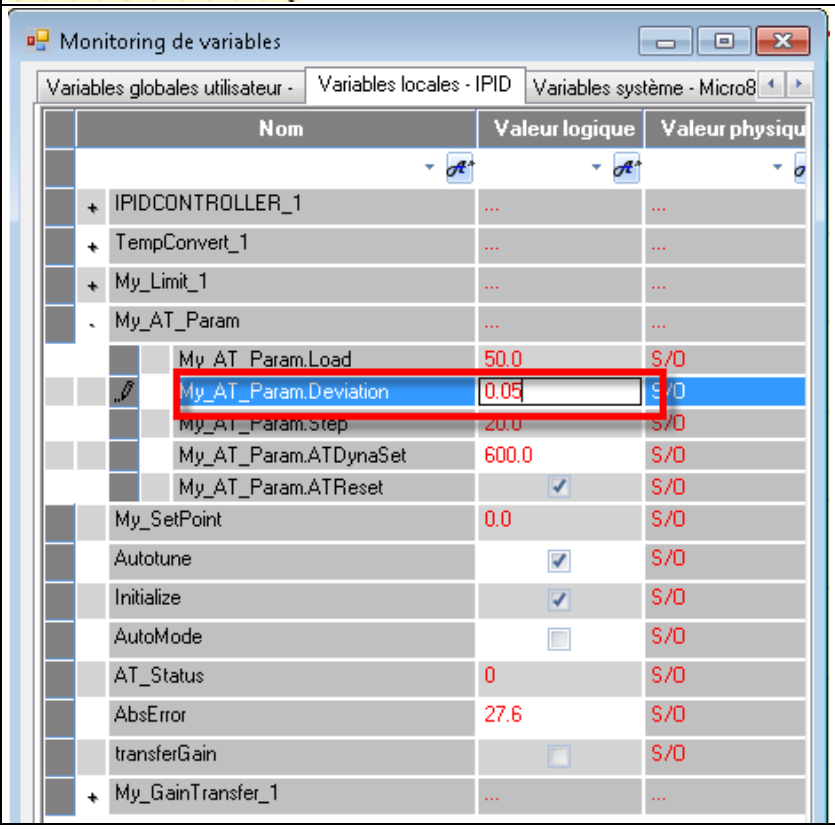
Configuration du réglage automatique

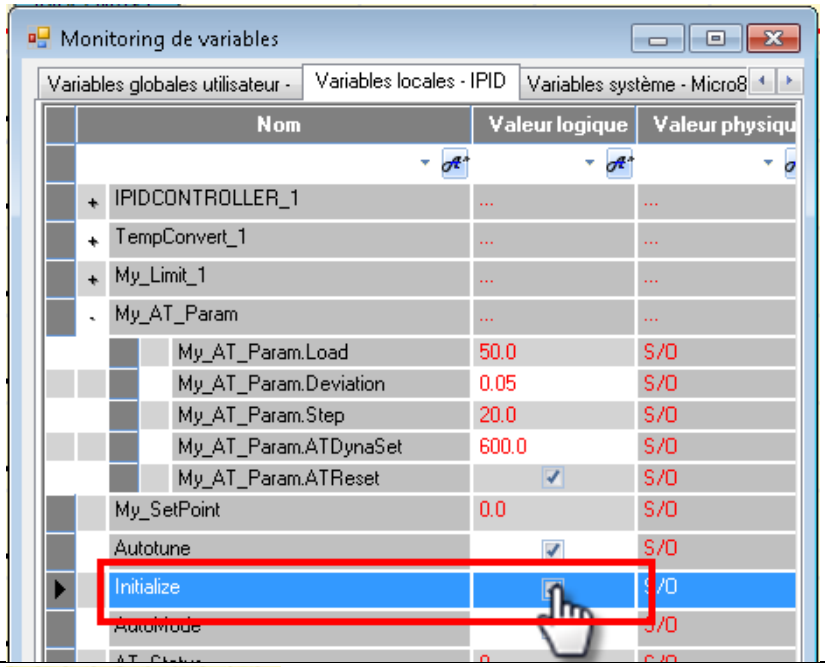
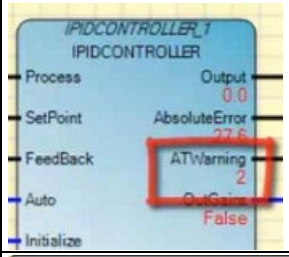
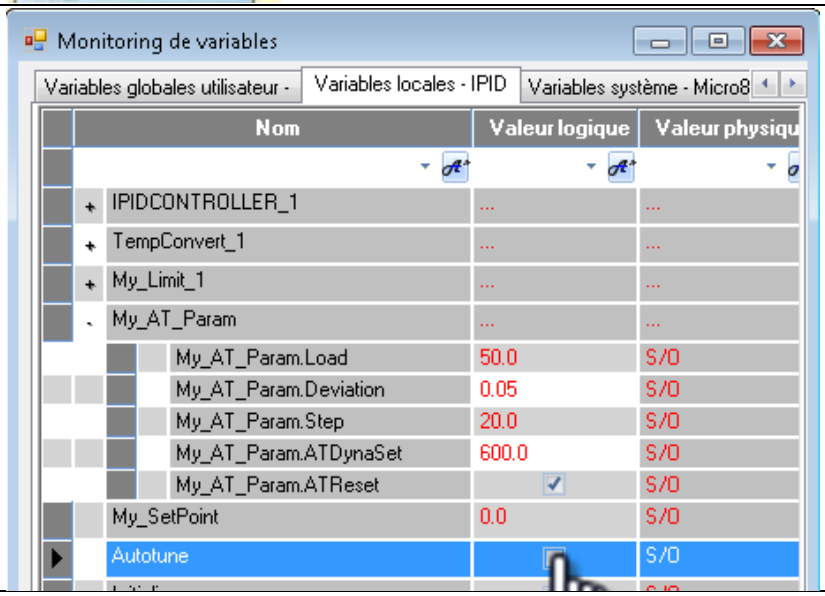
Utilisez cette procédure générale lors de la mise en œuvre du réglage automatique, à l'aide de la fonction IPIDController sur la page 537.

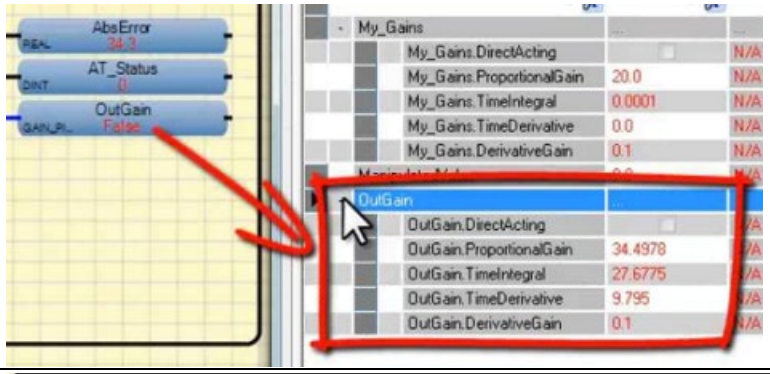
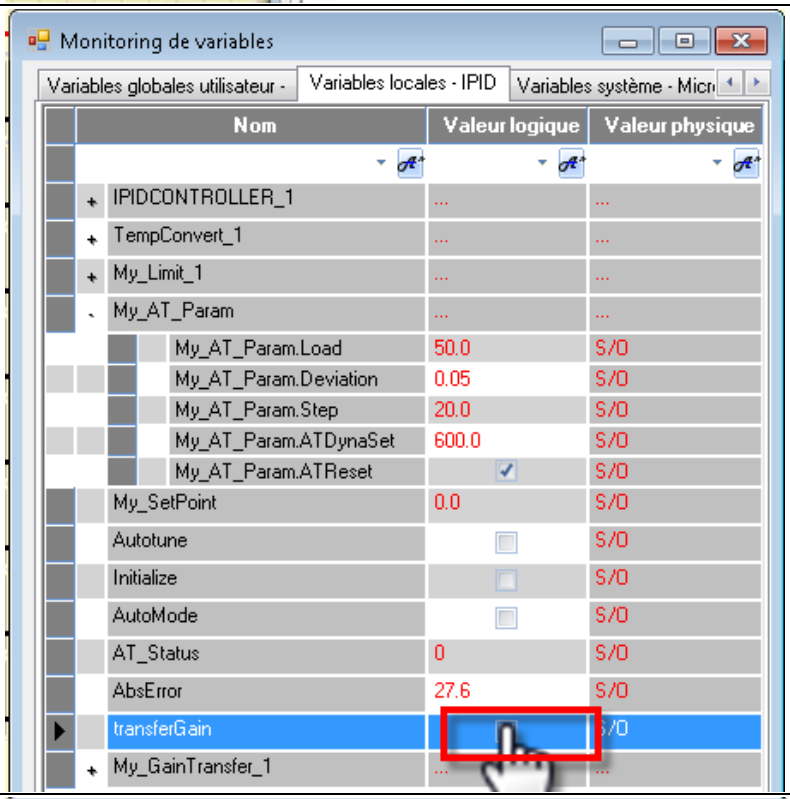
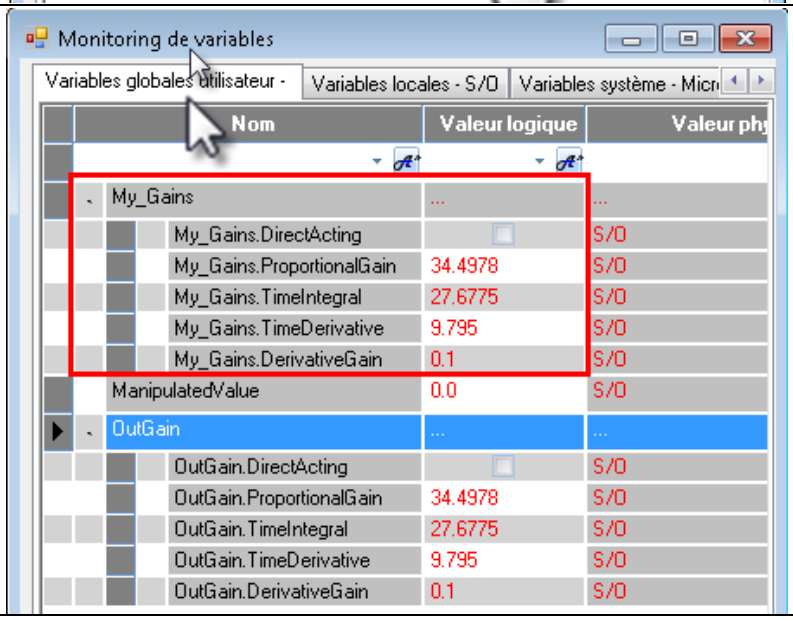
N°	Étape	Exemple
1	Réinitialisez le point de consigne.	

N°	Étape	Exemple
2	Définissez le mode Auto sur Faux.	
3	Définissez les paramètres de gain.	

N°	Étape	Exemple
4	Définissez les paramètres de réglage automatique.	Définissez les paramètres de réglage automatique, notamment la valeur de charge initiale, la modification d'étape pour la sortie, le temps d'exécution estimé du réglage automatique et la réinitialisation du réglage automatique. 
5	Définissez les valeurs Initialize et AutoTune sur Vrai.	

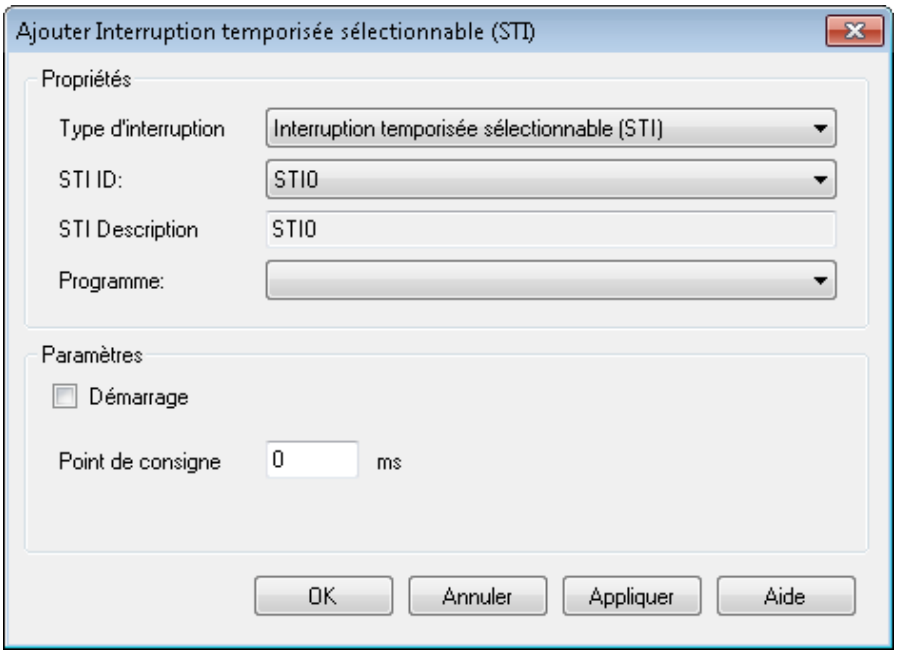
N°	Étape	Exemple
6	Notez les changements de sortie pour la valeur de charge, lorsque vous définissez AutoTune sur Vrai.	
7	Observez la hausse rapide de la valeur de traitement, jusqu'à ce qu'elle se rapproche du point de saturation.	
8	Observez la stabilisation de la valeur de traitement et sa fluctuation.	 
9	Définissez la déviation.	

N°	Étape	Exemple
10	Définissez Initialize sur Faux.	
11	Le contrôleur lance le réglage automatique. Attendez que ATWarning passe à 2.	
12	Définissez AutoTune sur Faux.	

N°	Étape	Exemple
13	Observez l'apparition des valeurs réglées dans OutGains.	
14	Transférez le paramètre de OutGain à My_Gains.	
15	Observez la mise à jour du contrôleur avec le paramètre de gain réglé.	

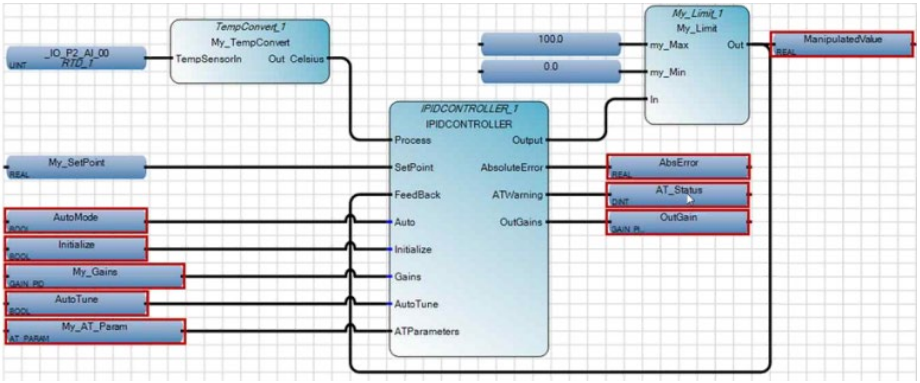
Utilisation d'un bloc d'instruction programmable en Interruption temporisée (STI) avec la fonction de réglage automatique

Bien qu'une instruction PID fonctionne correctement lorsqu'elle n'est pas contrôlée par un bloc d'instruction programmable en Interruption temporisée (STI), l'utilisation d'une instruction STI augmente les chances de succès du réglage automatique, car celui-ci s'exécute selon un cycle fixe.



Exemple : IPIDController avec réglage automatique

L'exemple de programme suivant présente les variables utilisées pour configurer les paramètres, en vue d'un réglage automatique.



Paramètres de réglage automatique

Le tableau suivant décrit les variables utilisées avec chaque paramètre, dans l'exemple de configuration du réglage automatique.

Paramètres d'entrée		
Variable	**Paramètre**	**Description**
AutoMode	Auto	Mode de fonctionnement du contrôleur PID : TRUE - le contrôleur fonctionne en mode normal. FALSE - le terme dérivatif est ignoré, forçant la sortie du contrôleur à suivre le retour codeur dans les limites du contrôleur et permettant à celui-ci de revenir en mode auto sans secouer la sortie.

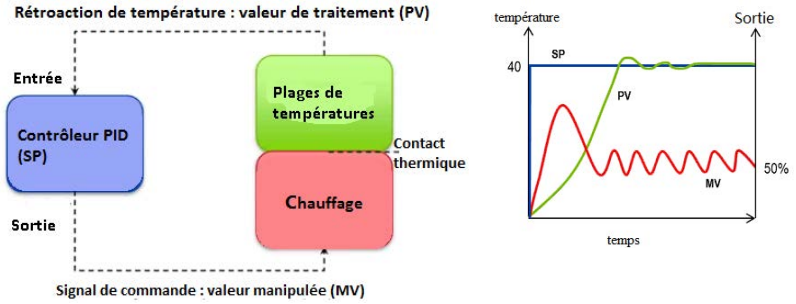
Initialiser	Initialiser	Initialise la séquence AutoTune. Un changement apporté à la valeur (de TRUE à FALSE ou de FALSE à TRUE) a entraîné la suppression, par le contrôleur, des gains proportionnels au cours du cycle.
My_Gains	Gains	Établir la valeur PID des gains pour IPIDController.
My_Gains.DirectActing	DirectActing	Définit le type d'action pour la sortie. TRUE : action directe (la sortie va dans la même direction que l'erreur). Autrement dit, la valeur de traitement réelle est supérieure au point de consigne et la mesure de contrôle adéquate consiste à augmenter la sortie. Par exemple, refroidir. FALSE : action inverse (la sortie va dans le sens opposé à l'erreur). Autrement dit, la valeur de traitement réelle est supérieure au point de consigne et la mesure de contrôle adéquate consiste à diminuer la sortie. Par exemple, chauffer.
My_Gains.ProportionalGain	ProportionalGain	Gain proportionnel pour PID (>= 0,0001).
My_Gains.TimeIntegral	TimeIntegral	Valeur temporelle intégrale pour PID (>= 0,0001). La tendance à l'oscillation augmente à mesure que la valeur TI diminue.
My_Gains.TimeDerivative	TimeDerivative	Valeur temporelle dérivative pour PID (> 0,0). L'amortissement augmente parallèlement au temps dérivatif, mais baisse si le temps dérivatif atteint une valeur trop élevée.
My_Gains.DerivativeGain	Gain dérivé pour PID (> 0,0).	
AutoTune	Lorsque fixé à TRUE et que Auto et Initialize sont FAUX, la séquence de réglage automatique démarre.	
ATParameters		
Load	• Valeur de sortie initiale durant le réglage automatique. • Permet de stabiliser la valeur de traitement à la charge.	
Deviation	• Déviation standard pour une série de valeurs de traitement stabilisées. Par exemple, si la valeur de traitement est stabilisée entre 31,4 et 32,0, la valeur de déviation sera (32,0-31,4)/2 = 0,3. • Certaines valeurs de traitement, telles que la température, mettent très longtemps à se stabiliser.	
Étape	• Le traitement de réglage automatique prend en compte la manière dont la valeur de traitement réagit aux modifications apportées à la valeur d'étape et en dérive les paramètres de gain.	
ATDynaSet	• Temps alloué à l'ensemble du traitement de réglage automatique. Cette valeur doit être supérieure au temps réel nécessaire à la réalisation du réglage automatique. • La valeur commune pour de nombreux systèmes est 600 secondes, toutefois, certains systèmes nécessitent davantage de temps.	
ATReset	• Si cette valeur est définie sur TRUE, la sortie est réinitialisée (0) une fois le réglage automatique terminé. • Si elle est définie sur FALSE, la sortie reste à la valeur de charge une fois le réglage auto terminé.	
Paramètres de sortie		
Paramètre	**Description**	
AbsoluteError	Erreur absolue (processus – point de consigne) issue du contrôleur.	
ATWarning	Avertissement pour la séquence de réglage automatique. Les valeurs possibles sont les suivantes : • 0 - pas de réglage automatique. • 1 : en mode Réglage automatique. • 2 : réglage automatique terminé. • -1 : entrée ERREUR 1 automatiquement définie sur TRUE, aucun réglage automatique possible. • -2 : erreur de réglage automatique ERREUR 2, expiration de ATDynaSet.	
OutGains	Gains calculés suite aux séquences de réglage automatique	

Exemple : Création d'une boucle de rétroaction pour la valeur manipulée

L'ajout d'une boucle de rétroaction pour la valeur manipulée empêche un dépassement excessif, en fournissant des valeurs minimale et maximale pour MV.

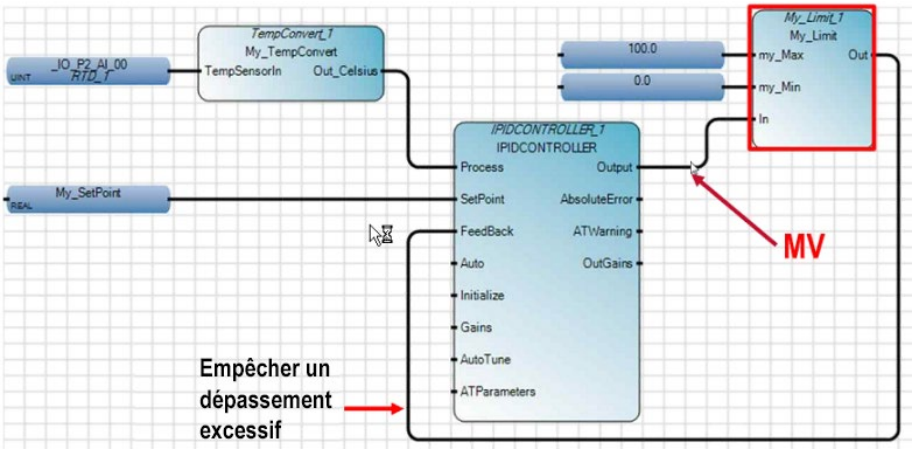
Exemple de boucle de rétroaction de température

Au début du traitement de contrôle de la température, la différence entre la valeur de traitement (PV) et la valeur du point de consigne (SP) est importante, comme illustré dans le graphique suivant. Dans cet exemple de boucle de rétroaction de température, la valeur PV commence à 0 degré Celsius et continue à monter vers la valeur SP de 40 °C. Notez également que la fluctuation entre la valeur manipulée (MV) haute et basse se stabilise avec le temps. Le comportement de MV dépend des valeurs utilisées dans chaque paramètre : P, I et D.



IPIDController avec boucle de rétroaction

Le diagramme de bloc fonctionnel suivant comprend une boucle de rétroaction pour la valeur manipulée, qui empêche un dépassement excessif en fournissant des valeurs minimale et maximale pour MV.



Exemple : Ajout d'un UDFB à un programme PID

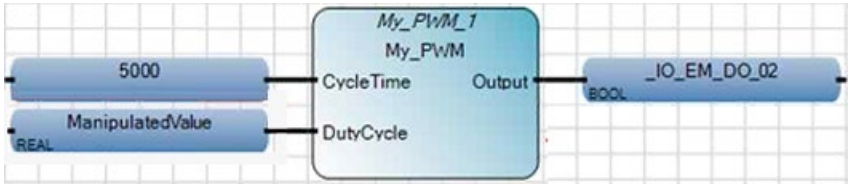
Pour les programmes PID, utilisez des blocs de fonctions définis par l'utilisateur (UDFB) en dehors du programme principal, afin d'exécuter des fonctions spécialisées, telles que la conversion d'unités ou le transfert de valeurs. Les exemples suivants sont des UDFB.

Transfert de la valeur de gain du réglage automatique

Ce bloc UDFB transfère la valeur de gain Autotune à My_GainTransfer, afin que le contrôleur puisse l'utiliser.

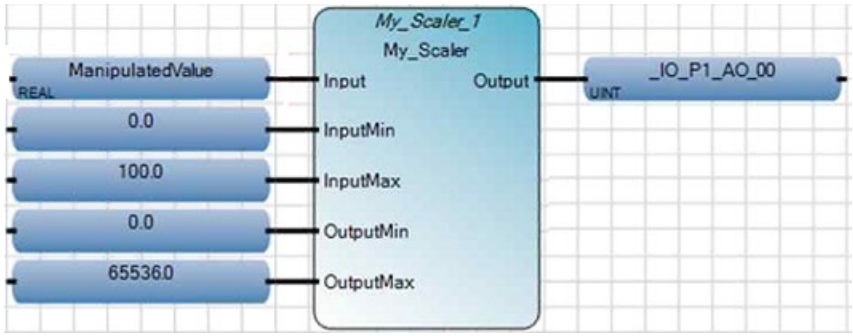
Conversion d'une valeur manipulée en sortie numérique

Cet UDFB convertit une valeur manipulée (MV) en une sortie numérique (DO), afin de pouvoir l'utiliser pour contrôler une entrée numérique (DI).



Conversion d'une valeur manipulée en sortie analogique

Cet UDFB convertit une valeur manipulée (MV) en une sortie analogique (AO), afin de pouvoir l'utiliser pour contrôler une entrée analogique (AI).



Exemple : Création d'un programme IPIDController pour contrôler la température

Le programme de contrôle de la température maintient la température au sein de la zone de contrôle.

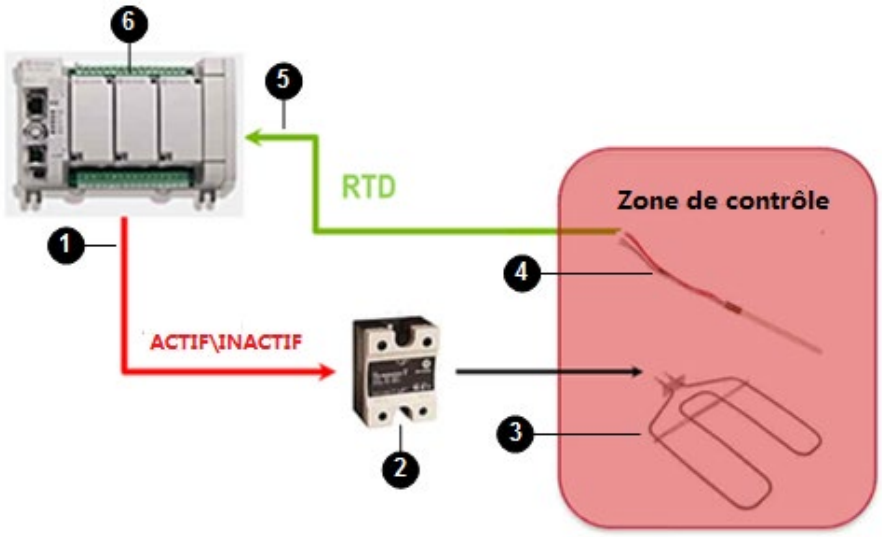
Valeurs manipulées, de point de consigne et de traitement

Le tableau suivant définit les valeurs SP, PV et MV utilisées dans le programme de contrôle de la température.

Élément	Description
Point de consigne (SP)	Mesure de température en degrés Celsius, qui définit la température pour la zone de contrôle.
Valeur de traitement (PV)	Doit être convertie dans la même unité que la valeur SP (mesure en degrés Celsius).
Valeur manipulée (MV)	Doit être convertie en valeur analogique pour la sortie au PWM, afin de contrôler l'élément chauffant.

Système de contrôle de la température

Le diagramme et le tableau suivants définissent les composants dans le système de contrôle de la température qui sont contrôlés par le programme de contrôle de la température et les événements qui se produisent lors de l'exécution du programme de contrôle.



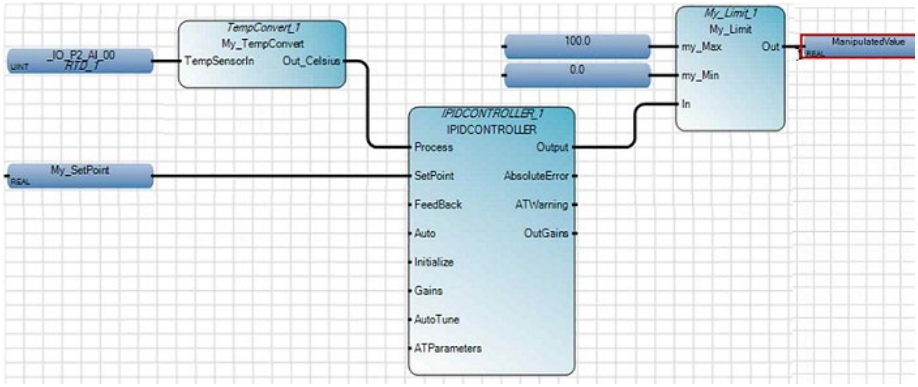
Séquence d'événements pour le programme de contrôle de la température

Le tableau suivant indique les composants du système de contrôle de la température et décrit la séquence des événements qui se produit dans le système lors de l'exécution du programme de contrôle de la température.

Non	Élément	Description
1	Sortie du contrôleur	Envoie la valeur MV au PWM (Activé/Désactivé).
2	Contrôleur de température PWM (modulation de la largeur d'impulsion)	Relais statique qui contrôle l'élément chauffant.
3	Élément chauffant	Accroît la température dans la zone de contrôle.
4	Détecteur de température de résistance (RTD)	Mesure la température au sein de la zone de contrôle et envoie la valeur PV (signal RTD) à l'entrée du contrôleur.
5	Entrée du contrôleur	Reçoit la valeur PV (signal RTD).
6	Programme PLC	Convertit la valeur PV (signal RTD) dans la même unité que la valeur SP (degrés Celsius) et détermine la différence entre les valeurs PV et SP, puis ajuste la valeur MV en fonction des valeurs définies pour les paramètres P, I et D.

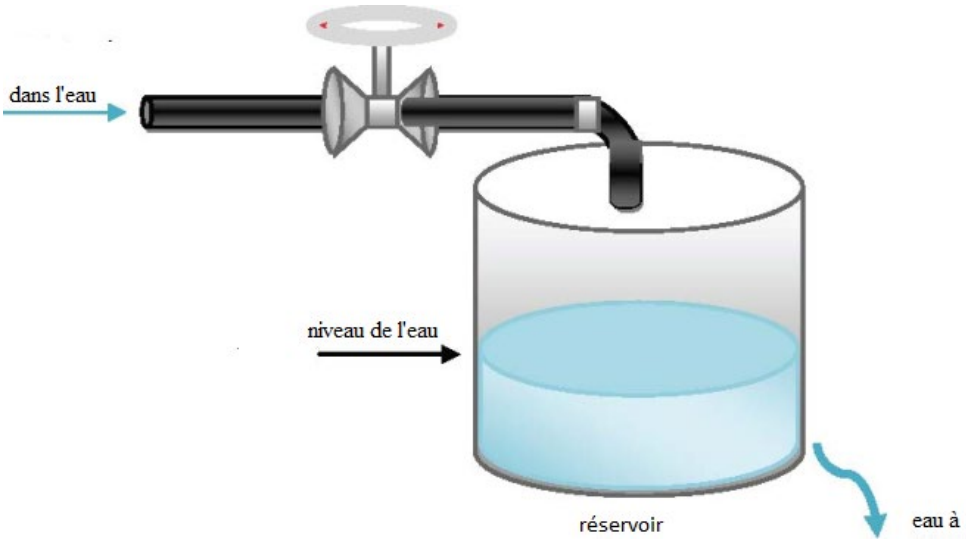
Exemple : Diagramme de bloc fonctionnel pour contrôler la température

Ce diagramme de bloc fonctionnel indique les blocs fonctionnels, prédéfinis et définis par l'utilisateur, utilisés par l'application pour surveiller la température dans une zone de contrôle.



Exemple : Création d'un programme IPIDController pour contrôler le niveau d'alimentation en eau

Le programme de contrôle du niveau d'alimentation en eau maintient un niveau d'eau suffisant dans un réservoir d'alimentation équipé d'un trop-plein. Une électrovanne contrôle l'arrivée d'eau et remplit le réservoir à un débit prédéfini ; l'eau en trop-plein est également contrôlée à ce débit prédéfini.



Informations sur l'exemple de programme

L'exemple de programme de contrôle du niveau d'alimentation en eau comprend les informations suivantes.

- Séquence d'événements qui se produit lors du traitement de contrôle
- Utilisation des valeurs manipulées, de point de consigne et de traitement par le programme de contrôle
- Exemple de diagramme de bloc fonctionnel, indiquant IPIDController et d'autres blocs d'instructions

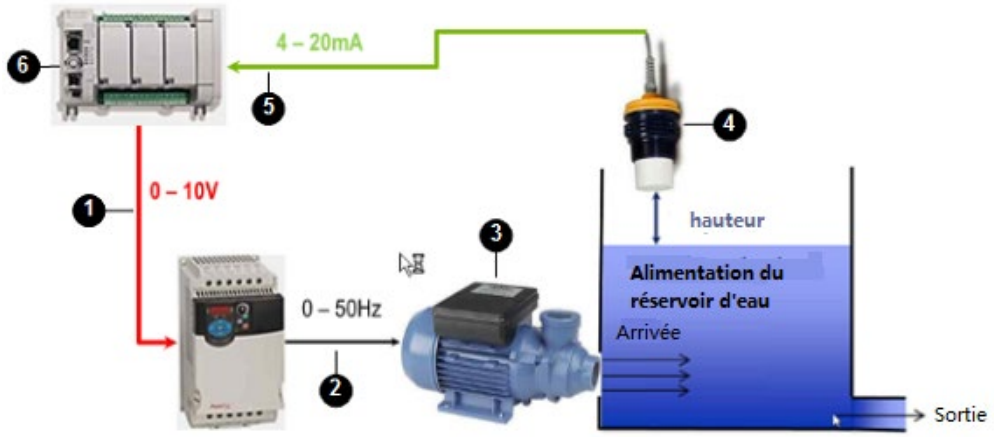
Valeurs manipulées, de point de consigne et de traitement

Le tableau suivant indique comment les valeurs SP, PV et MV sont utilisées dans le programme de contrôle du niveau d'alimentation en eau.

Élément	Description
Point de consigne (SP)	Mesure de hauteur qui définit le niveau d'alimentation en eau ciblé.
Valeur de traitement (PV)	La valeur 4-20 mA doit être convertie dans la même unité que la valeur SP (mesure de hauteur).
Valeur manipulée (MV)	Doit être convertie en valeur analogique pour la sortie au niveau du variateur, afin de contrôler la pompe.

Système de contrôle du niveau d'alimentation en eau

Le diagramme suivant présente les composants du système de contrôle du niveau d'alimentation en eau, qui sont contrôlés par le programme de contrôle du niveau d'alimentation en eau. Le tableau présenté après ce diagramme décrit les événements qui se produisent lors de l'exécution du programme de contrôle.



Séquence d'événements du système de contrôle du niveau d'alimentation en eau

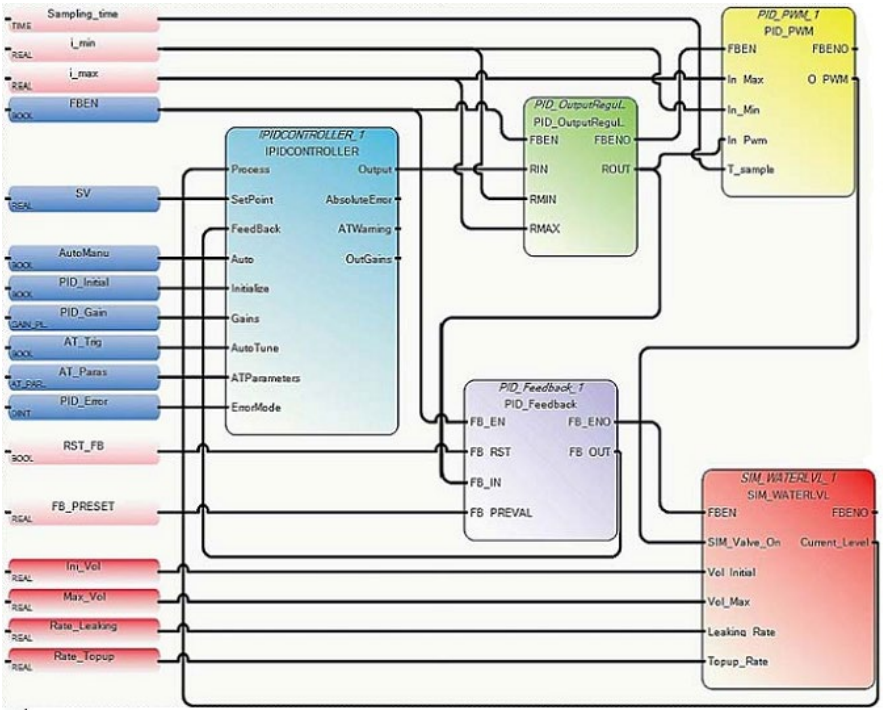
Le tableau suivant indique les composants du système de contrôle du niveau d'alimentation en eau et décrit, dans l'ordre de séquence, les événements qui se produisent dans le système lors de l'exécution du programme de contrôle du niveau d'alimentation en eau.

Non	Élément	Description
1	Sortie du contrôleur	Envoie la valeur MV au variateur PowerFlex (0-10 V).
2	Variateur PowerFlex	Contrôle la pompe à eau (0-50 Hz).
3	Pompe à eau	Contrôle le niveau d'eau dans le réservoir d'alimentation.
4	Appareil de transfert de sortie	Mesure la hauteur du niveau d'eau (4-20 mA) et envoie la valeur PV au contrôleur.
5	Entrée du contrôleur	Reçoit la valeur PV (niveau d'alimentation en eau, 4-20 mA).

Non	Élément	Description
6	Programme PLC	Convertit la valeur PV dans la même unité que la valeur SP (mesure de hauteur) et détermine la différence entre les valeurs PV et SP, puis ajuste la valeur MV en fonction des valeurs définies pour les paramètres P, I et D.

Exemple : Diagramme de bloc fonctionnel pour contrôler le niveau d'alimentation en eau

Le diagramme de bloc fonctionnel suivant indique les blocs fonctionnels, prédéfinis et définis par l'utilisateur, qui permettront au programme de contrôler le niveau de l'alimentation en eau.



Blocs fonctionnels et UDFB utilisés dans le diagramme FBD de niveau d'eau

Cette application, développée dans le langage FBD (Diagramme de Bloc Fonction), utilise les instructions décrites dans le tableau suivant.

Bloc de fonctions	Description
Bloc fonctionnel IPIDController	Fournit le contrôle de procédés PID.
UDFB PID_OutputRegulator	Régule la sortie de IPIDCONTROLLER dans la plage autorisée, afin de garantir que le matériel utilisé dans le procédé n'est pas endommagé. **Exemple de code :** IF RMIN ≤ RIN ≤ RMAX, then ROUT = RIN, IF RIN < RMIN, then ROUT = RMIN, IF RIN > RMAX, then ROUT = RMAX
UDFB PID_Feedback	Joue le rôle de multiplexeur. **Exemple de code :** IF "FB_RST" is false, FB_OUT=FB_IN; If "FB_RST" is true, then FB_OUT=FB_PREVAL.

Bloc de fonctions	Description
Bloc fonctionnel IPIDController	Fournit le contrôle de procédés PID.
UDFB PID_PWM	Offre une fonction PWM, qui convertit une valeur réelle en une sortie Activé/Désactivé à composante temporelle.
UDFB SIM_WATERLVL	Simule le procédé dans l'exemple d'application.

PID (dérivé intégral proportionnel)

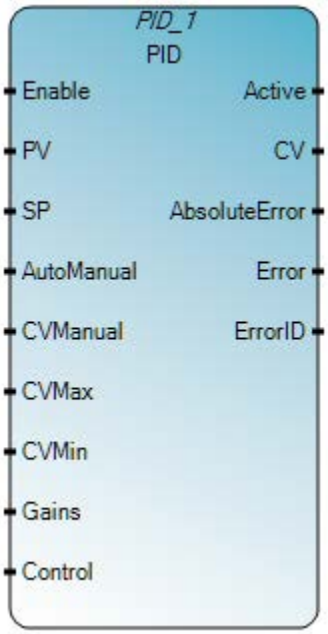
Une instruction de sortie qui contrôle les propriétés physiques telles que la température, la pression, le niveau de liquide ou le débit utilisant les boucles de processus.

