

Unity Pro

Communication

Bibliothèque de blocs

05/2010

Le présent document comprend des descriptions générales et/ou des caractéristiques techniques générales sur la performance des produits auxquels il se réfère. Le présent document ne peut être utilisé pour déterminer l'aptitude ou la fiabilité de ces produits pour des applications utilisateur spécifiques et n'est pas destiné à se substituer à cette détermination. Il appartient à chaque utilisateur ou intégrateur de réaliser, sous sa propre responsabilité, l'analyse de risques complète et appropriée, et d'évaluer et de tester les produits dans le contexte de leur application ou utilisation spécifique. Ni la société Schneider Electric, ni aucune de ses filiales ou sociétés dans lesquelles elle détient une participation, ne peut être tenue pour responsable de la mauvaise utilisation des informations contenues dans le présent document. Si vous avez des suggestions, des améliorations ou des corrections à apporter à cette publication, veuillez nous en informer.

Aucune partie de ce document ne peut être reproduite sous quelque forme ou par quelque moyen que ce soit, électronique, mécanique ou photocopie, sans l'autorisation écrite expresse de Schneider Electric.

Toutes les réglementations locales, régionales et nationales en matière de sécurité doivent être respectées lors de l'installation et de l'utilisation de ce produit. Pour des raisons de sécurité et afin de garantir la conformité aux données système documentées, seul le fabricant est habilité à effectuer des réparations sur les composants.

Lorsque des équipements sont utilisés pour des applications présentant des exigences de sécurité techniques, suivez les instructions appropriées.

La non-utilisation du logiciel Schneider Electric ou d'un logiciel approuvé avec nos produits peut entraîner des blessures, des dommages ou un fonctionnement incorrect.

Le non-respect de cette consigne peut entraîner des lésions corporelles ou des dommages matériels.

© 2010 Schneider Electric. Tous droits réservés.

Table des matières



	Consignes de sécurité	9
	A propos de ce manuel	11
Partie I	Généralités	13
Chapitre 1	Types de module et leur utilisation.	15
	Types de bloc	16
	Structure d'un FFB	18
	EN et ENO.	21
Chapitre 2	Disponibilité des modules sur les différentes plates- formes	25
	Disponibilité des blocs sur les différentes plates-formes.	25
Chapitre 3	Fonctionnement des EF de communication	27
3.1	Informations générales sur les fonctions de communication Premium et Atrium	27
	Règles d'utilisation des fonctions de communication des automates Premium et Atrium.	28
	Les fonctions de communication sur automates Premium et Atrium	29
	Structure des fonctions de communication Premium et Atrium.	31
	Adresse destinataire	32
	Structure des paramètres de gestion	33
	Paramètres de gestion : comptes rendus de communication et d'opération.	34
	Paramètres de gestion : longueur et timeout	37
	Fonction de serveur.	39
Partie II	Etendu	41
Chapitre 4	ADDM: conversion d'adresse	43
	Description	43
Chapitre 5	ADDR : conversion d'adresse	47
	Description	47
Chapitre 6	CANCEL : arrêt d'un échange en cours	49
	Description	50
	Exemple d'annulation d'un échange	52

Chapitre 7	CREAD_REG : lecture de registre continu	53
	Description	54
	Types de données dérivés	56
	Mode de fonctionnement	58
	Description des paramètres	59
Chapitre 8	CWRITE_REG : écriture de registre continu	61
	Description	62
	Types de données dérivés	65
	Mode de fonctionnement	67
	Description des paramètres	68
Chapitre 9	DATA_EXCH : échange de données entre applications	69
	Description	70
	Ecran de saisie assistée	75
	Exemple d'utilisation d'un réseau Fipway	77
Chapitre 10	INPUT_BYTE : réception de chaînes de caractères	79
	Description	79
Chapitre 11	INPUT_CHAR : réception de chaînes de caractères	83
	Description	84
	Ecran de saisie assistée	89
	Exemple de lecture de chaînes de caractères via un réseau Fipway	91
	Exemple de lecture de chaînes de caractères via une liaison série de processeurs Modicon M340	93
Chapitre 12	MBP_MSTR : maître Modbus Plus	95
	Description du bloc	97
	Codes fonction des opérations	100
	Structures du bloc de commande de réseau	101
	Lecture de données	104
	Ecriture de données	106
	Extraction de statistiques locales	108
	Suppression de statistiques locales	110
	Ecriture de données globales	111
	Lecture de données globales	112
	Lire statistiques distantes	113
	Effacer statistiques distantes	115
	Validité de Peer Cop	116
	Réinitialisation du module optionnel	117
	Lecture de la CTE	118
	Ecriture de la CTE	120
	Envoi de message électronique	122
	Lire/écrire données	124
	Etat de validité des communications Peer Cop	126
	Statistiques du réseau Modbus Plus	128
	Statistiques de réseau Ethernet TCP/IP	134
	Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP	137

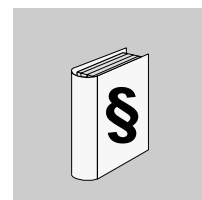
	Codes d'erreur spécifiques SY/MAX	142
	Codes d'erreur Ethernet TCP/IP	144
	Codes d'erreur CTE pour Ethernet SY/MAX et TCP/IP	148
	Codes d'erreur du service de messagerie.	149
Chapitre 13	ModbusP_ADDR : adresse Modbus Plus	151
	Description	152
	Description détaillée	155
Chapitre 14	OUT_IN_CHAR : envoi/réception de chaînes de caractères	157
	Description	158
	Ecran de saisie assistée	162
	Exemple d'envoi/réception d'une chaîne de caractères	164
Chapitre 15	OUT_IN_MBUS : fonction de communication Modbus	167
15.1	Présentation générale du bloc de communication OUT_IN_MBUS	168
	Description de la fonction	169
	Cas d'utilisation	170
	Fonctionnalités	172
15.2	Description du bloc de communication OUT_IN_MBUS	173
	Représentations et paramètres	174
	Le paramètre MbusCmd	177
	Le paramètre RetryLmt	180
	Le paramètre DataBits	181
	Le paramètre RespTout	182
	Le paramètre MasterDataArea	183
	Le paramètre Status	184
15.3	Mise en oeuvre du bloc de communication OUT_IN_MBUS	185
	Configuration de la liaison série	186
	Marche à suivre pour la programmation	189
	Utilisation d'un modem	191
15.4	Exemple d'utilisation du bloc de communication OUT_IN_MBUS	193
	Description de l'exemple	194
	Structure de la programmation	195
	Déclaration des variables	197
	Programmation	198
Chapitre 16	PRINT_CHAR : envoi de chaînes de caractères	205
	Description	206
	Ecran de saisie assistée	210
	Exemple d'envoi de chaînes de caractères via un réseau Fipway	212
	Exemple d'envoi de chaînes de caractères via une liaison série de processeurs Modicon M340	214
Chapitre 17	RCV_TLG : réception de télégrammes	217
	Description	218
	Exemple de réception d'un télégramme	221

Chapitre 18	READ_ASYN : lecture de données de façon asynchrone	223
	Description	223
Chapitre 19	READ_GDATA : lecture des données globales Modbus Plus	227
	Description	227
Chapitre 20	READ_REG : lecture de registres.	229
	Description	230
	Types de données dérivés	233
	Mode de fonctionnement	235
	Description des paramètres	236
Chapitre 21	READ_VAR : lecture de variables	239
	Description	240
	Ecran de saisie assistée	245
	Exemple d'utilisation sur un bus Uni-Telway	247
	Exemple de lecture de bits	249
	Exemple d'utilisation au sein d'un réseau	251
	Exemple de lecture de mots via la liaison série des processeurs Modicon M340	253
	Exemple de vérification d'exécution	255
Chapitre 22	SEND_EMAIL : messagerie électronique	257
	Send Email	257
Chapitre 23	SEND_REQ : envoi de requêtes	261
	Description	262
	Liste de requêtes UNI-TE	266
	Ecran de saisie assistée	273
	Exemple d'envoi d'une requête UNI-TE	275
	Modification des paramètres IP avec SEND_REQ (exemple)	277
	Utilisation de la fonction SEND_REQ	278
Chapitre 24	SEND_TLG : envoi de télégrammes.	279
	Description	280
	Exemple d'envoi d'un télégramme	283
Chapitre 25	SYMAX_IP_ADDR : adresse IP SY/MAX	285
	Description	286
	Description détaillée	288
Chapitre 26	TCP_IP_ADDR : adresse TCP/IP.	291
	Description	292
	Description détaillée	295
Chapitre 27	UNITE_SERVER : serveur immédiat	297
	Description	298
	Exemple de serveur immédiat	301
Chapitre 28	WRITE_ASYN : écriture de données de façon asynchrone.	303
	Description	303

Chapitre 29	WRITE_GDATA : écriture des données globales	
	Modbus Plus	307
	Description	307
Chapitre 30	WRITE_REG : écriture de registre.	309
	Description	310
	Types de données dérivés	313
	Mode de fonctionnement.	315
	Description des paramètres	316
Chapitre 31	WRITE_VAR : écriture de variables	319
	Description	320
	Ecran de saisie assistée	325
	Exemple d'écriture de mots sur un réseau	327
	Exemple d'écriture de mots via la liaison série des processeurs Modicon M340.	329
	Exemple de vérification d'exécution	331
Chapitre 32	XXMIT : transmettre	333
32.1	Introduction au bloc XXMIT.	334
	Fonctionnalités du bloc XXMIT	334
32.2	Fonctions XXMIT.	335
	Description sommaire	336
	Représentation	337
	Description détaillée des paramètres	341
	Fonctions de communication du bloc XXMIT	352
	Fonctions ASCII du bloc XXMIT	353
	Fonctions de modem du bloc XXMIT	359
	Fonctions Modbus du bloc XXMIT	361
	Tampon FIFO et contrôle de flux.	368
	Exemples d'application	372
32.3	XXMIT : Règles de programmation.	383
	Règles de programmation du bloc XXMIT	383
32.4	Références techniques XXMIT	385
	Limites des paramètres de requête/réponse Modbus.	386
	Configuration de XXMIT à l'aide de modems à numérotation automatique compatibles Hayes (uniquement)	387
	Exemple d'application Hayes	392
32.5	Informations sur le câblage	396
	Brochage des câbles.	397
	Kits d'adaptateurs de câble.	409
Annexes		413
Annexe A	codes et valeurs d'erreur des EFB	415
	Tableau des codes d'erreur pour la bibliothèque de communications	416
	Erreurs courantes relatives aux valeurs en virgule flottante	419

Annexe B	Objets système	421
	Présentation des bits système	422
	Description des bits système %S15 à %S21	423
	Description des mots système %SW12 à %SW29	426
Glossaire	433
Index	455

Consignes de sécurité



Informations importantes

AVIS

Lisez attentivement ces instructions et examinez le matériel pour vous familiariser avec l'appareil avant de tenter de l'installer, de le faire fonctionner ou d'assurer sa maintenance. Les messages spéciaux suivants que vous trouverez dans cette documentation ou sur l'appareil ont pour but de vous mettre en garde contre des risques potentiels ou d'attirer votre attention sur des informations qui clarifient ou simplifient une procédure.



L'apposition de ce symbole à un panneau de sécurité Danger ou Avertissement signale un risque électrique pouvant entraîner des lésions corporelles en cas de non-respect des consignes.



Ceci est le symbole d'une alerte de sécurité. Il vous avertit d'un risque de blessures corporelles. Respectez scrupuleusement les consignes de sécurité associées à ce symbole pour éviter de vous blesser ou de mettre votre vie en danger.

DANGER

DANGER indique une situation immédiatement dangereuse qui, si elle n'est pas évitée, **entraînera** la mort ou des blessures graves.

AVERTISSEMENT

L'indication **AVERTISSEMENT** signale une situation potentiellement dangereuse et susceptible **d'entraîner la** mort ou des blessures graves.

ATTENTION

L'indication **ATTENTION** signale une situation potentiellement dangereuse et susceptible **d'entraîner des** blessures d'ampleur mineure à modérée.

ATTENTION

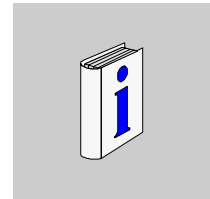
L'indication **ATTENTION**, utilisée sans le symbole d'alerte de sécurité, signale une situation potentiellement dangereuse et susceptible **d'entraîner des** dommages aux équipements.

REMARQUE IMPORTANTE

L'installation, l'utilisation, la réparation et la maintenance des équipements électriques doivent être assurées par du personnel qualifié uniquement. Schneider Electric décline toute responsabilité quant aux conséquences de l'utilisation de cet appareil.

Une personne qualifiée est une personne disposant de compétences et de connaissances dans le domaine de la construction et du fonctionnement des équipements électriques et installations et ayant bénéficié d'une formation de sécurité afin de reconnaître et d'éviter les risques encourus.

A propos de ce manuel



Présentation

Objectif du document

Ce document décrit les fonctions et blocs fonction de la bibliothèque de communication.

Champ d'application

Ce document est applicable à partir de Unity Pro version 5.0.

Document à consulter

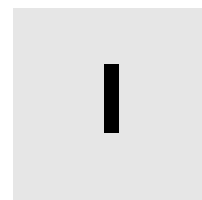
Titre de documentation	Référence
Bloc fonction XXMIT pour Quantum avec Unity Pro, Manuel utilisateur	-

Vous pouvez télécharger ces publications et autres informations techniques depuis notre site web à l'adresse : www.schneider-electric.com.

Commentaires utilisateur

Envoyez vos commentaires à l'adresse e-mail techpub@schneider-electric.com

Généralités



Objet de cette partie

Ce chapitre contient des informations d'ordre général concernant la bibliothèque Communication.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre	Titre du chapitre	Page
1	Types de module et leur utilisation	15
2	Disponibilité des modules sur les différentes plates-formes	25
3	Fonctionnement des EF de communication	27

Types de module et leur utilisation



Vue d'ensemble

Ce chapitre décrit les différents types de module et leur utilisation.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Types de bloc	16
Structure d'un FFB	18
EN et ENO	21

Types de bloc

Types de bloc

Différents types de bloc sont utilisés dans Unity Pro. FFB est le terme générique pour tous les types de bloc.

Une différence est faite entre les types de bloc suivants :

- Fonction élémentaire (EF)
- les blocs fonction élémentaires (EFB)
- Blocs fonction dérivés (DFB)
- Procédure

NOTE : les blocs fonction de mouvement ne sont pas disponibles sur la plate-forme Quantum.

Fonction élémentaire

ATTENTION

COMPORTEMENT INATTENDU DE L'EQUIPEMENT

N'utilisez pas de liens pour connecter les sorties des blocs fonction lorsque votre application repose sur des données de sortie persistantes d'un bloc EF.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

Les fonctions élémentaires (EF) ne disposent pas d'état interne et possèdent une seule sortie. Si les valeurs des entrées sont similaires, la valeur de la sortie est identique pour les exécutions de la fonction. Par exemple, l'addition de deux valeurs donne le même résultat à chaque exécution de la fonction.

Une fonction élémentaire est représentée dans les langages graphiques (FBD et LD) sous la forme d'un rectangle avec des entrées et une sortie. Les entrées sont toujours représentées à gauche du rectangle et les sorties à droite. Le nom de la fonction, c'est-à-dire le type de fonction, est affiché au centre du rectangle.

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

NOTE : la désactivation d'une fonction élémentaire (EN=0) entraîne la réinitialisation des liens associés à ses entrées/sorties. Pour transférer l'état du signal, n'utilisez pas de lien. Une variable doit être connectée à la sortie de la fonction élémentaire et être utilisée pour connecter l'entrée de l'élément.

Bloc fonction élémentaire

Les blocs fonction élémentaires (EFB) possèdent un état interne. Si les valeurs des entrées sont identiques, les valeurs des sorties peuvent différer à chaque exécution du bloc fonction. Pour un compteur, par exemple, la valeur de la sortie est incrémentée.

Un bloc fonction élémentaire est représenté dans les langages graphiques (FBD et LD) sous la forme d'un rectangle avec des entrées et des sorties. Les entrées sont toujours représentées à gauche du rectangle et les sorties à droite. Le nom du bloc fonction, c'est-à-dire le type de bloc fonction, est affiché au centre du rectangle. Le nom d'instance est affiché au-dessus du cadre.

Bloc fonction dérivé

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

Procédure

Les procédures correspondent à des fonctions proposant plusieurs sorties. Elles ne disposent pas d'état interne.

L'unique différence par rapport aux fonctions élémentaires est que les procédures peuvent avoir plus d'une sortie et qu'elles supportent des variables du type de donnée `VAR_IN_OUT`.

Les procédures ne renvoient aucune valeur.

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

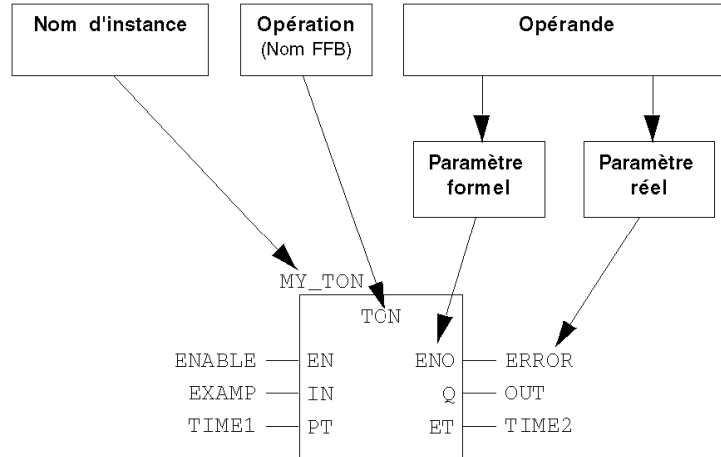
Visuellement, il n'existe aucune différence entre les procédures et les fonctions élémentaires.

Structure d'un FFB

Structure

Un FFB se compose d'une opération (nom du FFB), des opérands nécessaires à l'opération (paramètres réels et formels) et d'un nom d'instance pour les blocs fonction élémentaires ou dérivés.

Appel d'un bloc fonction dans le langage de programmation FBD :



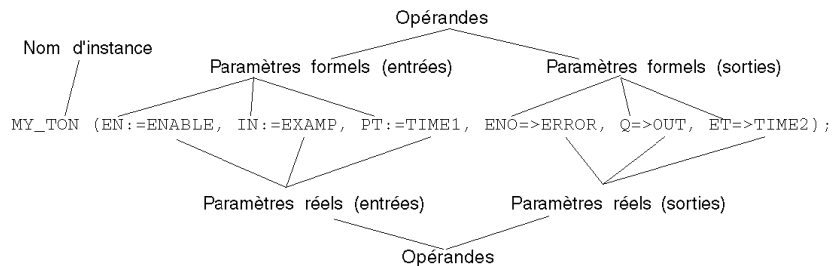
⚠ ATTENTION

COMPOTEMENT INATTENDU DE L'APPLICATION

N'appellez pas plusieurs fois la même instance de bloc pendant un cycle d'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

Appel formel d'un bloc fonction dans le langage de programmation ST :



Opération

L'opération détermine la fonction qui doit être exécutée par le FFB, par exemple registre à décalage ou opérations de conversion.

Opérande

L'opérande détermine les éléments sur lesquels porte l'opération qui est exécutée. Dans les FFB, il est constitué de paramètres formels et de paramètres réels.

Paramètres formels et réels

Des entrées et des sorties permettent de transférer les valeurs vers ou depuis un FFB. Ces entrées et ces sorties sont appelées « paramètres formels ».