Détails de l'opération :

- Lorsqu'il est activé, PIC contrôle le processus utilisant les paramètres d'entrée, notamment SP et Gains du contrôleur PID.
- Transition du mode Exécuter ou mode Programme : l'instruction PID est désactivée, les valeurs des paramètres sont conservées.
- Transition du mode Programme au mode Exécuter : l'instruction PID reste désactivée jusqu'à ce qu'un utilisateur réinitialise Activer en VRAI.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activer l'instruction. VRAI : lancement de l'exécution avec les paramètres d'entrée actuels. FAUX : pas d'exécution de PID. Définissez CV sur 0, puis calculez AbsoluteError.

Paramètre	Type de paramètre	Type de données	Description
PV	Entrée	REAL	Valeur de traitement. Cette valeur est généralement lue à partir d'un module d'entrée analogique. L'unité SI doit être la même que le Point de consigne.
SP	Entrée	REAL	La valeur du point de consigne pour le processus.
AutoManual	Entrée	BOOL	Sélection du mode automatique ou manuel. VRAI : CV est contrôlé par PID. FAUX : PID s'exécute et CV est contrôlé par l'entrée CVManual.
CVManual	Entrée	REAL	Entrée de la valeur de contrôle définie pour l'opération en mode manuel. La plage valide pour CVManual est : CVMin < CVManual < CVMax
CVMin	Entrée	REAL	Limite minimum de la valeur de contrôle. Si CV < CVMin, alors CV = CVMin. Si CVMin > CVMax, une erreur se produit.
CVMax	Entrée	REAL	Limite maximum de la valeur de contrôle. Si CV > CVMax, alors CV = CVMax. Si CVMax < CVMin, une erreur se produit.
Gains	Entrée	PID_GAINS	Gains PID pour le contrôleur. Utilisez le type de données PID_GAINS pour configurer le paramètre Gains.
Control	Entrée	BOOL	Direction de contrôle du processus : VRAI : action directe, comme le refroidissement. FAUX : action inverse, comme le réchauffement.
Active	Sortie	BOOL	État du contrôleur PID. VRAI : PID est actif. FAUX : PID est arrêté.
CV	Sortie	REAL	La sortie de la valeur de contrôle. Si une erreur se produit, CV est 0.
AbsoluteError	Sortie	REAL	L'erreur absolue est la différence entre la valeur de processus (PV) et la valeur du point de consigne (SP).
Error	Sortie	BOOL	Indique l'existence d'une condition d'erreur. VRAI : l'opération a rencontré une erreur. FAUX : l'opération s'est terminée avec succès ou l'instruction ne s'exécute pas.
ErrorID	Sortie	USINT	Valeur numérique unique permettant d'identifier l'erreur. Les erreurs sont définies dans les codes d'erreur PID.

Type de données PID_GAINS

Le tableau suivant décrit le type de données PID_GAINS de l'instruction PID sur la page 527.

Paramètre	Type de paramètre	Type de données	Description
Kc	Entrée	REAL	Gain de contrôleur pour PID. Les paramètres Proportionnel et Intégral dépendent de ce gain. (>= 0,0001). Augmenter Kc améliore le temps de réponse, mais également le dépassement et l'oscillation du PID. Si Kc n'est pas valide, une erreur survient.
Ti	Entrée	REAL	Constante intégrale de temps en secondes (>= 0,0001). Augmenter Ti réduit le dépassement et l'oscillation du PID. Si Ti n'est pas valide, une erreur survient.

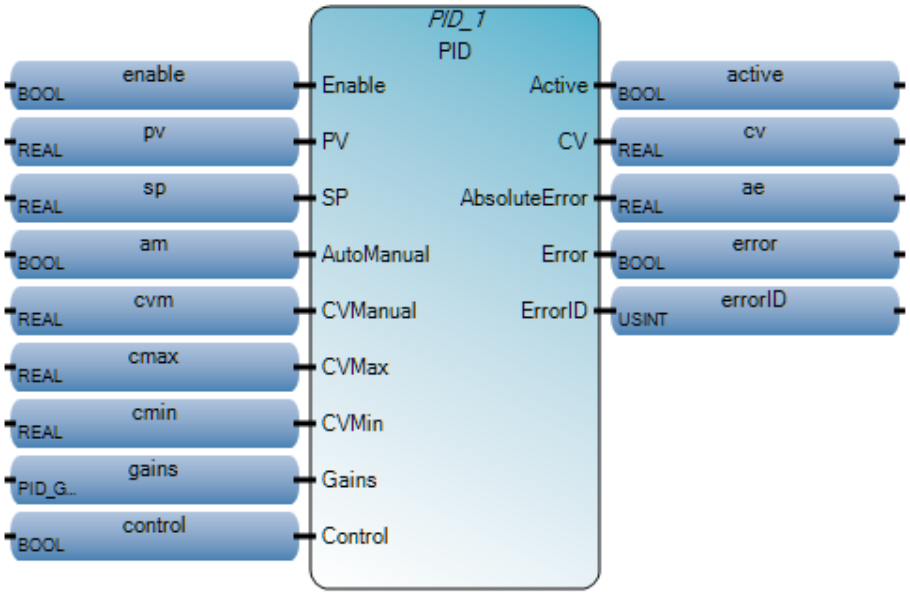
Paramètre	Type de paramètre	Type de données	Description
Kc	Entrée	REAL	Gain de contrôleur pour PID. Les paramètres Proportionnel et Intégral dépendent de ce gain. (>= 0,0001). Augmenter Kc améliore le temps de réponse, mais également le dépassement et l'oscillation du PID. Si Kc n'est pas valide, une erreur survient.
Td	Entrée	REAL	Constante dérivative de temps en secondes (>= 0,0). Lorsque Td égale 0, aucune action dérivative n'est exécutée, et PID devient un contrôleur PI. Augmenter Td réduit le dépassement et supprime l'oscillation du contrôleur PID. Si Td n'est pas valide, une erreur survient.
FC	Entrée	REAL	Constante de filtre (>= 0,0). La plage recommandée pour FC est comprise entre 0 et 20. Augmenter FC harmonise la réponse du contrôleur PID. Si FC n'est pas valide, une erreur survient.

Codes d'erreur PID

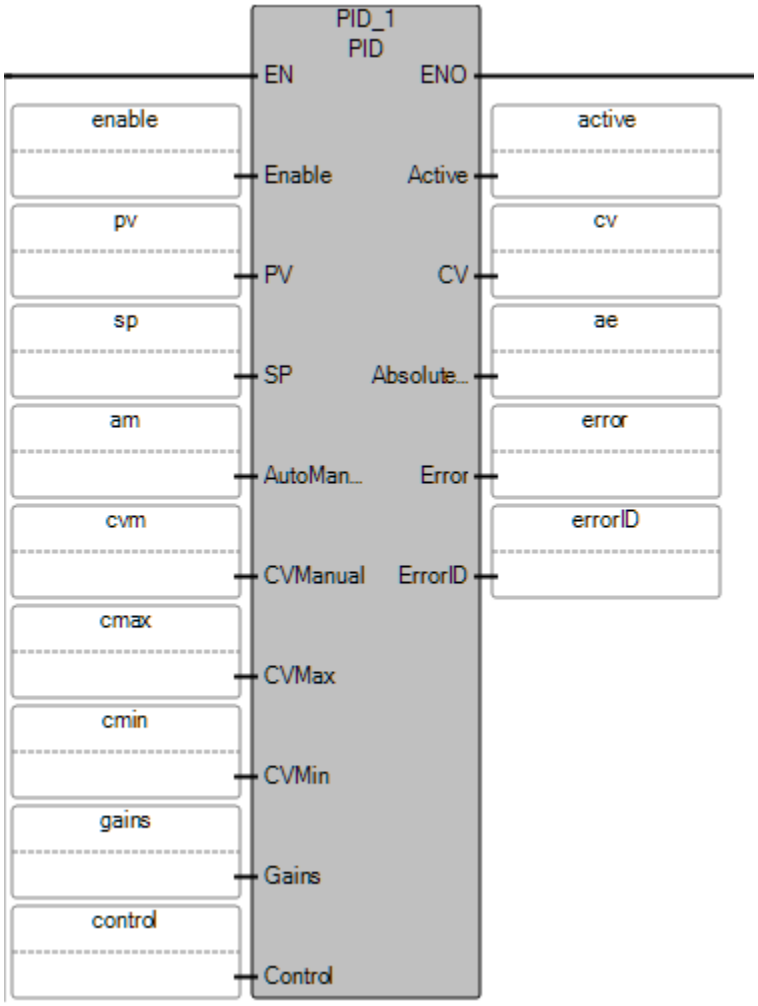
Utilisez ce tableau pour déterminer les codes d'erreur PID et les descriptions.

Code d'erreur	Description de l'erreur
0	PID fonctionne normalement.
1	Kc n'est pas valide.
2	Ti n'est pas valide.
3	Td n'est pas valide.
4	FC n'est pas valide.
5	CVMin > CVMax ou CVMax < CVMin
6	CVManual < CVMin CVManual n'est pas valide.
7	CVManual > CVMax CVManual n'est pas valide.

Exemple de diagramme de bloc de fonctions PID



Exemple de langage à relais PID

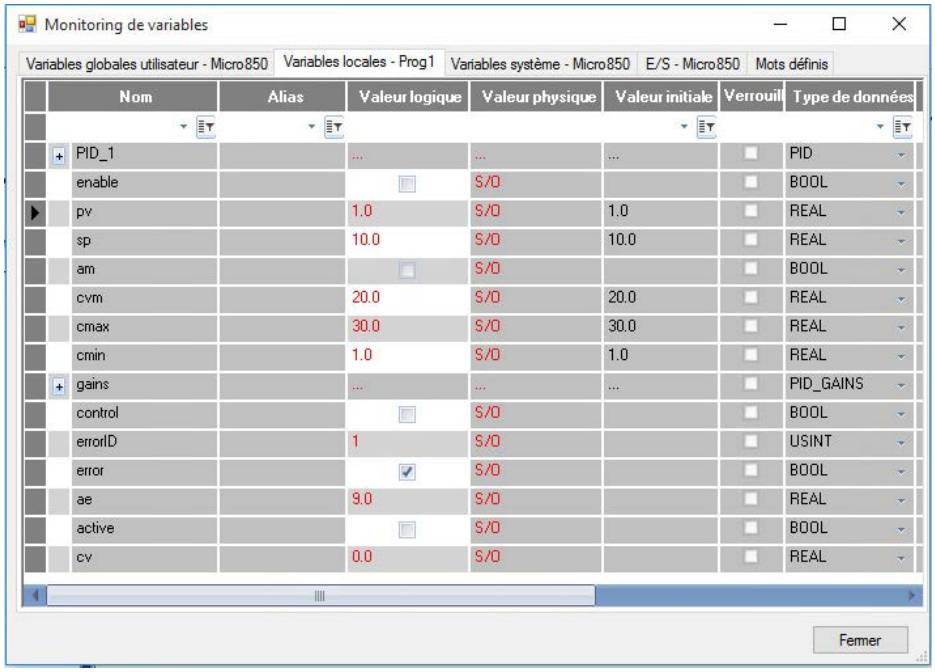


Exemple de texte structuré PID

```
1  PID_1(enable, pv, sp, am, cvm, cmax, cmin, gains, control);
2  active := PID_1.Active;
3  cv := PID_1.CV;
4  ae := PID_1.AbsoluteError;
5  error := PID_1.Error;
6  errorID := PID_1.ErrorID;
```

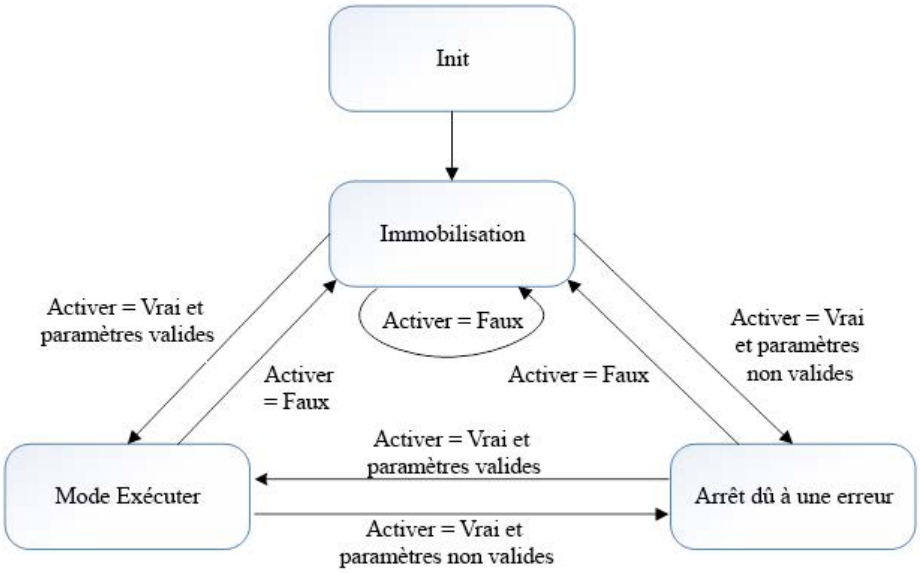
```
PID_1(
void PID_1(BOOL Enable, REAL PV, REAL SP, BOOL AutoManual, REAL CVManual, REAL CVMax, REAL CVMin, PID_GAINS Gains, BOOL Control)
Type: PID, Dérivé proportionnel intégral
```

Résultats PID



Machine d'état d'instruction PID

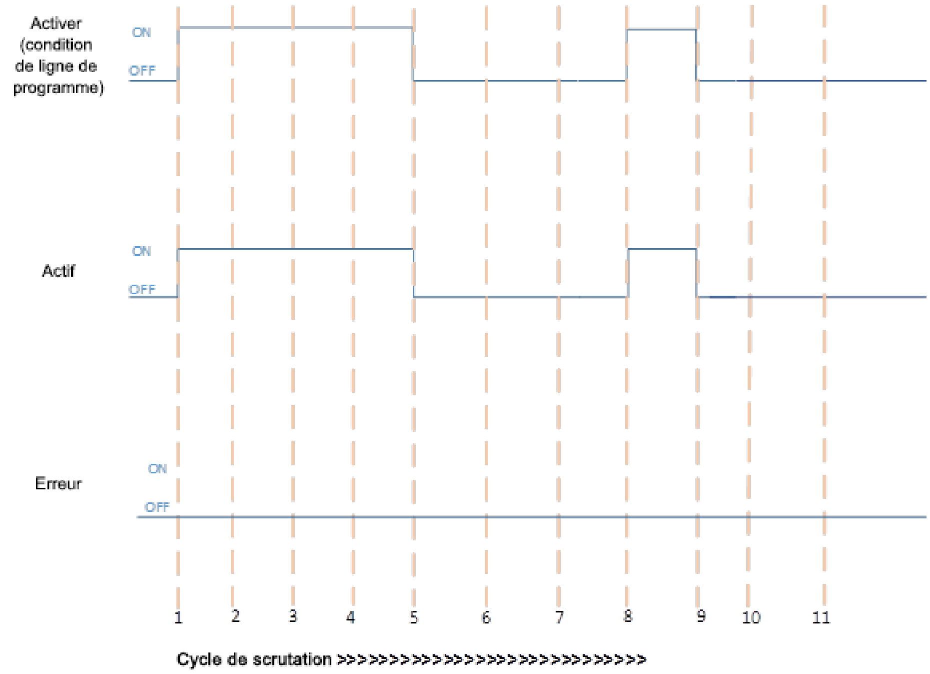
Le diagramme de machine d'état PID décrit les états de traitement de l'instruction PID sur la page 552.



Diagrammes de temporisation d'instructions PID

Les exemples de diagrammes de temporisation suivants décrivent les scénarios d'exécution pour l'instruction Dérivé intégral proportionnel (Proportional Integral Derivative, PID) sur la page 552.

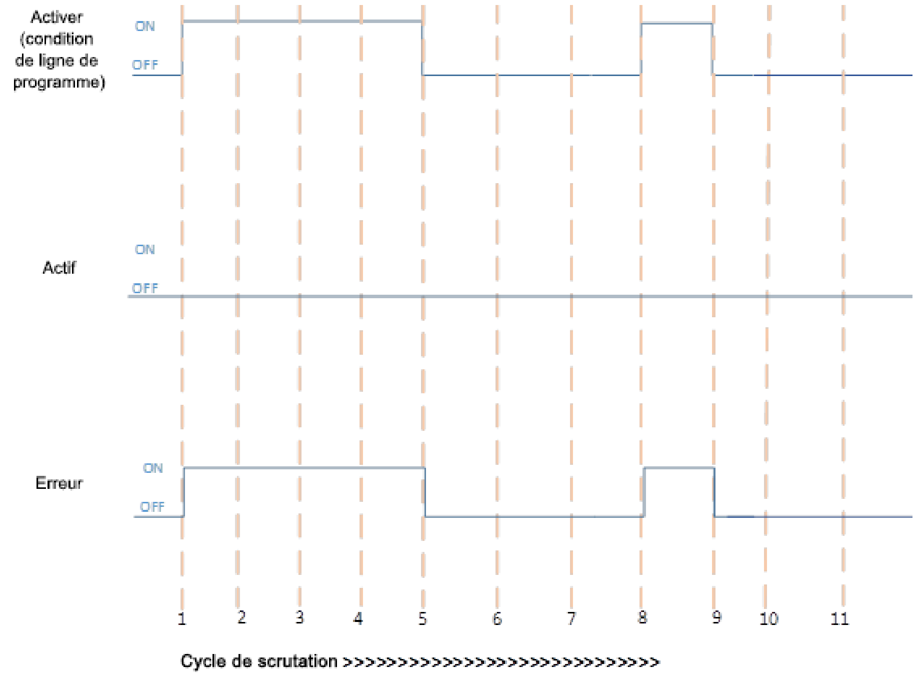
Exécution réussie de PID



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée sont valides. • Le bit Activer est VRAI. • Le bit Erreur est FAUX.
2,3,4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée sont valides. • Met à jour les paramètres de sortie PID.
5, 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
6, 7, 10, 11	Aucun changement de la condition de ligne. • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.

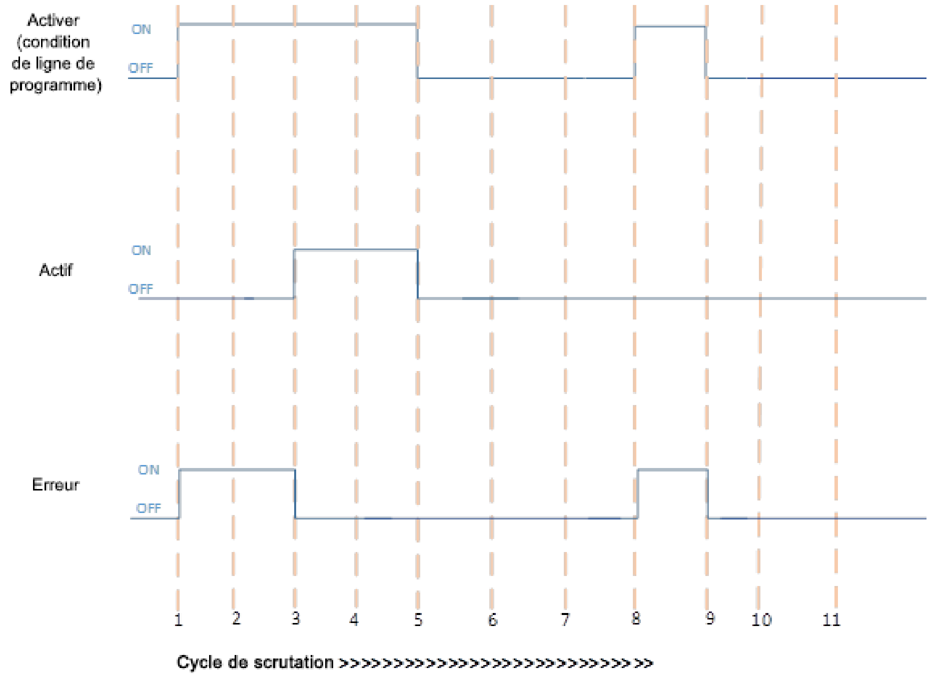
Exécution de PID avec erreur



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée ne sont pas valides. • Le bit Activer est FAUX. • Le bit Erreur est VRAI. La sortie ID d'erreur est définie. • La sortie CV est définie sur 0. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
2,3,4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée ne sont pas valides. • Met à jour les paramètres de sortie PID.
5, 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
6, 7, 10, 11	Aucun changement de la condition de ligne. • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.

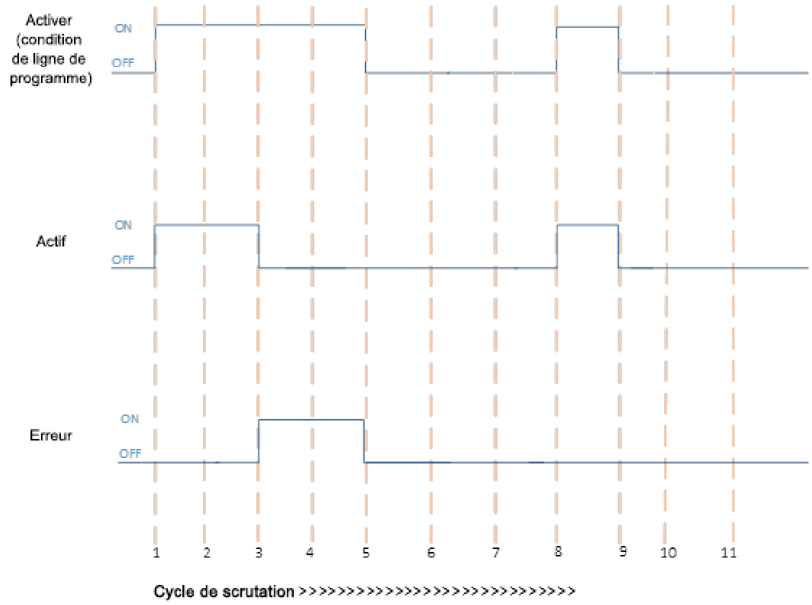
Exécution de PID avec Erreur et exécution réussie



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée ne sont pas valides. • Le bit Activer est FAUX. • Le bit Erreur est VRAI. La sortie ID d'erreur est définie. • La sortie CV est définie sur 0. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
2	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée ne sont pas valides. • Met à jour les paramètres de sortie PID.
3, 4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée sont valides. • Le bit Activer est VRAI. • Le bit Erreur est FAUX. • Met à jour les paramètres de sortie PID.
5, 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
6, 7, 10, 11	Aucun changement de la condition de ligne. • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.

Exécution de PID réussie et Erreur



Utilisez ce tableau pour déterminer les valeurs des paramètres pour chaque cycle de scrutation.

Cycle de scrutation	Description
1, 8	La condition de ligne devient VRAI lorsque : • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée sont valides. • Le bit Activer est VRAI. • Le bit Erreur est FAUX. • Mettez à jour les paramètres de sortie PID.
2	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée sont valides. • Mettez à jour les paramètres de sortie PID.
3, 4	Aucun changement de la condition de ligne. • Le bit d'entrée Activer est VRAI. • Les paramètres d'entrée ne sont pas valides. • Le bit Erreur est VRAI. La sortie ID d'erreur est définie. • La sortie CV est définie sur 0. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
5, 9	La condition de ligne devient FAUX lorsque : • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.
6, 7, 10, 11	Aucun changement de la condition de ligne. • Le bit Activer est FAUX. • Efface les paramètres de sortie PID, à l'exception d'Erreur absolue. • Erreur absolue calcule en fonction des valeurs d'entrée PV et SP.

Instructions d'horloge en temps réel (RTC)

Utilisez les instructions d'horloge en temps réel (RTC) pour configurer le calendrier et l'horloge.

Instruction	Description
RTC_READ sur la page 383	Lit les informations du module d'horloge en temps réel (RTC).
RTC_SET sur la page 385	Définit les données d'horloge en temps réel pour les informations du module RTC.

RTC_READ (lecture de l'horloge en temps réel)

Lit les informations du module d'horloge en temps réel (RTC).

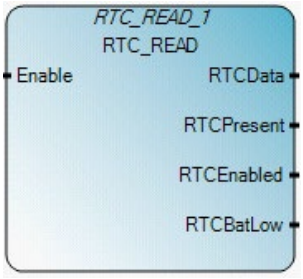
Détails de l'opération :

- Contrôleur Micro810 ou Micro820 avec RTC intégrée :
  - RTCBatLow est toujours défini sur zéro (0).
  - RTCEnabled est toujours défini sur un (1).
- Lorsque l'horloge en temps réel intégrée est déchargée ou que sa mémoire a été vidée en raison d'une panne électrique :
  - RTCData est défini sur 2000/1/1/0/0/0.
  - RTCEnabled est défini sur un (1).

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de la lecture des informations RTC.

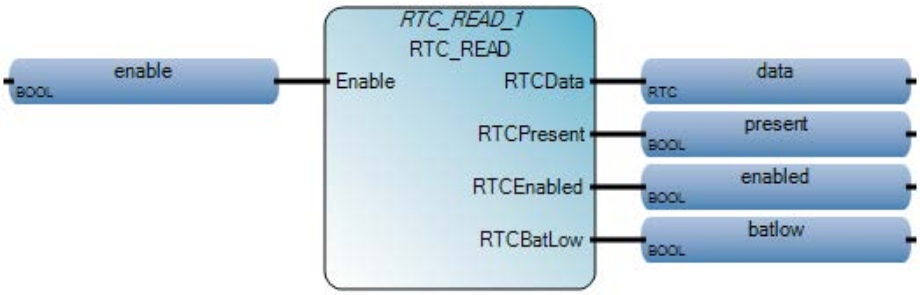
Paramètre	Type de paramètre	Type de données	Description
			FAUX : aucune opération de lecture n'est réalisée et les données RTC de sortie ne sont pas valides.
RTCData	Sortie	RTC	Information de données RTC: aa/mm/jj, hh/mm/ss, semaine. La sortie RTCData est définie à l'aide du type de données RTC.
RTCPresent	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est branché. FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC n'est pas branché.
RTCEnabled	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est activé (temporisation). FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC est désactivé (aucune temporisation).
RTCBatLow	Sortie	BOOL	VRAI : le niveau de la batterie RTC est faible. FAUX : le niveau de la batterie RTC n'est pas faible.
ENO	Sortie	BOOL	Activation de la sortie. Ne s'applique qu'aux programmes de langage à relais.

Type de données RTC

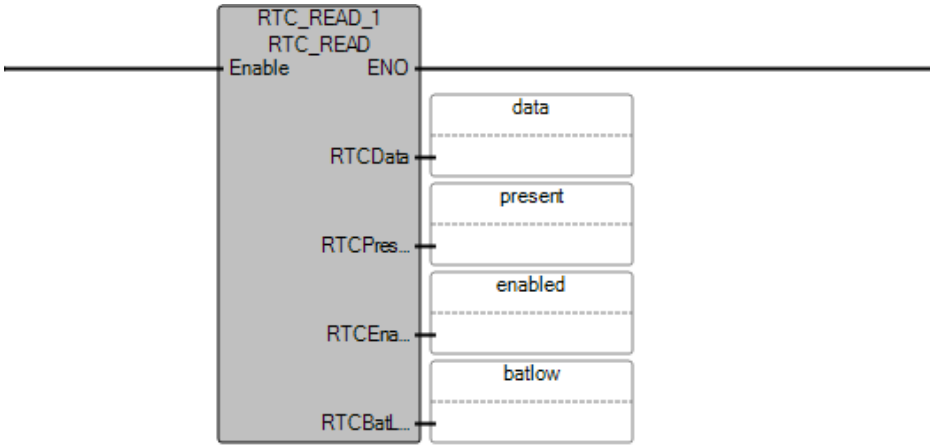
Ce tableau permet de déterminer les valeurs de paramètre pour le type de données RTC.

Paramètre	Type de données	Description
Year	UINT	Paramétrage de l'année pour l'horloge RTC. Valeur 16 bits ; les valeurs correctes sont comprises entre 2000 (01 Jan, 00:00:00) et 2098 (31 Déc, 23:59:59)
Month	UINT	Paramétrage du mois pour l'horloge RTC.
Day	UINT	Paramétrage du jour pour l'horloge RTC.
Hour	UINT	Paramétrage de l'heure pour l'horloge RTC.
Minute	UINT	Paramétrage de la minute pour l'horloge RTC.
Second	UINT	Paramétrage de la seconde pour l'horloge RTC.
DayOfWeek	UINT	Paramétrage du jour de la semaine pour l'horloge RTC. Ce paramètre n'est pas pris en compte pour RTC_SET.

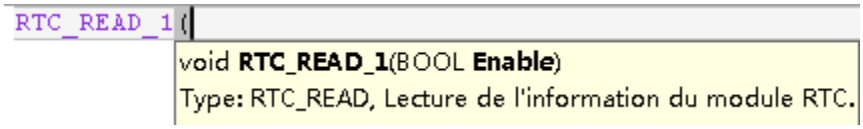
Exemple de diagramme de bloc de fonctions RTC_READ



Exemple de langage à relais RTC_READ



Exemple de texte structuré RTC_READ




```
1  RTC_READ_1(enable);
2  data := RTC_READ_1.RTCData;
3  present := RTC_READ_1.RTCPresent;
4  enabled := RTC_READ_1.RTCEnabled;
5  batlow := RTC_READ_1.RTCBatLow;
```

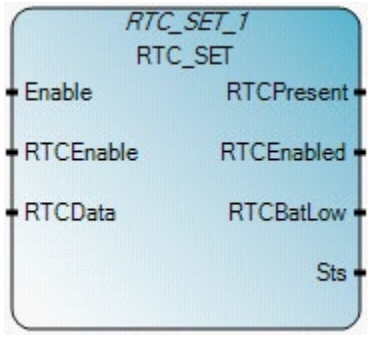
RTC_SET (définir l'horloge en temps réel)

Définit les données RTC (horloge en temps réel) pour les informations du module RTC.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

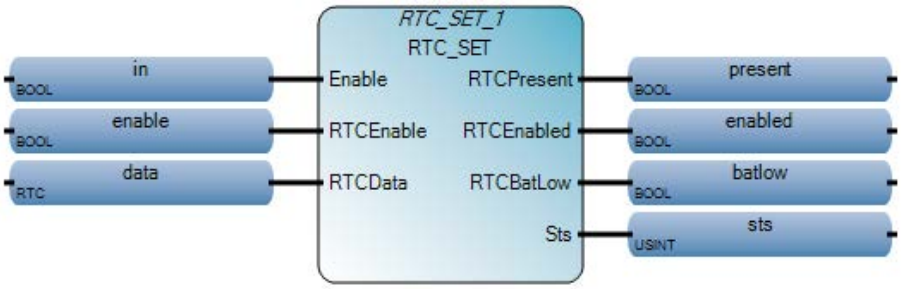
Paramètre	Type de paramètre	Type de données	Description
Enable	Entrée	BOOL	Activation du bloc d'instruction. VRAI : exécution de RTC_SET avec les informations RTC de l'entrée. En général, s'exécute uniquement pour 1 scrutation de programme lors de la mise à jour de RTC. FAUX : n'exécute pas RTC_SET. Défini sur FAUX pour exécuter RTC normalement.
RTCEnable	Entrée	BOOL	VRAI : activation de l'horloge RTC avec les données RTC spécifiées. FALSE - Désactivation de l'horloge RTC.
RTCData	Entrée	RTC	Informations sur les données RTC : aa/mm/jj, hh/mm/ss, semaine, tel que défini dans le type de données RTC. RTCData est ignoré lorsque RTCEnable = 0.
RTCPresent	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est branché. FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC n'est pas branché.
RTCEnabled	Sortie	BOOL	VRAI : l'horloge d'exécution libre est utilisée, ou le matériel RTC est activé (temporisation). FAUX : l'horloge d'exécution libre n'est pas utilisée, ou le matériel RTC est désactivé (aucune temporisation).
RTCBatLow	Sortie	BOOL	VRAI : le niveau de la batterie RTC est faible. FAUX : le niveau de la batterie RTC n'est pas faible.
Sts	Sortie	USINT	État de l'opération de lecture. Valeurs d'état RTC_Set (Sts) : • 0x00 - Bloc de fonctions non activé (aucune opération). • 0x01 - Réussite de l'opération de la définition RTC. • 0x02 - Échec de l'opération de la définition RTC.

Type de données RTC

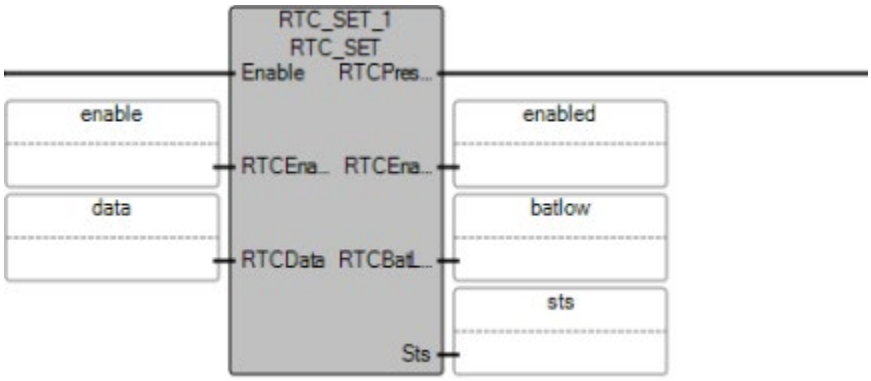
Ce tableau permet de déterminer les valeurs de paramètre pour le type de données RTC.

Paramètre	Type de données	Description
Year	UINT	Paramétrage de l'année pour l'horloge RTC. Valeur 16 bits ; les valeurs correctes sont comprises entre 2000 (01 Jan, 00:00:00) et 2098 (31 Déc, 23:59:59)
Month	UINT	Paramétrage du mois pour l'horloge RTC.
Day	UINT	Paramétrage du jour pour l'horloge RTC.
Hour	UINT	Paramétrage de l'heure pour l'horloge RTC.
Minute	UINT	Paramétrage de la minute pour l'horloge RTC.
Second	UINT	Paramétrage de la seconde pour l'horloge RTC.
DayOfWeek	UINT	Paramétrage du jour de la semaine pour l'horloge RTC. Ce paramètre n'est pas pris en compte pour RTC_SET.

Exemple de diagramme de bloc de fonctions RTC_SET



Exemple de langage à relais RTC_SET



Exemple de texte structuré RTC_SET

```
RTC_SET_1(
```

void **RTC_SET_1**(BOOL **Enable**, BOOL **RTCEnable**, RTC **RTCData**)
Type: RTC_SET, Fixe les données RTC au module RTC.

```
1  RTC_SET_1(in, enable, data);
2  present := RTC_SET_1.RTCPresent;
3  enabled := RTC_SET_1.RTCEnabled;
4  batlow := RTC_SET_1.RTCBatLow;
5  sts := RTC_SET_1.Sts;
```

Instructions de socket

Utilisez le protocole de sockets pour les communications Ethernet avec les appareils qui ne prennent pas en charge Modbus TCP et EtherNet/IP. Les sockets prennent en charge le client, le serveur, le protocole TCP et le protocole UDP. Les applications types incluent la communication avec des imprimantes, les lecteurs de code-barres et les PC.

Comportement de l'instruction de socket en mode **Modification en mode Exécuter** :

- Si une opération de suppression est effectuée sur une instruction SOCKET_OPEN, SOCKET_ACCEPT, SOCKET_READ ou SOCKET_WRITE en mode **Modification en mode Exécuter**, l'instance de socket est supprimée.
- En mode **Modification en mode Exécuter**, tout changement apporté à une entrée SOCKET_READ pendant une opération à l'état BUSY entraîne une erreur, et le paquet reçu est ignoré. Les paramètres d'entrée SOCKET_READ sont : Longueur, Décalage, Taille du tableau de données, Variable du tableau de données.
- SOCKET_READ est la seule instruction de SOCKET qui prend en charge l'ajout ou la modification des opérations en mode **Modification en mode Exécuter**.
- Si les paramètres Ethernet IP sont modifiés à l'aide d'une **Modification en mode Exécuter**, toutes les instances de socket créées sont supprimées, de la même manière qu'avec SOCKET_DELETEALL.

Traitement de l'instruction et mises à jour de la sortie pour les instructions de socket :

- **Asynchrone** : Instructions correspondantes où toutes les sorties sont mises à jour de manière asynchrone avec l'analyse du programme utilisateur, par exemple une analyse de schéma à relais. Une sortie asynchrone ne peut pas être utilisée pour détecter un front de déclenchement. Les paramètres d'une sortie asynchrone ne sont pas verrouillés et peuvent être mis à jour à la fin des instructions de support correspondantes.
- **Synchrone** : Instructions correspondantes où toutes les sorties sont mises à jour de manière synchrone avec les scrutations du programme utilisateur. Les paramètres d'une sortie synchrone sont verrouillés et ne peuvent pas être modifiés à la fin d'une instruction de socket correspondante.

- **Hybride** : Instructions correspondantes où quelques sorties sont mises à jour de manière synchrone avec l'analyse du programme utilisateur. Les autres sorties sont mises à jour de manière asynchrone avec l'analyse du programme utilisateur.
- **Exécution immédiate d'une instruction** : L'instruction termine la fonction souhaitée avant de passer à l'instruction suivante.
- **Exécution non immédiate d'une instruction** : L'instruction nécessite plusieurs analyses du programme pour terminer la fonction souhaitée. Les instructions réalisent une capture des paramètres d'entrée lorsqu'une transition **Faux > Vrai** est détectée.

Utilisez cette table pour déterminer l'utilisation des instructions de socket.

Instruction	Description	Client TCP	Serveur TCP	UDP avec ouverture	UDP sans ouverture	Traitement des instructions	Mise à jour de la sortie des instructions
SOCKET_ACCEPT sur la page 568	Accepte une demande de connexion TCP à partir d'une destination à distance et renvoie une instance de socket utilisée pour envoyer et recevoir des données sur la connexion nouvellement créée.	NO	OUI	NO	NO	Non immédiat	Hybride
SOCKET_CREATE sur la page 571	Crée une instance du support et renvoie un numéro d'instance qui utilise les opérations de support suivantes.	OUI	OUI	OUI	OUI	Immédiat	Synchrone
SOCKET_DELETE sur la page 576	Supprime une instance de socket créée. Les connexions TCP sont fermées avant la suppression.	OUI	OUI	OUI	OUI	Non immédiat	Synchrone
SOCKET_DELETEALL sur la page 578	Supprime toutes les instances de socket créées.	OUI	OUI	OUI	OUI	Non immédiat	Synchrone
SOCKET_INFO sur la page 580	Renvoie des informations sur le socket, comme des codes d'erreur et des états d'exécution.	OUI	OUI	OUI	OUI	Immédiat	Synchrone
SOCKET_OPEN sur la page 586	Les connexions TCP s'ouvrent avec l'adresse de destination indiquée. Les connexions UDP associent une adresse IP de destination et un numéro de port au support indiqué.	OUI	NO	OUI	NO	Non immédiat	Synchrone
SOCKET_READ sur la page 590	Lit des données sur un socket. Tente de recevoir le nombre d'octets spécifié et renvoie le nombre d'octets reçus.	OUI	OUI	OUI	OUI	Non immédiat	Hybride
SOCKET_WRITE sur la page 594	Envoie des données sur un socket. Tente d'envoyer le nombre d'octets demandé et renvoie le nombre d'octets envoyés.	OUI	OUI	OUI	OUI	Non immédiat	Hybride

SOCKET_ACCEPT

Pour les connexions TCP uniquement. Accepte une demande de connexion TCP à partir d'une destination à distance et renvoie une instance de socket utilisée pour envoyer et recevoir des données sur la connexion nouvellement créée.

Détails de l'opération :

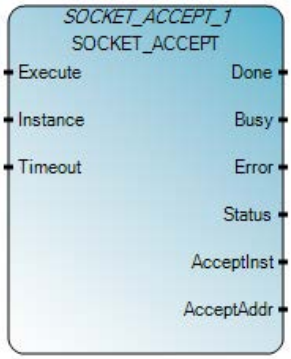
- Avant d'exécuter Socket_Accept, exécutez SOCKET_CREATE et spécifiez le numéro de port local pour accepter la connexion.