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

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

Le type de données des paramètres réels doit correspondre au type de données des entrées/sorties (paramètres formels). La seule exception concerne les entrées/sorties génériques dont le type de données est déterminé par le paramètre réel. On choisira un type de données adapté pour le bloc fonction, si les paramètres réels sont constitués de valeurs littérales.

Appel de FFB dans le langage IL/ST

Les FFB peuvent être appelés de deux manières dans les langages textuels IL et ST : formelle ou informelle. Pour plus d'informations, consultez le *Manuel de référence*.

Exemple d'un appel de fonction formel :

```
out:=LIMIT (MN:=0, IN:=var1, MX:=5);
```

Exemple d'un appel de fonction informel :

```
out:=LIMIT (0, var1, 5);
```

NOTE : Les paramètres EN et la sortie ENO peuvent uniquement être utilisés pour des appels formels.

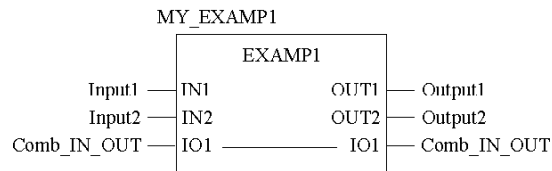
Variable VAR_IN_OUT

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

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

La relation entre la variable d'entrée et la variable de sortie est représentée dans les langages graphiques (FBD et LD) par une ligne.

Bloc fonction avec la variable VAR_IN_OUT dans le langage FBD :



Bloc fonction avec la variable VAR_IN_OUT dans le langage ST :

```
MY_EXAMP1 (IN1:=Input1, IN2:=Input2, IO1:=Comb_IN_OUT,
           OUT1=>Output1, OUT2=>Output2);
```

Tenez compte des points suivants lorsque vous utilisez des FFB avec les variables VAR_IN_OUT :

- Une variable doit être affectée à toutes les entrées VAR_IN_OUT.
- Aucune valeur littérale ou constante ne doit être affectée aux entrées/sorties VAR_IN_OUT.

Les limitations supplémentaires de ces langages graphiques (FBD et LD) sont les suivantes :

- Les liaisons graphiques permettent uniquement de relier des sorties VAR_IN_OUT à des entrées VAR_IN_OUT.
- Seule une liaison graphique peut être associée à une entrée/sortie VAR_IN_OUT.
- Des variables/composantes de variables différentes peuvent être reliées à l'entrée VAR_IN_OUT et à la sortie VAR_IN_OUT. Dans un tel cas, la valeur de la variable/composante de variable en entrée est copiée dans la variable/composante de variable en sortie.
- Il est interdit d'utiliser des négations au niveau des entrées/sorties VAR_IN_OUT.
- Une combinaison de variable/adresse et de liaisons graphiques n'est pas possible pour les sorties VAR_IN_OUT.

EN et ENO

Description

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

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

Si la valeur de **EN** est déjà réglée sur 1, lors de l'appel de FFB, les algorithmes définis par FFB sont exécutés. Une fois les algorithmes exécutés, la valeur de la sortie **ENO** est réglée sur « 1 ». Si certaines conditions d'erreur sont détectées durant l'exécution de ces algorithmes, **ENO** est réglé sur 0.

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

Une fois les algorithmes exécutés, la valeur de **ENO** est réglée sur « 1 », sinon la valeur de **ENO** est réglée sur « 0 ».

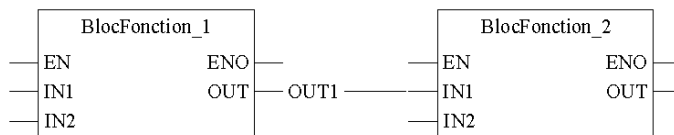
Si la valeur de **ENO** est réglée sur 0 (car **EN** = 0 ou en raison d'une condition d'erreur détectée lors de l'exécution ou de l'échec de l'exécution des algorithmes) :

- Blocs fonction
 - Traitement des paramètres EN/ENO avec des blocs fonction qui possèdent (uniquement) une liaison en tant que paramètre de sortie :



Si l'entrée **EN** de **BlocFonction_1** est réglée sur « 0 », la connexion de sortie **OUT** de **BlocFonction_1** conserve l'état qu'elle avait lors du dernier cycle correctement exécuté.

- Traitement des paramètres EN/ENO avec des blocs fonction qui possèdent une variable et une liaison en tant que paramètres de sortie :



Si l'entrée **EN** de **BlocFonction_1** est réglée sur « 0 », la connexion de sortie **OUT** de **BlocFonction_1** conserve l'état qu'elle avait lors du dernier cycle correctement exécuté. La variable **OUT1** présente sur la même broche conserve son état précédent ou peut être modifiée de manière externe sans incidence sur la connexion. La variable et la liaison sont enregistrées indépendamment l'une de l'autre.

- Fonctions/procédures

Comme spécifié dans la norme CEI 61131-3, les sorties de fonctions désactivées (entrée EN réglée sur « 0 ») ne sont pas définies. (Cette caractéristique s'applique également aux procédures.)

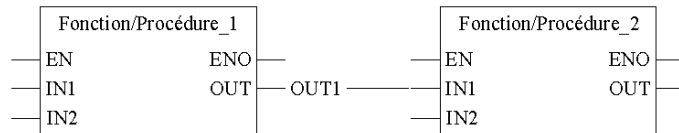
Voici une explication des états des sorties dans un tel cas :

- Traitement des paramètres EN/ENO avec des fonctions/procédures qui possèdent (uniquement) une liaison en tant que paramètre de sortie :



Si l'entrée EN de Fonction/Procédure_1 est réglée sur 0, la connexion de sortie OUT de Fonction/Procédure_1 est également réglée sur 0.

- Traitement des paramètres EN/ENO avec des blocs fonction qui possèdent une variable et une liaison en tant que paramètres de sortie :



Si l'entrée EN de Fonction/Procédure_1 est réglée sur 0, la connexion de sortie OUT de Fonction/Procédure_1 est également réglée sur 0. La variable OUT1 présente sur la même broche conserve son état précédent ou peut être modifiée de manière externe sans incidence sur la connexion. La variable et la liaison sont enregistrées indépendamment l'une de l'autre.

Le comportement de la sortie des FFB ne dépend pas de la façon dont les FFB sont appelés (sans EN/ENO ou avec EN=1).

Appel de FFB conditionnel/inconditionnel

Un FFB peut être appelé de manière "conditionnelle" ou "inconditionnelle". La condition est établie en pré-connectant l'entrée EN.

- Entrée EN connectée
appels conditionnels (le FFB est exécuté uniquement si EN = 1)
- Entrée EN affichée, masquée et marquée comme TRUE, ou affichée et non occupée
appels inconditionnels (le FFB est traité indépendamment de l'entrée EN)

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

 **ATTENTION****EQUIPEMENT D'APPLICATION IMPREVU**

Ne désactivez pas les blocs fonction équipés d'une fonction d'horloge interne en cours de fonctionnement.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

Remarque concernant les langages IL et ST

Les paramètres `EN` et la sortie `ENO` peuvent uniquement être utilisés dans les langages textuels et dans le cadre d'un appel de FFB formel, par exemple :

```
MY_BLOCK (EN:=enable, IN1:=var1, IN2:=var2,  
ENO=>error, OUT1=>result1, OUT2=>result2);
```

L'affectation de variables à `ENO` doit être effectuée à l'aide de l'opérateur `=>`.

L'entrée `EN` et la sortie `ENO` ne peuvent pas être utilisées pour un appel informel.

Disponibilité des modules sur les différentes plates-formes

2

Disponibilité des blocs sur les différentes plates-formes

Introduction

Les blocs ne sont pas disponibles sur toutes les plates-formes matérielles. Le tableau ci-après indique les modules disponibles sur les différentes plates-formes matérielles.

NOTE : les fonctions, procédures et blocs fonction de cette bibliothèque ne sont pas définis par la norme CEI 61131-3.

Description

Disponibilité des modules :

Nom du bloc	Type de bloc	M340	Premium	Quantum
ADDR	EF	-	+	-
ADDM	EF	+	-	-
CANCEL	Procédure	+	+	-
CREAD_REG	EFB	-	-	+
CWRITE_REG	EFB	-	-	+
DATA_EXCH	Procédure	+	+	-
INPUT_BYTE	Procédure	+	+	-
INPUT_CHAR	Procédure	+	+	-
MBP_MSTR	EFB	-	-	+
MODBUSP_ADDR	EFB	-	-	+
OUT_IN_CHAR	Procédure	-	+	-
OUT_IN_MBUS	DFB	-	+	-
PRINT_CHAR	Procédure	+	+	-
RCV_TLG	Procédure	-	+	-

Nom du bloc	Type de bloc	M340	Premium	Quantum
READ_ASYN	Procédure	-	+	-
READ_GDATA	Procédure	-	+	-
READ_REG	EFB	-	-	+
READ_VAR	Procédure	+	+	-
SEND_REQ	Procédure	-	+	-
SEND_EMAIL	EF	+	-	-
SEND_TLG	Procédure	-	+	-
SYMAX_IP_ADDR	EFB	-	-	+
TCP_IP_ADDR	EFB	-	-	+
UNITE_SERVER	Procédure	-	+	-
WRITE_ASYN	Procédure	-	+	-
WRITE_GDATA	Procédure	-	+	-
WRITE_REG	EFB	-	-	+
WRITE_VAR	Procédure	+	+	-
XXMIT	EFB	-	-	+
Légende :				
+	Oui			
-	Non			

Fonctionnement des EF de communication

3

3.1 Informations générales sur les fonctions de communication Premium et Atrium

Objet de cette partie

Cette section décrit le fonctionnement et la gestion des fonctions de communication pour les automates Premium et Atrium.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Règles d'utilisation des fonctions de communication des automates Premium et Atrium	28
Les fonctions de communication sur automates Premium et Atrium	29
Structure des fonctions de communication Premium et Atrium	31
Adresse destinataire	32
Structure des paramètres de gestion	33
Paramètres de gestion : comptes rendus de communication et d'opération	34
Paramètres de gestion : longueur et timeout	37
Fonction de serveur	39

Règles d'utilisation des fonctions de communication des automates Premium et Atrium

Vue d'ensemble

Les fonctions de communication utilisées avec les automates Premium et Atrium présentent certaines caractéristiques uniques qui les différencient des autres fonctions de la bibliothèque. Le présent document respecte la charte de documentation relative à la bibliothèque de fonctions. Il contient également des informations supplémentaires concernant les particularités des modules métiers de communication.