- Les sorties sont mises à jour de manière synchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).

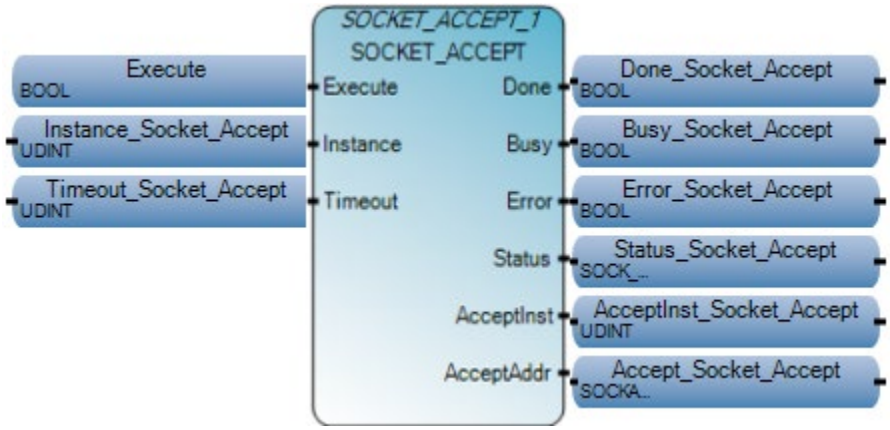


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

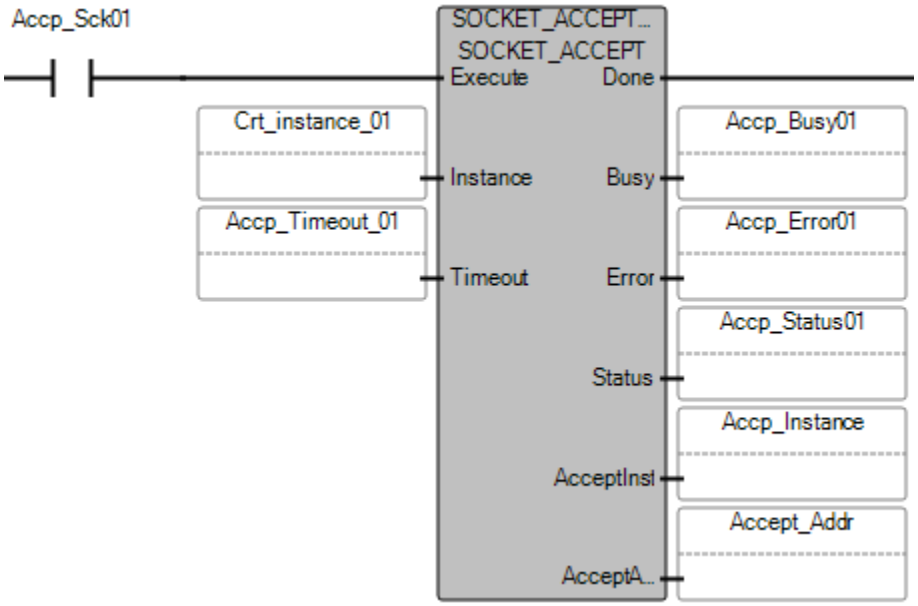
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FALSE : aucun front montant détecté, bloc d'instruction non démarré.
Instance	Entrée	UDINT	Identifie l'instance de socket. Copiez le gestionnaire de socket renvoyé depuis l'instruction **SOCKET_CREATE**.
Timeout	Entrée	UDINT	Temps dépassé pour les instances du bloc d'instruction **SOCKET_ACCEPT**. Le bloc de fonctions renvoie une erreur si la valeur du paramètre Timeout est inférieure à la valeur minimale. Plage du paramètre Timeout : 1 000 à 86 400 000 millisecondes Définissez le temps dépassé sur 0 pour utiliser la valeur par défaut 10 000 (10 secondes).
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.

Paramètre	Type de paramètre	Type de données	Description
AcceptInst	Sortie	UDINT	Contient l'instance Accept pour cette instance de socket. Utilisez le numéro d'instance Accept unique pour les opérations **SOCKET_READ** et **SOCKET_WRITE** ultérieures pour cette connexion. La sortie est mise à jour de manière synchrone par l'analyse du programme.
AcceptAddr	Sortie	SOCKADDR_CFG	Une structure de données qui contient l'adresse Accept pour le socket. Pour plus d'informations, reportez-vous au type de données SOCKADDR_CFG sur la page 598. Pour spécifier l'adresse IP 192.168.2.100 et le port = 12000 : • AcceptAddr.IPAddress[0]=192 • AcceptAddr.IPAddress[1]=168 • AcceptAddr.IPAddress[2]=2 • AcceptAddr.IPAddress[3]=100 • AcceptAddr.Port = 12000 La sortie est mise à jour de manière synchrone par l'analyse du programme.

Exemple de diagramme de bloc de fonctions SOCKET_ACCEPT



Exemple de langage à relais SOCKET_ACCEPT



Exemple de texte structuré SOCKET_ACCEPT

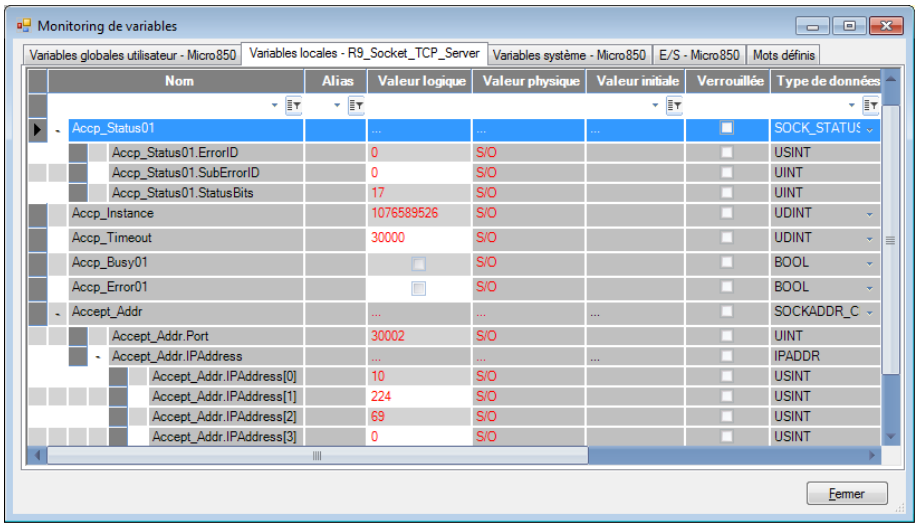
```
SOCKET_ACCEPT_1(
```

void **SOCKET_ACCEPT_1**(BOOL **Execute**, UDINT **Instance**, UDINT **Timeout**)
Type: SOCKET_ACCEPT, Accepter le socket

```
SOCKET_ACCEPT_1(Execute_SOCKET_ACCEPT, Instance _SOCKET_ACCEPT, Timout_SOCKET_ACCEPT);
Done_SOCKET_ACCEPT :- SOCKET_ACCEPT. Done;
Busy_SOCKET_ACCEPT :- SOCKET_ACCEPT. Busy;
Error_SOCKET_ACCEPT :- SOCKET_ACCEPT. Error;
Status_SOCKET_ACCEPT :- SOCKET_ACCEPT. Status;
AcceptInst_SOCKET_ACCEPT :- SOCKET_ACCEPT. AcceptInst;
AcceptAddr_SOCKET_ACCEPT :- SOCKET_ACCEPT. AccptAddr;
```

Résultats

Serveur TCP



SOCKET_CREATE

Crée une instance du socket et renvoie un numéro d'instance qui est utilisé comme entrée dans toutes les opérations de socket suivantes.

Détails de l'opération :

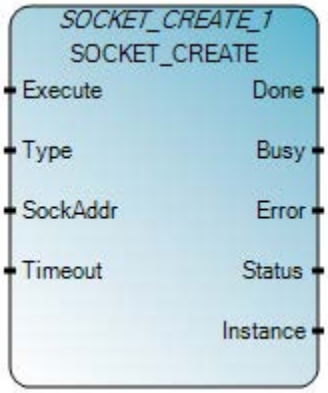
- Les instructions de socket sur la page 567 prennent en charge la communication en duplex intégral avec les appareils distants.
- Les connexions UDP prennent en charge un maximum de huit paquets de datagrammes UDP en file d'attente. La file d'attente contient les paquets les plus récents.
- Les contrôleurs Micro820 et Micro850 de la version 9 ou d'une version ultérieure prennent en charge jusqu'à huit instances de socket. Les instances de socket prennent en charge les sockets UDP et TCP :
  - Utilisez les huit instances pour les connexions aux clients TCP.
  - Utilisez les huit instances pour écouter les connexions TCP entrantes, puis acceptez huit connexions d'autres appareils.
  - Exécutez les opérations serveur et client TCP.
  - Exécutez les opérations TCP et UDP.

- Pour accepter les connexions TCP entrantes, sur le même port, créez une nouvelle instance de socket.
- Perte de connexion TCP :
  - Le programme d'application de l'utilisateur doit détecter la perte de connexion TCP et gérer l'événement. En fonction de l'application de l'utilisateur, envisagez l'option de défaut du contrôleur.
    - Défaut du contrôleur.
    - Essayez de rétablir la connexion.
  - Pour rétablir la communication avec un autre appareil :
    - Supprimez l'instance de socket pour la connexion perdue.
    - Si la connexion est un client TCP, créez une nouvelle instance de socket avec SOCKET_CREATE, puis exécutez SOCKET_OPEN sur l'appareil cible.
    - Si la connexion est un serveur TCP, créez une nouvelle instance de socket avec SOCKET_CREATE, puis exécutez SOCKET_ACCEPT pour attendre une autre connexion de l'appareil distant.
  - Messages de l'application pour les connexions TCP :
    - Une connexion TCP est un flux d'octets entre deux applications. Le protocole de l'application détermine les formats de message.
    - Les messages peuvent avoir une taille fixe ou variable.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).

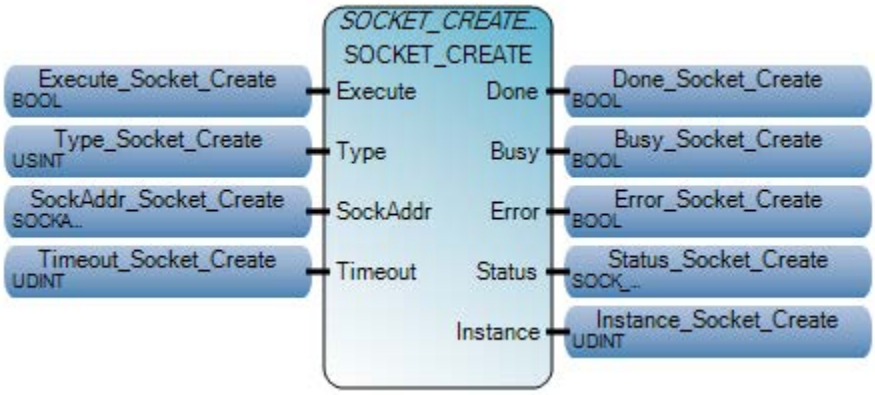


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

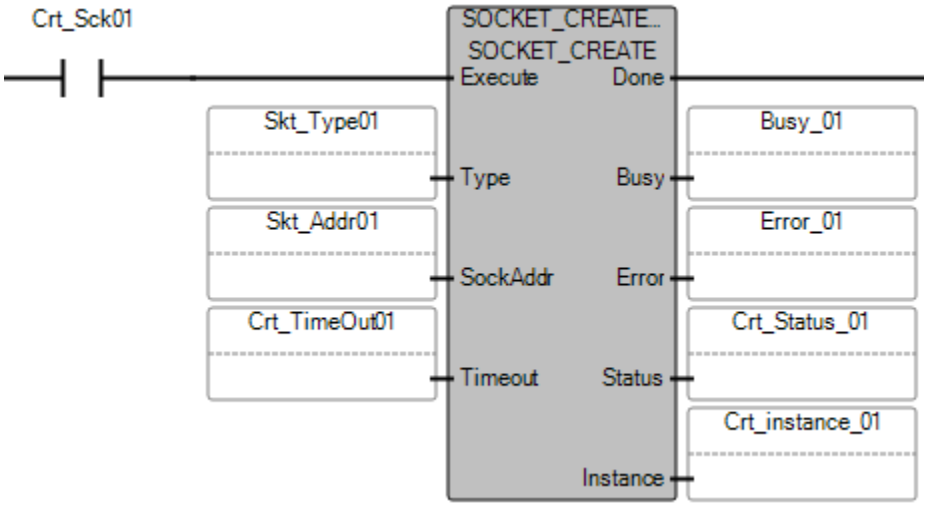
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : aucun front montant détecté.
Type	Entrée	USINT	Spécifiez le type de socket : • Protocole TCP (Transmission Control Protocol). • Protocole UDP (User Datagram Protocol).
SockAddr	Entrée	SOCKADDR_CFG	Spécifie la configuration d'adresse du socket. Module EtherNet/IP pour choisir le numéro de port local ; SockAddr défini sur 0. Indiquez le numéro de port local sur lequel une application écoute et reçoit, ou : • Les éléments du tableau doivent tous être 0 • Pour les opérations de client TCP, spécifiez 0, sauf si vous souhaitez un numéro de port local spécifique. • Pour la communication de serveur TCP, spécifiez le numéro de port pour accepter des demandes de connexions entrantes. • Pour UDP, spécifiez un numéro de port local pour recevoir des datagrammes sur un port spécifique. Plage de ports locaux : 1 à 65535. Une erreur se produit si le numéro de port local est déjà utilisé par le contrôleur Micro820 ou Micro850. Le contrôleur utilise les numéros de port suivants : Ports TCP : • EtherNet/IP : 44818 • ModbusTCP : 502 • Serveur DHCP : 67 • Client DHCP : 68 Ports UDP : • EtherNet/IP : 2222 • Serveur DHCP : 67 • Client DHCP : 68 Voir Type de données SOCKADDR_CFG sur la page 598.
Timeout	Entrée	UDINT	Indique le temps dépassé pour l'inactivité du socket. Si une instance de socket ne reçoit pas de demandes dans le délai d'inactivité spécifié dans le paramètre de temps dépassé, l'instance de socket est supprimée. Si une demande est envoyée après la suppression de l'instance de socket, l'erreur « Socket instance not supported » (Instance de socket non prise en charge) est renvoyée. Le bloc d'instruction renvoie une erreur si la valeur de temps dépassé est inférieure à la valeur minimale. Définissez le temps dépassé pour qu'il soit supérieur à l'intervalle le plus long entre des opérations de socket. Si la valeur de Timeout pour l'inactivité est trop court, des instances de socket risquent d'être arrêtées pour inactivité. Plage du paramètre Timeout : 1000 à 86400000 millisecondes Définissez le **Temps dépassé** sur 0 pour utiliser la valeur par défaut de 300 000 (5 minutes).
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur.

Paramètre	Type de paramètre	Type de données	Description
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données **SOCK_STATUS** qui contient les informations sur ErrorID, SubErrorID et StatusBits. Voir Type de données SOCK_STATUS sur la page 598, Bits d'état d'instruction de socket sur la page 602 et Codes d'erreur du socket sur la page 598.
Instance	Sortie	UDINT	Contient un gestionnaire de socket. Utilisez le paramètre Instance pour les instructions Socket ultérieures.

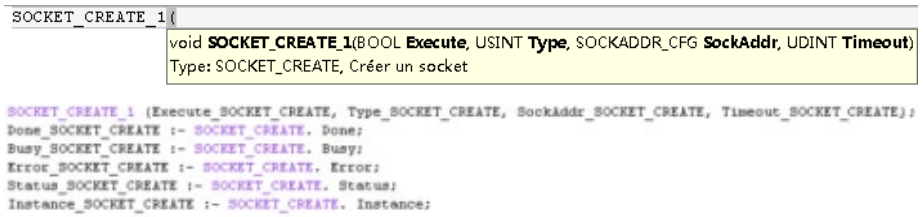
Exemple de diagramme de bloc de fonctions SOCKET_CREATE



Exemple de langage à relais SOCKET_CREATE

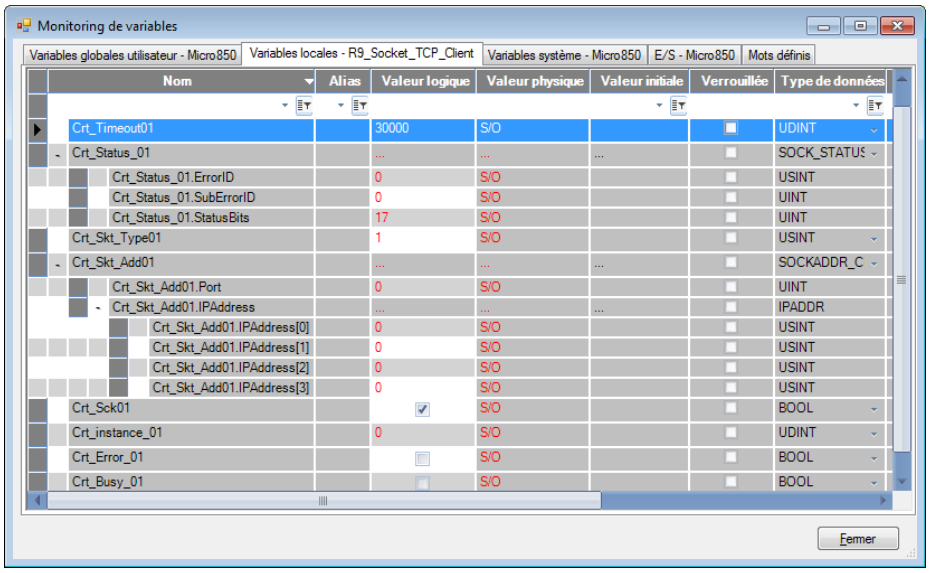


Exemple de texte structuré SOCKET_CREATE

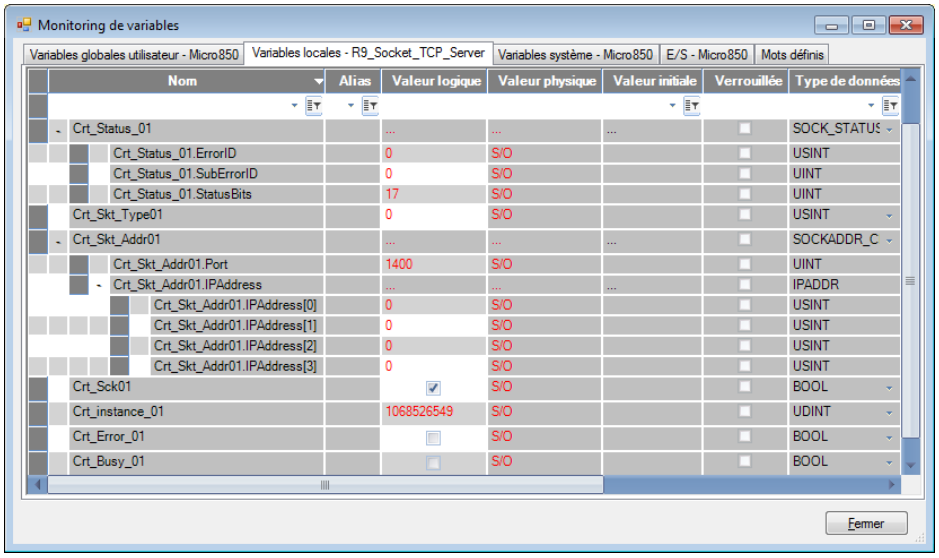


Résultats

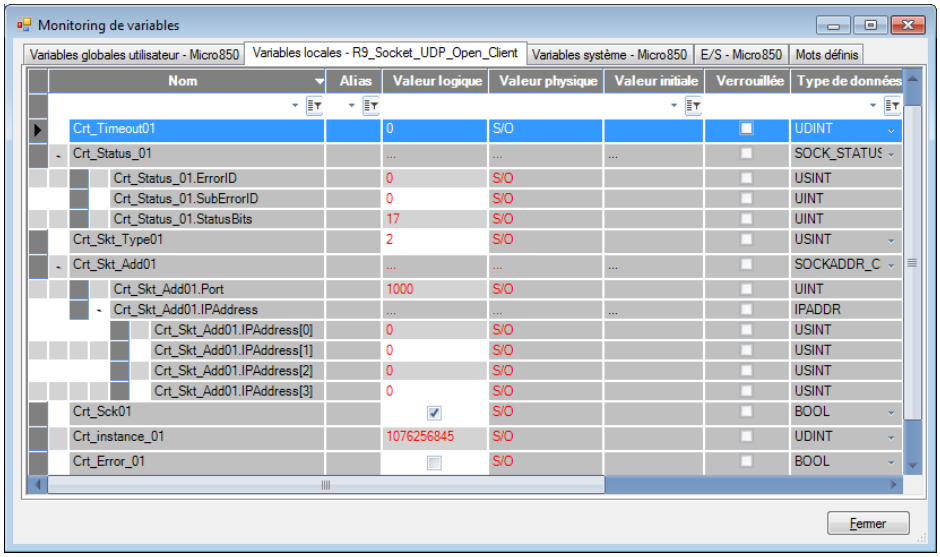
Exemple de client TCP



Exemple de serveur TCP



Exemple d'UDP



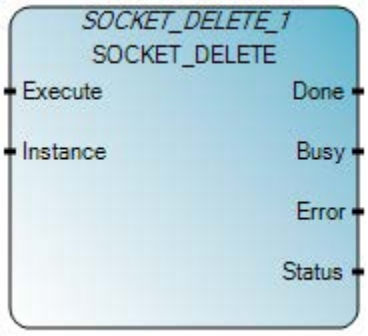
SOCKET_DELETE

Supprime une instance de socket créée. Pour les connexions TCP (Transmission Control Protocol), **SOCKET_DELETE** ferme également (fermeture passive) la connexion avant de supprimer l'instance. Les sorties sont mises à jour de manière synchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).

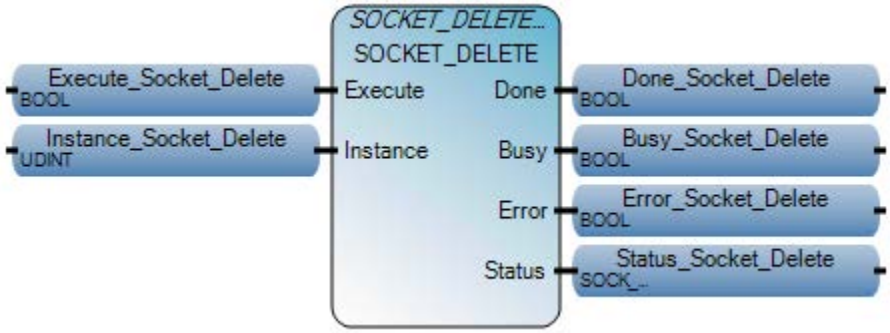


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

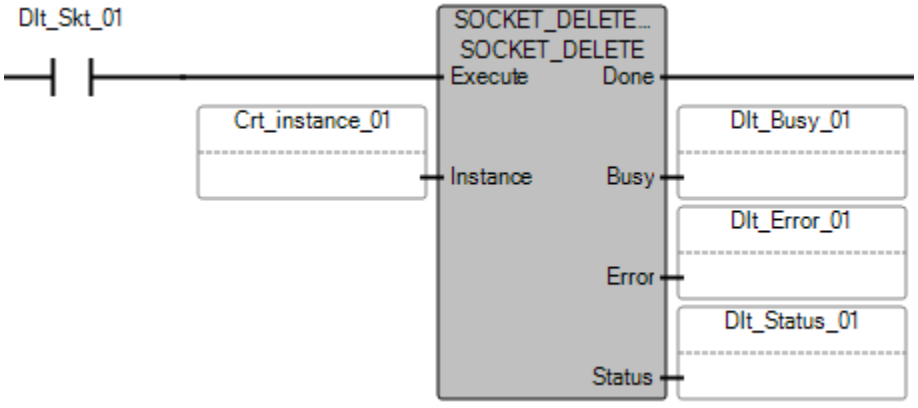
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : aucun front montant détecté.

Paramètre	Type de paramètre	Type de données	Description
Instance	Entrée	UDINT	Copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE** ou **SOCKET_ACCEPT** pour supprimer le socket respectif. • Pour les types de socket de client UDP et TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE**. • Pour le type de socket de serveur TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_ACCEPT**.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.

Exemple de diagramme de bloc de fonctions SOCKET_DELETE



Exemple de langage à relais SOCKET_DELETE



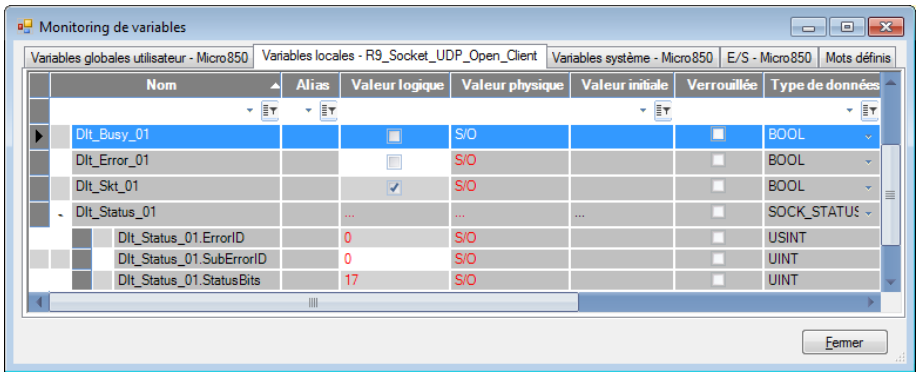
Exemple de texte structuré SOCKET_DELETE

```
SOCKET_DELETE_1(
```

void **SOCKET_DELETE_1**(BOOL **Execute**, UDINT **Instance**)
Type: SOCKET_DELETE, Supprimer un socket

```
SOCKET_DELETE_1 (Execute_SOCKET_DELETE, Instance_SOCKET_DELETE);
Done_SOCKET_DELETE := SOCKET_DELETE. Done;
Busy_SOCKET_DELETE := SOCKET_DELETE. Busy;
Error_SOCKET_DELETE := SOCKET_DELETE. Error;
Status_SOCKET_DELETE := SOCKET_DELETE. Status;
```

Résultats



SOCKET_DELETEALL

Supprime toutes les instances de socket créées.

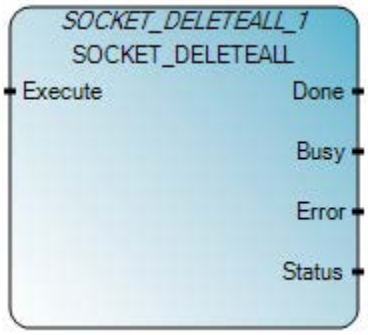
Détails de l'opération :

- Si le câble Ethernet est déconnecté du contrôleur ou si l'adresse IP du contrôleur est modifiée, vous pouvez exécuter SOCKET_DELETEALL pour supprimer toutes les instances de socket précédemment créées.
- Les sorties sont mises à jour de manière synchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

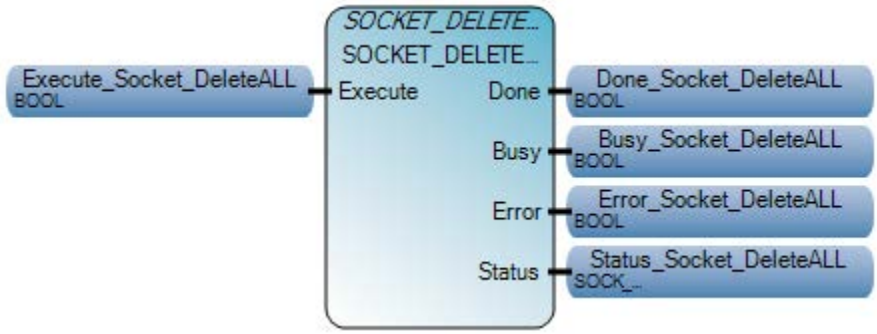
Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



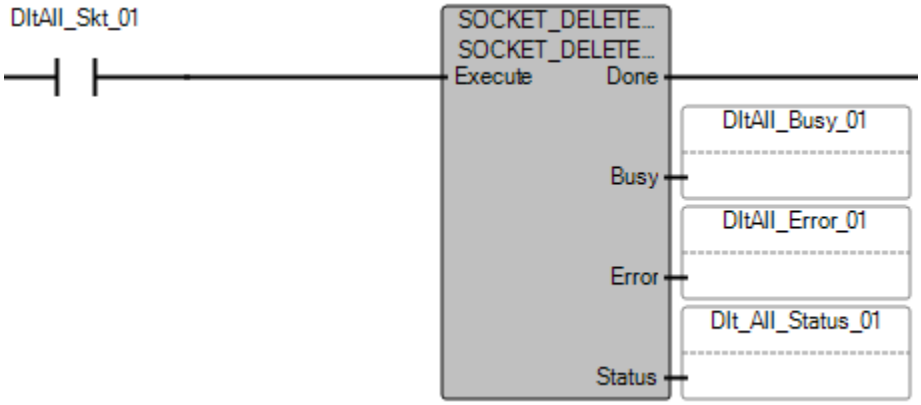
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : aucun front montant détecté.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Ce champ est défini sur « TRUE » lorsque l'exécution du bloc de fonctions rencontre une erreur. Pour en savoir plus, reportez-vous aux codes d'erreur des sockets sur la page 598. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.

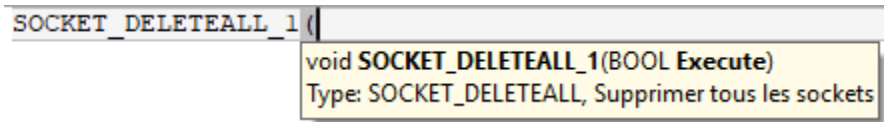
Exemple de diagramme de bloc de fonctions SOCKET_DELETEALL



Exemple de langage à relais SOCKET_DELETEALL

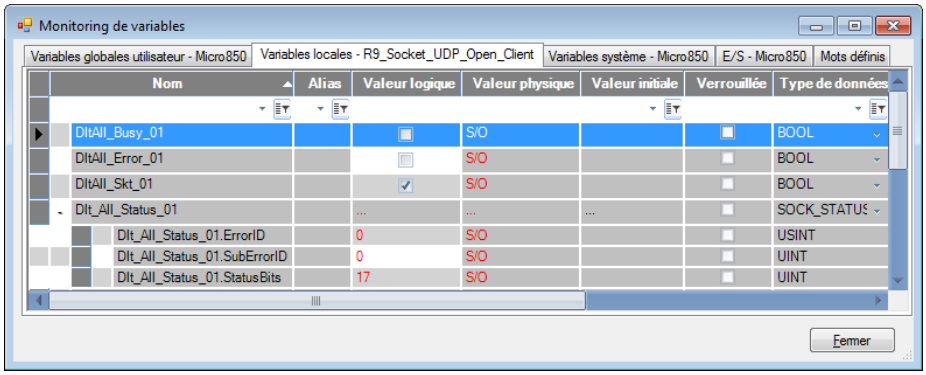


Exemple de texte structuré SOCKET_DELETEALL




```
SOCKET_DELETEALL (Execute_SOCKET_DELETEALL);
Done_Socket_DeleteAll := SOCKET_DELETE. Done;
Busy_Socket_DeleteAll := SOCKET_DELETE. Busy;
Error_Socket_DeleteAll := SOCKET_DELETE. Error;
Status_Socket_DeleteAll := SOCKET_DELETE. Status;
```

Résultats



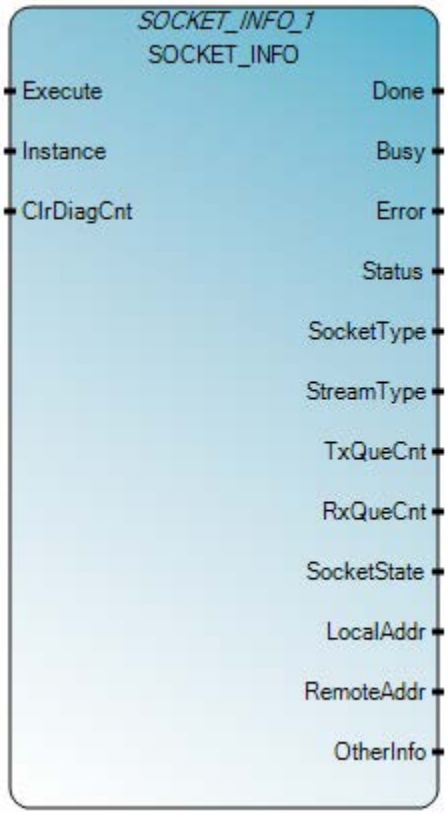
SOCKET_INFO

Renvoie des informations pour une instance de socket, comme les codes d'erreur et les états d'exécution. Les sorties sont mises à jour de manière synchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



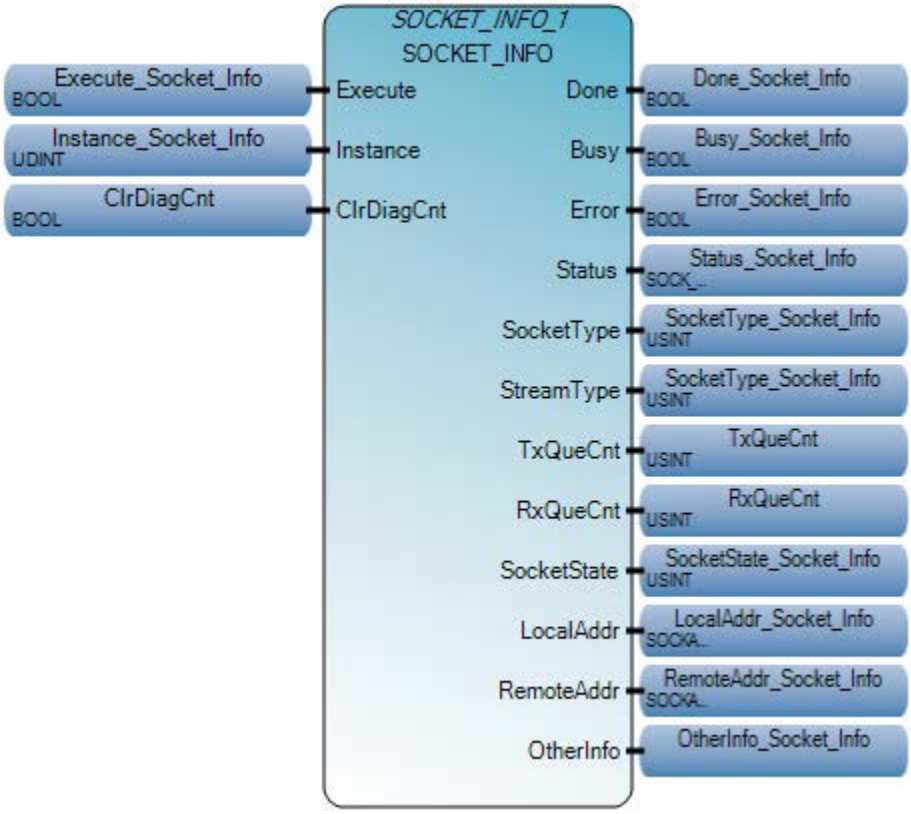
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : aucun front montant détecté.
Instance	Entrée	UDINT	Copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE** ou **SOCKET_ACCEPT** pour supprimer le socket respectif. • Pour les types de socket de client UDP et TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE**. • Pour le type de socket de serveur TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_ACCEPT**. Lorsque l'instance est définie sur 0, un récapitulatif de toutes les instances de socket est renvoyé.
ClrDiagCnt	Entrée	BOOL	VRAI : suppression des informations de compteur de diagnostic TCP (Transmission Control Protocol) ou UDP (User Datagram Protocol). FAUX : les informations de compteur TCP ou UDP ne sont pas supprimées. Comme pour TCP et UDP, index du tableau **OtherInfo** 1 à 6. Lorsque l'Instance est définie sur 0, effacez les index du tableau **OtherInfo** 7 à 14.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée.

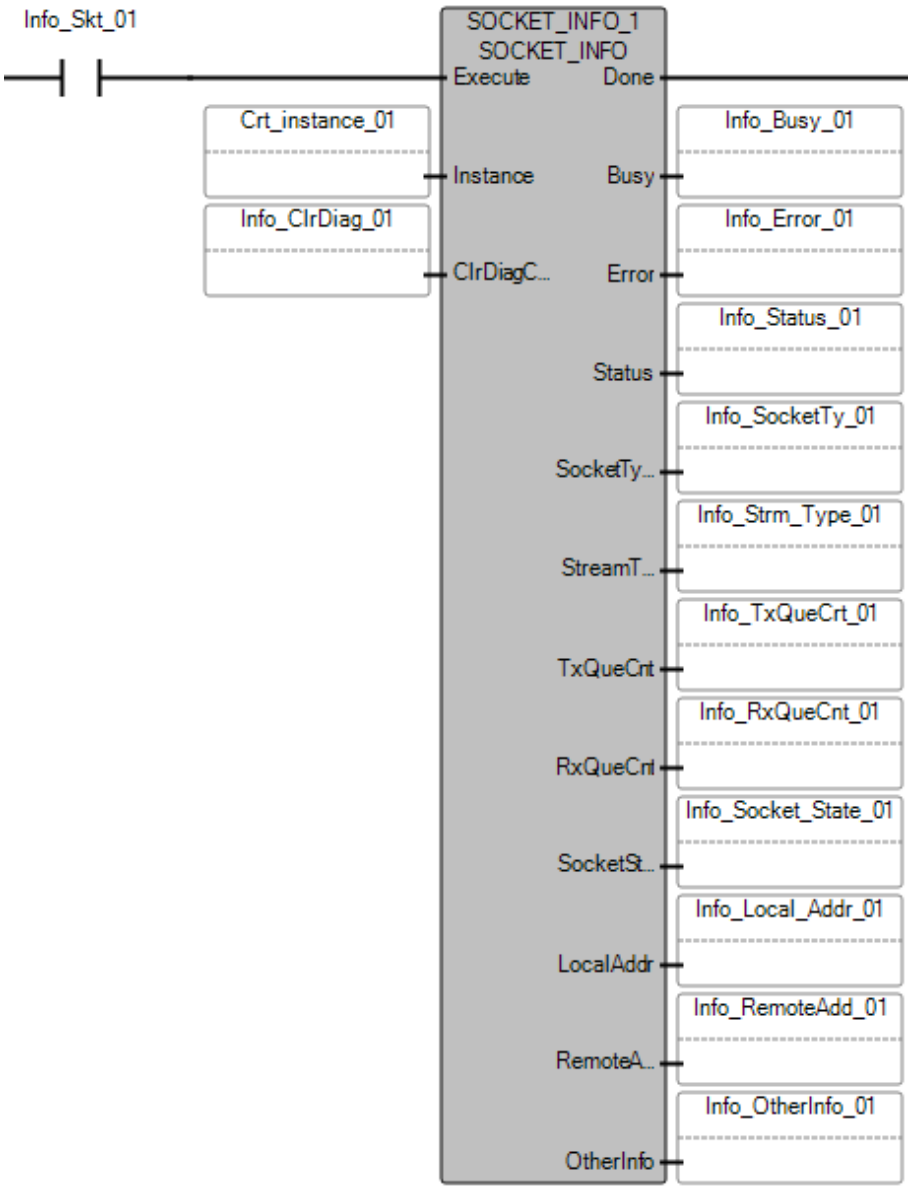
Paramètre	Type de paramètre	Type de données	Description
			FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.
SocketType	Sortie	USINT	Type d'instance de socket : • 0 - Non utilisée • 1 - TCP • 2 - UDP Lorsque l'**Instance** Socket_Info est définie sur 0, **SocketType** s'affiche comme 0.
StreamType	Sortie	USINT	Type de flux de socket : • 0 - Aucun • 1 - Serveur TCP • 2 - Client TCP Lorsque l'**Instance** Socket_Info est définie sur 0, **StreamType** s'affiche comme 0.
TxQueCnt	Sortie	USINT	Nombre de messages Tx actuellement dans la file d'attente. Lorsque l'**Instance** Socket_Info est définie sur 0, **TxQueCnt** s'affiche comme 0.
RxQueCnt	Sortie	USINT	Nombre de messages Rx actuellement dans la file d'attente. Lorsque l'**Instance** Socket_Info est définie sur 0, **RxQueCnt** s'affiche comme 0.
SocketState	Sortie	USINT	Information d'état d'instruction du socket Pour en savoir plus, reportez-vous aux états de machine du socket. Lorsque l'**Instance** Socket_Info est définie sur 0, **SocketState** s'affiche comme 0.
LocalAddr	Sortie	SOCKETADDR_CFG	Adresse locale pour le socket. Pour plus d'informations, reportez-vous au type de données SOCKADDR_CFG sur la page 598. Lorsque l'**Instance** Socket_Info est définie sur 0, **LocalAddr** s'affiche comme 0.
RemoteAddr	Sortie	SOCKETADDR_CFG	Adresse distante pour le socket. Pour plus d'informations, reportez-vous au type de données SOCKADDR_CFG sur la page 598. **RemoteAddr** s'affiche comme 0 dans les cas suivants : • L'**Instance** Socket_Info est définie sur 0 • Connexions UDP (User Datagram Protocol) sans **SOCKET_OPEN** • UDP avec **SOCKET_OPEN** et **RxFilter** désactivé.
OtherInfo		UDINT[1..15]	L'**Instance** de socket est configurée comme TCP et la description de l'index du tableau est : • 1 - Paquets envoyés : nombre total de paquets TCP envoyés sur un socket. • 2 - Octets envoyés : nombre total d'octets TCP envoyés sur un socket. • 3 - Paquets reçus : nombre total de paquets TCP reçus sur un socket. • 4 - Octets reçus : nombre total d'octets TCP reçus sur un socket. • 5 - Paquets retransmis : nombre total de retransmissions de paquets TCP. • 6 - Erreurs de somme de contrôle : nombre total de paquets TCP contenant des erreurs de somme de contrôle sur un socket. • 7 - État TCP : état actuel d'un socket. • (8 à 11) - OtherInfo n'est pas pris en charge par TCP, il s'affiche comme 0. • 12 ,13,14,15 - S'affiche comme 0. Instance de socket configurée comme UDP, la description de l'index du tableau est : • 1 - Paquets envoyés : nombre total de paquets UDP envoyés sur un socket. • 2 - Octets envoyés : nombre total d'octets UDP envoyés sur un socket. • 3 - Paquets reçus : nombre total de paquets UDP reçus sur un socket.