Variables affectées

Toutes les fonctions de communication ne nécessitant pas de programmation au niveau de l'automate du serveur (`READ_VAR`, `WRITE_VAR`, etc.) donnent accès aux variables affectées des automates distants. Les variables non affectées ne sont pas accessibles.

NOTE : pour transférer des variables non affectées entre automates, il est nécessaire d'utiliser la fonction `DATA_EXCH`. Une autre solution consiste à réaliser des copies locales dans les zones de variables affectées.

Sauf pour le premier exemple concernant la fonction `WRITE_VAR` (Unity Pro et versions ultérieures uniquement) tous les autres exemples utilisent des variables d'adressage direct (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) (utilisation d'adresses, de variables affectées).

Langage de programmation

Le langage de programmation le plus concis pour la création d'applications de communication est le langage littéral structuré (ST - Structured Text). Tous les exemples, à l'exception de la fonction `READ_VAR` (*voir page 239*), sont écrits en ST.

Les fonctions de communication sur automates Premium et Atrium

Présentation

Ces fonctions permettent la communication d'un équipement vers un autre. Certaines sont communes à plusieurs types de voie de communication, d'autres peuvent être spécifiques à une seule voie de communication.

NOTE : Le traitement des fonctions de communication est asynchrone par rapport au traitement de la tâche applicative qui a permis de les activer. Seules les fonctions d'émission/réception de télégramme et d'arrêt opération sont des exceptions puisque leur exécution est totalement synchrone avec l'exécution de la tâche d'activation.

Fonctions de communication asynchrone

Une fonction de communication est dite asynchrone quand elle est exécutée pendant un ou plusieurs cycles après celui qui l'a activée.

Le tableau suivant présente les fonctions de communication dont l'exécution est asynchrone :

Fonction	Son rôle est...
READ_VAR	la lecture d'objets langage de base : mots et bits internes, mots et bits système, temporisateurs, monostables, programmeurs cycliques, registres, compteurs.
WRITE_VAR	l'écriture d'objets langage de base : mots et bits internes, mots et bits système.
SEND_REQ	l'émission des requêtes UNI-TE.
DATA_EXCH	l'émission et/ou demande de réception de données.
PRINT_CHAR	l'écriture d'une chaîne de caractères.
INPUT_CHAR	la lecture d'une chaîne de caractères.
OUT_IN_CHAR	l'émission d'une chaîne de caractères et attente d'une réponse.
READ_GDATA	la lecture des données communes Modbus Plus.
WRITE_GDATA	l'écriture des données communes Modbus Plus.

Fonction	Son rôle est...
SERVER	de traiter des requêtes READ_VAR et WRITE_VAR sur Modbus de manière immédiate (Serveur immédiat).
READ_Asyn	la lecture de 1 Koctets en messagerie.
WRITE_Asyn	l'écriture de 1 Koctets en messagerie.

NOTE : Il est recommandé de déclencher les fonctions asynchrones sur front et non sur état afin de ne pas saturer les buffers des équipements en envoyant de multiples requêtes. Il est également conseillé, pour cette même raison, de gérer le bit d'activité (voir page 33) et les mots de compte rendu (voir page 34) lors de l'exécution de chaque fonction de communication.

Fonctions de communication synchrone

Une fonction de communication est dite synchrone quand elle est entièrement exécutée pendant la tâche automate qui l'a activée.

Le tableau suivant présente les fonctions de communication dont l'exécution est synchrone :

Fonction	Son rôle est ...
SEND_TLG	l'émission d'un télégramme.
RCV_TLG	la réception d'un télégramme.
CANCEL	l'arrêt d'un échange en cours.
ADR	la conversion d'une chaîne de caractères en une adresse (tableau de 6 entiers) directement exploitable par la fonction de communication.

Structure des fonctions de communication Premium et Atrium

Présentation

Une fonction de communication sur les automates Premium et Atrium utilise :

- un paramètre d'adresse,
- des paramètres spécifiques à une opération de communication,
- des paramètres de gestion.

Syntaxe

La syntaxe d'une fonction de communication se présente sous la forme suivante :

Fonction (adresse cible, paramètres spécifiques, paramètres de gestion)

Le tableau suivant décrit les différentes entités composant une fonction :

Entité	Description
Fonction	Type de fonction de communication.
Adresse cible	Adresse du destinataire de l'échange.
Paramètres spécifiques	Dépendent du type de fonction de communication. Une description est fournie pour chaque fonction de communication.
Paramètres de gestion	Communs à toutes les fonctions de communication asynchrones. Ils sont constitués : <ul style="list-style-type: none"> ● d'un paramètre fournissant des données relatives à l'activité de la fonction, ● d'un paramètre indiquant le numéro d'échange identifiant la transaction en cours, ● d'un paramètre contenant le compte rendu de l'échange (comptes rendus de communication et d'opération), ● d'un paramètre de timeout pouvant être utilisé pour vérifier l'absence de réponse, ● d'un paramètre de longueur permettant de stocker le nombre d'octets à envoyer ou le nombre d'octets reçus.

Adresse destinataire

Vue d'ensemble

Ce paramètre indique l'adresse de l'équipement destinataire (*voir Premium, Atrium et Quantum sous Unity Pro, Architectures et services de communication, Manuel de référence*) de l'échange.

Il peut être localisé :

- soit par des mots internes (%MW) ou des constantes internes (%KW),
- soit être écrit directement en valeur immédiate.

Pour faciliter la phase de préparation de l'échange, il existe la fonction `ADDR` (uniquement Unity Pro ou version ultérieure) qui convertit une valeur immédiate de type adresse (chaîne de caractère) en un tableau comportant toujours six mots internes (%MW).

Exemple

```
%MWi:6:=ADDR(' {2.4}SYS');
```


Structure des paramètres de gestion

Présentation

Les paramètres de gestion sont regroupés dans un tableau de quatre entiers. Les valeurs contenues dans ce tableau servent à gérer les fonctions de communication.

NOTE : dans la documentation technique, ces paramètres sont également désignés par les termes « table de gestion » ou « compte rendu de gestion ».

NOTE : les deux premiers mots sont gérés par le système. Vous êtes responsable de la gestion des deux derniers mots.

Structure

Le tableau suivant décrit la structure des données de la table de gestion des fonctions de communication :

	Ordre du mot	Octet de poids fort	Octet de poids faible
Données gérées par le système	1	Numéro d'échange	Bit d'activité
	2	Compte rendu d'opération (voir page 35)	Compte rendu de communication (voir page 34)
Données gérées par l'utilisateur	3	Timeout (voir page 37)	
	4	Longueur (voir page 37)	

Bit d'activité

Ce bit indique l'état d'exécution de la fonction de communication.

Il prend la valeur 1 lorsqu'il est lancé et revient à 0 une fois l'exécution terminée.

Il s'agit du premier bit du premier élément de la table.

Exemple : Si la table de gestion a été déclarée de la façon suivante :

```
Tab_Gest ARRAY [1..4] OF INT, le bit d'activité est le bit avec la notation Tab_Gest[1].0.
```

NOTE : la notation utilisée précédemment nécessite de configurer les propriétés du projet de façon à autoriser l'extraction de bits sur les types d'entier. Dans le cas contraire, l'accès à `Tab_Gest[1].0` ne peut pas être réalisé de cette manière.

Numéro d'échange

Lorsqu'une fonction de communication est envoyée, le système lui attribue automatiquement un numéro, permettant ainsi l'identification de l'échange.

Ce numéro peut être utilisé lorsqu'il est nécessaire d'arrêter l'échange en cours (à l'aide de la fonction `CANCEL` (voir page 50)).

Paramètres de gestion : comptes rendus de communication et d'opération

Vue d'ensemble

Les comptes rendus de communication et d'opération font partie des paramètres de gestion.

NOTE : il est recommandé de toujours tester les comptes rendus sur la fonction de communication à la fin de leur exécution et avant l'activation suivante. Lors d'un démarrage à froid, il est impératif de vérifier tous les paramètres de gestion de la fonction de communication et de les remettre à 0.

Compte rendu de communication

Ce compte rendu est commun à toutes les fonctions. Il est pertinent lorsque la valeur du bit d'activité passe de 1 à 0.

Les comptes rendus dont la valeur est comprise entre 16#01 et 16#FE concernent des erreurs détectées par le processeur qui a exécuté la fonction.

Le tableau suivant indique les différentes valeurs que peut prendre ce compte rendu :

Valeur	Rapport de communication (octet de poids faible)
16#00	Echange correct
16#01	Echange interrompu à expiration du délai
16#02	Echange interrompu sur demande de l'utilisateur (CANCEL)
16#03	Format d'adresse incorrect
16#04	Adresse cible incorrecte
16#05	Format du paramètre de gestion incorrect
16#06	Paramètres spécifiques incorrects
16#07	Problème lors de l'envoi à la cible
16#08	Réservé
16#09	Taille du tampon de réception insuffisante
16#0A	Taille du tampon d'envoi insuffisante
16#0B	Pas de ressources système du processeur
16#0C	Numéro d'échange incorrect
16#0D	Aucun télégramme reçu
16#0E	Longueur incorrecte
16#0F	Service de télégramme non configuré

Valeur	Rapport de communication (octet de poids faible)
16#10	Module réseau manquant
16#11	Requête manquante
16#12	Serveur d'application déjà actif
16#13	Numéro de transaction UNI-TE V2 incorrect
16#FF	Message refusé

NOTE : la fonction peut détecter une erreur de paramètre avant d'activer l'échange. Dans ce cas, le bit d'activité reste à 0 et le compte rendu est initialisé avec les valeurs correspondant à l'erreur.

NOTE : la valeur 16#FF est renvoyée, indiquant un échange correct effectué avec la fonction `WRITE_VAR` dans une requête de diffusion Modbus. Cette valeur de rapport est mise en œuvre dans SCY 21601 à partir de la version 2.8 IE46, dans SCY 11601 à partir de la version 1.2 IE11 et dans TSX SCP 111/114 à partir de la version 3.2 IR25.

Compte rendu d'opération

Cet octet de compte rendu est spécifique à chaque fonction et indique le résultat de l'opération sur l'application distante.

Il n'est pertinent que si le compte rendu de communication a les valeurs suivantes :

- 16#00 (échange correct),
- 16#FF (message refusé).

Si la valeur du compte rendu de communication est 16#00, le compte rendu d'opération a les valeurs suivantes :

Valeur	Rapport d'opération (octet de poids fort)
16#00	Résultat positif
16#01	Requête non traitée
16#02	Réponse incorrecte
16#03	Réservé REMARQUE : pour Premium, ce rapport signale une taille erronée de tampon de réception (le tampon de réception est trop petit pour contenir la réponse).
Code requête + 16#30	Lors d'une réponse positive à certaines requêtes
16#FB	Lors d'une réponse à une requête mineure
16#FD	Erreur de fonctionnement
16#FE	Lors d'une réponse positive à certaines requêtes

Si la valeur du compte rendu de communication est 16#FF, le compte rendu d'opération a les valeurs suivantes :

Valeur	Rapport d'opération (octet de poids fort)
16#01	Pas de ressources vers le processeur
16#02	Pas de ressources de ligne
16#03	Aucun équipement ou équipement sans ressource (*)
16#04	Erreur de ligne
16#05	Erreur de longueur
16#06	Voie de communication défectueuse
16#07	Erreur d'adressage
16#08	Erreur d'application
16#0B	Pas de ressources système
16#0C	Fonction de communication non active
16#0D	Cible manquante
16#0F	Problème de routage intrastation ou voie non configurée
16#11	Format d'adresse non pris en charge
16#12	Aucune ressource cible
16#14	Connexion non opérationnelle (exemple : Ethernet TCP/IP)
16#15	Aucune ressource sur la voie locale
16#16	Accès non autorisé (exemple : Ethernet TCP/IP)
16#17	Configuration réseau incorrecte (exemple : Ethernet TCP/IP)
16#18	Connexion temporairement indisponible
16#21	Serveur d'application arrêté
16#30	Erreur d'émission
Légende :	
(*)	Code uniquement géré par les cartes PCMCIA : TSX FPP20 et TSX FPP10

Paramètres de gestion : longueur et timeout

Vue d'ensemble

Vous devez définir ces deux paramètres.

Longueur

Le paramètre de longueur est utilisé pour définir le nombre de caractères (en octets) à envoyer lors de l'émission, mais également pour stocker le nombre de caractères (en octets) reçus après la réception d'un message.

Avant le lancement de certaines fonctions de communication (`SEND_REQ`, `DATA_EXCH`, `PRINT_CHAR` et `SEND_TLG`), il est recommandé, voire obligatoire pour certaines d'entre elles, de mettre à jour le paramètre de longueur.

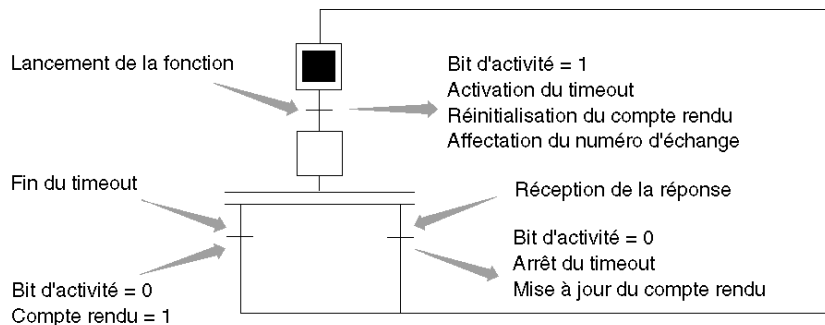
NOTE : Exemple de la fonction `PRINT_CHAR`. Si une autre fonction de l'application utilise la même table de compte rendu dans laquelle le nombre d'octets à envoyer est différent de celui de la fonction précédente, il est impératif d'initialiser le paramètre de longueur avec le nouveau nombre d'octets à émettre. Dans le cas contraire, le nombre d'octets envoyés par la fonction précédente est conservé.

Timeout

Le timeout définit le temps d'attente maximum pour la réponse. La base de temps de ce paramètre est de 100 ms (la valeur 0 correspond à une valeur d'attente infinie).

Une fois le timeout expiré, un compte rendu d'erreur est renvoyé pour l'échange. En outre, le système n'accepte aucune réponse après la fin du timeout.

Exemple



NOTE : La valeur de timeout d'une fonction de communication doit être suffisamment élevée pour garantir la réception de la réponse à la question posée (utiliser un modem externe sur une liaison basée sur un protocole, par exemple).

NOTE : Pour les communications de maître Modbus, un timeout d'application défini dans les fonctions de communication doit être supérieur au timeout de l'écran de configuration multiplié par le nombre de tentatives (timeout matériel).

NOTE : Le timeout des EF de communication (tel que `WRITE_VAR` ou `READ_VAR`) doit être supérieur au timeout de l'équipement maître de communication (délai de réponse).

Fonction de serveur

Présentation

La fonction de serveur peut servir à répondre aux requêtes émanant d'équipements clients.

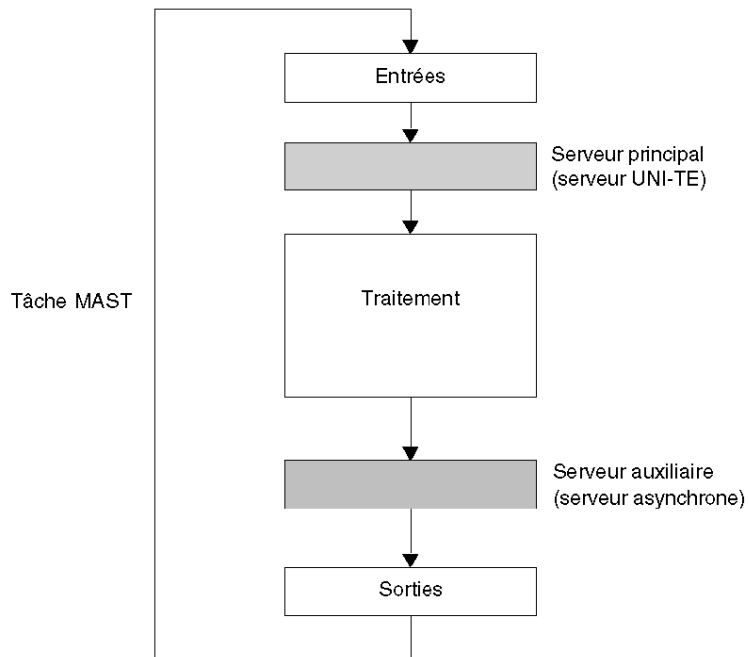
Les processeurs TSX 57 35• et PCX 57 35• disposent de deux serveurs de requêtes :

- un serveur principal (recommandé pour les requêtes inférieures à 256 octets),
- un serveur auxiliaire (recommandé pour les requêtes de 1 024 octets maximum).

Ces deux serveurs peuvent être activés simultanément.

Illustration

Le diagramme suivant illustre l'envoi des requêtes aux serveurs au cours du cycle d'automate :



Serveur principal

Ce serveur correspond au port 0 (serveur UNI-TE). Il est activé au démarrage du cycle MAST de l'automate.

Le temps de réponse de l'automate client varie en fonction de la durée du cycle de l'automate du serveur. Il est ainsi possible de traiter jusqu'à 4 requêtes simultanées par cycle d'automate.

Toutes les requêtes UNI-TE sont prises en charge. Une requête doit être inférieure à 256 octets.

Cette entité peut être adressée à l'adresse topologique `SYS` ou `{réseau.station}SYS`.

Serveur auxiliaire

Ce serveur correspond au port 7 (serveur asynchrone). Il est activé uniquement pour des tâches périodiques à la fin du cycle d'automate, une fois la tâche MAST traitée alors que le cycle suivant est sur le point de démarrer.

Le démarrage du cycle suivant, avec une priorité plus élevée, peut interrompre une requête en cours. C'est pourquoi l'accès à ce serveur est réservé aux applications pour lesquelles les opérations de lecture/écriture de données ne doivent pas nécessairement correspondre.

Le temps de réponse de l'application dépend essentiellement de la durée du cycle de l'automate. La taille d'une requête peut atteindre jusqu'à 1 024 octets. Ce serveur n'est pas accessible via une fonction de communication. Il traite les requêtes de lecture et écriture des objets (bit ou mot), etc.