Paramètre	Type de paramètre	Type de données	Description
			• 4 - Octets reçus : nombre total d'octets UDP reçus sur un socket. • 5 - Paquet abandonnés : nombre total de paquets UDP reçus abandonnés pour un socket en raison d'un dépassement de la taille maximale de la file d'attente (8). • 6 - Erreurs de somme de contrôle : nombre total de paquets UDP contenant des erreurs de somme de contrôle sur un socket. • 7 à 15 - S'affiche comme 0. Instance de socket configurée comme 0, la description de l'index du tableau est : • 1 - Nombre pour l'instance de socket disponible. Nombre maximal de sockets pris en charge. • 2 - Nombre d'instances de socket utilisées. Nombre de sockets créés. • 3 - Nombre d'instances de socket créées comme TCP. • 4 - Nombre d'instances de socket créées comme client TCP. • 5 - Nombre d'instances de socket créées comme serveur TCP. • 6 - Nombre d'instances de socket créées comme UDP. • 7 - **SOCKET_READ** Nombre de réussites lorsque l'instance de socket est configurée comme TCP. • 8 - **SOCKET_WRITE** Nombre de réussites lorsque l'instance de socket est configurée comme TCP. • 9 - **SOCKET_READ** Nombre d'échecs lorsque l'instance de socket est configurée comme TCP. • 10 - **SOCKET_WRITE** Nombre d'échecs lorsque l'instance de socket est configurée comme TCP. • 11 - **SOCKET_READ** Nombre de réussites lorsque l'instance de socket est configurée comme UDP. • 12 - **SOCKET_WRITE** Nombre de réussites lorsque l'instance de socket est configurée comme UDP. • 13 - **SOCKET_READ** Nombre d'échecs lorsque l'instance de socket est configurée comme UDP. • 14 - **SOCKET_WRITE** Nombre d'échecs lorsque l'instance de socket est configurée comme UDP. • 15 - S'affiche comme 0.

Exemple de diagramme de bloc de fonctions SOCKET_INFO



Exemple de langage à relais SOCKET_INFO



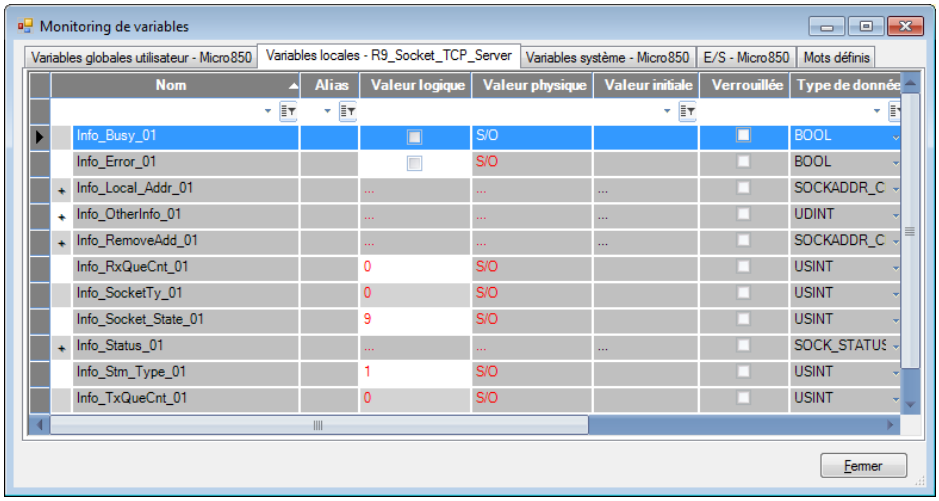
Exemple de texte structuré SOCKET_INFO

```
SOCKET_INFO_1(
```

void **SOCKET_INFO_1**(BOOL **Execute**, UDINT **Instance**, BOOL **ClrDiagCnt**)
Type: SOCKET_INFO, Informations sur l'instance de socket

```
SOCKET_INFO_1 (Execute_SOCKET_INFO, Instance_SOCKET_INFO, ClrDiagC_SOCKET_INFO);
Done_SOCKET_INFO := SOCKET_INFO. Done;
Busy_SOCKET_INFO := SOCKET_INFO. Busy;
Error_SOCKET_INFO := SOCKET_INFO. Error;
Status_SOCKET_INFO := SOCKET_INFO. Status;
SocketType_SOCKET_INFO := SOCKET_INFO. SocketTy;
StreamType_SOCKET_INFO := SOCKET_INFO. StreamType;
TxQueCnt_SOCKET_INFO := SOCKET_INFO. TxQueCnt;
RxQueCnt_SOCKET_INFO := SOCKET_INFO. RxQueCnt;
SocketState_SOCKET_INFO := SOCKET_INFO. SocketState;
LocalAddr_SOCKET_INFO := SOCKET_INFO. LocalAddr;
RemoteAddr_SOCKET_INFO := SOCKET_INFO. RemoteAddr;
OtherInfo SOCKET INFO := SOCKET INFO. OtherInfo;
```

Résultats



SOCKET_OPEN

Ouvre la connexion pour l'adresse de destination spécifiée pour les connexions TCP (Transmission Control Protocol). Pour les connexions UDP (User Datagram Protocol),associe une adresse IP de destination et un numéro de port au socket spécifié.

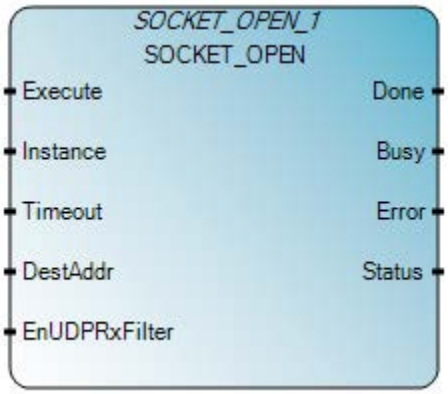
Détails de l'opération :

- Pour les connexions UDP (User Datagram Protocol) utilisant **SOCKET_OPEN**, l'adresse IP et le numéro de port ne sont pas nécessaires à chaque envoi de données.
- Pour les connexions UDP n'utilisant pas **SOCKET_OPEN**, l'adresse de destination doit être spécifiée à chaque utilisation de **SOCKET_WRITE** pour l'envoi de données.
- Lors de l'utilisation de **SOCKET_READ**, outre les données reçues, l'adresse de l'émetteur est renvoyée. L'adresse de l'émetteur peut être utilisée pour envoyer une réponse avec **SOCKET_WRITE**.
- Une opération **SOCKET_OPEN** peut être renvoyée avant la période de temps dépassé sans création d'une connexion TCP (Transmission Control Protocol). Si l'appareil de destination est connecté, mais qu'il n'écoute pas les connexions sur le numéro de port spécifié, **SOCKET_OPEN** est renvoyé avec une erreur avant l'expiration du temps dépassé.
- Les sorties sont mises à jour de manière synchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).

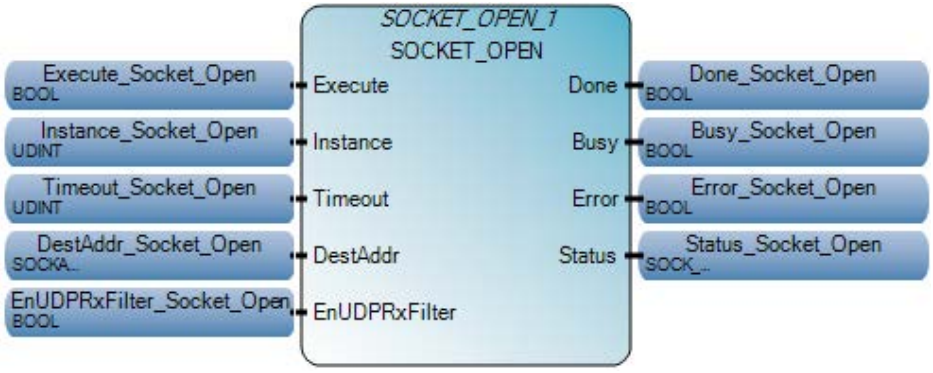


Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

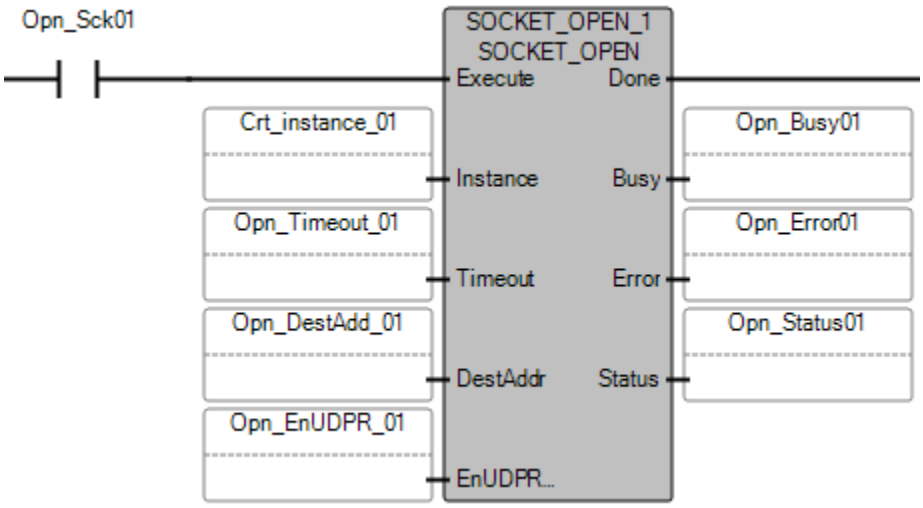
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : aucun front montant détecté.
Instance	Entrée	UDINT	Copiez le Gestionnaire de socket renvoyé depuis un bloc de fonctions **SOCKET_CREATE**.
Timeout	Entrée	UDINT	Temps dépassé pour le bloc de fonctions SOCKET_OPEN. Le bloc de fonctions renvoie une erreur si la valeur de temps dépassé est inférieure à la valeur minimale. Plage du paramètre Timeout : 1000 à 1800000 millisecondes Définissez le temps dépassé sur 0 pour utiliser la valeur par défaut de 10 000 (10 secondes).
DestAddr	Entrée	SOCKADDR_CFG sur la page 598	L'adresse de la connexion de destination. Une connexion est requise entre l'adresse IP et le numéro de port de l'hôte distant. Les adresses IP suivantes ne sont pas prises en charge par DestAddr : • Adresse IP de réponse • Adresse de boucle • 0.0.0.0 • Diffuse l'adresse (uniquement pris en charge pour TCP), adresses d'exception : • Adresse multicast de classe D (224.x.x.x) • Adresse de liaison locale (169.254.x.x) Exemple d'adresse IP 192.168.2.100 et de port 12000 : • DestAddr.IPAddress[0]=192 • DestAddr.IPAddress[1]=168 • DestAddr.IPAddress[2]=2 • DestAddr.IPAddress[3]=100 • DestAddr.Port = 12000
EnUDPRxFilter	Entrée	BOOL	Pour le socket UDP, lorsque vous utilisez SOCKET_OPEN et Enable EnUDPRxFilter, un filtre de paquets pour l'adresse IP et le numéro de port spécifiés n'est pas requis à chaque lecture de données. • Un socket UDP avec ouverture est créé. DestAddr comme IP 192.168.1.157 / Port 161.

Paramètre	Type de paramètre	Type de données	Description
			EnUDPRxFilter Enable, exécutez Socket_Read. • Lorsque le contrôleur reçoit des données de DestAddr (192.168.1.157 / 161), Socket_Read termine l'opération. Si le contrôleur reçoit des données d'une autre adresse IP ou d'un autre port, Socket_Read ignore ce paquet et attend un paquet DestAddr. EnUDPRxFilter désactivé : • Exécutez SOCKET_READ. Le contrôleur reçoit des données sur la configuration du port UDP, SOCKET_READ termine l'opération. • L'application vérifie si le paquet entrant arrive de l'appareil attendu ou pas. • L'application gère le filtrage en fonction du paramètre de sortie SOCKET_READ, UDPAddr.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.

Exemple de diagramme de bloc de fonctions SOCKET_OPEN



Exemple de langage à relais SOCKET_OPEN



Exemple de texte structuré SOCKET_OPEN

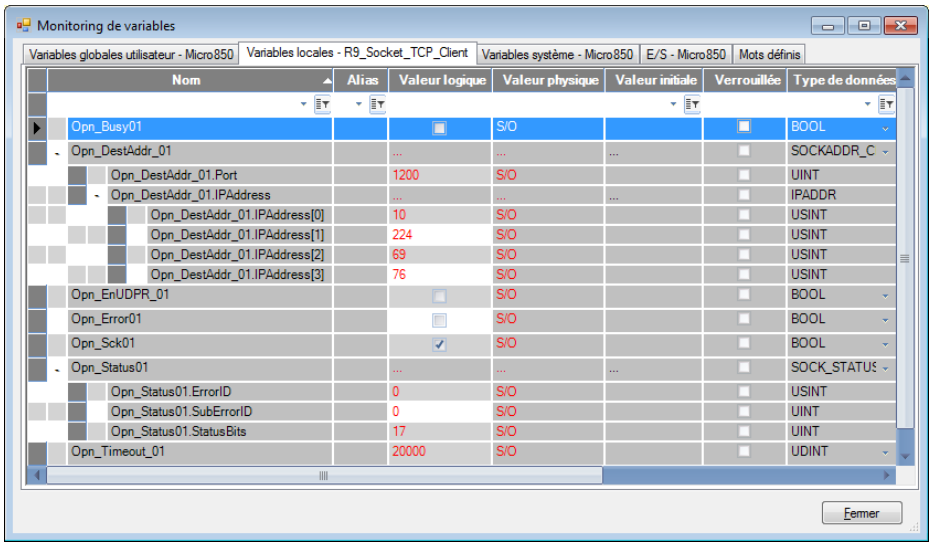
```
SOCKET_OPEN_1(
```

void **SOCKET_OPEN_1**(BOOL **Execute**, UDINT **Instance**, UDINT **Timeout**, SOCKADDR_CFG **DestAddr**, BOOL **EnUDPRxFilter**)
Type: SOCKET_OPEN, Ouvrir un socket

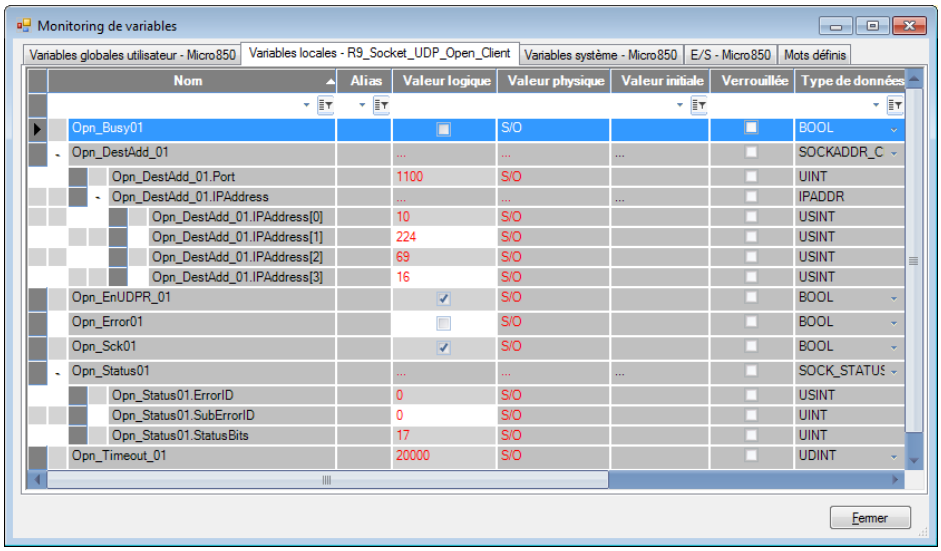
```
SOCKET_OPEN_1 (Execute_SOCKET_OPEN. Instance_SOCKET_OPEN, Timeout_SOCKET_OPEN, BestAddr_SOCKET_OPEN, EnVDPR_SOCKET_OPEN)
Done_SOCKET_OPEN := _SOCKET_OPEN. Done:
Busy_SOCKET_OPEN := _SOCKET_OPEN. Busy:
Error_SOCKET_OPEN := _SOCKET_OPEN. Error:
Status_SOCKET_OPEN := _SOCKET_OPEN. Status:
```

Résultats

Exemple de TCP



Exemple d'UDP



SOCKET_READ

Lit les données sur un socket et renvoie le nombre d'octets spécifié. Pour le protocole TCP (Transmission Control Protocol), renvoie des octets jusqu'au nombre requis lorsque des données sont reçues. Pour le protocole UDP (User Datagram Protocol), se termine lorsqu'un datagramme est disponible.

Détails de l'opération :

Le comportement **SOCKET_READ** suivant peut avoir un impact sur les communications existantes, y compris la communication qui ne concerne pas les sockets.

- Si l'opération **SOCKET_READ** n'est pas exécutée de manière synchronisée avec l'appareil distant, le contrôleur conserve le paquet reçu de l'appareil distant jusqu'à ce que l'un des événements suivants se produise :
  - **SOCKET_READ** s'exécute.
  - Le temps de dépassement du socket expire.
  - RST est reçu à partir d'un appareil distant.
  - **SOCKET_DELETE** ou **SOCKET_DELETEALL** s'exécute.
  - L'utilisateur exécute une **Modification en mode Exécuter** qui supprime toutes les instances de socket créées.
  - Le contrôleur passe du mode Exécuter au mode Programmer, ce qui supprime toutes les instances de socket créées.
  - Le contrôleur passe du mode Exécuter au mode Programmer, ce qui efface les informations du compteur de diagnostic du socket et les informations des compteurs individuels.
- Si la valeur des paramètres Longueur ou Décalage est modifiée pendant l'opération SOCKET_READ (BUSY = True) une erreur se produit et le paquet reçu est rejeté.
- L'instruction **SOCKET_READ** peut renvoyer moins d'octets que le nombre demandé. RxLength contient le nombre d'octets de données reçus. Écrivez des programmes pour vérifier RxLength, puis envoyez

des demandes de lecture supplémentaires pour recevoir un message d'application entier.

- En mode **Modification en mode Exécuter**, tout changement apporté à une entrée **SOCKET_READ** pendant une opération à l'état BUSY entraîne une erreur, et le paquet reçu est ignoré.
- Les sorties sont mises à jour de manière synchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

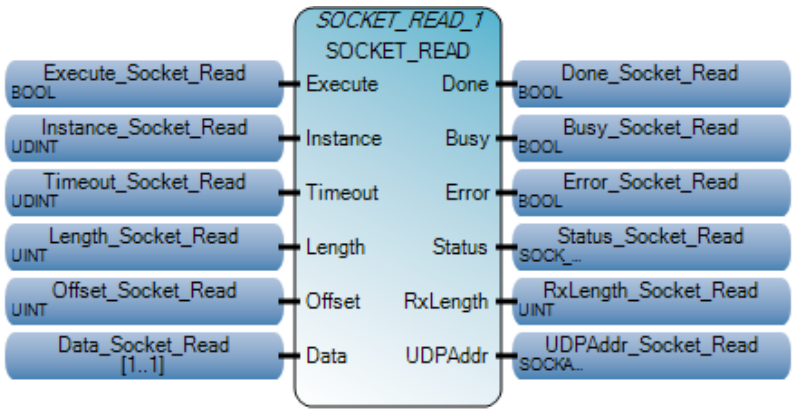
Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

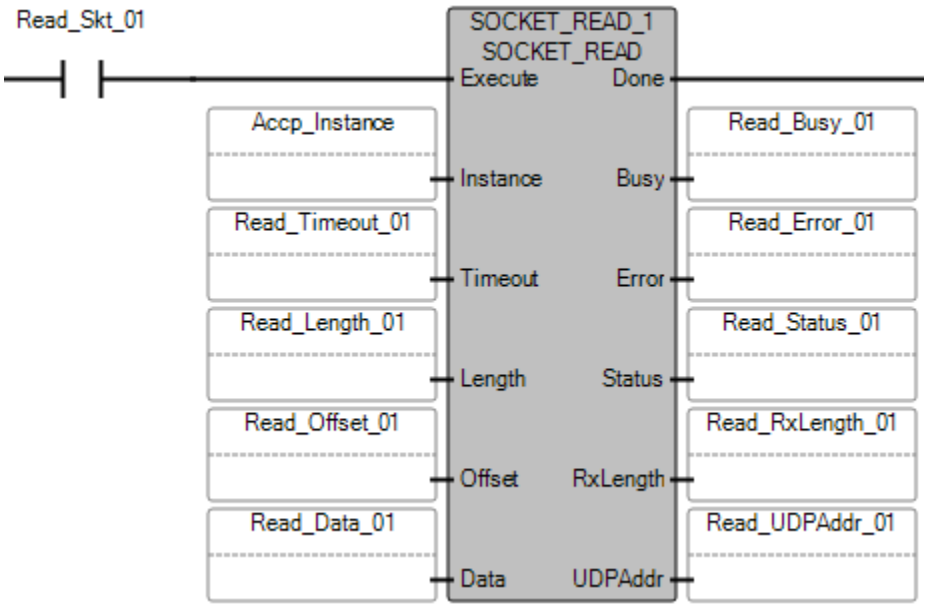
Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation de l'instruction. VRAI : le front montant est détecté, démarrez l'instruction à condition que la dernière opération soit terminée. FAUX : l'instruction ne démarre pas.
Instance	Entrée	UDINT	Copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE** ou **SOCKET_ACCEPT**. • Pour les types de socket de client UDP et TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE**. • Pour le type de socket de serveur TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_ACCEPT**.
Timeout	Entrée	UDINT	Temps dépassé pour **SOCKET_READ**. Le bloc d'instruction renvoie une erreur si la valeur de temps dépassé est inférieure à la valeur minimale. Plage du paramètre Timeout : 1 000 à 86 400 000 millisecondes Définissez le **Temps dépassé** sur 0 pour utiliser la valeur par défaut 10 000 (10 secondes).

Paramètre	Type de paramètre	Type de données	Description
Length	Entrée	UINT	Définit le nombre d'octets à lire. Vérifiez que **RxLength** contient le nombre réel d'octets lus. **SOCKET_READ** peut renvoyer moins d'octets que le nombre demandé. Prend en charge jusqu'à 256 octets.
Offset	Entrée	UNIT	Décalage dans le tableau de **Données**. Commencez la lecture des données depuis cet emplacement.
Data	Sortie	USINT[1..1]	Un tableau utilisé pour stocker les données lues à partir de **SOCKET_READ**. • La taille du tableau de données doit être >= (**Décalage** + **Longueur**). • Le tableau de données peut être plus grand que la longueur du socket lu. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.
RxLength	Sortie	UNIT	Contient le nombre d'octets de données reçus.
UDPAddr	Sortie	SOCKADDR_CFG sur la page 598	Adresse de l'appareil envoyant des données UDP (User Datagram Protocol). Exemple d'adresse UDPAddr 192.168.2.100 et de port 12000 : UDPAddr.IPAddress[0]=192 UDPAddr.IPAddress[1]=168 UDPAddr.IPAddress[2]=2 UDPAddr.IPAddress[3]=100 UDPAddr.Port = 12000 Pour TCP (Transmission Control Protocol), cette structure n'est pas utilisée et contient des 0. La connexion TCP transmet les informations de l'adresse distante.

Exemple de diagramme de bloc de fonctions SOCKET_READ



Exemple de langage à relais SOCKET_READ



Exemple de texte structuré SOCKET_READ

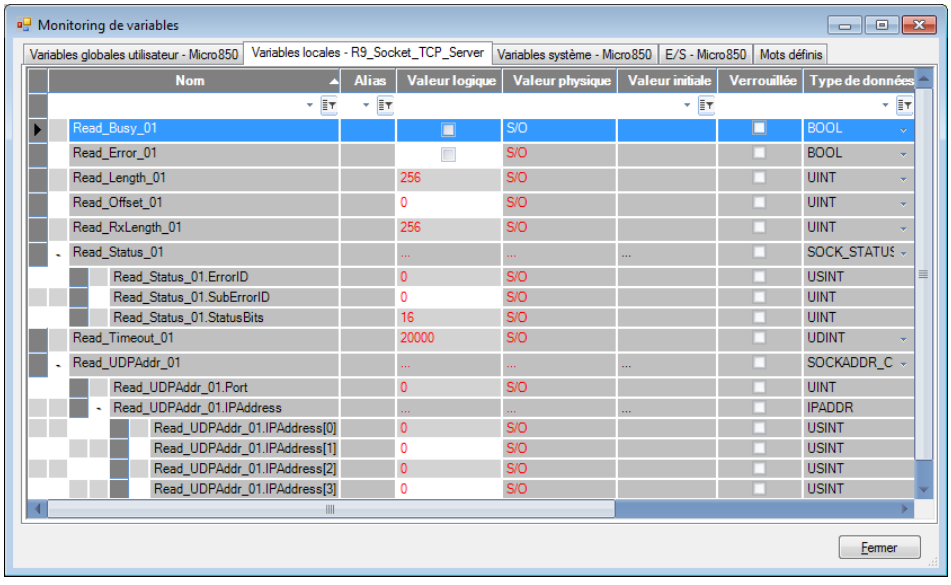
```
SOCKET_READ_1(
```

void **SOCKET_READ_1**(BOOL **Execute**, UDINT **Instance**, UDINT **Timeout**, UINT **Length**, UINT **Offset**, USINT[1..1] **Data**)
Type: SOCKET_READ, Lire un socket

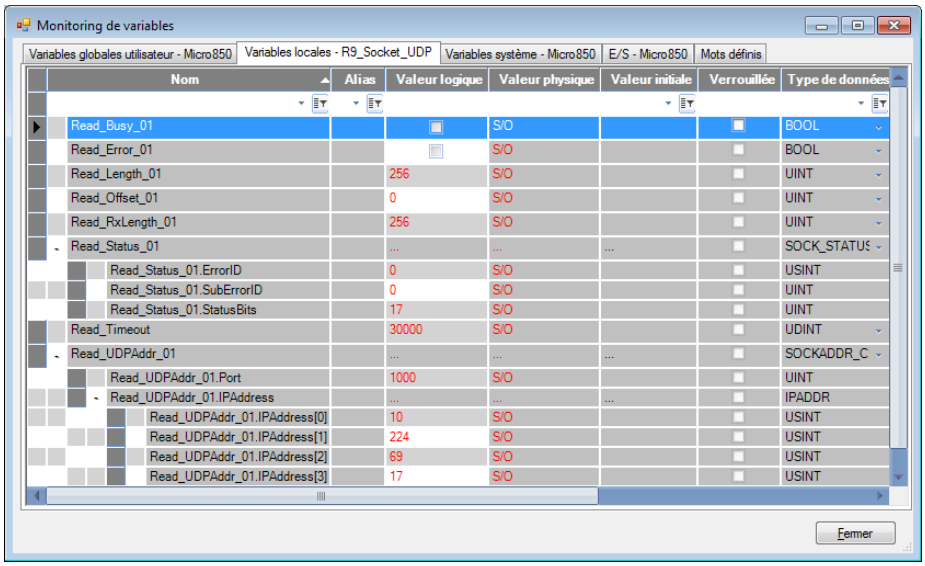
```
SOCKET_READ_1 (Execute_SOCKET_READ, Instance_SOCKET_READ, Timeout_SOCKET_READ, Length_SOCKET_READ, Offset_SOCKET_READ, Data_SOCKET_READ)
Done_SOCKET_READ := SOCKET_READ. Done;
Busy_SOCKET_READ := SOCKET_READ. Busy;
Error_SOCKET_READ := SOCKET_READ. Error;
Status_SOCKET_READ := SOCKET_READ. Status;
RxLength_SOCKET_READ := SOCKET_READ. RxLength;
UDPAddr_SOCKET_READ := SOCKET_READ. UDPAddr;
```

Résultats

Exemple de TCP :



Exemple d'UDP :



SOCKET_WRITE

Envoie des données sur un socket.

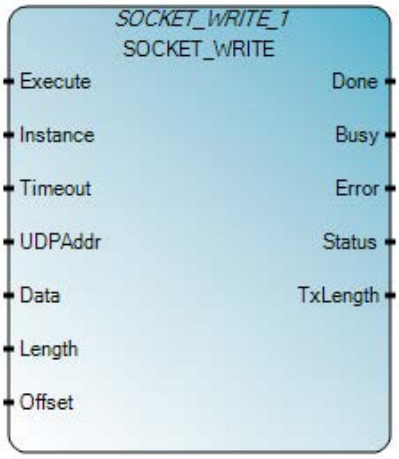
Détails de l'opération :

- **SOCKET_WRITE** tente d'envoyer le nombre requis d'octets de données spécifié dans le paramètre Longueur. Lorsque l'opération d'envoi se termine, **SOCKET_WRITE** renvoie le nombre d'octets de données écrit dans le paramètre TxLength.
- La sortie est mise à jour de manière asynchrone par la scrutation du programme.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro820, Micro850 et Micro870. Pour le Micro800 Simulator, cette instruction peut être ajoutée à un programme, mais n'est qu'un signet pour empêcher que l'instruction ne soit supprimée lors d'un changement de contrôleur.

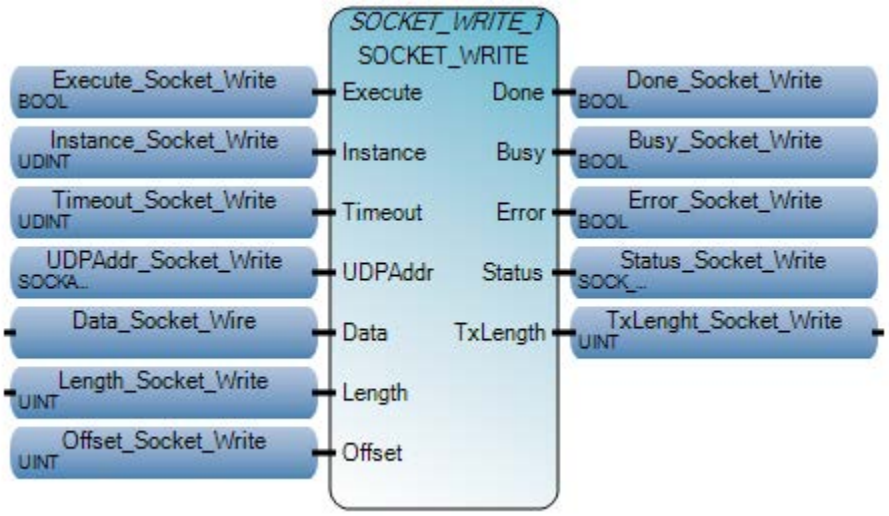
Les sorties sont toujours réinitialisées lorsque l'instruction est appliquée au contrôleur simulé (2080-LC50-48QWB-SIM).



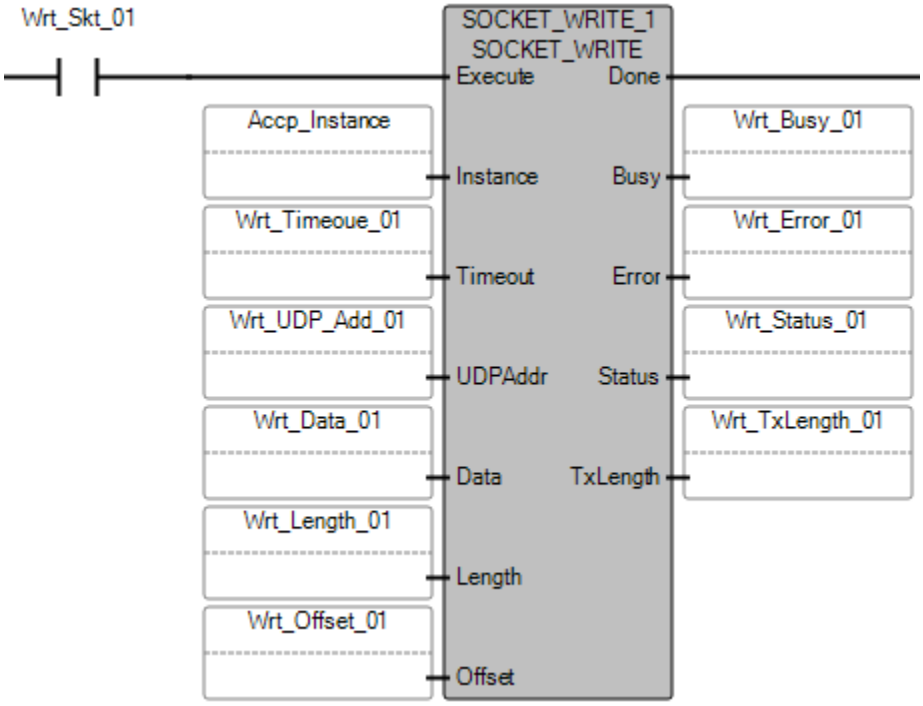
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
Execute	Entrée	BOOL	Activation du bloc d'instruction. VRAI : lorsque le front montant est détecté, démarrez le bloc d'instruction à condition que la dernière opération soit terminée. FAUX : aucun front montant détecté.
Instance	Entrée	UDINT	Copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE** ou **SOCKET_ACCEPT**. • Pour les types de socket de client UDP et TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_CREATE**. • Pour le type de socket de serveur TCP, copiez le Gestionnaire de socket renvoyé depuis une instruction **SOCKET_ACCEPT**.
Timeout	Entrée	UDINT	Temps dépassé pour les instances **SOCKET_WRITE**. L'instruction renvoie une erreur si la valeur du paramètre Timeout est inférieure à la valeur minimale. Plage du paramètre Timeout : 1 000 à 1800000 millisecondes Définissez le temps dépassé sur 0 pour utiliser la valeur par défaut 10 000 (10 secondes).
UDPAddr	Entrée	SOCKADDR_CFG sur la page 598	L'adresse de destination UDP sur laquelle les données sont écrites lorsqu'une instruction **SOCKET_OPEN** n'a pas été exécutée depuis la création du socket. Pour TCP, ou lorsque l'instruction **SOCKET_OPEN** a été exécutée pour UDP, cette structure n'est pas utilisée et doit contenir des zéros. La connexion TCP et l'instruction **SOCKET_OPEN** pour UDP véhiculent toutes les informations de l'adresse distante. Exemple d'adresse UDPAddr 192.168.2.100 et de port 12000 : UDPAddr.IPAddress[0]=192 UDPAddr.IPAddress[1]=168 UDPAddr.IPAddress[2]=2 UDPAddr.IPAddress[3]=100 UDPAddr.Port = 12000 Utilisez le type de données **SOCKADDR_CFG** pour définir UDPAddr.
Data	Entrée	USINT[1..1]	Tableau utilisé pour stocker l'écriture de données dans les instances de socket respectives à l'aide de l'instruction **SOCKET_WRITE**. • La taille du tableau de données doit être >= (**Décalage** + **Longueur**). • Le tableau de données peut être plus grand que le paramètre Length **SOCKET_WRITE**.
Length	Entrée	UINT	Le nombre d'octets de données à écrire. Le maximum est 256 octets.
Offset	Entrée	UNIT	Décalage dans le tableau de données. L'écriture de données à partir de SOCKET_WRITE commence depuis cet emplacement.
Done	Sortie	BOOL	Indique la fin de l'opération. VRAI : l'opération s'est déroulée correctement. FAUX : l'opération est en cours ou a rencontré une erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Busy	Sortie	BOOL	VRAI : l'opération n'est pas terminée. FAUX : l'opération est terminée. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Error	Sortie	BOOL	Indique une erreur. VRAI : erreur détectée. FAUX : aucune erreur. La sortie est mise à jour de manière synchrone par l'analyse du programme.
Status	Sortie	SOCK_STATUS	L'état est défini à l'aide du type de données SOCK_STATUS sur la page 598 qui contient les informations sur ErrorID sur la page 598, SubErrorID et StatusBits sur la page 602. La sortie est mise à jour de manière synchrone par l'analyse du programme.
TxLength	Sortie	UNIT	Le nombre d'octets de données écrites.