Etendu



Objet de cette partie

Cette section décrit les fonctions et blocs fonction élémentaires de la famille Etendu.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre	Titre du chapitre	Page
4	ADDM: conversion d'adresse	43
5	ADDR : conversion d'adresse	47
6	CANCEL : arrêt d'un échange en cours	49
7	CREAD_REG : lecture de registre continu	53
8	CWRITE_REG : écriture de registre continu	61
9	DATA_EXCH : échange de données entre applications	69
10	INPUT_BYTE : réception de chaînes de caractères	79
11	INPUT_CHAR : réception de chaînes de caractères	83
12	MBP_MSTR : maître Modbus Plus	95
13	ModbusP_ADDR : adresse Modbus Plus	151
14	OUT_IN_CHAR : envoi/réception de chaînes de caractères	157
15	OUT_IN_MBUS : fonction de communication Modbus	167
16	PRINT_CHAR : envoi de chaînes de caractères	205
17	RCV_TLG : réception de télégrammes	217
18	READ_ASYN : lecture de données de façon asynchrone	223
19	READ_GDATA : lecture des données globales Modbus Plus	227
20	READ_REG : lecture de registres	229
21	READ_VAR : lecture de variables	239
22	SEND_EMAIL : messagerie électronique	257
23	SEND_REQ : envoi de requêtes	261

Chapitre	Titre du chapitre	Page
24	SEND_TLG : envoi de télégrammes	279
25	SYMAX_IP_ADDR : adresse IP SY/MAX	285
26	TCP_IP_ADDR : adresse TCP/IP	291
27	UNITE_SERVER : serveur immédiat	297
28	WRITE_ASYN : écriture de données de façon asynchrone	303
29	WRITE_GDATA : écriture des données globales Modbus Plus	307
30	WRITE_REG : écriture de registre	309
31	WRITE_VAR : écriture de variables	319
32	XXMIT : transmettre	333

ADDM: conversion d'adresse

4

Description

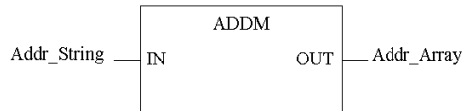
Description de la fonction

La fonction `ADDM` permet de convertir une chaîne de caractères en une adresse pouvant être utilisée directement par les fonctions de communication suivantes : `READ_VAR`, `WRITE_VAR`, `INPUT_CHAR`, `PRINT_CHAR`, `DATA_EXCH`, `SEND_EMAIL`.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

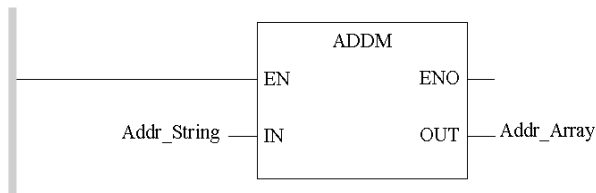
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

```

Représentation :
LD Addr_String
ADDM
ST Addr_Array
    
```

Représentation en ST

```

Représentation :
Addr_Array := ADDM(Addr_String);
    
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
IN	Chaîne de caractères.	Adresse de l'équipement sur un bus ou sur un réseau.

Pour effectuer l'adressage d'une station sur Ethernet, le paramètre IN prend la forme suivante :

- 'lien_réseau{adresse_hôte}'
- 'lien_réseau{adresse_hôte}TCP.MBS'
- 'lien_réseau{adresse_hôte}nœud'
- 'r.m.c{adresse_hôte}'
- 'r.m.c{adresse_hôte}TCP.MBS'
- 'r.m.c{adresse_hôte}'
- 'r.m.c{adresse_hôte}'
- 'r.m.c{adresse_hôte}TCP.MBS'
- 'r.m.c{adresse_hôte}'

Il y a aussi la possibilité d'utiliser la liaison par défaut à l'aide de la notation simplifiée pour adresser une station sur Ethernet :

- ADDM'{adresse_hôte}'
La liaison par défaut est la liaison configurée la plus proche de l'UC.

Avec :

- lien_réseau : nom de réseau défini dans le champ Lien réseau de la voie Ethernet
- adresse_hôte : adresse IP de l'équipement
- r : numéro de rack
- m : position du module
- c : numéro de voie
- nœud : nœud Modbus ou CANopen derrière une passerelle (passerelle identifiée par adresse_hôte)
- TCP.MBS : pour adressage d'un serveur Modbus TCP

Pour effectuer l'adressage d'un équipement sur un bus CANopen, le paramètre IN prend la forme 'r.m.c.e', où :

- r : numéro de rack
- m : position du module
- c : numéro de voie du port CANopen (2)
- e : nœud esclave CANopen (équipement, de 1 à 127)

Pour effectuer l'adressage d'un équipement à l'aide du protocole Modbus, le paramètre IN prend la forme :

- 'r.m.c.e.MBS'

Avec :

- r : numéro de rack
- m : position du module
- c : numéro de voie du port Modbus (0)
- e : numéro d'esclave Modbus (équipement, de 1 à 247)
- MBS : pour adressage d'un serveur Modbus

Pour l'adressage d'un appareil utilisant le protocole en mode caractère, le paramètre IN prend la forme 'r.m.c' ou 'r.m.c.SYS", où :

- r : numéro de rack
- m : position du module
- c : numéro de voie du port Mode caractère (0)
- SYS : mot clé utilisé pour indiquer le système du serveur de la station. SYS n'est pas obligatoire.

Pour plus d'informations, reportez-vous à la présentation de l'adressage des automates M340 dans le manuel utilisateur Architectures de communication.

Le tableau suivant décrit le paramètre de sortie :

Paramètre	Type	Commentaire
OUT	ADDM_TYPE Tableau de 8 entiers simples	Tableau représentant l'adresse d'un équipement. Ce paramètre est utilisable par plusieurs fonctions de communication comme paramètre d'entrée.

Le bloc ADDM analyse la syntaxe de la chaîne d'adressage (paramètre IN) et place le résultat dans un tableau de 8 entiers simples qui définit l'adresse de destination. L'adresse de destination peut être fournie par le paramètre OUT de la fonction ADDM ou directement par un tableau de 8 objets INT. Cependant, il est vivement recommandé d'utiliser la fonction ADDM pour effectuer l'adressage d'une fonction élémentaire de communication.

Structure de l'adresse de destination :

Champ	Taille	Valeur
Type	Octet	Réservé.
ClientID	Octet	Réservé.
Rack	Octet	Numéro d'emplacement du rack.
Emplacement	Octet	Numéro d'emplacement du module.
Voie	Octet	Numéro de voie.
ProtId	Octet	Réservé. 0 pour Modbus.
AddrLen	Octet	Cet octet peut avoir les valeurs suivantes : <ul style="list-style-type: none"> ● 0 si le serveur du module ou de la voie est adressé (UnitId et AddrExt non utilisés), ● 1 si le numéro de l'équipement est défini, ● >1 si AddrExt est également utilisé.
UnitId	Octet	Numéro d'équipement (équipement Modbus, par exemple).
AddrExt	Array[7]	Réservé. Permet de coder des informations supplémentaires concernant les adresses TCP/IP.

AVERTISSEMENT

COMPORTEMENT INATTENDU DE L'APPLICATION

L'utilisation de paramètres réservés relève de la responsabilité de l'utilisateur et peut provoquer des dysfonctionnements.

Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

ADDR : conversion d'adresse

5

Description

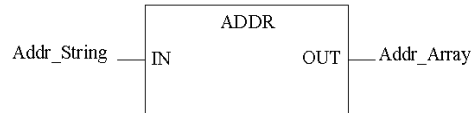
Description de la fonction

La fonction ADDR permet de convertir une chaîne de caractères en une adresse pouvant être utilisée directement par les fonctions de communication.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

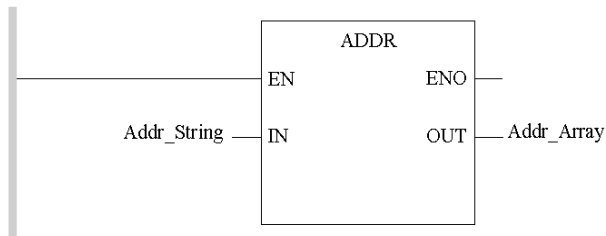
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Addr_String

ADDR

ST Addr_Array

Représentation en ST

Représentation :

Addr_Array := ADDR(Addr_String);

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
Addr_String	STRING	Variable sous forme de chaîne de caractères représentant l'adresse de l'équipement sur un bus ou un réseau. Exemples : '{5.6}SYS', '{1.2}0.4.5.2'

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Addr_Array	ADDR_TYPE ou ARRAY [0.0,5] OF INT	Tableau composé de 6 entiers représentant l'adresse Addr_String. Addr_Array peut être utilisé directement comme premier paramètre d'entrée des fonctions élémentaires de communication.

ATTENTION

COMPORTEMENT IMPREVU DE L'APPLICATION

N'utilisez pas des paramètres d'adresse erronés. Par exemple :

- Ne définissez pas un paramètre d'adresse ne correspondant pas à l'équipement cible.
- Ne définissez pas de valeurs supérieures à 98 dans la fonction ADDR (champ "e" de l'adresse de l'équipement) lorsque vous utilisez un port série intégré à l'UC ou une voie 0 ou 1 d'un module TSX SCY 21601.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

CANCEL : arrêt d'un échange en cours

6

Objet de ce chapitre

Ce chapitre décrit la fonction CANCEL.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	50
Exemple d'annulation d'un échange	52

Description

Description de la fonction

La fonction `CANCEL` permet d'interrompre une fonction de communication asynchrone en cours. Le numéro d'échange affecté à chaque communication permet d'identifier la fonction à arrêter.

Le laps de temps entre la requête de fonction `CANCEL` et l'action `CANCEL` dépend du nombre de fonctions de communication en cours. L'utilisation d'une fonction `CANCEL` signifie, pour toutes les voies et toutes les fonctions élémentaires (EF), que :

- les fonctions de communication asynchrones affectées qui sont en cours sont annulées,
- les trames dédiées stockées dans le tampon avant l'utilisation de la fonction `CANCEL` peuvent être envoyées.

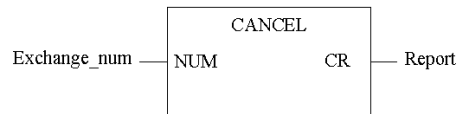
NOTE : en mode caractère, les fonctions élémentaires affectées `INPUT_CHAR`, `INPUT_BYTE` et/ou `OUT_IN_CHAR` en cours sont annulées et la voie de liaison série asynchrone est déverrouillée si elle est en attente de critères de fin.

NOTE : l'exécution de cette fonction est synchrone à l'exécution du programme automate (la fonction de communication est arrêtée dans le cycle automate où a été exécutée la fonction `CANCEL`).

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

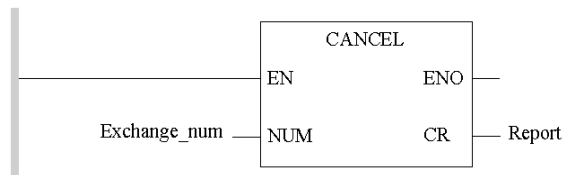
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Exchange_Num

CANCEL

ST Report

Représentation en ST

Représentation :

CANCEL(Exchange_Num, Report);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètres	Type	Commentaire
Exchange_Num	INT	Ce paramètre spécifie le numéro de l'échange dont l'exécution doit être interrompue.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Compte rendu	INT	Ce compte rendu d'opération prend l'une des deux valeurs suivantes : <ul style="list-style-type: none"> ● 16#00 : opération correcte. La communication est interrompue, le bit d'activité de la fonction interrompue est réglé sur 0 et son compte rendu prend la valeur 2. ● 16#0C : numéro d'échange incorrect.

Exemple d'annulation d'un échange

Présentation

Cet exemple concerne l'annulation d'un échange effectué avec la fonction `OUT_IN_CHAR`.

Lors du lancement de l'échange, un numéro unique lui est affecté. Ce numéro reste valide jusqu'à la fin de l'échange.

La fonction `CANCEL` utilise ce numéro pour interrompre l'échange correspondant.

Programmation de la fonction `OUT_IN_CHAR`

Programmation en ST :

```
IF RE(%I0.3.8) AND NOT %MW170.0 THEN
  (* initialisation des données à envoyer *)
  %MW173 := 10;
  (* fonction de communication *)

  OUT_IN_CHAR(ADDR('{20.5}0.0.0.SYS'),1,Str_Out,Str_In,%MW170:
4);
END_IF;
```

Programmation de la fonction `CANCEL`

Programmation en ST :

```
%MW180 := SHRZ_INT(%MW170,8);
IF RE(%I0.3.9) THEN
  CANCEL(%MW180,%MW185);
END_IF;
```

La fonction `CANCEL` comprend deux paramètres :

- en entrée : numéro de l'échange à annuler,
- en sortie : compte rendu.

Il est nécessaire d'initialiser le premier paramètre avec le numéro de l'échange à annuler. Ce numéro est situé dans l'octet de poids fort du premier mot de la table de gestion. Dans le cas présent, il s'agit de l'octet de poids fort de `%MW170`. Avant d'envoyer la fonction `CANCEL`, il est nécessaire de créer un décalage de 8 bits afin de récupérer les 8 bits de poids fort de `%MW170`.

Paramètres de la requête :

Paramètres	Description
<code>%MW180</code>	Le bit de poids faible contient le numéro de l'échange à annuler. ATTENTION , l'octet de poids fort doit être défini sur zéro.
<code>%MW185</code>	Compte rendu de fonction.

CREAD_REG : lecture de registre continu

7

Introduction

Ce chapitre décrit le bloc CREAD_REG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	54
Types de données dérivés	56
Mode de fonctionnement	58
Description des paramètres	59

Description

Description de la fonction

Ce bloc fonction est prévu pour pouvoir lire une zone de registre en continu. Il lit les données depuis un abonné adressé par Modbus Plus, Ethernet TCP/IP ou Ethernet SY/MAX.

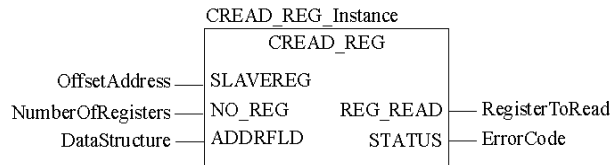
Les paramètres supplémentaires EN et ENO peuvent être configurés.

NOTE : Lorsque vous programmez une fonction CREAD_REG, il vous faut connaître les procédures de routage utilisées par votre réseau. Les structures des itinéraires de routage Modbus Plus sont décrites en détail dans le *Guide de planification et d'installation du réseau Modbus Plus*. Si un routage TCP/IP ou Ethernet SY/MAX est implémenté, vous devez obligatoirement utiliser des produits routeurs standard Ethernet IP. Vous trouverez une description détaillée du routage TCP/IP dans le *Guide utilisateur de Quantum avec la configuration Unity Pro TCP/IP*.

NOTE : Plusieurs exemplaires de ce bloc fonction peuvent être utilisés dans le programme. Il n'est cependant pas possible de procéder à une instanciation multiple de ces exemplaires.

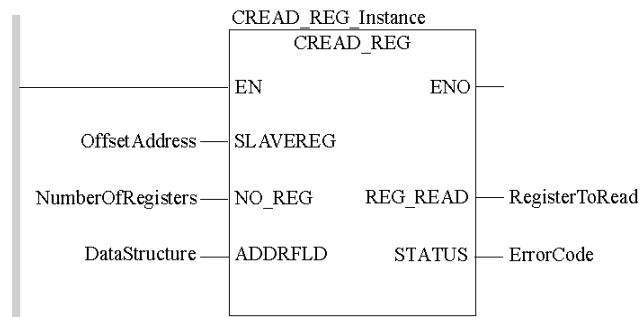
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL CREAD_REG_Instance (SLAVEREG:=OffsetAddress,
    NO_REG:=NumberOfRegisters, ADDRFLD:=DataStructure,
    REG_READ=>RegisterToRead, STATUS=>ErrorCode)
```