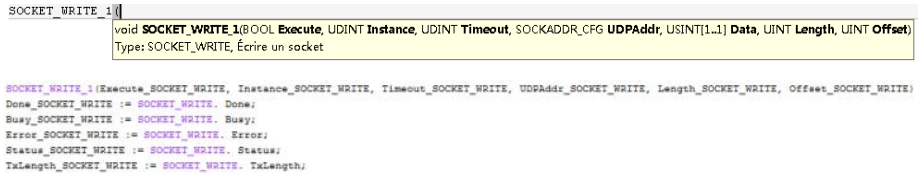
Exemple de diagramme de bloc de fonctions SOCKET_WRITE



Exemple de langage à relais SOCKET_WRITE

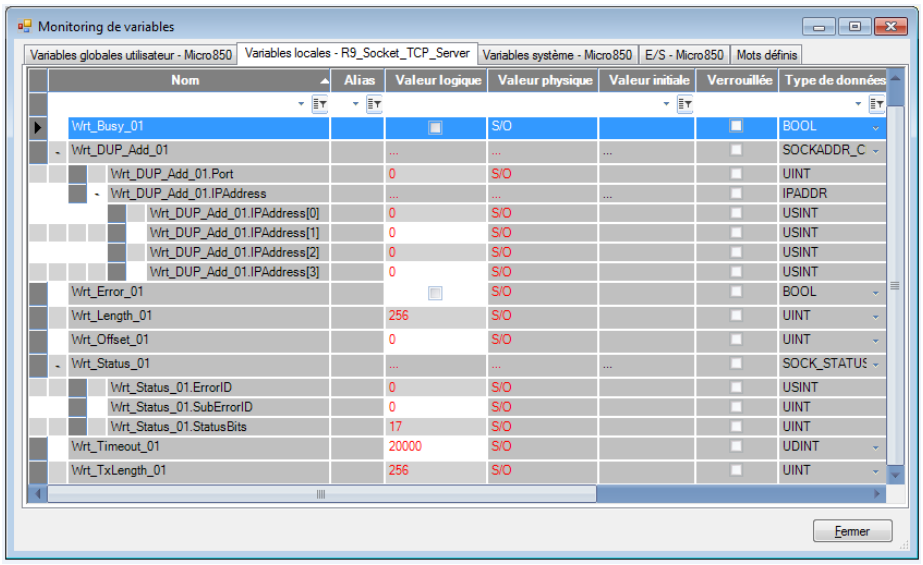


Exemple de texte structuré SOCKET_WRITE

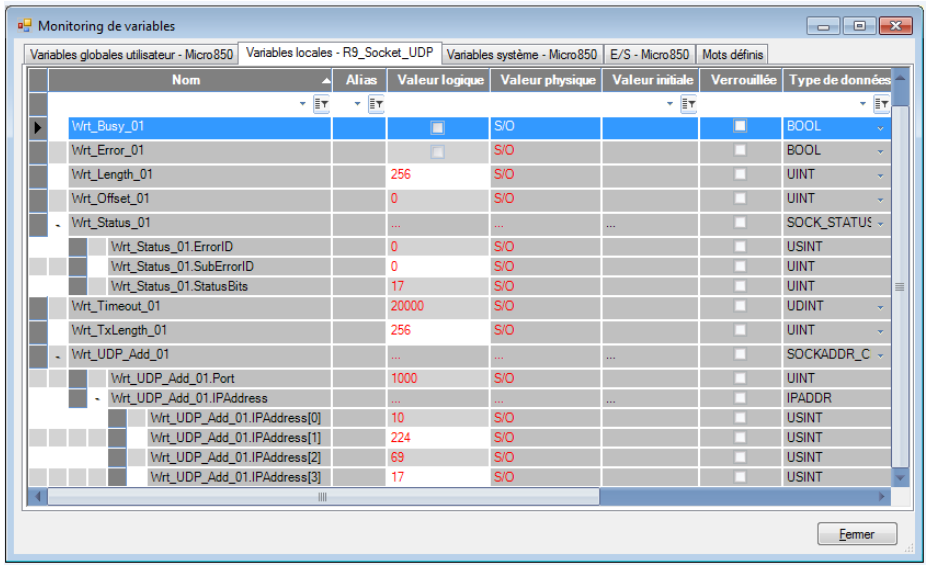


Résultats

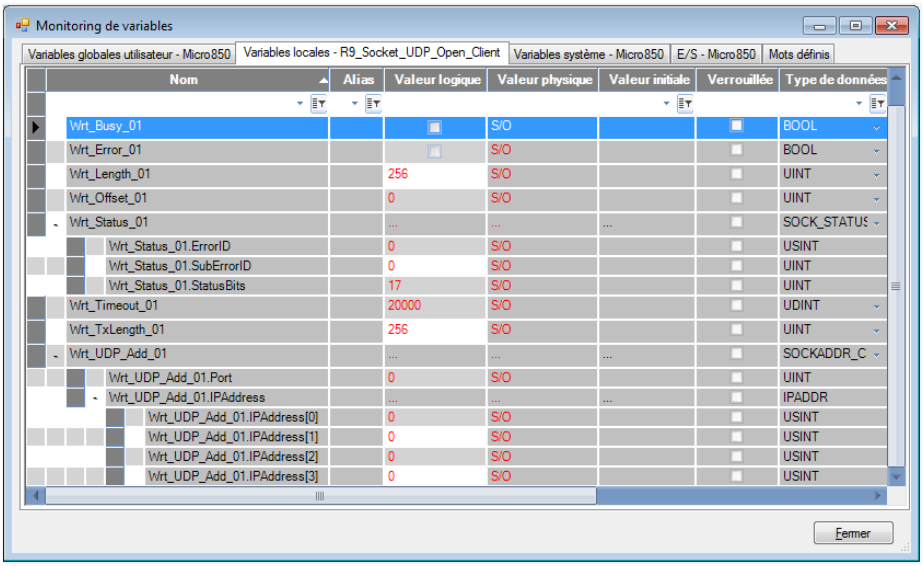
Exemple de TCP :



UDP sans exemple d'ouverture :



UDP avec exemple d'ouverture :



Type de données SOCKADDR_CFG

Le tableau suivant décrit le type de données **SOCKADDR_CFG**.

Paramètre	Type de données	Description
Port	UINT	Indiquez un numéro de port local sur lequel une application écoute et reçoit.
IP Address[4]	USINT	Indiquez une adresse IP. Par exemple, une adresse IP égale à 192.168.2.100 : • `IPAddress[0]=192` • `IPAddress[1]=168` • `IPAddress[2]=2` • `IPAddress[3]=100`

Type de données SOCK_STATUS

Le tableau suivant décrit le type de données **SOCK_STATUS**.

Paramètre	Type de données	Description
ErrorID	USINT	Valeur du code d'erreur sur la page 598.
SubErrorID	UINT	Sous-valeur du code d'erreur.
StatusBits	UINT	Bits d'état d'exécution pour les instructions du socket sur la page 602.

Codes d'erreur du socket

Le tableau suivant décrit les codes d'erreur d'état pour les instructions de socket :

Code ErrorID	Code SubErrorID	Description de l'erreur	Action corrective
0		L'instruction de socket a correctement terminé l'opération.	
1		L'instruction de socket est en attente.	
2		L'Instance de socket n'est pas disponible.	Confirmez que l'Instance de support n'est pas supprimée ou que la valeur de Timeout est dépassée.

Code ErrorID	Code SubErrorID	Description de l'erreur	Action corrective
3		L'opération **SOCKET_DELETEALL** est en cours.	Attendez que l'opération **SOCKET_DELETEALL** en attente se termine.
4	1	Paramètre incorrect, chaîne non valide.	Aucune action. Réservé pour une utilisation future.
4	2	Paramètre incorrect, adresse IP du socket non valide. • Une erreur se produit. **SOCKET_CREATE** contient un **SockAddr** avec une valeur différente de zéro. • Exécution de **SOCKET_OPEN** ou **SOCKET_WRITE** avec une adresse IP cible incorrecte. Les adresses IP cibles non valides sont : • Adresse IP de réponse • 0.0.0.0 • Adresse de boucle (127.x.x.x) • Adresse multicast de classe D (224.x.x.x) • Adresse de liaison locale (169.254.x.x) • Adresse de diffusion (uniquement applicable pour l'Instance de socket TCP)	Remplacez par une adresse IP valide.
4	3	Paramètre incorrect, adresse de port du socket non valide. Une erreur se produit lorsque les ports suivants sont indiqués dans **SOCKET_CREATE**, **SOCKET_OPEN** ou **SOCKET_WRITE** : • Ports TCP • 44818 – EtherNet/IP • 502 – ModbusTCP • 67 – Serveur DHCP • 68 – Client DHCP • 0 – Port non valide • Ports UDP • 2222 – EtherNet/IP • 67 – Serveur DHCP • 68 – Client DHCP • 0 – Port non valide	Modifiez l'adresse du port.
4	4	Paramètre incorrect, type du socket non valide. Les types de support valides sont : • TCP – 1 • UDP – 2	Modifiez le type du socket.
4	5	Paramètre incorrect, valeur de Timeout du socket non valide. Les valeurs de temporisation correctes sont les suivantes : • **SOCKET_CREATE**, **SOCKET_ACCEPT** et **SOCKET_READ** : • 0 (par défaut) ou toute valeur comprise entre 1 000 et 86 400 000 ms. • **SOCKET_OPEN** et **SOCKET_WRITE** : • 0 (par défaut) ou toute valeur comprise entre 1 000 et 86 400 000 ms.	Changez la valeur de Timeout sur une valeur valide.
4	6	Paramètre incorrect, instance du socket non valide. • Type de support de serveur TCP : • Utilisez l'Instance de socket renvoyée dans **SOCKET_ACCEPT** pour **SOCKET_WRITE**, **SOCKET_READ**, **SOCKET_DELETE** et **SOCKET_INFO**. • Types de socket UDP et de socket de client TCP : • Utilisez l'Instance de socket renvoyée dans **SOCKET_CREATE** pour **SOCKET_OPEN**, **SOCKET_WRITE**, **SOCKET_READ**, **SOCKET_DELETE** et **SOCKET_INFO**.	Utilisez le numéro d'Instance renvoyé dans des instructions **SOCKET_CREATE** et **SOCKET_ACCEPT** après une exécution réussie.

Code ErrorID	Code SubErrorID	Description de l'erreur	Action corrective
4	7	Paramètre incorrect, longueur du tableau non valide.	Augmentez la taille du tableau utilisé pour contenir les données lues et écrites du socket.
4	8	Paramètre incorrect, dimension du tableau non valide.	Utilisez un tableau unidimensionnel pour contenir les données lues et écrites du socket.
5		Le socket demande d'annuler l'opération. Une erreur se produit lorsque : • La liaison Ethernet est désactivée ou le câble Ethernet est débranché • L'opération **SOCKET_DELETE** est exécutée lorsqu'une opération **SOCKET_OPEN**, **SOCKET_ACCEPT**, **SOCKET_READ** ou **SOCKET_WRITE** est en cours. • L'opération Modification en mode Exécuter est exécutée lorsqu'une opération **SOCKET_OPEN**, **SOCKET_ACCEPT**, **SOCKET_READ** ou **SOCKET_WRITE** est en cours. • Collision d'adresses IP détectée lorsqu'une opération **SOCKET_OPEN**, **SOCKET_ACCEPT**, **SOCKET_READ** ou **SOCKET_WRITE** est en cours.	Redémarrez l'opération du socket en fonction du Type de socket. Reportez-vous à la Machine d'état correspondant pour redémarrer l'opération du support.
6	1	Séquence du socket incorrecte, opération d'ouverture du socket en cours ou socket connecté. Une erreur se produit lorsque : • **SOCKET_OPEN** est en cours (état BUSY) et l'utilisateur exécute une autre instance de SOCKET_OPEN avec la même instance de socket. • **SOCKET_OPEN** est en cours (état BUSY), et l'utilisateur exécute **SOCKET_ACCEPT** avec la même instance de socket.	Utilisez l'exécution unique de **SOCKET_OPEN** pour l'Instance de socket correspondante. N'exécutez pas une opération **SOCKET_ACCEPT** pour la même instance de socket.
6	2	Séquence du socket incorrecte, opération d'acceptation du socket en cours ou socket connecté. Une erreur se produit lorsque : • **SOCKET_ACCEPT** est en cours (état **BUSY**), et l'utilisateur exécute une autre instance de **SOCKET_ACCEPT** avec la même instance de socket. • **SOCKET_ACCEPT** est en cours (état **BUSY**), et l'utilisateur exécute **SOCKET_OPEN** avec la même instance de socket.	Utilisez l'exécution unique de **SOCKET_ACCEPT** pour l'Instance de socket correspondante. N'exécutez pas une opération **SOCKET_OPEN** pour la même instance de socket.
6	3	Séquence du socket incorrecte, socket configuré en tant que client TCP.	
6	4	Séquence du socket incorrecte, socket configuré en tant que serveur TCP.	
6	5	Séquence de socket incorrecte, le socket est connecté.	Assurez-vous que le socket n'est pas déjà connecté à un appareil à distance avant d'exécuter **SOCKET_OPEN** ou **SOCKET_ACCEPT** à nouveau.
6	6	Séquence du socket incorrecte, socket configuré en tant qu'UDP.	Assurez-vous que les Instances de socket UDP ne sont pas utilisées avec **SOCKET_ACCEPT**.
6	7	Séquence de socket incorrecte, le socket n'est pas connecté.	Confirmez que le support est connecté avec la cible.
7		Délai de temporisation d'Instance du socket.	Configurez les valeurs de temporisation pour **SOCKET_ACCEPT**, **SOCKET_OPEN**, **SOCKET_READ** et **SOCKET_WRITE** en conséquence.
8		Le module du socket n'est pas initialisé.	Assurez-vous que le lien Ethernet est activé ou que le câble Ethernet du contrôleur est connecté au réseau ou résolvez le conflit d'adresses IP du contrôleur.
9	1	Erreur fatale d'Instruction de socket, Instance de socket manquante.	
9	2	Erreur fatale d'Instruction de socket, Instance de socket non	

Code ErrorID	Code SubErrorID	Description de l'erreur	Action corrective
		valide.	
9	3	Erreur fatale d'Instruction de socket, Instance de socket de verrouillage non valide.	
9	4	Erreur fatale d'instruction de socket, type de socket non valide.	
9	5	Erreur fatale d'Instruction de socket, gestionnaire d'annulation manquant.	
10	1	Erreur de traitement en arrière-plan du socket, adresse en cours d'utilisation.	
10	2	Erreur de traitement en arrière-plan du socket, UDP a reçu un paquet important et le paquet est plus grand que la longueur du socket lu.	Assurez-vous que la longueur **SOCKET_READ** est égale ou supérieure à la taille du paquet reçu. La taille maximale de lecture est recommandée. La taille maximale de **SOCKET_READ** est de 256.
10	3	Erreur de traitement en arrière-plan du socket, TCP reçoit un paquet important. Le paquet reçu est plus grand que la longueur du socket lu.	Assurez-vous que la longueur **SOCKET_READ** est égale ou supérieure à la taille du paquet reçu. Dans la mesure où le client peut envoyer une taille égale à 256 à la fenêtre de réception, la valeur 256 est recommandée.
10	4	Erreur de traitement en arrière-plan du socket, RST reçu ou déconnecté de l'appareil à distance.	Redémarrez le client ou serveur TCP du contrôleur comme indiqué dans le diagramme d'état. Vérifiez l'appareil cible qui a envoyé le RST et assurez-vous que la cible a redémarré avec l'état correct.
10	5	Erreur de traitement en arrière-plan du socket, paquet UDP reçu à partir d'un appareil différent, paquet ignoré. Réservé pour une utilisation future.	
10	6	Erreur de traitement en arrière-plan du socket, la file d'attente est pleine. Une erreur se produit lorsque : • **SOCKET_READ** ou **SOCKET_WRITE** exécute quatre instructions pour la même instance de socket dans la même scrutation. • Effectuez un **SOCKET_READ** lorsque les quatre instructions **SOCKET_READ** sont à l'état BUSY pour la même instance de socket.	Attendez que la file d'attente **SOCKET_READ** et **SOCKET_WRITE** soit disponible pour l'instance de socket correspondante.
10	7	Erreur de traitement en arrière-plan du socket, le paramètre **SOCKET_READ** a changé (longueur, décalage, taille du tableau de données et variable du tableau de données).	Lorsque l'état **SOCKET_READ** est **BUSY**, ne modifiez pas les paramètres d'entrée.
11	1	Erreur fatale de traitement en arrière-plan du socket, pointeur de données manquant.	
11	2	Erreur fatale de traitement en arrière-plan du socket, pointeur de session manquant.	
11	3	Erreur fatale de traitement en arrière-plan du socket, pointeur du socket TCP ou UDP non valide.	
11	4	Erreur fatale de traitement en arrière-plan du socket, type d'Instance de socket non valide.	
11	5	Erreur fatale de traitement en arrière-plan du socket, Instance de socket manquante.	
11	6	Erreur fatale de traitement en arrière-plan du socket, Instance de socket non valide.	
11	7	Erreur fatale de traitement en arrière-plan du socket, état du socket non valide.	
11	8	Erreur fatale de traitement en arrière-plan du socket, type du socket non valide.	
11	9	Erreur fatale de traitement en arrière-plan du socket, échec de suppression TCP.	

Code ErrorID	Code SubErrorID	Description de l'erreur	Action corrective
11	10	Erreur fatale de traitement en arrière-plan du socket, échec de suppression UDP.	
128	1	Aucun paquet disponible pour le message de déconnexion.	
128	2	Espace insuffisant pour ajouter l'en-tête TCP au début.	
128	3	Le pointeur d'ajout de paquets n'est pas valide.	
128	7	Pointeur de socket non valide.	
128	10	Option de durée de vie, de fragment ou de type de service non valide.	
128	17	Appelant non valide pour ce service.	
128	18	Le paquet n'est pas valide.	
128	20	Ce composant n'a pas été activé.	
128	21	Ce composant a déjà été activé.	
128	22	L'écoute n'est pas activée pour le port indiqué.	
128	26	La suspension demandée a été interrompue.	
128	33	Adresse IP de serveur non valide.	
128	34	Ce socket est lié à un autre port.	
128	35	Le port est lié à un autre socket.	
128	36	Le socket n'est pas lié.	
128	38	Le lien du socket a été annulé lorsqu'il était suspendu en attente d'un paquet de réception.	
128	39	Le socket n'a pas été créé.	
128	51	Aucune structure de demande d'écoute supplémentaire n'est disponible.	
128	52	Il existe déjà une demande d'écoute active pour ce port.	
128	53	Le socket n'est pas dans un état fermé.	
128	54	Le socket du serveur fourni n'est pas dans un état d'écoute.	
128	55	Aucun temps d'attente n'a été indiqué, la tentative de connexion est en cours.	
128	56	La connexion a échoué.	
128	57	La demande est supérieure à la taille de fenêtre publiée du récepteur en octets.	
128	64	Un autre thread est suspendu. Un seul thread est autorisé.	
128	65	La déconnexion a échoué dans la période de temporisation.	
128	66	Le socket est lié.	
128	69	Aucun port disponible.	
128	70	Port non valide.	
128	71	Il existe déjà un pointeur de socket valide pour ce port ou le port indiqué ne dispose pas d'une demande d'écoute active.	
128	72	Identique à **NX_SUCCESS**, à l'exception qu'une demande de connexion en file d'attente a été traitée au cours de cet appel.	
128	73	Erreur de dépassement de la file d'attente de transmission TCP.	

Bits d'état d'instruction de socket

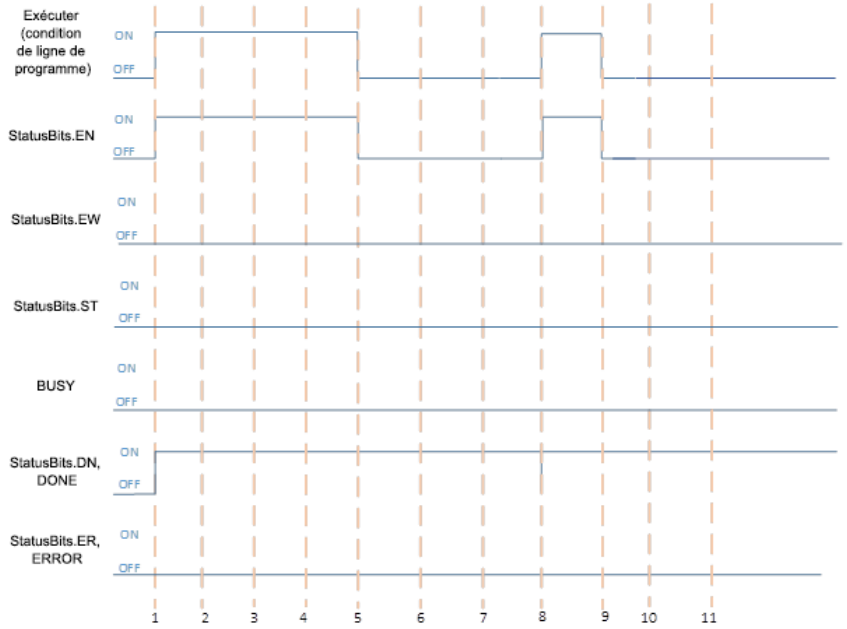
Le tableau suivant décrit les bits d'état d'exécution pour les instructions de socket sur la page 567.

Nombre de bits	Nom	Description

Nombre de bits	Nom	Description
0	EN – Bit d'activation	Le bit EN est défini lorsque l'instruction est activée en raison d'une transition Faux à Vrai, mais qu'elle n'est pas encore terminée ou a déclenché une erreur.
1	EW – Bit d'activation de la file d'attente	Le bit EW est défini lorsque le contrôleur détecte qu'une demande d'instruction de socket est entrée dans la file d'attente. Le contrôleur réinitialise le bit EW lorsque le bit ST est défini.
2	ST – Bit de démarrage	Le bit ST est défini lorsque l'instruction en file d'attente est en cours d'exécution. Le bit ST est réinitialisé lorsque le bit DN ou ER est défini.
3	ER – Bit d'erreur	Indique qu'une erreur est survenue lors de l'exécution de l'instruction. Le bit ER est réinitialisé la fois suivante où la condition de ligne passe de Faux à Vrai.
4	DN – Bit de réalisation	Le bit DN est défini lorsque l'instruction de socket se termine correctement. Le bit DN est réinitialisé la fois suivante où la condition de ligne passe de Faux à Vrai.

Diagrammes de temporisation d'instructions de socket

Exécution réussie pour les instructions de socket sur la page 567 lorsque le processus est immédiat



Condition A : La condition de ligne est VRAI lors de l'exécution des instructions

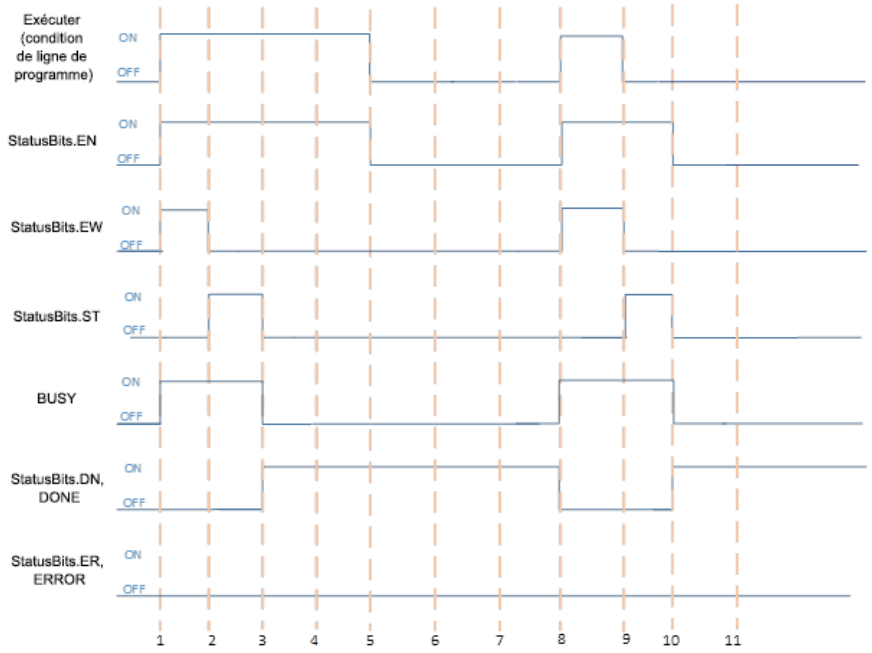
Condition de ligne	Description
1	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support terminent correctement l'exécution. • La sortie pour le bit DN et DONE est définie.
2,3,4	Aucun changement de la condition de ligne.

Condition de ligne	Description
5	La condition de ligne de programme devient FAUX lorsque le bit EN est effacé.
6,7	Aucun changement de la condition de ligne.

Condition B : La condition de ligne est FAUX lors de l'exécution des instructions

Condition de ligne	Description
8	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Les instructions de support terminent correctement l'exécution. • Le bit EN est défini et tous les autres bits sont effacés. • La sortie pour le bit DN et DONE est définie.
9	La condition de ligne devient FAUX lorsque : • Le bit EN est effacé.
10,11	Aucun changement de la condition de ligne.

Exécution réussie pour les instructions de support lorsque le processus n'est pas immédiat



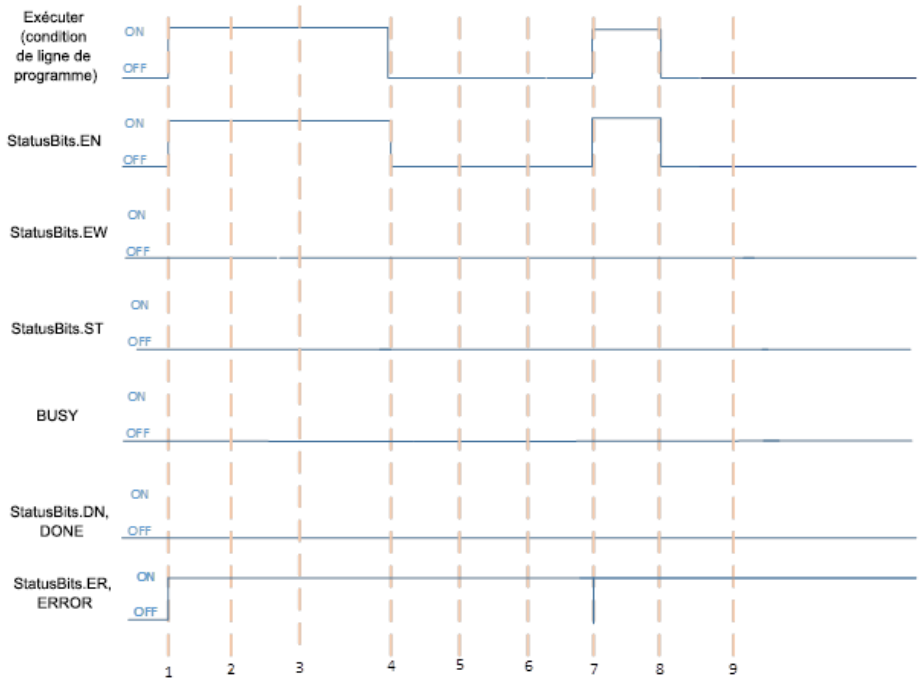
Condition A : La condition de ligne est VRAI lors de l'exécution des instructions

Condition de ligne	Description
1	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support exécutent le traitement en arrière-plan. • La sortie pour le bit EW et BUSY est définie. • Paramètre d'entrée des instructions de support verrouillé pour le traitement en arrière-plan.
2	Les instructions de support commencent l'exécution lorsque : • Le bit EW est effacé. • Le bit ST est défini.
3	Les instructions de support terminent correctement l'exécution lorsque : • La sortie pour le bit ST et BUSY est effacée. • La sortie pour le bit DN et DONE est définie.
4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit EN est effacé.
6,7	Aucun changement de la condition de ligne.

Condition B : La condition de ligne passe à FAUX lors de l'exécution des instructions

Condition de ligne	Description
8	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support envoyées pour le traitement en arrière-plan. • La sortie pour le bit EW et BUSY est définie. • Paramètre d'entrée des instructions de support verrouillé pour le traitement en arrière-plan.
9	Les instructions de support commencent l'exécution lorsque : • Le bit EW est effacé. • Le bit ST est défini. • La condition de ligne devient FAUX.
10	Les instructions de support terminent correctement l'exécution lorsque : • La sortie pour le bit ST et BUSY est effacée. • La sortie pour le bit DN et DONE est définie. • La condition de ligne est définie sur FAUX. • Le bit EN est effacé.

Les instructions de support échouent lorsque EN est VRAI et qu'EW et ST sont FAUX



Condition A : La condition de ligne est VRAI lors de l'exécution des instructions

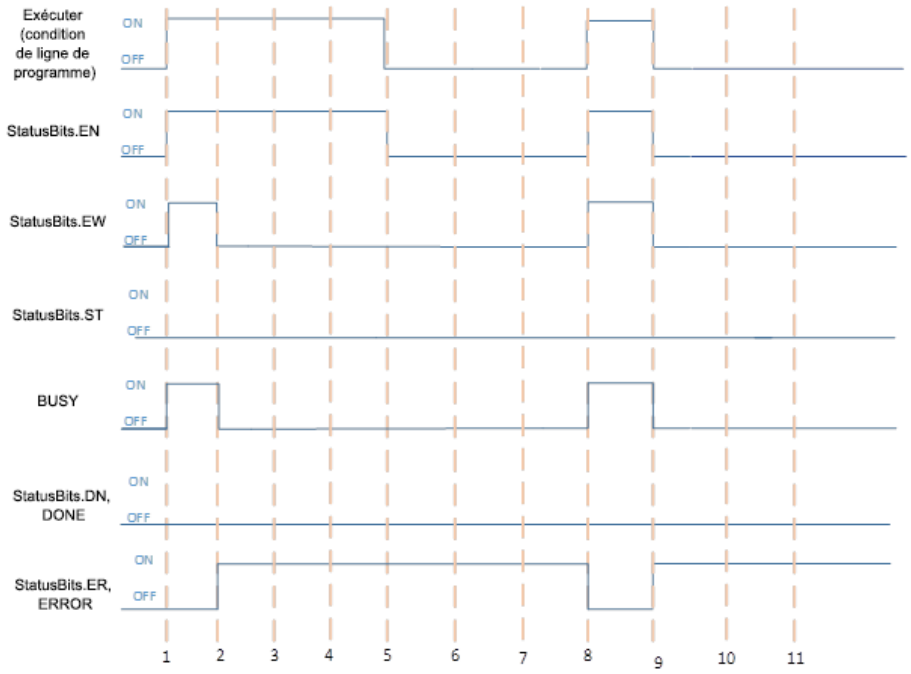
Condition de ligne	Description
1	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support terminent l'exécution avec une erreur. • La sortie pour le bit ER et l'erreur est définie.
2,3	Aucun changement de la condition de ligne.
4	La condition de ligne devient FAUX lorsque : • Le bit EN est effacé.
5,6	Aucun changement de la condition de ligne.

Condition B : La condition de ligne passe à FAUX lors de l'exécution des instructions

Condition de ligne	Description
7	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés.

Condition de ligne	Description
	• Les instructions de support terminent l'exécution avec une erreur. • La sortie pour le bit ER et l'erreur est définie.
8	La condition de ligne devient FAUX lorsque : • Le bit EN est effacé.
9	Aucun changement de la condition de ligne.

Les instructions de support échouent lorsque EW est VRAI et que le processus d'instruction n'est pas immédiat



Condition A : La condition de ligne est VRAI lors de l'exécution des instructions

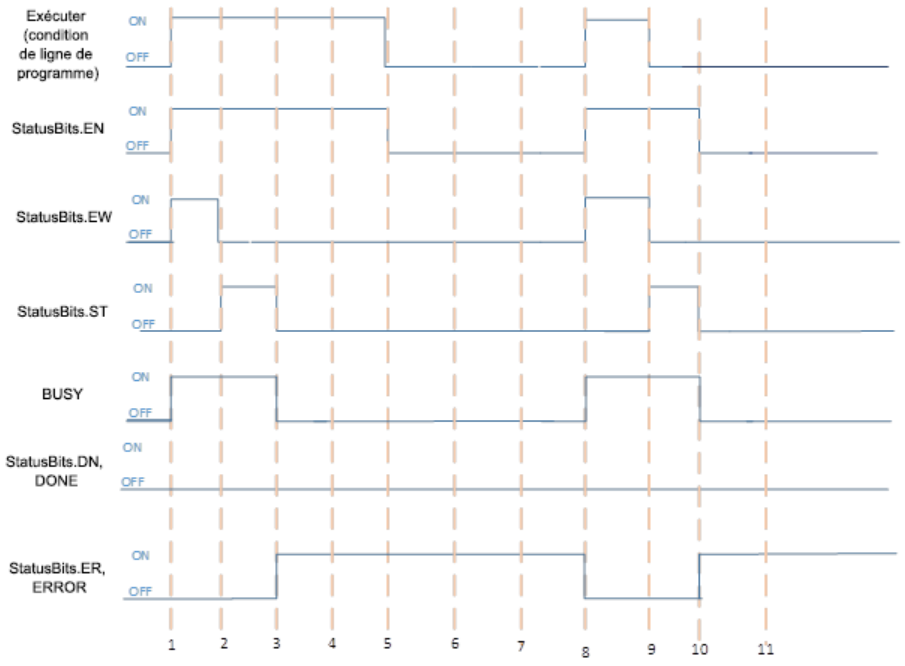
Condition de ligne	Description
1	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support envoyées pour le traitement en arrière-plan. • La sortie pour le bit EW et BUSY est définie. • Paramètre d'entrée des instructions de support verrouillé pour le traitement en arrière-plan.
2	Les instructions de support terminent l'exécution avec une erreur lorsque : • La sortie pour le bit EW et BUSY est effacée.

Condition de ligne	Description
	• La sortie pour le bit ER et l'erreur est définie.
3,4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit EN est effacé.
6,7	Aucun changement de la condition de ligne.

Condition B : La condition de ligne passe à FAUX lors de l'exécution des instructions

Condition de ligne	Description
8	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support envoyées pour le traitement en arrière-plan. • La sortie pour le bit EW et BUSY est définie. • Paramètre d'entrée des instructions de support verrouillé pour le traitement en arrière-plan.
9	Les instructions de support terminent l'exécution avec une erreur lorsque : • La sortie pour le bit EW et BUSY est effacée. • La sortie pour le bit ER et l'erreur est définie. • La condition de ligne devient FAUX. • Le bit EN est effacé.
10,11	Aucun changement de la condition de ligne.

Les instructions de support échouent lorsque ST est VRAI et que le processus d'instruction n'est pas immédiat



Condition A : La condition de ligne est VRAI lors de l'exécution des instructions

Condition de ligne	Description
1	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support envoyées pour le traitement en arrière-plan. • La sortie pour le bit EW et la sortie BUSY sont définies. • Paramètre d'entrée des instructions de support verrouillé pour le traitement en arrière-plan.
2	Les instructions de support commencent l'exécution lorsque : • Le bit EW est effacé • Le bit ST est défini.
3	Les instructions de support terminent l'exécution avec une erreur lorsque : • La sortie pour le bit ST et la sortie BUSY sont effacées. • La sortie pour le bit ER et l'erreur est définie.
4	Aucun changement de la condition de ligne.
5	La condition de ligne devient FAUX lorsque : • Le bit EN est effacé.
6,7	Aucun changement de la condition de ligne.

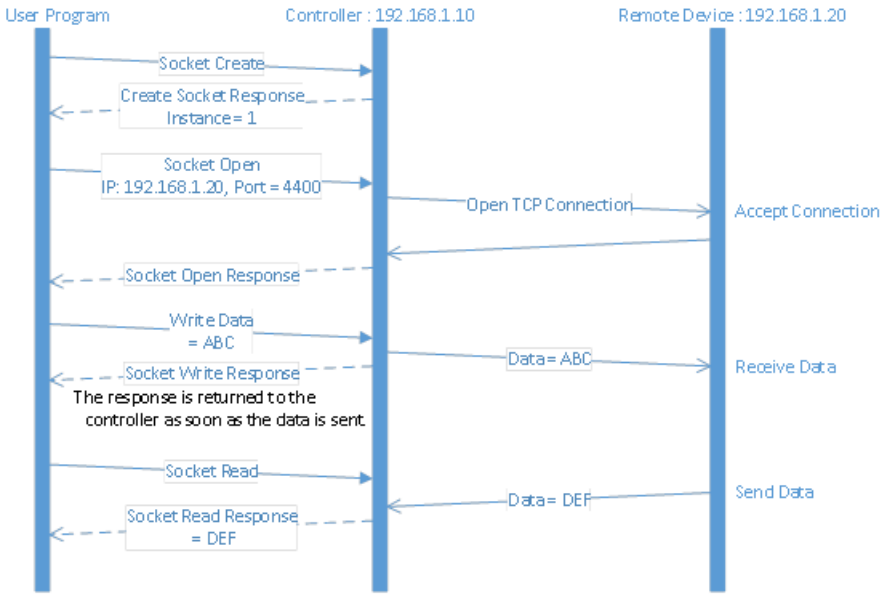
Condition B : La condition de ligne passe à FAUX lors de l'exécution des instructions

Condition de ligne	Description
8	La condition de ligne devient VRAI lorsque : • L'exécution des instructions de support est activée. • Le bit EN est défini et tous les autres bits sont effacés. • Les instructions de support envoyées pour le traitement en arrière-plan. • La sortie pour le bit EW et BUSY est définie. • Paramètre d'entrée des instructions de support verrouillé pour le traitement en arrière-plan.
9	Les instructions de support commencent l'exécution lorsque : • Le bit EW est effacé et le bit ST est défini.
10	Les instructions de support terminent l'exécution avec une erreur lorsque : • La sortie pour le bit ST et BUSY est effacée. • La sortie pour le bit ER et l'erreur est définie. • La condition de ligne devient FAUX. • Le bit EN est effacé.
11	Aucun changement de la condition de ligne.

Diagrammes de transaction d'instructions de socket

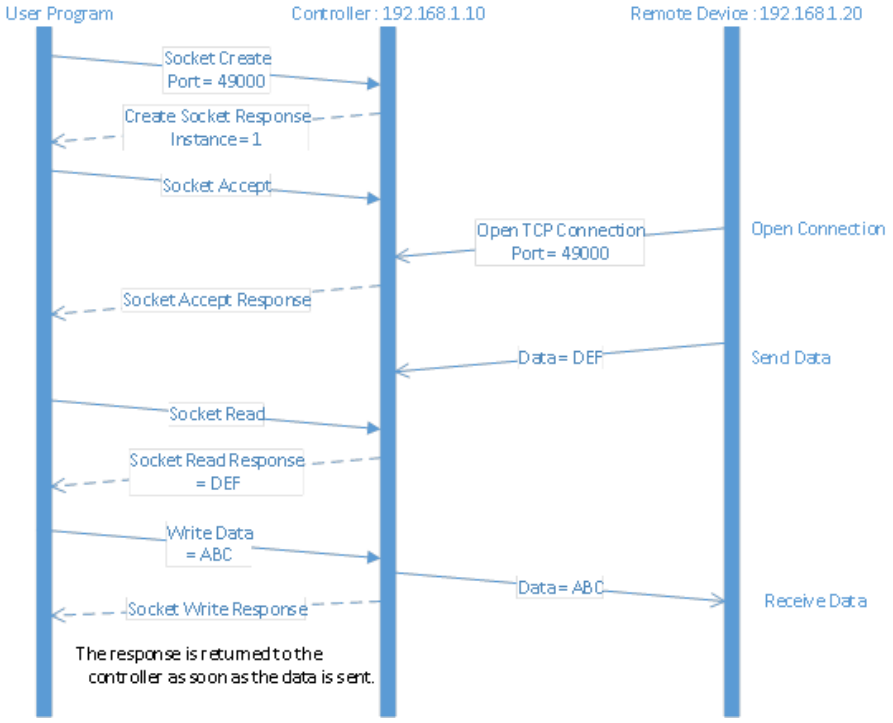
Transactions pour un client TCP

Le diagramme suivant représente une séquence type de transactions d'interface de support avec le contrôleur agissant comme un client TCP.



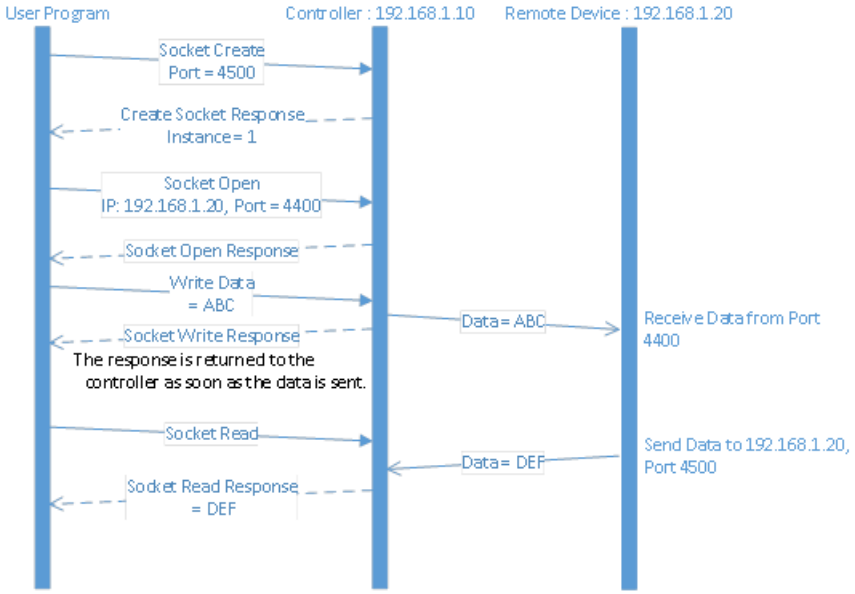
Transactions pour un serveur TCP

Le diagramme suivant représente une séquence type de transactions d'interface de support avec le contrôleur agissant comme un serveur TCP.



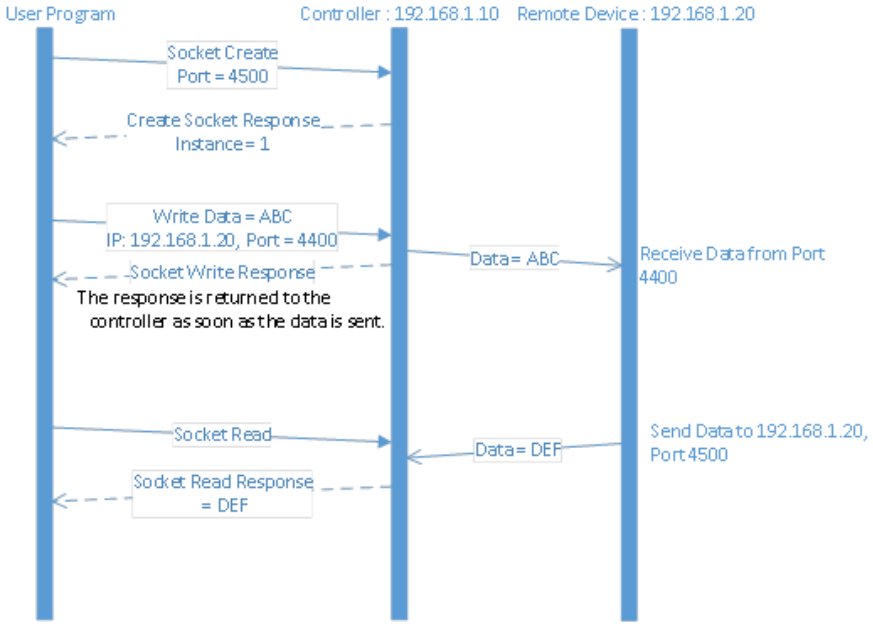
Transactions pour UDP avec une connexion ouverte

Le diagramme suivant représente une séquence type de transactions d'interface de support pour les communications UDP lors de l'utilisation du service de connexion ouverte pour indiquer l'adresse de destination.



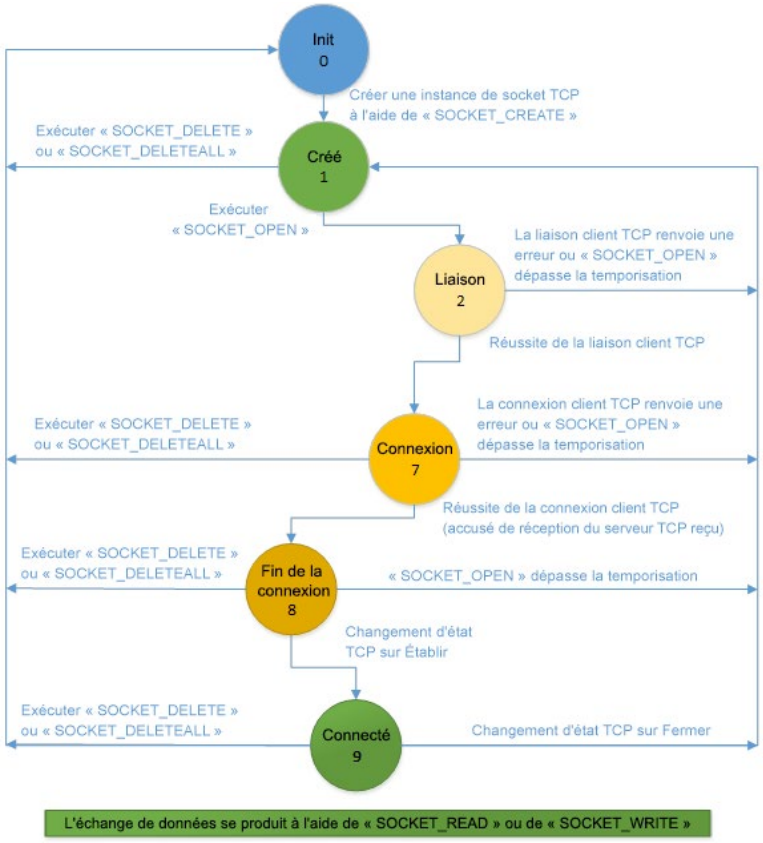
Transactions pour UDP sans connexion ouverte

Le diagramme suivant représente une séquence type de transactions d'interface de support pour les communications UDP sans utiliser le service de connexion ouverte pour indiquer l'adresse de destination. Dans ce cas, le contrôleur indique la destination de chaque datagramme et reçoit l'adresse de l'expéditeur avec chaque datagramme qu'il reçoit.



Diagrammes d'états de machine pour TCP

Client TCP utilisant les instructions desocket sur la page 567



Serveur TCP utilisant les instructions de support

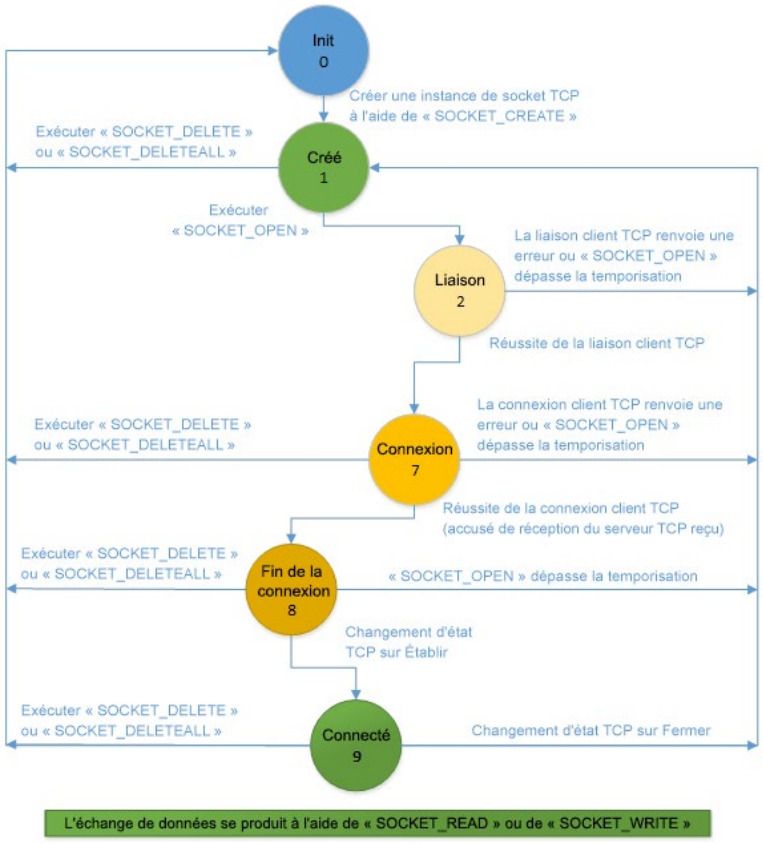
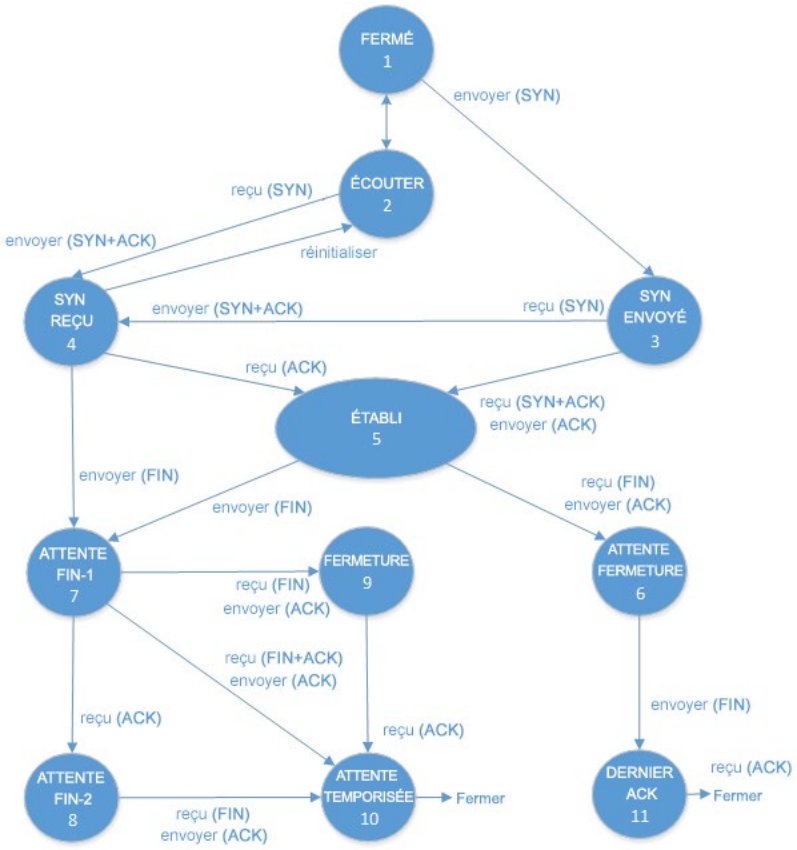
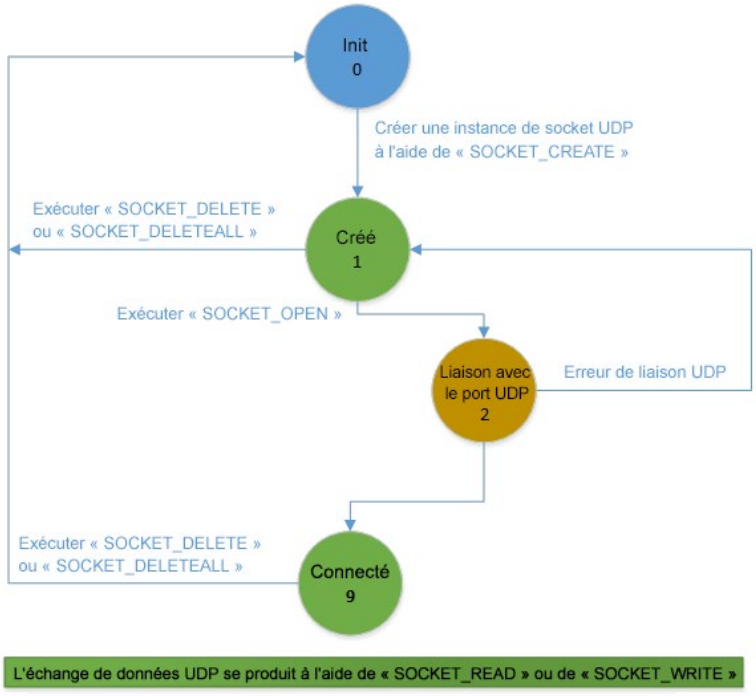


Diagramme d'état TCP

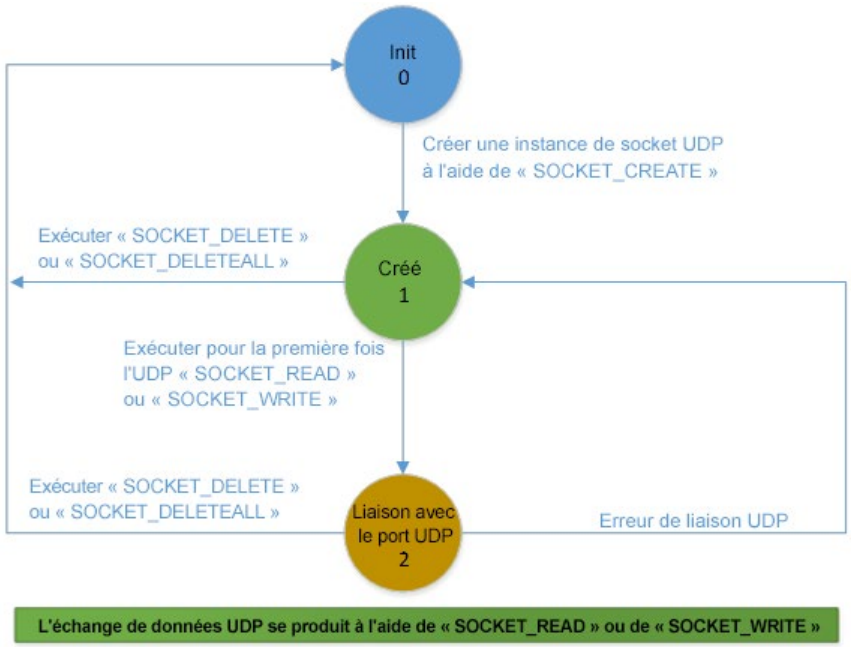


Diagrammes de machine à état pour UDP

Datagramme UDP avec instruction SOCKET_OPEN



Datagramme UDP sans instruction SOCKET_OPEN



Instructions de manipulation des chaînes

Utilisez les instructions de manipulation des chaînes pour modifier une séquence de symboles choisis à partir d'un jeu ou d'un alphabet, afin de modifier l'état de sortie. pour une lecture correcte des chaînes d'entrée contenant des caractères spéciaux, les caractères de la chaîne d'entrée après le programme contenant l'instance de bloc fonctionnel sont en ligne.

Instruction	Description
ASCII sur la page 615	Renvoie le code ASCII des caractères d'une chaîne. Caractère -> code ASCII.
CHAR sur la page 617	Renvoie une chaîne d'un caractère pour un code ASCII. Code ASCII -> caractère.
DELETE sur la page 618	Supprime des caractères dans une chaîne.
FIND sur la page 620	Localise et communique la position de sous-chaînes au sein de chaînes.
INSERT sur la page 622	Insère des sous-chaînes à des positions définies par l'utilisateur dans des chaînes.
LEFT sur la page 624	Extrait les caractères de la partie gauche d'une chaîne.
MID sur la page 626	Extrait les caractères de la partie centrale d'une chaîne.
MLEN sur la page 628	Calcule la longueur d'une chaîne.
REPLACE sur la page 631	Remplace les parties d'une chaîne par de nouveaux jeux de caractères.
RIGHT sur la page 629	Extrait les caractères de la partie droite d'une chaîne.

ASCII

Renvoie le code ASCII des caractères des chaînes.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

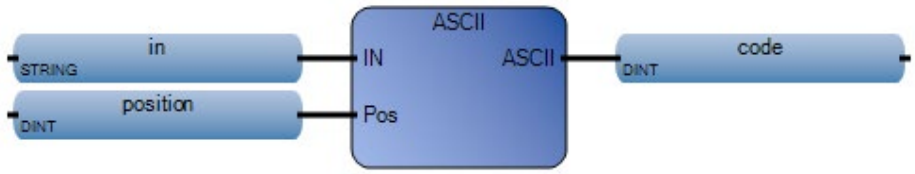
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

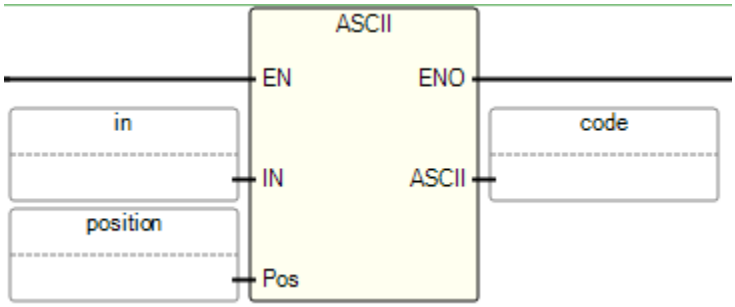
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : affiche le code ASCII des caractères. FAUX : aucune opération d'affichage. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne non vide, quelle qu'elle soit.

Paramètre	Type de paramètre	Type de données	Description
Pos	Entrée	DINT	Position du caractère sélectionné dans l'ensemble [1.. lon] (lon étant la longueur de la chaîne IN).
ASCII	Sortie	DINT	Code ASCII du caractère sélectionné (dans l'ensemble [0 .. 255]) ; renvoie 0 si Pos est à l'extérieur de la chaîne.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions ASCII



Exemple de logique à relais ASCII



Exemple de texte structuré ASCII

```
1  position := 1;
2  code := ASCII(in, position);
```

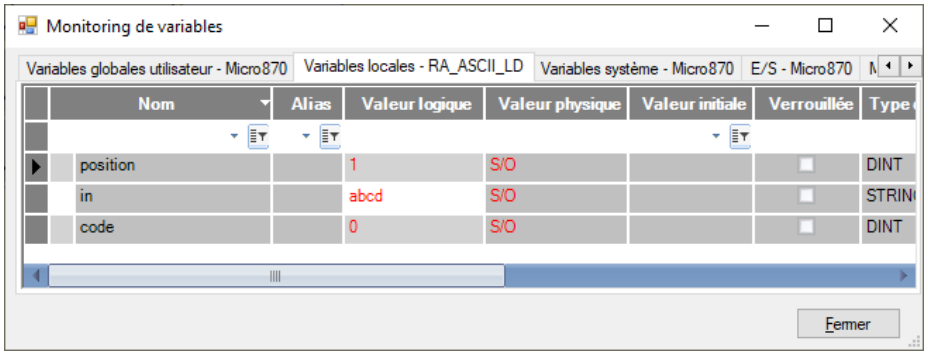
```
ASCII(
    DINT ASCII(STRING IN, DINT Pos)
    Caractère -> code ASCII
```

(* Équivalence ST : *)

FirstChr := ASCII (message, 1) ;

(* FirstChr est le code ASCII du premier caractère de la chaîne *)

Résultats



CHAR (code ASCII pour un caractère de chaîne)

Renvoie une chaîne d'un caractère pour un code ASCII. Code ASCII -> caractère.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

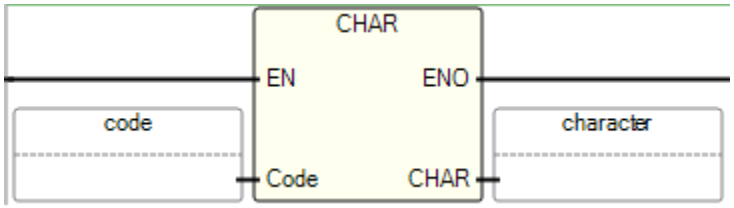
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : fournit une chaîne d'un caractère. FAUX : aucune opération. S'applique aux programmes de schémas à relais.
Code	Entrée	DINT	Code ASCII dans l'ensemble [0 .. 255].
CHAR	Sortie	STRING	Chaîne contenant un caractère. Le code ASCII du caractère est fourni dans le code d'entrée.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions CHAR

Exemple de logique à relais CHAR



Exemple de texte structuré CHAR

```
1  code := 97;
2  character := CHAR(code);
```

```
CHAR(
     STRING CHAR(DINT Code)
     Code ASCII -> Caractère
```

(* Équivalence ST : *)

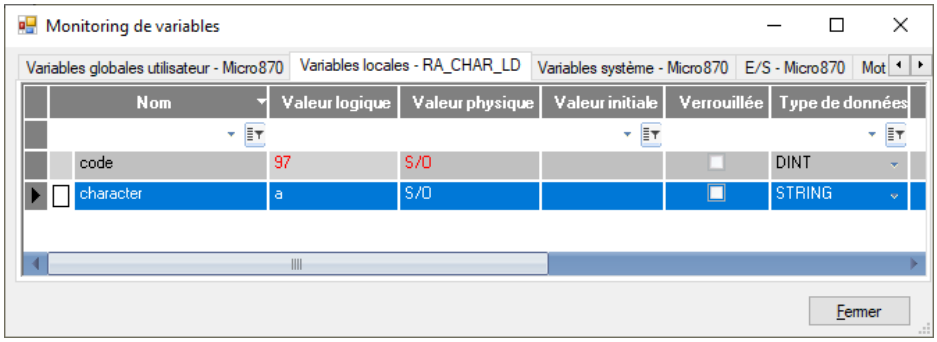
Affichage := CHAR (valeur + 48) ;

(* la valeur est comprise dans l'ensemble [0..9] *)

(* 48 est le code ASCII de '0' *)

(* le résultat correspond à une chaîne contenant un caractère compris entre 0 et 9 *)

Résultats



DELETE (supprimer une sous-chaîne)

Supprime des caractères dans une chaîne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

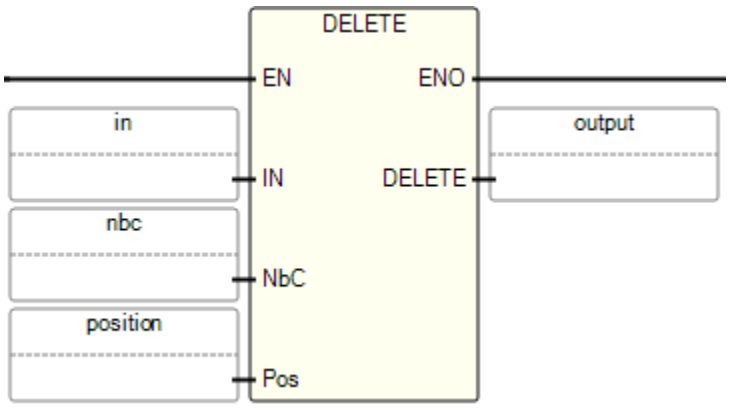
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : suppression de la partie spécifiée d'une chaîne. FAUX : aucune opération. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne non vide, quelle qu'elle soit.
NbC	Entrée	DINT	Nombre de caractères à supprimer.
Pos	Entrée	DINT	Position du premier caractère supprimé (le premier caractère de la chaîne occupe la position 1).
DELETE	Sortie	STRING	La sortie est : • Une chaîne modifiée. • Une chaîne vide (si Pos < 1). • La chaîne initiale (si Pos > longueur de la chaîne IN). • La chaîne initiale (si NbC <= 0).
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction DELETE

Exemple de logique à relais DELETE



Exemple de texte structuré DELETE

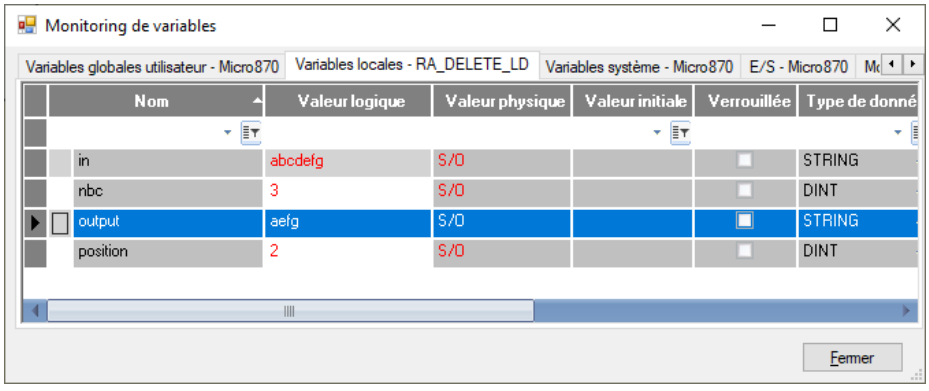
```
1  nbc := 3;
2  position := 2;
3  output := DELETE(in, nbc, position);
```

(* Équivalence ST : *)

complete_string := INSERT ('ABCD ', 'EFGH', 5) ; (* complete_string correspond à 'ABCDEFGH ' *)

sub_string := DELETE (complete_string, 4, 3); (* sub_string correspond à 'ABGH '*)

Résultats



FIND (recherche une sous-chaîne)

Localise et communique la position de sous-chaînes au sein de chaînes.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830,

Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

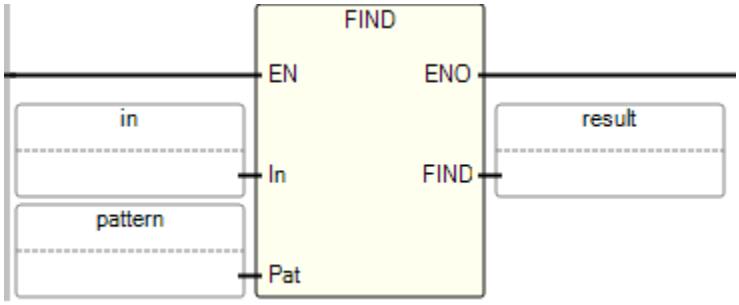
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

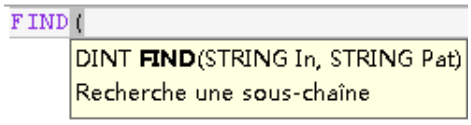
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : localisation de la position dans des chaînes. FAUX : aucune opération de localisation. S'applique aux programmes de schémas à relais.
In	Entrée	STRING	Chaîne non vide, quelle qu'elle soit.
Pat	Entrée	STRING	Chaîne non vide, quelle qu'elle soit (modèle).
FIND	Sortie	DINT	La sortie est : • 0 si le modèle de la sous-chaîne n'est pas trouvé. • position du premier caractère de la première occurrence du modèle de sous-chaîne (la première position est 1). Cette instruction est sensible à la casse.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction FIND

Exemple de logique à relais FIND



Exemple de texte structuré FIND



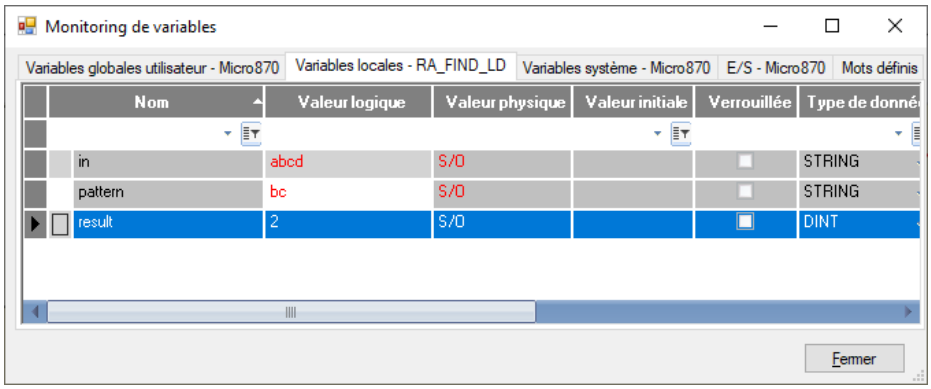

```
1  result := FIND(in, pattern);
```

(* Équivalence ST : *)

complete_string := 'ABCD' + 'EFGH' ; (* complete_string correspond à 'ABCDEFGH ' *)

résultat := FIND (complete_string, 'CDEF') ; (* résultat = 3 *)

Résultats



INSERT (insérer une chaîne)

Insère des sous-chaînes à des positions définies par l'utilisateur dans des chaînes.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

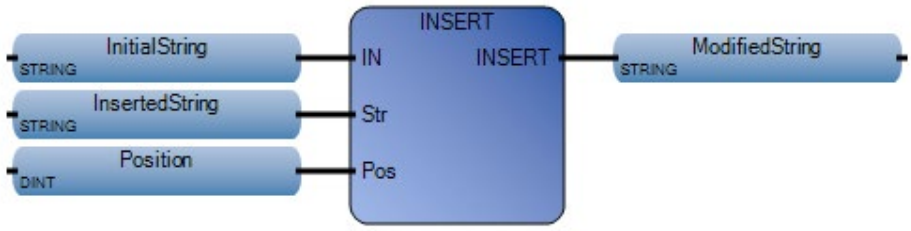
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

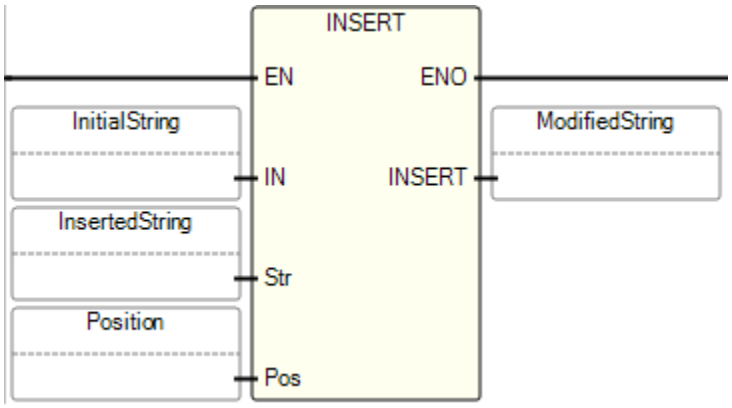
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : insertion de sous-chaînes dans une chaîne.

Paramètre	Type de paramètre	Type de données	Description
			FAUX : aucune opération. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne initiale.
Str	Entrée	STRING	Chaîne à insérer.
Pos	Entrée	DINT	Position de l'insertion ; l'insertion est réalisée avant la position (la première position valide est 1).
INSERT	Sortie	STRING	Chaîne modifiée. Il peut s'agir de l'une des valeurs suivantes : • chaîne vide si Pos <= 0 ; • concaténation des deux chaînes si la valeur Pos est supérieure à la longueur de la chaîne IN.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

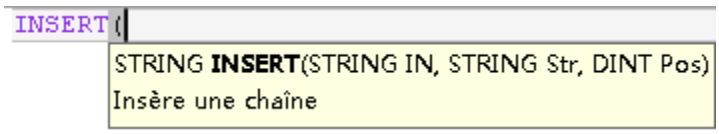
Exemple de Diagramme de Bloc Fonction INSERT



Exemple de langage à relais INSERT



Exemple de texte structuré INSERT



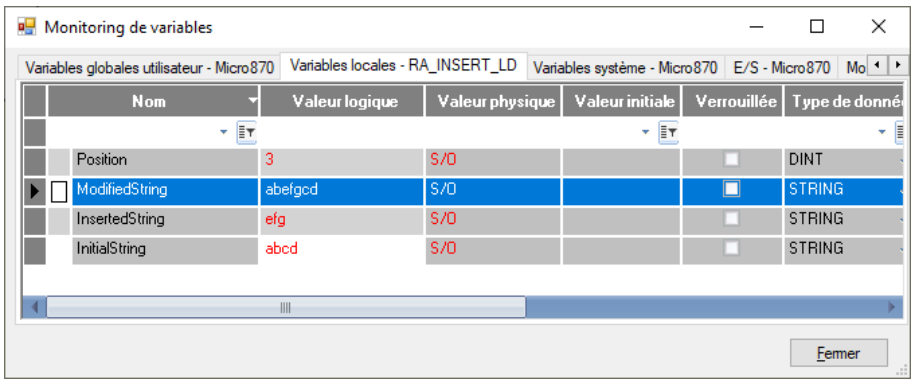

```
1  Position := 3;
2  ModifiedString := INSERT(InitialString, InsertedString, Position);
```

(* Équivalence ST : *)

MyName := INSERT ('Mr JONES', 'Frank ', 4) ;

(* MyName correspond à 'Mr Frank JONES' *)

Résultats



LEFT (extrait la partie gauche d'une chaîne)

Extrait les caractères de la partie gauche d'une chaîne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

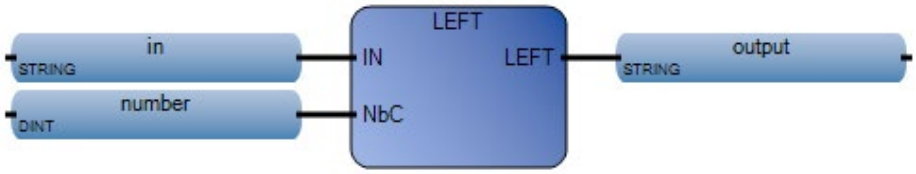
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

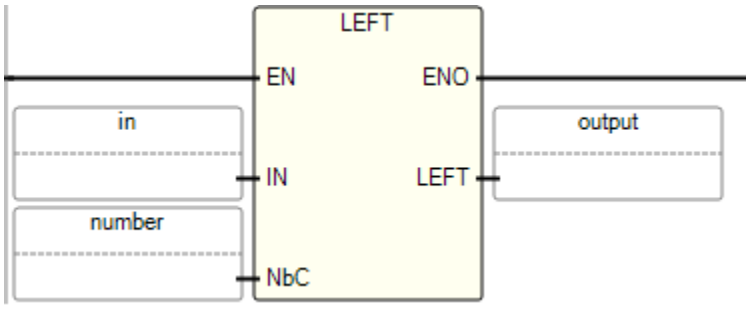
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : calcule le nombre de caractères à partir de la gauche de la chaîne. FAUX : aucune opération. S'applique aux programmes de schémas à relais.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	STRING	Chaîne non vide, quelle qu'elle soit.
NbC	Entrée	DINT	Nombre de caractères à extraire. Ce nombre ne peut pas être supérieur à la longueur de la chaîne IN.
LEFT	Sortie	STRING	Partie gauche de la chaîne IN (sa longueur = NbC). Il peut s'agir de l'une des valeurs suivantes : • chaîne vide si NbC <= 0 ; • chaîne IN complète si NbC >= longueur de la chaîne IN.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

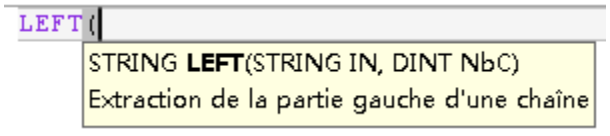
Exemple de diagramme de bloc de fonctions LEFT



Exemple de logique à relais LEFT



Exemple de texte structuré LEFT




```
1  number := 3;
2  output := LEFT(in, number);
```

(* Équivalence ST : *)

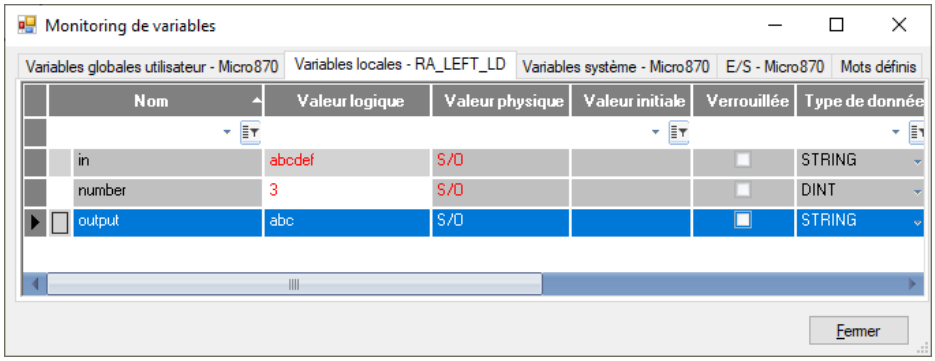
complete_string := RIGHT ('12345678', 4), LEFT ('12345678', 4), 5 ;

(* complete_string correspond à '56781234'

la valeur issue de l'appel RIGHT est égale à '5678'

la valeur issue de l'appel LEFT est égale à '1234'*)

Résultats



MID (extraction de la partie du centre d'une chaîne)

Extrait les caractères de la partie centrale d'une chaîne. Utilisez la position et le nombre de caractères fournis pour calculer les parties de chaînes requises.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

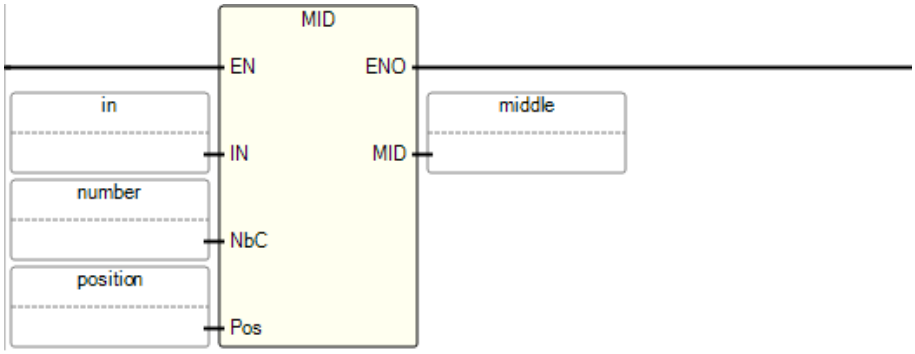
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : génération de la partie d'une chaîne. FAUX : aucune opération de génération. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne non vide, quelle qu'elle soit.
NbC	Entrée	DINT	Le nombre de caractères à extraire ne peut pas être supérieur à la longueur de la chaîne IN.
Pos	Entrée	DINT	Position de la sous-chaîne. Le premier caractère de la sous-chaîne correspond au caractère référencé par Pos (la première position valide correspond à la valeur 1).
MID	Sortie	STRING	Partie centrale de la chaîne (sa longueur = NbC). Lorsque le nombre de caractères à extraire dépasse la longueur de la chaîne IN, NbC est automatiquement calculé pour n'extraire que le reste de la chaîne. Lorsque les valeurs NbC ou Pos sont égales à zéro ou négatives, une chaîne vide est renvoyée.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de diagramme de bloc de fonctions MID

Exemple de langage à relais MID



Exemple de texte structuré MID

```
MID(
```

STRING **MID**(STRING IN, DINT NbC, DINT Pos)
Extraction de la partie du centre d'une chaîne.

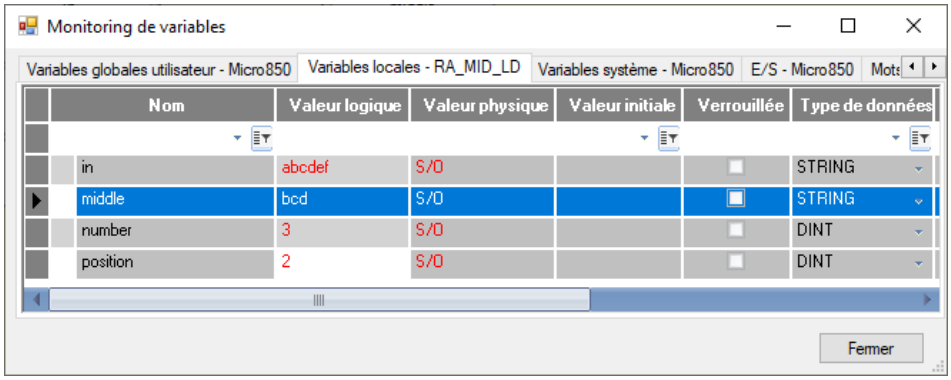
```
1  number := 3;
2  position := 2;
3  middle := MID(in, number, position);
```

(* Équivalence ST : *)

sub_string := MID ('abcdefgh', 2, 4) ;

(* sub_string correspond à 'de' *)

Résultats



MLEN (longueur de la chaîne)

Calcule la longueur d'une chaîne.

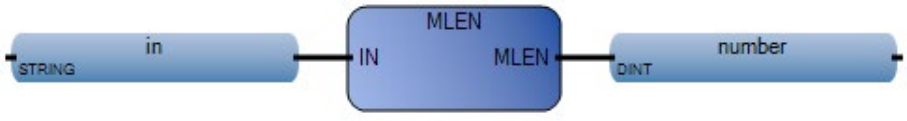
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

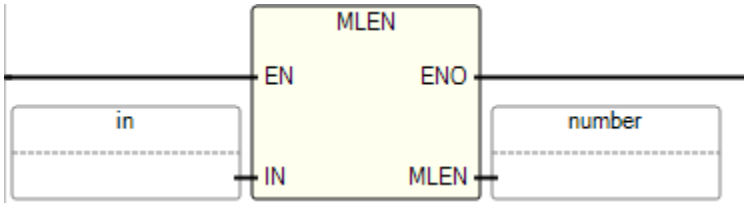
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : calcul de la longueur d'une chaîne. FAUX : aucune opération. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne, quelle qu'elle soit.
MLEN	Sortie	DINT	Nombre de caractères dans la chaîne IN.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

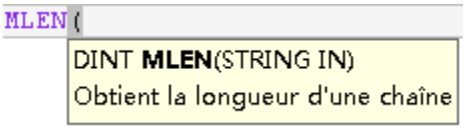
Exemple de Diagramme de Bloc Fonction MLEN



Exemple de langage à relais MLEN



Exemple de texte structuré MLEN




```
1  number := MLEN(in);
```

(* Équivalence ST : *)

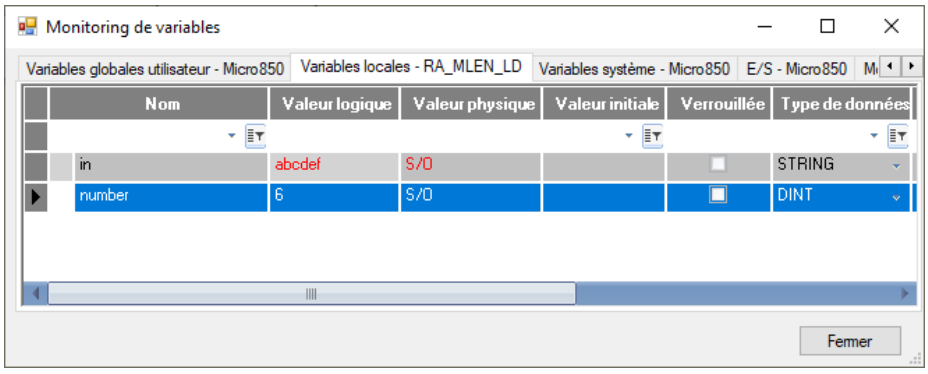
nbchar := MLEN (complete_string) ;

Si (nbchar < 3) alors renvoi ; End_if ;

préfixe := LEFT (complete_string, 3) ;

(* ce programme extrait les 3 caractères situés à gauche de la chaîne et insère le résultat dans la variable de la chaîne préfixe. Aucune opération n'est réalisée si la chaîne comporte moins de 3 caractères *)

Résultats



RIGHT (extrait la partie droite d'une chaîne)

Extrait les caractères de la partie droite d'une chaîne.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

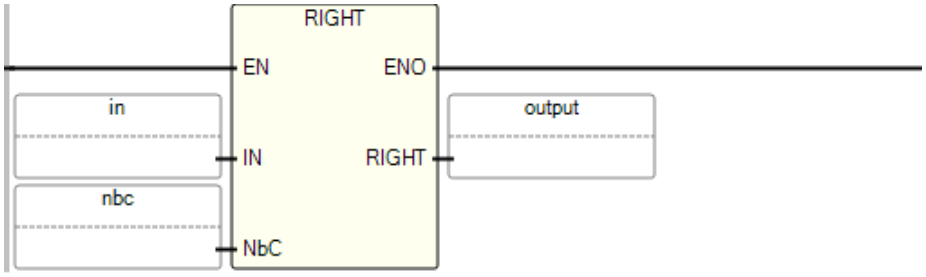
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : extraction du nombre de caractères spécifiés situés à l'extrémité droite de la chaîne. FAUX : aucune opération. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne non vide, quelle qu'elle soit.
NbC	Entrée	DINT	Nombre de caractères à extraire. Ce nombre ne peut pas être supérieur à la longueur de la chaîne IN.
RIGHT	Sortie	STRING	Partie droite de la chaîne (longueur = NbC). Il peut s'agir de l'une des valeurs suivantes : • chaîne vide si NbC <= 0 ; • chaîne complète si NbC >= longueur de la chaîne.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction RIGHT

Exemple de langage à relais RIGHT



Exemple de texte structuré RIGHT

```
RIGHT(
```

STRING **RIGHT**(STRING IN, DINT NbC)
Extraction de la partie droite d'une chaîne

```
1  nbc := 2;
2  output := RIGHT(in, nbc);
```

(* Équivalence ST : *)

complete_string := RIGHT ('12345678', 4), LEFT ('12345678', 4),5 ;

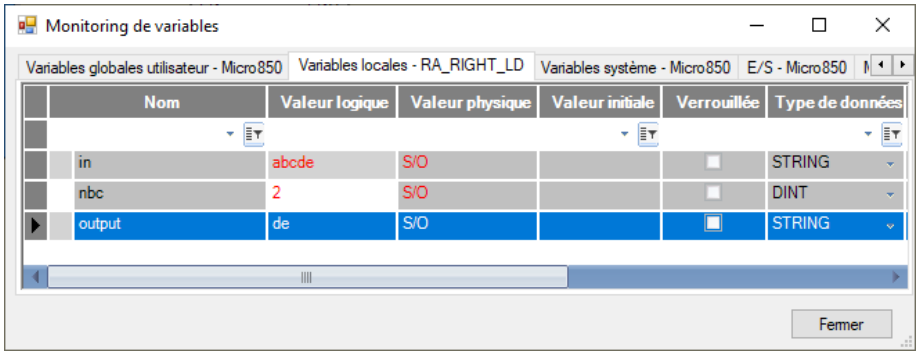
(* complete_string correspond à '56781234'

la valeur issue de l'appel RIGHT est égale à '5678'

la valeur issue de l'appel LEFT est égale à '1234'

*)

Résultats

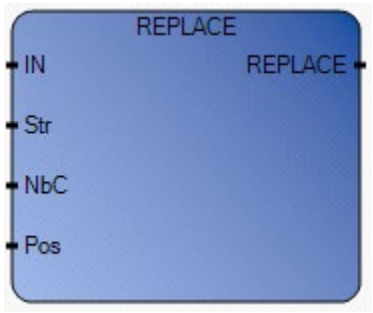


REPLACE (remplacer une sous-chaîne)

Remplace les parties d'une chaîne par de nouveaux jeux de caractères.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

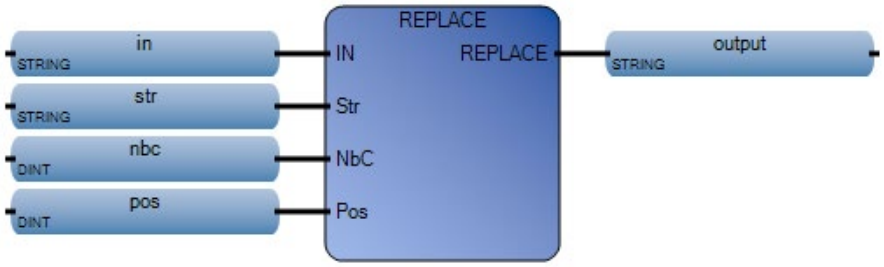
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



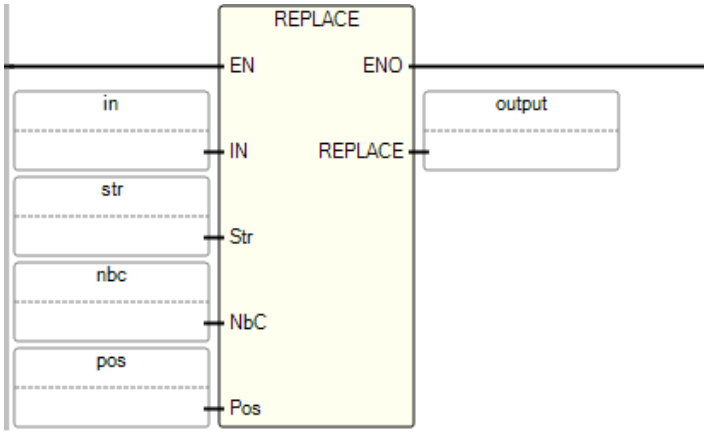
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de la fonction VRAI : remplacement de parties des chaînes par de nouveaux caractères. FAUX : aucune opération. S'applique aux programmes de schémas à relais.
IN	Entrée	STRING	Chaîne, quelle qu'elle soit.
Str	Entrée	STRING	Chaîne à insérer (pour remplacer les caractères NbC).
NbC	Entrée	DINT	Nombre de caractères à supprimer.
Pos	Entrée	DINT	Position du premier caractère modifié (la première position valide correspond à la position 1).
REPLACE	Sortie	STRING	Chaîne modifiée. Les caractères NbC sont supprimés de la position Pos, puis la sous-chaîne Str est insérée à cette position. Il peut s'agir de l'une des valeurs suivantes : • chaîne vide si Pos <= 0 ; • concaténation de chaînes (IN+Str) si la valeur Pos est supérieure à la longueur de la chaîne IN ; • chaîne initiale IN si NbC <= 0.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.

Exemple de Diagramme de Bloc Fonction REPLACE



Exemple de langage à relais REPLACE



Exemple de texte structuré REPLACE

```
REPLACE(
```

STRING **REPLACE**(STRING IN, STRING Str, DINT NbC, DINT Pos)
Remplace une sous-chaîne

```
1  nbc := 4;
2  pos := 2;
3  output := REPLACE(in, str, nbc, pos);
```

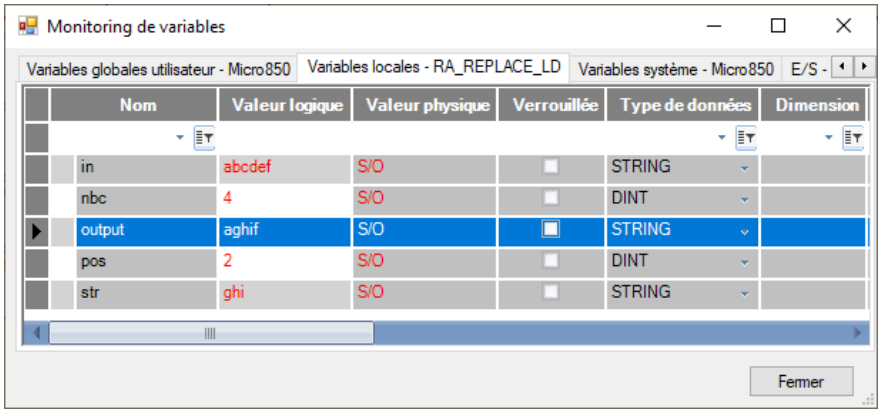
Remplacement d'une partie de chaîne par un nouveau jeu de caractères.

(* Équivalence ST : *)

MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4) ;

(* MyName correspond à 'Mr Frank JONES' *)

Résultats



Instructions de temporisateur

Utilisez les instructions de temporisateur pour contrôler les opérations basées sur le temps.

Instruction	Description
TOF sur la page 635	Temporisation interne sur front descendant. Augmente un temporisateur interne jusqu'à une valeur donnée.
TON sur la page 637	Temporisation interne sur front montant. Augmente un temporisateur interne jusqu'à une valeur donnée.
TONOFF sur la page 640	Retard dans l'activation d'une sortie sur une ligne Vrai, puis retard dans la désactivation de la sortie sur la ligne Faux.
TP sur la page 642	Temporisation interne avec comptage. Dans le cas d'un front montant, incrémente un temporisateur interne jusqu'à une valeur donnée.
RTO sur la page 644	Temporisation retenue. Incrémente un temporisateur interne lorsque son entrée est active, mais ne le réinitialise pas lorsque son entrée devient inactive.
DOY sur la page 646	Active une sortie si la valeur de l'horloge en temps réel est comprise dans la plage de valeurs propre au paramétrage de l'année.
TDF sur la page 648	Calcule la différence de temps entre TimeA et TimeB.
TOW sur la page 650	Active une sortie si la valeur de l'horloge en temps réel est comprise dans la plage de valeurs propre au paramétrage de la semaine.

Configuration de l'instruction de temporisateur

Le terme de « précision temporelle » se réfère au laps de temps situé entre le moment où le processeur active une instruction de temporisateur et celui où l'intervalle spécifié se termine.

Le processeur utilise les informations suivantes, dans l'instruction de temporisateur sur la page 635 :

- **Temporisateur (Timer) :** adresse de contrôle du temporisateur dans la zone temporisateur du stockage des données.
- **Base temporelle (Time Base) :** détermine le mode de fonctionnement du temporisateur.
- **Valeur prédéfinie (Preset) :** indique la valeur que le temporisateur doit atteindre pour que le processeur définisse le bit Done sur 1.
- **Valeur cumulée (Accumulated value) :** nombre d'incréments de temps comptés par l'instruction. Lorsque ce paramètre est activé, le temporisateur met à jour cette valeur en continu.

TOF (temporisateur de retard inactif)

Incrémente un temporisateur interne jusqu'à une valeur donnée.

Détails de l'opération :

- Si le paramètre EN est utilisé avec ce bloc d'instruction, le temporisateur commence l'incrémentation lorsque le paramètre EN

est défini sur VRAI, puis continue l'incrémentation même si EN est défini sur FAUX.

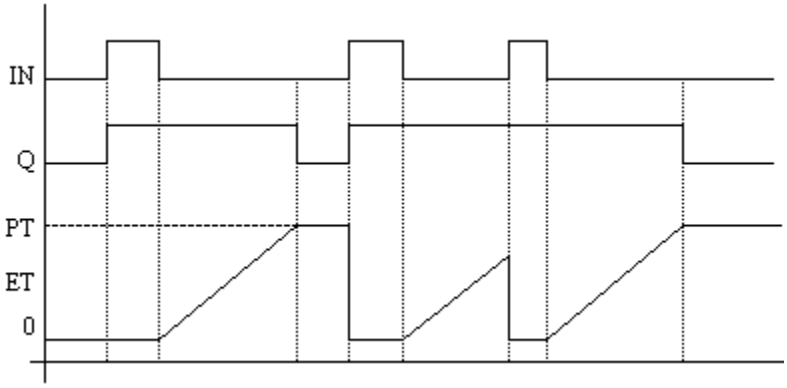
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

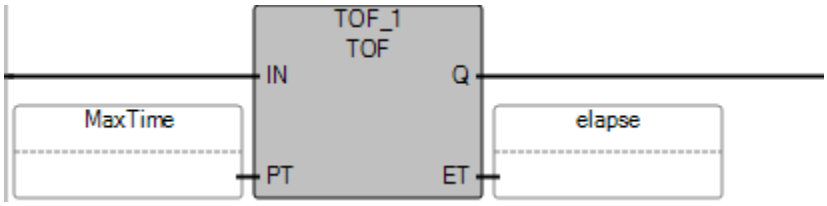
Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	Contrôle de l'entrée. TRUE : front descendant détecté, commence à incrémenter le temporisateur interne. FALSE : front montant détecté, arrête et réinitialise le temporisateur interne.
TI	Entrée	TIME	Temps programmé maximum. Voir Type de données TIME.
Q	Sortie	BOOL	VRAI : le temps total n'est **pas** écoulé. FAUX : le temps total est écoulé.
ET	Sortie	TIME	Temps actuel écoulé. Les valeurs possibles sont comprises entre 0 ms et 1193 h 2 m 47 s 294 ms.

Diagramme de temporisation TOF



Exemple de diagramme de bloc de fonctions TOF

Exemple de langage à relais TOF



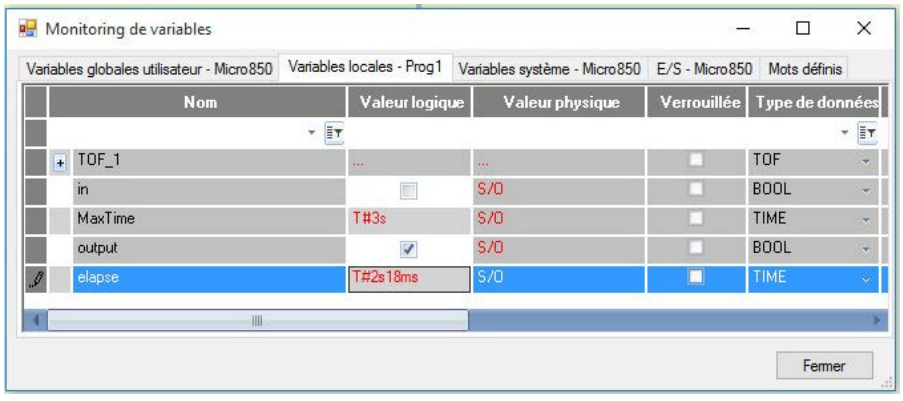
Exemple de texte structuré TOF

```
TOF_1(
```

void **TOF_1**(BOOL **IN**, TIME **PT**)
Type: TOF, Temporisation interne sur front descendant

```
1  MaxTime := T#3s;
2  TOF_1(in, MaxTime);
3  output := TOF_1.Q;
4  elapse := TOF_1.ET;
```

Résultats



TOF (temporisateur de retard actif)

Incrémente un temporisateur interne jusqu'à une valeur donnée.

Détails de l'opération :

- N'utilisez pas de saut pour sauter un bloc de fonctions TON dans un langage à relais (LD). Si un saut est utilisé, le temporisateur TON continue au-delà du temps écoulé.

- Par exemple : La ligne de programme 1 contient un saut ; la ligne de programme 2 contient un bloc d'instruction TON avec un temps écoulé de 10 secondes ; activez le saut de la ligne 1 à la ligne 3 ; désactivez le saut après 30 secondes ; le temps écoulé s'élève à 30 secondes et non 10 secondes comme défini dans le temps écoulé.
- Si le paramètre EN est utilisé avec TON, le temporisateur commence l'incrémentation lorsque le paramètre EN est défini sur VRAI, puis continue l'incrémentation même si EN est défini sur FAUX.

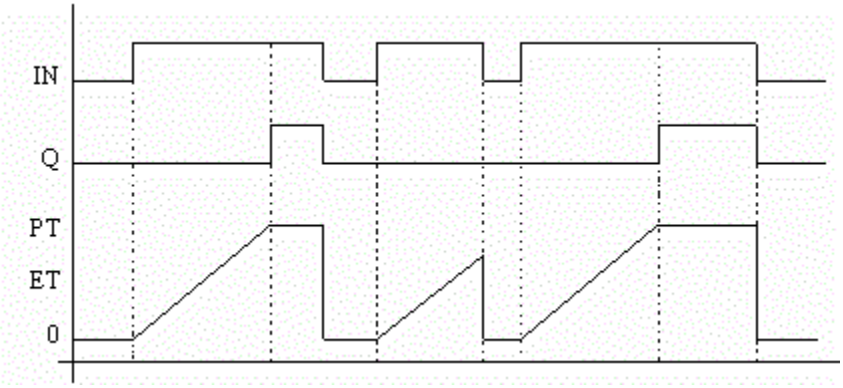
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

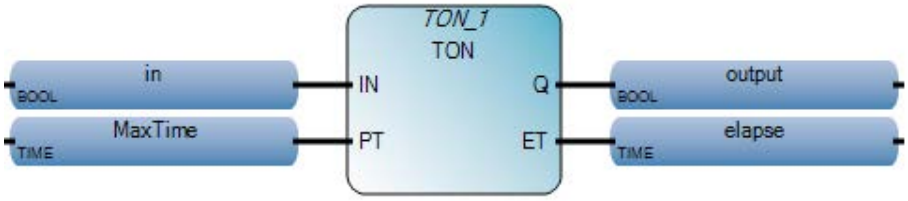
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	Contrôle de l'entrée. VRAI : dans le cas d'un front montant, commence à augmenter le temporisateur interne. FAUX : dans le cas d'un front descendant, arrête et réinitialise le temporisateur interne.
TI	Entrée	TIME	Temps maximal programmé défini en utilisant le type de données Temps.
Q	Sortie	BOOL	VRAI : le temps programmé est écoulé. FAUX : le temps programmé n'est pas écoulé.
ET	Sortie	TIME	Temps actuel écoulé. Les valeurs possibles sont comprises entre 0 ms et 1193 h 2 m 47 s 294 ms.

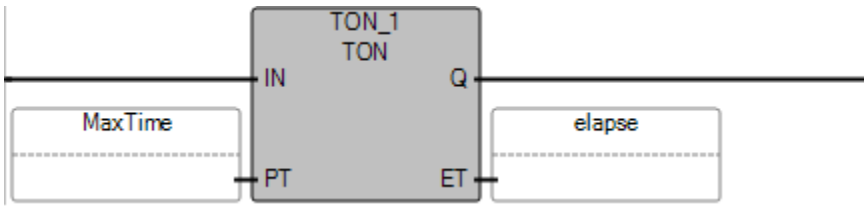
Diagramme de temporisation TON



Exemple de diagramme de bloc de fonctions TON



Exemple de langage à relais TON



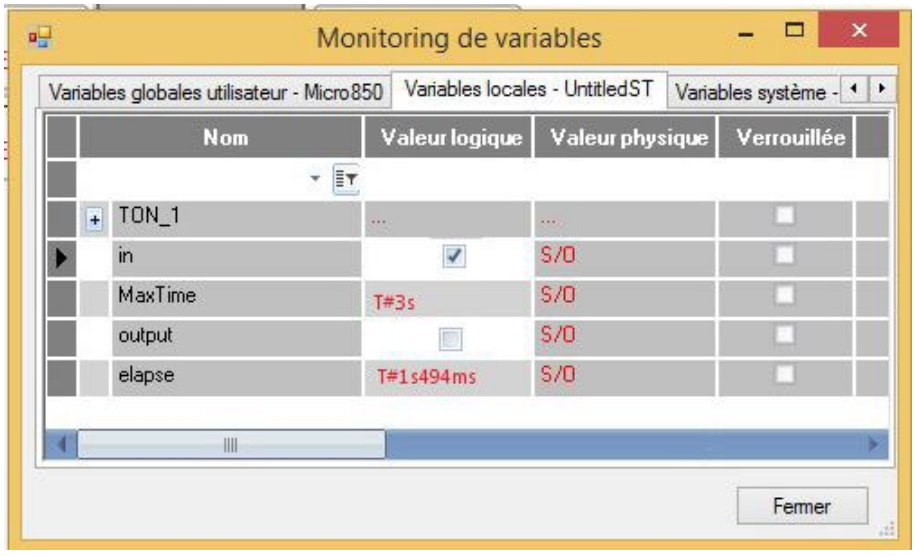
Exemple de texte structuré TON

```
TON_1(
```

void **TON_1**(BOOL **IN**, TIME **PT**)
Type: TON, Temporisation interne sur front montant

```
1  MaxTime := T#3s;
2  TON_1(in, MaxTime);
3  output := TON_1.Q;
4  elapse := TON_1.ET;
```

Résultats



TONOFF (retard, actif, inactif)

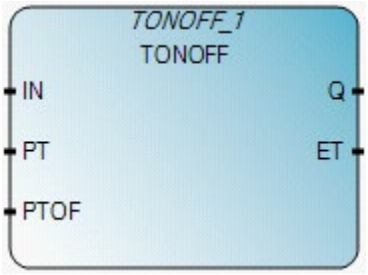
Retards dans l'activation d'une sortie sur une ligne de programme Vrai, puis retards dans la désactivation de la sortie sur la ligne de programme Faux.

Détails de l'opération :

- Si le paramètre EN est utilisé avec TONOFF, le temporisateur commence l'incrémentation lorsque le paramètre EN est défini sur VRAI, puis continue l'incrémentation même si EN est défini sur FAUX.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

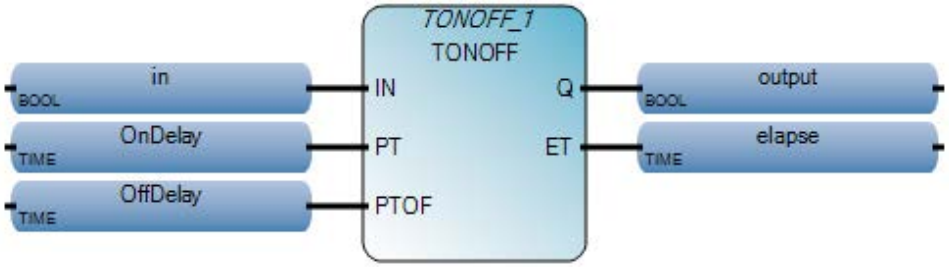
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



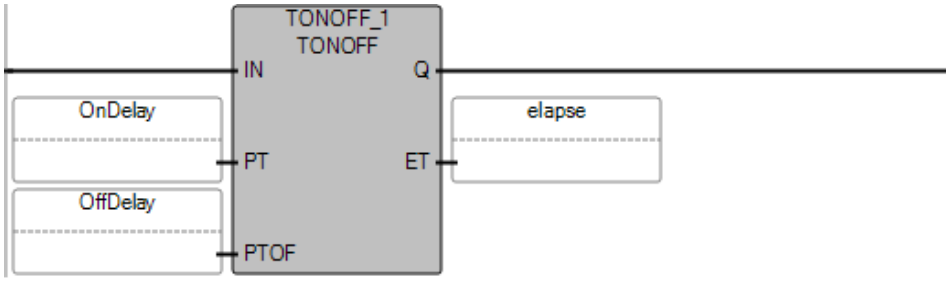
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	Contrôle de l'entrée. VRAI : front montant détecté (IN passe de 0 à 1) : • démarrez le temporisateur de retard actif (PT). • Si le temps de Retard au Déclenchement Programmé (PTOF) n'est pas écoulé, redémarrez le temporisateur de retard actif (PT). VRAI : front descendant détecté (IN passe de 1 à 0) : • si le temps de retard actif programmé (PT) n'est pas écoulé, arrêtez le temporisateur PT et réinitialisez ET ; • si le temps de retard actif programmé (PT) est écoulé, démarrez le temporisateur de retard inactif (PTOF).
TI	Entrée	TIME	Paramètre de temps de retard actif défini en utilisant le type de données Temps.
PTOF	Entrée	TIME	Paramètre de temps de retard inactif défini en utilisant le type de données Temps.
Q	Sortie	BOOL	VRAI : le temps de retard actif programmé est écoulé alors que le temps de Retard au Déclenchement Programmé ne l'est pas.
ET	Sortie	TIME	Temps actuel écoulé. Les valeurs possibles sont comprises entre 0 ms et 1193 h 2 m 47 s 294 ms. Si le délai programmé de retardement de l'activation est écoulé et que le temporisateur de retard inactif ne démarre pas, le temps écoulé (ET) correspond à la valeur de retardement de l'activation (PT). Si la durée de retard inactif programmé est écoulée et que le temporisateur de retardement de la désactivation ne démarre pas, le temps écoulé (ET) correspond à la valeur de retard inactif (PTOF) jusqu'au prochain front montant.

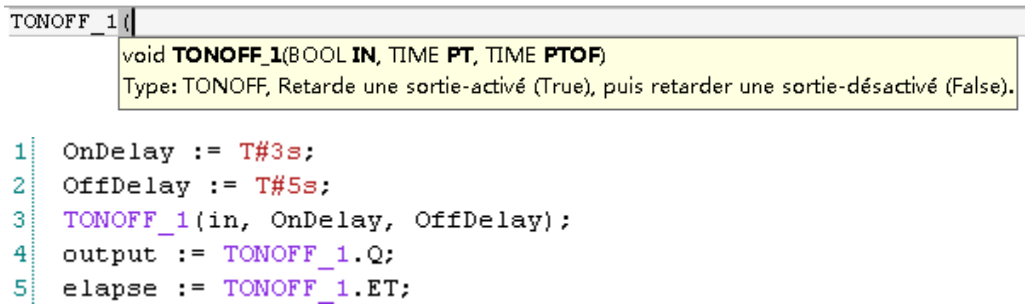
Exemple de diagramme de bloc de fonctions TONOFF



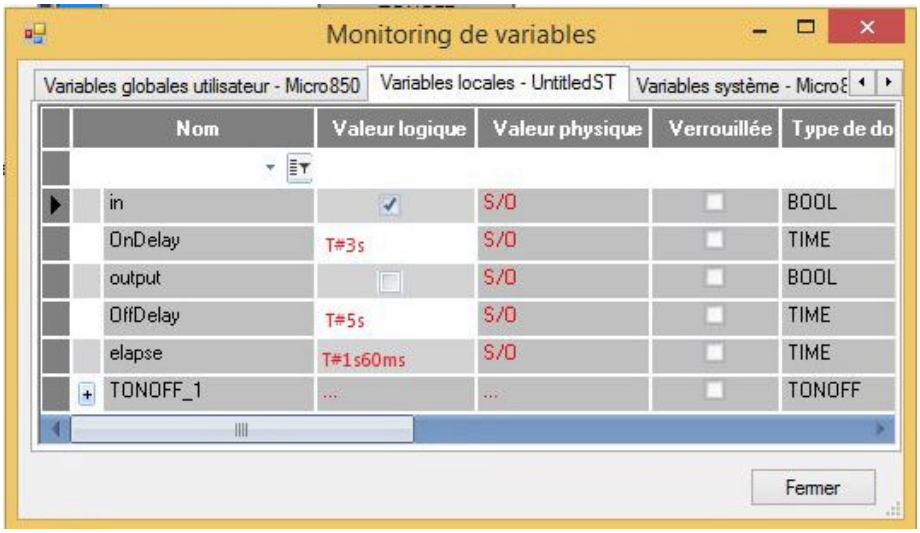
Exemple de langage à relais TONOFF

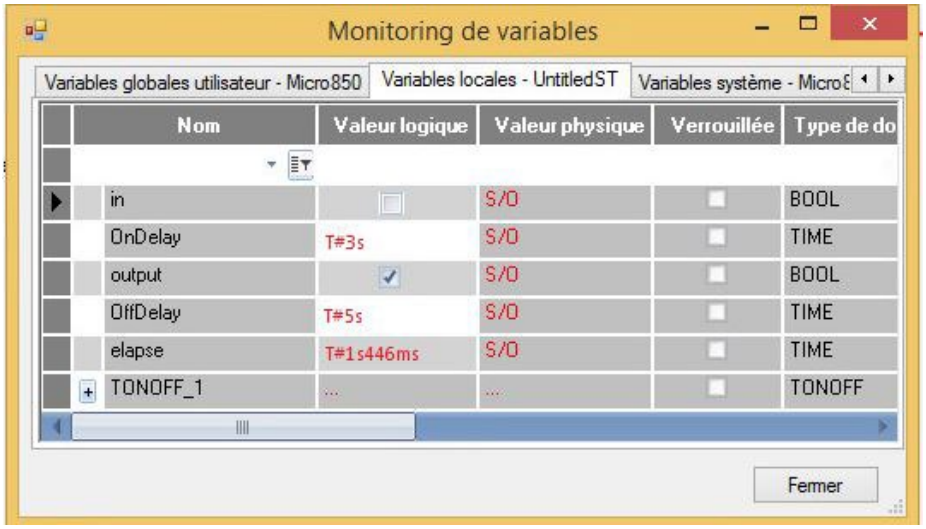


Exemple de texte structuré TONOFF



Résultats





TP (temporisation d'impulsion)

Dans le cas d'un front montant, incrémente un temporisateur interne jusqu'à une valeur donnée. Si le délai du temporisateur est écoulé, le temps interne est réinitialisé.

Détails de l'opération :

- Si le paramètre EN est utilisé avec TP, le temporisateur commence l'incrémentation lorsque le paramètre EN est défini sur VRAI, puis continue l'incrémentation même si EN est défini sur FAUX.

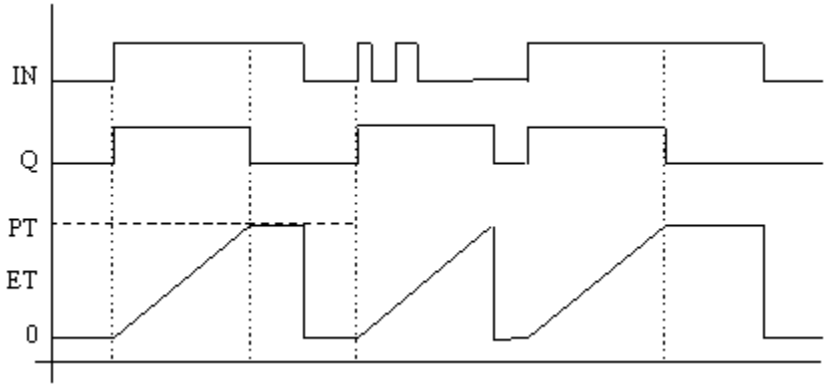
Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

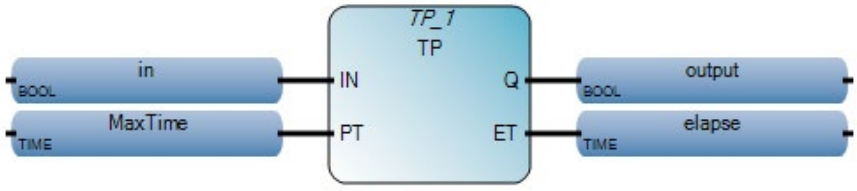
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	VRAI : dans le cas d'un front montant, commence à incrémenter le temporisateur interne (si l'incrémentation n'a pas encore été lancée). FAUX : si le délai du temporisateur est écoulé, réinitialise le temporisateur interne. Les changements apportés sur IN lors du comptage n'ont aucun effet.
TI	Entrée	TIME	Temps maximal programmé défini en utilisant le type de données Temps.
Q	Sortie	BOOL	VRAI : le temporisateur décompte. FALSE : le temporisateur ne compte pas.
ET	Sortie	TIME	Temps actuel écoulé. Les valeurs possibles sont comprises entre 0 ms et 1193 h 2 m 47 s 294 ms.

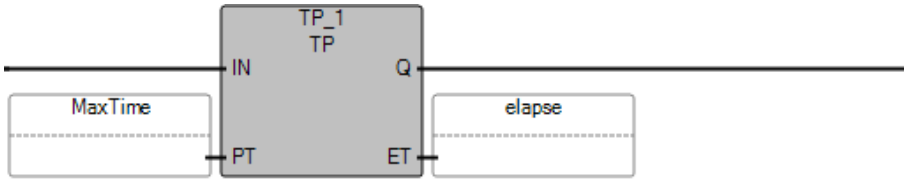
Diagramme de temporisation TP



Exemple de diagramme de bloc de fonctions TP



Exemple de langage à relais TP



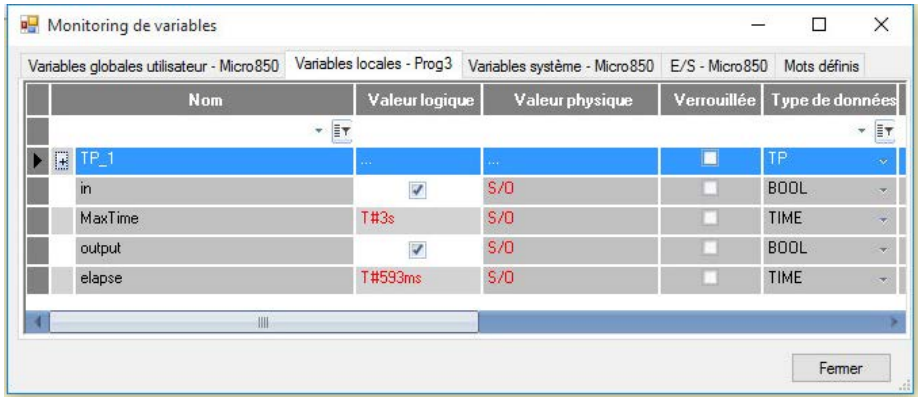
Exemple de texte structuré TP

```
TP_1(
```

void **TP_1**(BOOL **IN**, TIME **PT**)
Type: TP, Temporisation interne avec comptage

```
1  MaxTime := T#3s;
2  TP_1(in, MaxTime);
3  output := TP_1.Q;
4  elapse := TP_1.ET;
```

Résultats



RTO (temporisateur rémanent, retard actif)

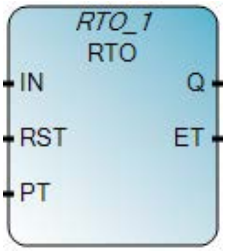
Incrémente un temporisateur interne lorsque son entrée est active, mais ne le réinitialise pas lorsque son entrée devient inactive.

Détails de l'opération :

- Avec un contrôleur Micro810 ou Micro820, le temporisateur interne du RTO n'est pas conservé après un arrêt et une remise sous tension (par défaut). Pour conserver le temporisateur interne, définissez le paramètre de configuration Conservé sur Vrai.
- Avec un contrôleur Micro830 ou Micro850, le temporisateur interne du RTO est conservé après un arrêt et une remise sous tension.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

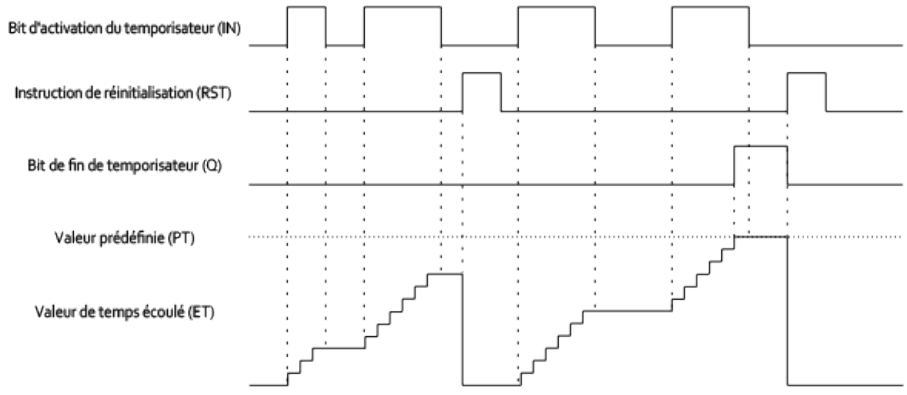
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



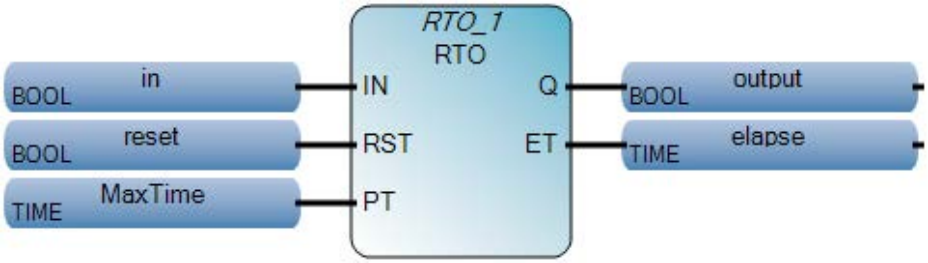
Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

Paramètre	Type de paramètre	Type de données	Description
IN	Entrée	BOOL	Contrôle de l'entrée. VRAI : dans le cas d'un front montant, commence à augmenter le temporisateur interne. FAUX : dans le cas d'un front descendant, arrête et ne réinitialise pas le temporisateur interne.
RST	Entrée	BOOL	VRAI : dans le cas d'un front montant, réinitialise le temporisateur interne. FAUX : ne réinitialise pas le temporisateur interne.
TI	Entrée	TIME	Temps de retard actif programmé maximal. PT est défini à l'aide du type de données Temps.
Q	Sortie	BOOL	VRAI : le temps de retard actif programmé est écoulé. FAUX : le temps de retard actif programmé n'est pas écoulé.
ET	Sortie	TIME	Temps actuel écoulé. Les valeurs sont comprises entre 0 ms et 1 193 h 2 m 47 s 294 ms. ET est défini à l'aide du type de données Temps.

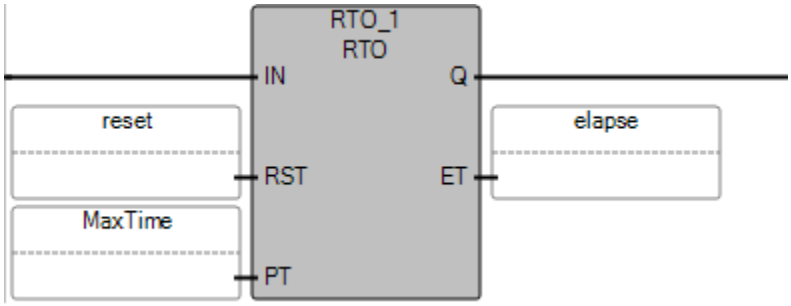
Exemple de diagramme de temporisation RTO



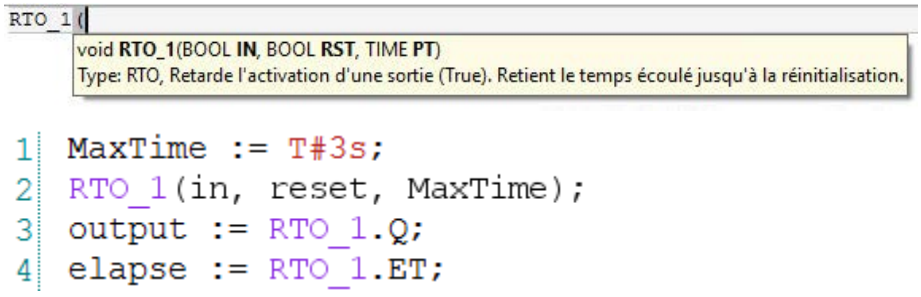
Exemple de diagramme de bloc de fonctions RTO



Exemple de langage à relais RTO



Exemple de texte structuré RTO



DOY (vérifier l'année pour l'horloge temps réel)

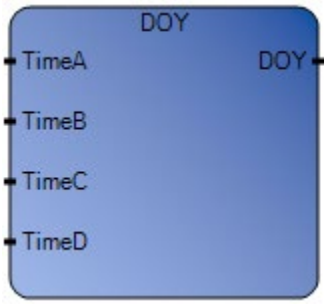
Active une sortie lorsque la valeur de l'horloge en temps réel (RTC) est comprise dans la plage de valeurs propre au paramétrage de l'année.

Détails de l'opération :

- En cas d'absence RTC, la sortie est toujours désactivée.
- Configure les paramètres d'entrée Temps avec des valeurs de plages valides, telles que définies dans le tableau Type de données DOYDATA. Si TimeX.Enable est défini sur VRAI et que RTC est présent et activé, une valeur non valide empêche le bon fonctionnement du contrôleur.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.



Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

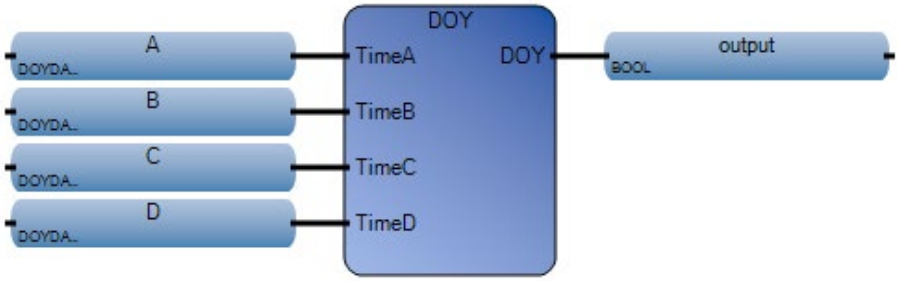
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution de l'opération. FAUX : aucune exécution de l'opération.
TimeA	Entrée	DOYDATA	Paramétrage de l'année du canal A. Utilisez le type de données DOYDATA pour configurer TimeA.
TimeB	Entrée	DOYDATA	Paramétrage de l'année du canal B. Utilisez le type de données DOYDATA pour configurer TimeB.
TimeC (HeureC)	Entrée	DOYDATA	Paramétrage de l'année du canal C. Utilisez le type de données DOYDATA pour configurer TimeC.
TimeD (HeureD)	Entrée	DOYDATA	Paramétrage de l'année du canal D. Utilisez le type de données DOYDATA pour configurer TimeD.
DOY	Sortie	BOOL	Si TRUE, la valeur de l'horloge en temps réel est comprise dans la plage propre au paramétrage de l'année de l'un des quatre canaux.

Type de données DOYDATA

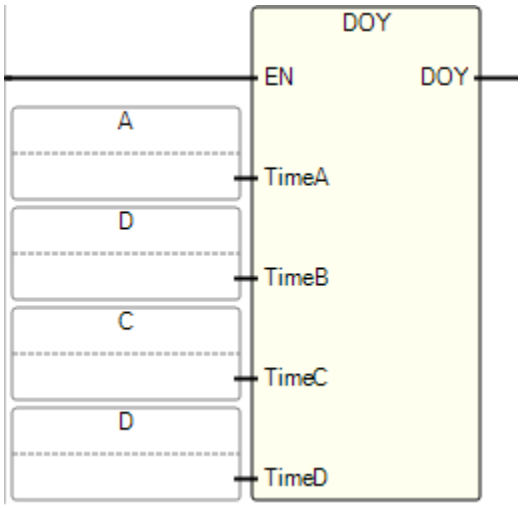
Le tableau suivant décrit le type de données DOYDATA.

Paramètre	Type de données	Description
Enable	BOOL	TRUE : activation ; FALSE : désactivation
YearlyCenturial	BOOL	Type de temporisateur (0 : temporisateur annuel ; 1 : temporisateur séculaire).
YearOn	UINT	Valeur d'activation de l'année (doit être comprise dans l'ensemble [2000...2098]).
MonthOn	USINT	Valeur d'activation du mois (doit être comprise dans l'ensemble [1...12]).
DayOn	USINT	Valeur d'activation du jour (doit être comprise dans l'ensemble [1...31], déterminée par la valeur « MonthOn »).
YearOff	UINT	Valeur de désactivation de l'année (doit être comprise dans l'ensemble [2000...2098]).
MonthOff	USINT	Valeur de désactivation du mois (doit être comprise dans l'ensemble [1...12]).
DayOff	USINT	Valeur de désactivation du jour (doit être comprise dans l'ensemble [1...31], déterminée par la valeur « MonthOff »).

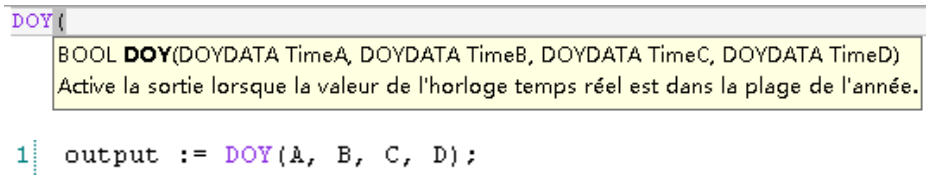
Exemple de Diagramme de Bloc Fonction DOY



Exemple de langage à relais DOY



Exemple de texte structuré DOY




```
1  output := DOY(A, B, C, D);
```

(* Équivalence ST : *)

TESTOUTPUT := DOY(TIMEA1, TIMEB1, TIMEC1, TIMED1) ;

TDF (différence de temps)

Calcule la différence de temps entre TimeA et TimeB.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

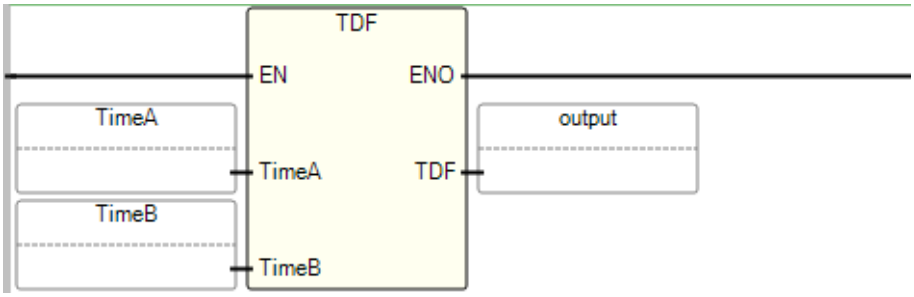
Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

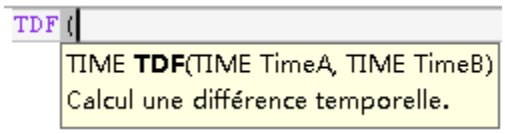
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. VRAI : exécution du calcul actuel. FAUX : aucun calcul. S'applique aux programmes de schémas à relais.
TimeA	Entrée	TIME	Temps de départ pour le calcul de la différence temporelle.
TimeB	Entrée	TIME	Temps de fin pour le calcul de la différence temporelle.
ENO	Sortie	BOOL	Activation de la sortie. S'applique aux programmes de schémas à relais.
TDF	Sortie	TIME	Différence de temps entre les deux entrées temporelles. TDF correspond au nom ou à l'ID PIN

Exemple de Diagramme de Bloc Fonction TDF

Exemple de logique à relais TDF



Exemple de texte structuré TDF

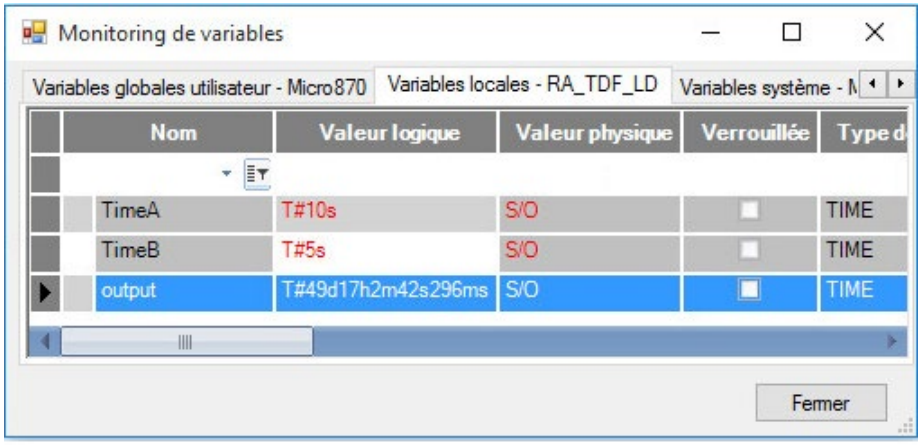



```
1  TimeA := T#10s;
2  TimeB := T#5s;
3  output := TDF(TimeA, TimeB);
```

(* Équivalence ST : *)

TESTTIMEDIFF := TDF(TESTTIME1, TESTTIME2) ;

Résultats



TOW (vérifier la semaine pour l'horloge temps réel)

Active une sortie si la valeur de l'horloge en temps réel (RTC) est comprise dans la plage de valeurs propre au paramétrage de la semaine.

Détails de l'opération :

- En cas d'absence RTC, la sortie est toujours désactivée.
- Configure les paramètres d'entrée Temps avec des valeurs de plages valides, telles que définies dans le tableau Type de données TOWDATA. Si TimeX.Enable est défini sur VRAI et que RTC est présent et activé, une valeur non valide empêche le bon fonctionnement du contrôleur.

Langages pris en charge : Diagramme de Bloc Fonction, diagramme de schéma à relais et texte structuré.

Cette instruction s'applique aux contrôleurs Micro810, Micro820, Micro830, Micro850 et Micro870 ainsi qu'à Micro800 Simulator.

Ce tableau permet de déterminer les valeurs de paramètre pour cette instruction.

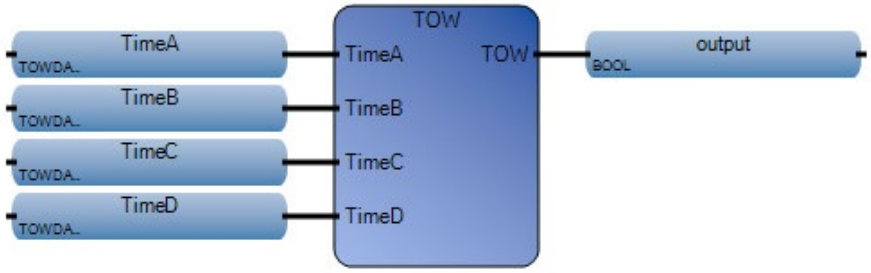
Paramètre	Type de paramètre	Type de données	Description
EN	Entrée	BOOL	Activation de l'instruction. Lorsque EN = VRAI, exécute l'opération. Lorsque EN = FAUX, n'exécute pas l'opération.
TimeA	Entrée	TOWDATA	Paramétrage du jour sur le canal A. Utilisez le type de données TOWDATA pour définir TimeA.
TimeB	Entrée	TOWDATA	Paramétrage du jour sur le canal B. Utilisez le type de données TOWDATA pour définir TimeB.
TimeC (HeureC)	Entrée	TOWDATA	Paramétrage du jour sur le canal C. Utilisez le type de données TOWDATA pour définir TimeC.
TimeD (HeureD)	Entrée	TOWDATA	Paramétrage du jour sur le canal D. Utilisez le type de données TOWDATA pour définir TimeD.
TOW	Sortie	BOOL	Si TRUE, la valeur de l'horloge en temps réel est comprise dans la plage propre au paramétrage du jour de l'un des quatre canaux.

Type de données TOWDATA

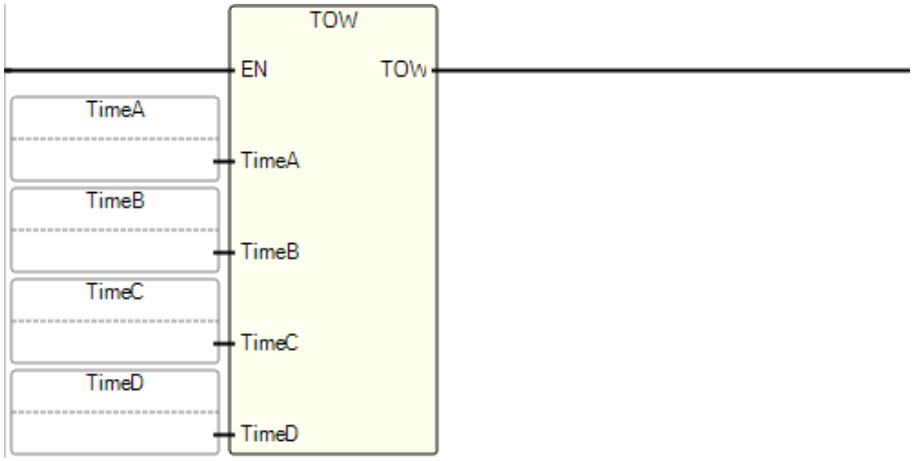
Le tableau suivant décrit le type de données TOWDATA :

Paramètre	Type de données	Description
Enable	BOOL	TRUE : activation ; FALSE : désactivation.
DailyWeekly	BOOL	Type de temporisateur (0 : temporisateur quotidien ; 1 : temporisateur hebdomadaire).
DayOn	USINT	Valeur d'activation du jour de la semaine (doit être comprise dans l'ensemble [0...6]).
HourOn	USINT	Valeur d'activation de l'heure (doit être comprise dans l'ensemble [0...23]).
MinOn	USINT	Valeur d'activation de la minute (doit être comprise dans l'ensemble [0...59]).
DayOff	USINT	Valeur de désactivation du jour de la semaine (doit être comprise dans l'ensemble [0...6]).
HourOff	USINT	Valeur de désactivation de l'heure (doit être comprise dans l'ensemble [0...23]).
MinOff	USINT	Valeur de désactivation de la minute (doit être comprise dans l'ensemble [0...59]).

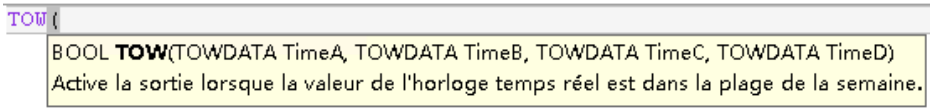
Exemple de Diagramme de Bloc Fonction TOW



Exemple de langage à relais TOW



Exemple de texte structuré TOW




```
1  output := TOW(TimeA, TimeB, TimeC, TimeD);
```

(* Équivalence ST : *)

TESTOUTPUT := TOW(TIMEA, TIMEB, TIMEC, TIMED) ;

Résultats

Nom	Valeur logique	Valeur physique	Verrouillée	Type de données
TimeA	...	...	☐	TOWDATA
TimeA.Enable	☑	S/O	☐	BOOL
TimeA.DailyWeekly	☑	S/O	☐	BOOL
TimeA.DayOn	1	S/O	☐	USINT
TimeA.HourOn	10	S/O	☐	USINT
TimeA.MinOn	20	S/O	☐	USINT
TimeA.DayOff	2	S/O	☐	USINT
TimeA.HourOff	15	S/O	☐	USINT
TimeA.MinOff	30	S/O	☐	USINT
TimeB	...	...	☐	TOWDATA
TimeB.Enable	☑	S/O	☐	BOOL
TimeB.DailyWeekly	☑	S/O	☐	BOOL
TimeB.DayOn	2	S/O	☐	USINT
TimeB.HourOn	15	S/O	☐	USINT
TimeB.MinOn	20	S/O	☐	USINT
TimeB.DayOff	3	S/O	☐	USINT
TimeB.HourOff	10	S/O	☐	USINT
TimeB.MinOff	30	S/O	☐	USINT
TimeC	...	...	☐	TOWDATA
TimeC.Enable	☑	S/O	☐	BOOL
TimeC.DailyWeekly	☑	S/O	☐	BOOL
TimeC.DayOn	3	S/O	☐	USINT
TimeC.HourOn	20	S/O	☐	USINT
TimeC.MinOn	10	S/O	☐	USINT
TimeC.DayOff	4	S/O	☐	USINT
TimeC.HourOff	25	S/O	☐	USINT
TimeC.MinOff	55	S/O	☐	USINT
TimeD	...	...	☐	TOWDATA
TimeD.Enable	☑	S/O	☐	BOOL
TimeD.DailyWeekly	☑	S/O	☐	BOOL
TimeD.DayOn	5	S/O	☐	USINT
TimeD.HourOn	10	S/O	☐	USINT
TimeD.MinOn	15	S/O	☐	USINT
TimeD.DayOff	6	S/O	☐	USINT
TimeD.HourOff	35	S/O	☐	USINT
TimeD.MinOff	40	S/O	☐	USINT
output	☑	S/O	☐	BOOL

Indice

A

à impulsion sur front descendant (négatif)
bobines 44
contacts 51
à impulsion sur front montant (positif)
bobines 45
contacts 51
contacts dans FBD 51
AFI 523
alarmes
bloc de fonctions LIM_ALRM 61
appel
blocs fonctionnels 21
arithmétique
fonction ABS 65
fonction ACOS 67
fonction ACOS_LREAL 68
fonction ASIN 71
fonction ASIN_LREAL 73
fonction ATAN 74
fonction ATAN_LREAL 75
fonction COS 77
fonction COS_LREAL 78
fonction EXPT 81
fonction LOG 83
fonction MOD 84
fonction POW 89
fonction RAND 91
fonction SIN 93
fonction SQRT 96
fonction TAN 98
fonction TAN_LREAL 100
fonction TRUNC 101
opérateur d'addition 70
opérateur de division 80
opérateur de multiplication 87
opérateur de soustraction 97
opérateur d'inv. 89
opérateur MOV 86
SIN_LREAL 94

B

bloc de fonctions ABL 103
bloc de fonctions ACB 109
bloc de fonctions ACL 105
bloc de fonctions AHL 107
bloc de fonctions ARD 111
bloc de fonctions ARL 114
bloc de fonctions AVERAGE 275
bloc de fonctions AWA 116
bloc de fonctions AWT 118
bloc de fonctions COP 276
bloc de fonctions CTD 249
bloc de fonctions CTU 251
bloc de fonctions CTUD 252
bloc de fonctions DERIVATE 477
bloc de fonctions F_TRIG 143
bloc de fonctions HSC 288
bloc de fonctions HSC_SET_STS 304
bloc de fonctions HYSTER 492
bloc de fonctions IIM 350
bloc de fonctions INTEGRAL 494
bloc de fonctions IOM 352
bloc de fonctions IPIDCONTROLLER 527
bloc de fonctions KEY_READ 355
bloc de fonctions LIM_ALRM 61
bloc de fonctions MC_AbortTrigger 413
bloc de fonctions MC_Halt 416
bloc de fonctions MC_Home 419
bloc de fonctions MC_MoveAbsolute 422
bloc de fonctions MC_MoveRelative 426
bloc de fonctions MC_MoveVelocity 430
bloc de fonctions MC_Power 435
bloc de fonctions MC_ReadAxisError 445
bloc de fonctions MC_ReadBoolParameter 449
bloc de fonctions MC_ReadParameter 451
bloc de fonctions MC_ReadStatus 454
bloc de fonctions MC_Reset 458
bloc de fonctions MC_SetPosition 461
bloc de fonctions MC_Stop 463
bloc de fonctions MC_TouchProbe 466
bloc de fonctions MC_WriteBoolParameter 470
bloc de fonctions MC_WriteParameter 473
bloc de fonctions MMINFO 359
bloc de fonctions MSG_CIPGENERIC 163
bloc de fonctions MSG_CIPSYMBOLIC 171
bloc de fonctions MSG_MODBUS 175
bloc de fonctions MSG_MODBUS2 181
bloc de fonctions PLUGIN_INFO 373
bloc de fonctions PLUGIN_READ 375
bloc de fonctions PLUGIN_RESET 377
bloc de fonctions PLUGIN_WRITE 379
bloc de fonctions R_TRIG 145
bloc de fonctions RS 146
bloc de fonctions RTC_READ 383
bloc de fonctions RTC_SET 385
bloc de fonctions SCALER 507
bloc de fonctions SR 150
bloc de fonctions STACKINT 509
bloc de fonctions SUS 523
bloc de fonctions SYS_INFO 387

bloc de fonctions TOF 635
bloc de fonctions TON 637
bloc de fonctions TONOFF 640
bloc de fonctions TP 642
bloc de fonctions TRIMPOT_READ 390
Bloc fonctionnel RTO 644
blocs de contrôle avancés
SCALER 507
Blocs de fonctions contrôle du mouvement 403
blocs fonctionnels
ABL 103
ACB 109
ACL 105
AHL 107
appel 21
ARD 111
ARL 114
AVERAGE 275
AWA 116
AWT 118
COP 276
CTD 249
CTU 251
CTUD 252
DERIVATE 477
F_TRIG 143
FFL 479
FFU 487
HSC 288
HSC_SET_STS 304
HYSTER 492
IIM 350
INTEGRAL 494
IOM 352
IPIDCONTROLLER 527
KEY_READ 355
LIM_ALRM 61
MC_AbortTrigger 413
MC_Halt 416
MC_Home 419
MC_MoveAbsolute 422
MC_MoveRelative 426
MC_MoveVelocity 430
MC_Power 435
MC_ReadAxisError 445
MC_ReadBoolParameter 449
MC_ReadParameter 451
MC_ReadStatus 454
MC_Reset 458
MC_SetPosition 461
MC_Stop 463
MC_TouchProbe 466
MC_WriteBoolParameter 470
MC_WriteParameter 473
MMINFO 359
MODULE_INFO 362
MSG_CIPGENERIC 163
MSG_CIPSYMBOLIC 171
MSG_MODBUS 175
MSG_MODBUS2 181
noms et paramètres pour 21
PID 552
PLUGIN_INFO 373
PLUGIN_READ 375
PLUGIN_RESET 377
PLUGIN_WRITE 379
R_TRIG 145
RS 146
RTC_READ 383
RTC_SET 385
RTO 644
SCALER 507
SOCKET_ACCEPT 568
SOCKET_CREATE 571
SOCKET_DELETE 576
SOCKET_DELETEALL 578
SOCKET_INFO 580
SOCKET_OPEN 586
SOCKET_READ 590
SOCKET_WRITE 594
SR 150
STACKINT 509
SUS 523
SYS_INFO 387
TOF 635
TON 637
TONOFF 640
TP 642
TRIMPOT_READ 390
bobine de déclenchement dans FBD 46
bobines
à impulsion sur front descendant (positif) 44
à impulsion sur front montant (positif) 45
déclenchement 46
déclenchement dans FBD 46
enclenchement 45
enclenchement dans FBD 45
insertion 40
insertion dans FBD 40
modification des types dans FBD 40
modification des types de 40
parallèle, insertion 40
parallèle, insertion dans FBD 40
type direct 43
type direct dans FBD 43
type inversé 43
type inversé dans FBD 43
utilisation et types disponibles de 40
utilisation et types disponibles pour FBD 40
booléen
fonction MUX4B 158

fonction MUX8B 154
fonction TTABLE 152
branches
insertion 37
pour LD 37
branches parallèles pour diagrammes de LD 37

C

COM_IO_WDOG 161
communications
bloc de fonctions ABL 103
bloc de fonctions ACL 105
bloc de fonctions AHL 107
bloc de fonctions ARD 111
bloc de fonctions ARL 114
bloc de fonctions AWA 116
bloc de fonctions AWT 118
bloc de fonctions MSG_CIPGENERIC 163
bloc de fonctions MSG_CIPSYMBOLIC 171
bloc de fonctions MSG_MODBUS 175
bloc de fonctions MSG_MODBUS2 181
blocs de fonctions ACB 109
compteur
bloc de fonctions CTD 249
bloc de fonctions CTU 251
bloc de fonctions CTUD 252
contacts
à impulsion sur front descendant (positif) 51
à impulsion sur front montant (positif) 51
à impulsion sur front montant (positif) dans FBD 51
direct 50
directs dans FBD 50
insertion 47
inversé 50
inversés dans FBD 50
modification du type de 47
parallèle, insertion 47
utilisation et types disponibles dans FBD 47
utilisation et types disponibles de 47
contrôle de processus
bloc de fonctions DERIVATE 477
bloc de fonctions HYSTER 492
bloc de fonctions INTEGRAL 494
bloc de fonctions IPIDCONTROLLER 527
bloc de fonctions STACKINT 509
fonction LIMIT 519
LFL(LIFO load) 500
LFU (déchargement FIFO) 502
contrôle de programme
bloc de fonctions SUS 523
fonction TND 518
conversion de données
opérateur ANY_TO_BOOL 255
opérateur ANY_TO_BYTE 256
opérateur ANY_TO_DATE 257
opérateur ANY_TO_DINT 259
opérateur ANY_TO_DWORD 260
opérateur ANY_TO_INT 261
opérateur ANY_TO_LINT 262
opérateur ANY_TO_LREAL 263
opérateur ANY_TO_LWORD 264
opérateur ANY_TO_REAL 265
opérateur ANY_TO_SINT 266
opérateur ANY_TO_STRING 267
opérateur ANY_TO_TIME 268
opérateur ANY_TO_UDINT 269
opérateur ANY_TO_UINT 270
opérateur ANY_TO_ULINT 271
opérateur ANY_TO_USINT 272
opérateur ANY_TO_WORD 273

D

déclenchement
bobines 46
dénomination des conventions
blocs fonctionnels 21
Description des blocs d'instruction 163, 171, 181, 275, 276
Description des blocs d'instruction pour
ABL 103
ACB 109
ACL 105
AHL 107
ARD 111
ARL 114
AVERAGE 275
AWA 116
AWT 118
COP 276
CTD 249
CTU 251
CTUD 252
DERIVATE 477
F_TRIG 143
HSC 288
HSC_SET_STS 304
HYSTER 492
IIM 350
INTEGRAL 494
IOM 352
IPIDCONTROLLER 527
KEY_READ 355
LIM_ALRM 61
MC_AbortTrigger 413
MC_Halt 416
MC_Home 419
MC_MoveAbsolute 422
MC_MoveRelative 426

MC_MoveVelocity 430
MC_Power 435
MC_ReadAxisError 445
MC_ReadBoolParameter 449
MC_ReadParameter 451
MC_ReadStatus 454
MC_Reset 458
MC_SetPosition 461
MC_Stop 463
MC_TouchProbe 466
MC_WriteBoolParameter 470
MC_WriteParameter 473
MM_INFO 359
MSG_CIPGENERIC 163
MSG_CIPSYMBOLIC 171
MSG_MODBUS 175
MSG_MODBUS2 181
PLUGIN_INFO 373
PLUGIN_READ 375
PLUGIN_RESET 377
PLUGIN_WRITE 379
R_TRIG 145
RST 146
RTC_READ 383
RTC_SET 385
RTO 644
SCALER 507
SRT 150
STACKINT 509
SUS 523
SYS_INFO 387
TOF 635
TON 637
TONOFF 640
TP 642
TRIMPOT_READ 390

direct
bobines 43
bobines dans FBD 43
contacts 50
contacts dans FBD 50

E

entrée/sortie
bloc de fonctions HSC 288
bloc de fonctions HSC_SET_STS 304
bloc de fonctions IIM 350
bloc de fonctions IOM 352
bloc de fonctions KEY_READ 355
bloc de fonctions MMINFO 359
bloc de fonctions PLUGIN_INFO 373
bloc de fonctions PLUGIN_READ 375
bloc de fonctions PLUGIN_RESET 377
bloc de fonctions PLUGIN_WRITE 379
bloc de fonctions RTC_READ 383
bloc de fonctions RTC_SET 385
bloc de fonctions SYS_INFO 387
bloc de fonctions TRIMPOT_READ 390
fonction LCD 340
fonction RHC 346
fonction RPC 348

étiquettes
lignes, insertion 33

F

FBD (diagramme de bloc de fonctions)
bobines de déclenchement 46
bobines d'enclenchement 45
bobines directes 43
bobines inversées 43
bobines, utilisation et types disponibles de 40
contacts à impulsion sur front montant (positifs) 51
contacts normalement fermés 50
contacts normalement ouverts 50
contacts, utilisation et types disponibles de 47
lignes, insertion 33
retours, insertion 52
sauts vers étiquettes, insertion 53
FFL 479
FFU 487
fonction ABS 65
fonction ACOS 67
fonction ACOS_LREAL 68
fonction AND_MASK 123
fonction ASCII 615
fonction ASIN 71
fonction ASIN_LREAL 73
fonction ATAN 74
fonction ATAN_LREAL 75
fonction CHAR 617
fonction COS 77
fonction COS_LREAL 78
fonction DOY 646
fonction EXPT 81
fonction FIND 620
fonction INSERT 622
fonction LCD 340
fonction LEFT 624
fonction LIMIT 519
fonction LOG 83
fonction MAX 283
fonction MID 626
fonction MIN 282
fonction MLEN 628
fonction MOD 84
fonction MUX4B 158
fonction MUX8B 154
fonction NOT_MASK 131
fonction OR_MASK 132
fonction POW 89
fonction RAND 91

fonction REPLACE 631
fonction RHC 346
fonction RIGHT 629
fonction ROL 134
fonction ROR 135
fonction RPC 348
fonction SHL 137
fonction SHR 139
fonction SIN 93
fonction SQRT 96
fonction STIS 393
fonction TAN 98
fonction TAN_LREAL 100
fonction TDF 648
fonction TND 518
fonction TOW 650
fonction TRUNC 101
fonction TTABLE 152
fonction UIC 395
fonction UID 396
fonction UIE 398
fonction UIF 400
fonction XOR_MASK 140
fonctions
ABS 65
ACOS 67
ACOS_LREAL 68
AND_MASK 123
ASCII 615
ASIN 71
ASIN_LREAL 73
ATAN 74
ATAN_LREAL 75
CHAR 617
COS 77
COS_LREAL 78
DELETE 618
DOY 646
EXPT 81
FIND 620
INSERT 622
LCD 340
LEFT 624
LIMIT 519
LOG 83
MAX 283
MID 626
MIN 282
MLEN 628
MOD 84
MUX4B 158
MUX8B 154
NOT_MASK 131
OR_MASK 132
POW 89
RAND 91
REPLACE 631
RHC 346
RIGHT 629
ROL 134
ROR 135
RPC 348
SHL 137
SHR 139
SIN 93
SIN_LREAL 94
SQRT 96
STIS 393
TAN 98
TAN_LREAL 100
TDF 648
TND 518
TOW 650
TRUNC 101
TTABLE 152
UIC 395
UID 396
UIE 398
UIF 400
XOR_MASK 140

I

insertion
étiquettes, lignes 33
lignes (éléments de FBD) 33
retours (éléments de FBD) 52
sauts vers étiquettes (éléments de FBD) 53
instructions 29
instructions de retour
insertion 52
pour diagrammes de LD 52
interruptions
fonction STIS 393
fonction UIC 395
fonction UID 396
fonction UIE 398
fonction UIF 400
inversé
bobines 43
bobines dans FBD 43
contacts 50
contacts dans FBD 50

L

langage de LD
raccourcis clavier 57
langue
Ladder Diagram (LD) 29
LD (logique à relais)
bobines à impulsion sur front descendant (négatives) 44

bobines à impulsion sur front montant (positives) 45
bobines de déclenchement 46
bobines d'enclenchement 45
bobines directes 43
bobines inversées 43
bobines, utilisation et types disponibles de 40
branches 37
contacts à impulsion sur front descendant (négatifs) 51
contacts à impulsion sur front montant (positifs) 51
contacts normalement fermés 50
contacts normalement ouverts 50
contacts, utilisation et types disponibles de 47
instructions de retour 52
lignes, insertion 33
sauts 53

lignes
diagramme de bloc de fonctions (FBD), insertion 33
étiquettes, insertion 33
insertion 33

Logique à relais (LD)
instructions 29
langue 29

M

manipulation de chaînes
fonction ASCII 615
fonction CHAR 617
fonction DELETE 618
fonction FIND 620
fonction INSERT 622
fonction LEFT 624
fonction MID 626
fonction MLEN 628
fonction REPLACE 631
fonction RIGHT 629

manipulation des données
bloc de fonctions AVERAGE 275
bloc de fonctions COP 276
fonction MAX 283
fonction MIN 282

MC_Read 442
MC_ReadActualPosition 438
MODULE_INFO 362

mouvement
bloc de fonctions MC_AbortTrigger 413
bloc de fonctions MC_Halt 416
bloc de fonctions MC_Home 419
bloc de fonctions MC_MoveAbsolute 422
bloc de fonctions MC_MoveRelative 426
bloc de fonctions MC_MoveVelocity 430
bloc de fonctions MC_Power 435
bloc de fonctions MC_ReadAxisError 445
bloc de fonctions MC_ReadBoolParameter 449
bloc de fonctions MC_ReadParameter 451
bloc de fonctions MC_ReadStatus 454
bloc de fonctions MC_Reset 458
bloc de fonctions MC_SetPosition 461
bloc de fonctions MC_Stop 463
bloc de fonctions MC_TouchProbe 466
bloc de fonctions MC_WriteBoolParameter 470
bloc de fonctions MC_WriteParameter 473

O

opérateur - 97
opérateur * 87
opérateur / 80
opérateur + 70
opérateur < 244
opérateur <= 246
opérateur <> 247
opérateur = 239
opérateur > 242
opérateur >= 243
opérateur 1Gain 86
opérateur AND 148
opérateur ANY_TO_BOOL 255
opérateur ANY_TO_BYTE 256
opérateur ANY_TO_DATE 257
opérateur ANY_TO_DINT 259
opérateur ANY_TO_DWORD 260
opérateur ANY_TO_INT 261
opérateur ANY_TO_LINT 262
opérateur ANY_TO_LREAL 263
opérateur ANY_TO_LWORD 264
opérateur ANY_TO_REAL 265
opérateur ANY_TO_SINT 266
opérateur ANY_TO_STRING 267
opérateur ANY_TO_TIME 268
opérateur ANY_TO_UDINT 269
opérateur ANY_TO_UINT 270
opérateur ANY_TO_ULINT 271
opérateur ANY_TO_USINT 272
opérateur ANY_TO_WORD 273
opérateur d'addition 70
opérateur de division 80
opérateur de multiplication 87
opérateur de soustraction 97
opérateur d'inv. 89
opérateur égal 239

opérateur inférieur à 244
opérateur inférieur ou égal à 246
opérateur MOV 86
opérateur non égal 247
opérateur NOT 150
opérateur OR 148
opérateur supérieur à 242
opérateur supérieur ou égal à 243
opérateur XOR 149
opérateurs
1Gain 86
addition 70
AND 148
ANY_TO_BOOL 255
ANY_TO_BYTE 256
ANY_TO_DATE 257
ANY_TO_DINT 259
ANY_TO_DWORD 260
ANY_TO_INT 261
ANY_TO_LINT 262
ANY_TO_LREAL 263
ANY_TO_LWORD 264
ANY_TO_REAL 265
ANY_TO_SINT 266
ANY_TO_STRING 267
ANY_TO_TIME 268
ANY_TO_UDINT 269
ANY_TO_UINT 270
ANY_TO_ULINT 271
ANY_TO_USINT 272
ANY_TO_WORD 273
division 80
égal 239
inférieur à 244
inférieur ou égal à 246
Inv. 89
MOV 86
multiplication 87
non égal 247
NOT 150
OR 148
soustraction 97
supérieur à 242
supérieur ou égal à 243
XOR 149
opérations arithmétiques
opérateur 1Gain 86
opérations binaires
fonction AND_MASK 123
fonction NOT_MASK 131
fonction OR_MASK 132
fonction ROL 134
fonction ROR 135
fonction SHL 137
fonction SHR 139
fonction XOR_MASK 140
opérations booléennes
bloc de fonctions F_TRIG 143
bloc de fonctions R_TRIG 145
bloc de fonctions RS 146
bloc de fonctions SR 150
opérateur AND 148
opérateur NOT 150
opérateur OR 148
opérateur XOR 149
opérations de comparaison
opérateur égal 239
opérateur inférieur à 244
opérateur inférieur ou égal à 246
opérateur non égal 247
opérateur supérieur à 242
opérateur supérieur ou égal à 243

P

paramètres
pour des blocs fonctionnels 21
PID 552
pochoir
éléments de contact disponibles pour les diagrammes de LD 47
lignes, insertion dans des conteneurs de LD 33
types de bobines disponibles pour FBD 40
types de bobines disponibles pour les diagrammes de LD 40

R

raccourcis clavier
langage de LD 57

S

sauts
diagramme de bloc de fonctions (FBD), insertion 53
échelon, insertion 33
logiques à relais, insertion 53
pour diagrammes de LD 53
SIN_LREAL 94
SOCKET_ACCEPT 568
SOCKET_CREATE 571
SOCKET_DELETE 576
SOCKET_DELETEALL 578
SOCKET_INFO 580
SOCKET_OPEN 586
SOCKET_READ 590
SOCKET_WRITE 594
symboles de retour
insertion, éléments du FBD 52

T

temps
bloc de fonctions TOF 635
bloc de fonctions TON 637
bloc de fonctions TONOFF 640
bloc de fonctions TP 642
Bloc fonctionnel RTO 644
fonction DOY 646
fonction TDF 648
fonction TOW 650

Assistance de Rockwell Automation

Utilisez ces ressources pour accéder à l'information de soutien.

Centre de soutien technique	Trouvez de l'aide pour les vidéos, faq, chat, forums d'utilisateurs et mises à jour de notification de produits.	rok.auto/support
Base de connaissances	Accédez aux articles knowledgebase.	rok.auto/base de connaissances
Numéros de téléphone de soutien technique local	Localisez le numéro de téléphone de votre pays.	rok.auto/phonesupport Rok.auto/phonesupport rok.auto/phonesupport rok
Bibliothèque de documents	Trouvez des instructions d'installation, des manuels, des brochures et des publications de données techniques.	rok.auto/littérature
Centre de compatibilité et de téléchargement de produits (PCDC)	Obtenez de l'aide pour déterminer comment les produits interagissent, vérifiez les fonctionnalités et les capacités et trouvez le firmware associé.	rok.auto/pcdc

Commentaires concernant la documentation

Vos commentaires nous permettent d'améliorer votre documentation. Si vous avez des suggestions concernant l'amélioration de notre contenu, complétez le formulaire à l'adresse rok.auto/docfeedback.

Déchets d'équipements électriques et électroniques (DEEE)

En fin de vie, cet équipement doit être collecté séparément de tout déchet municipal non trié.

Rockwell Automation maintient les informations environnementales actuelles sur son site Web à rok.auto/pec.

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

rockwellautomation.com — expanding **human possibility**®

AMÉRIQUES : Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 États-Unis, Tél. : +(1) 414.382.2000, Fax :+(1) 414.382.4444
EUROPE / MOYEN-ORIENT / AFRIQUE : Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgique, Tél. : +(32) 2 663 0600, Fax : +(32) 2 663 0640
ASIE PACIFIQUE : Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tél. : +(852) 2887 4788, Fax : +(852) 2508 1846
CANADA : Rockwell Automation, 3043 rue Joseph A. Bombardier, Laval, Québec, H7P 6C5, Tél: +1 (450) 781-5100, Fax: +1 (450) 781-5101, www.rockwellautomation.ca
FRANCE : Rockwell Automation SAS – 2, rue René Caudron, Bât. A, F-78960 Voisins-le-Bretonneux, Tél: +33 1 61 08 77 00, Fax : +33 1 30 44 03 09
SUISSE : Rockwell Automation AG, Av. des Baumettes 3, 1020 Renens, Tél: 021 631 32 32, Fax: 021 631 32 31, Customer Service Tél: 0848 000 278

Publication Rockwell Automation 2080-RM001L-FR-E - Novembre 2023
Annule et remplace la publication 2080-RM001K-FR-E - Mars 2022
Copyright © 2023 Rockwell Automation Technologies, Inc. Tous droits réservés. Imprimé aux États-Unis