Représentation en ST

Représentation :

```
CREAD_REG_Instance (SLAVEREG:=OffsetAddress,
    NO_REG:=NumberOfRegisters, ADDRFLD:=DataStructure,
    REG_READ=>RegisterToRead, STATUS=>ErrorCode) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Signification
SLAVEREG	DINT	Adresse de la première adresse %MW devant être lue dans l'esclave.
NO_REG	INT	Nombre d'adresses à lire par l'esclave.
ADDRFLD	WordArr5	Structure de données décrivant l'adresse Modbus Plus, l'adresse TCP/IP ou l'adresse SY/MAX-IP.

Description des paramètres de sortie :

Paramètres	Type de données	Signification
REG_READ	ANY	Données à lire Une structure de données doit être déclarée en tant que variable localisée pour les données à lire.
STATUS	WORD	Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie. Code d'erreur, voir : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137 ● Codes d'erreur spécifiques SY/MAX, page 142 ● Codes d'erreur Ethernet TCP/IP, page 144

Erreur d'exécution

Pour obtenir une liste de tous les codes et valeurs d'erreur du bloc, voir *Booléen étendu*, page 416.

Types de données dérivés

Type de données dérivé `WordArr5` sur Modbus Plus

Description des éléments :

Élément	Type de données	Description
<code>WordArr5 [1]</code>	WORD	Octet de poids faible : Registre 1 de routage, sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau. Le dernier octet différent de zéro de l'itinéraire de routage est l'abonné cible. Octet de poids fort : Adresse de l'abonné source. <ul style="list-style-type: none"> ● Position de l'emplacement du module lors de l'utilisation du port Modbus Plus sur le module NOM. ● Si vous utilisez le port Modbus Plus de l'UC, cet octet doit être réglé sur 0 (pour tous les emplacements de l'UC).
<code>WordArr5 [2]</code>	WORD	Registre 2 de routage
<code>WordArr5 [3]</code>	WORD	Registre 3 de routage
<code>WordArr5 [4]</code>	WORD	Registre 4 de routage
<code>WordArr5 [5]</code>	WORD	Registre 5 de routage

Description de `WordArr5` sur Ethernet TCP/IP

Description de `WordArr5` sur Ethernet TCP/IP :

Élément	Type de données	Description
<code>WordArr5 [1]</code>	WORD	Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET) Octet de poids fort : Emplacement du module NOE
<code>WordArr5 [2]</code>	WORD	Octet 4 (octet de poids fort) de l'adresse IP cible 32 bits
<code>WordArr5 [3]</code>	WORD	Octet 3 de l'adresse IP cible 32 bits
<code>WordArr5 [4]</code>	WORD	Octet 2 de l'adresse IP cible 32 bits
<code>WordArr5 [5]</code>	WORD	Octet 1 (octet de poids faible) de l'adresse IP cible 32 bits

Description de WordArr5 sur Ethernet SY/MAX

Description de WordArr5 sur Ethernet SY/MAX :

Élément	Type de données	Description
WordArr5[1]	WORD	Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET) Octet de poids fort : Emplacement du module NOE
WordArr5[2]	WORD	Numéro de station cible (ou mettre FF en hexadécimal)
WordArr5[3]	WORD	Terminaison (ou mettre FF en hexadécimal)
WordArr5[4]	WORD	Réservé
WordArr5[5]	WORD	Réservé

Mode de fonctionnement

Mode de fonctionnement du bloc CREAD_REG

Un grand nombre de blocs fonction CREAD_REG peut être programmé, mais seules quatre opérations de lecture peuvent être actives en même temps. Que celles-ci soient déclenchées par ce bloc fonction ou par d'autres (p. ex. MBP_MSTR, MSTR, READ_REG), tous les blocs fonction utilisent la même session de transaction de données et nécessitent plusieurs cycles de programme pour achever une commande.

NOTE : Une communication TCP/IP entre un API Quantum (NOE 211 00) et un API Momentum (toutes les UC TCP/IP et tous les modules d'E/S TCP/IP) n'est possible que si **une seule** tâche de lecture ou d'écriture est effectuée dans chaque cycle d'API. Si plusieurs tâches sont envoyées par cycle, la communication est stoppée, sans qu'un message d'erreur soit généré dans le registre d'état.

L'information complète de routage est contenue dans la structure de données WordArr5 de l'entrée ADDRFLD. Le type du bloc fonction lié à cette entrée se règle en fonction du réseau utilisé.

Vous devez utiliser pour :

- Modbus Plus le bloc fonction ModbusP_ADDR
- Ethernet TCP/IP le bloc fonction TCP_IP_ADDR
- Ethernet SY/MAX le bloc fonction SYMAX_IP_ADDR

NOTE : Vous pouvez également utiliser la structure de données WordArr5 avec des constantes.

NOTE : Ce bloc fonction produit une lourde charge sur le réseau ; il est donc conseillé de contrôler soigneusement la performance du réseau. Si ce dernier est surchargé, le programme devra être restructuré afin de travailler avec le bloc fonction READ_REG, une variante du présent bloc fonction, qui fonctionne sur demande et non en mode continu.

Description des paramètres

SLAVEREG

Début de la zone de l'escalve adressé dans laquelle les données sont lues. La zone source réside toujours dans la zone d'adresse %MW.

NOTE : Pour les esclaves d'un automate **non-Unity Pro** :

La zone source réside toujours dans la zone de registre 4x. SLAVEREG voit l'adresse source comme un décalage à l'intérieur de la zone 4x. Le "4" de tête doit être omis (p. ex. 59 (contenu de la variable ou valeur du littéral) = 40059).

Ce paramètre peut être indiqué comme adresse, variable localisée, variable non localisée ou littéral .

NO_REG

Nombre d'adresses à lire dans l'équipement esclave adressé (1 ... 100).

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

REG_READ

Pour ce paramètre un `ARRAY` de la taille de la transmission demandée doit être spécifié (\geq NO_REG). Le nom de ce tableau est transmis comme paramètre. Si le tableau est défini sur une taille trop réduite, la quantité de données transmise sera limitée par la place proposée dans le tableau.

Ce paramètre doit être indiqué comme variable localisée.

STATUS

Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie.

Code d'erreur, voir :

- *Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137*
- *Codes d'erreur spécifiques SY/MAX, page 142*
- *Codes d'erreur Ethernet TCP/IP, page 144*

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

CWRITE_REG : écriture de registre continu



8

Introduction

Ce chapitre décrit le bloc CWRITE_REG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	62
Types de données dérivés	65
Mode de fonctionnement	67
Description des paramètres	68

Description

Description de la fonction

Ce bloc fonction est prévu pour pouvoir écrire une zone de registre en continu. Il transmet des données depuis l'API par Modbus Plus, Ethernet TCP/IP ou Ethernet SY/MAX sur un esclave adressé.

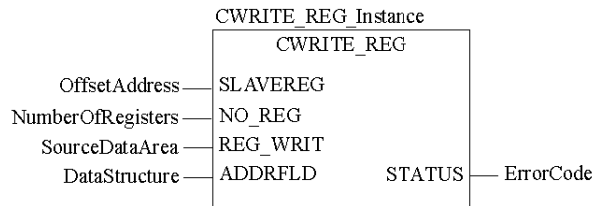
Les paramètres supplémentaires EN et ENO peuvent être configurés.

NOTE : Lorsque vous programmez une fonction CWRITE_REG, il vous faut connaître les procédures de routage utilisées par votre réseau. Les structures des itinéraires de routage Modbus Plus sont décrites en détail dans le *Guide de planification et d'installation du réseau Modbus Plus*. Si un routage TCP/IP ou Ethernet SY/MAX est implémenté, vous devez obligatoirement utiliser des produits routeurs standard Ethernet IP. Vous trouverez une description détaillée du routage TCP/IP dans le *Guide utilisateur de Quantum avec la configuration Unity Pro TCP/IP*.

NOTE : Plusieurs exemplaires de ce bloc fonction peuvent être utilisés dans le programme. Il n'est cependant pas possible de procéder à une instanciation multiple de ces exemplaires.

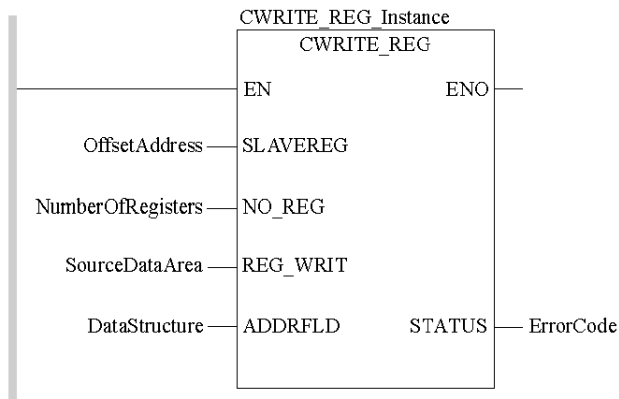
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```

CAL CWRITE_REG_Instance (SLAVEREG:=OffsetAddress,
  NO_REG:=NumberOfRegisters, REG_WRIT:=SourceDataArea,
  ADDRFLD:=DataStructure, STATUS=>ErrorCode)
  
```

Représentation en ST

Représentation :

```

CWRITE_REG_Instance ( SLAVEREG:=OffsetAddress,
  NO_REG:=NumberOfRegisters, REG_WRIT:=SourceDataArea,
  ADDRFLD:=DataStructure, STATUS=>ErrorCode) ;
  
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Signification
SLAVEREG	DINT	Adresse de la première adresse %MW devant être écrite dans l'esclave.
NO_REG	INT	Nombre d'adresses à écrire dans l'esclave
REG_WRIT	ANY	Données source (Une structure de données doit être déclarée en tant que variable localisée pour les données source.)
ADDRFLD	WordArr5	Structure de données pour la transmission de l'adresse Modbus Plus, de l'adresse TCP/IP ou de l'adresse SY/MAX-IP.

Description des paramètres de sortie :

Paramètres	Type de données	Signification
STATUS	WORD	Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie. Code d'erreur, voir : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137 ● Codes d'erreur spécifiques SY/MAX, page 142 ● Codes d'erreur Ethernet TCP/IP, page 144

Erreur d'exécution

Pour obtenir une liste de tous les codes et valeurs d'erreur du bloc, voir *Booléen étendu*, page 416.

Types de données dérivés

Description de WordArr5 sur Modbus Plus

Description de WordArr5 sur Modbus Plus :

Élément	Type de données	Description
WordArr5[1]	WORD	<p>Octet de poids faible :</p> <p>Registre 1 de routage, sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau.</p> <p>Le dernier octet différent de zéro de l'itinéraire de routage est l'abonné cible.</p> <p>Octet de poids fort :</p> <p>Adresse de l'abonné source.</p> <ul style="list-style-type: none"> ● Position de l'emplacement du module lors de l'utilisation du port Modbus Plus sur le module NOM. ● Si vous utilisez le port Modbus Plus de l'UC, cet octet doit être réglé sur 0 (pour tous les emplacements de l'UC).
WordArr5[2]	WORD	Registre 2 de routage
WordArr5[3]	WORD	Registre 3 de routage
WordArr5[4]	WORD	Registre 4 de routage
WordArr5[5]	WORD	Registre 5 de routage

Description de WordArr5 sur Ethernet TCP/IP

Description de WordArr5 sur Ethernet TCP/IP :

Élément	Type de données	Description
WordArr5[1]	WORD	<p>Octet de poids faible :</p> <p>Index de mappage MBP sur Ethernet Transporter (MET)</p> <p>Octet de poids fort :</p> <p>Numéro d'emplacement du module NOE</p>
WordArr5[2]	WORD	Octet 4 (octet de poids fort) de l'adresse IP cible 32 bits
WordArr5[3]	WORD	Octet 3 de l'adresse IP cible 32 bits
WordArr5[4]	WORD	Octet 2 de l'adresse IP cible 32 bits
WordArr5[5]	WORD	Octet 1 (octet de poids faible) de l'adresse IP cible 32 bits

Description de wordArr5 sur Ethernet SY/MAX

Description de WordArr5 sur Ethernet SY/MAX :

Élément	Type de données	Description
WordArr5 [1]	WORD	Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET) Octet de poids fort : Emplacement du module NOE
WordArr5 [2]	WORD	Numéro de station cible (ou mettre FF en hexadécimal)
WordArr5 [3]	WORD	Terminaison (ou mettre FF en hexadécimal)
WordArr5 [4]	WORD	Réservé
WordArr5 [5]	WORD	Réservé

Mode de fonctionnement

Mode de fonctionnement du module CWRITE_REG

Un grand nombre de blocs fonction CWRITE_REG peut être programmé, mais seules quatre commandes d'écriture peuvent être actives en même temps. Que celles-ci soient déclenchées par ce bloc fonction ou par d'autres (p. ex. MBP_MSTR, MSTR, WRITE_REG) n'est pas significatif. Tous les blocs fonction utilisent la même session de transaction de données et nécessitent plusieurs cycles de programme pour achever une commande.

Si plusieurs blocs fonction CWRITE_REG sont utilisés dans une application, ils doivent se différencier entre eux au moins par les paramètres NO_REG ou REG_WRITE.

NOTE : Une communication TCP/IP entre un API Quantum (NOE 211 00) et un API Momentum (toutes les UC TCP/IP et tous les modules d'E/S TCP/IP) n'est possible que si **une** seule tâche de lecture ou d'écriture est effectuée dans chaque cycle d'API. Si plusieurs tâches par cycle sont envoyées, la communication est stoppée, sans qu'un message d'erreur soit généré dans le registre d'état.

L'information complète de routage est contenue dans la structure de données WordArr5 de l'entrée ADDRFLD. Le type du bloc fonction lié à cette entrée se règle en fonction du réseau utilisé.

Vous devez utiliser pour :

- Modbus Plus le bloc fonction ModbusP_ADDR
- Ethernet TCP/IP le bloc fonction TCP_IP_ADDR
- Ethernet SY/MAX le bloc fonction SYMAX_IP_ADDR

NOTE : Vous pouvez également utiliser la structure de données WordArr5 avec des constantes.

NOTE : Ce bloc fonction produit une lourde charge sur le réseau ; il est donc conseillé de contrôler soigneusement la performance du réseau. Si ce dernier est surchargé, le programme devra être restructuré afin de travailler avec le bloc fonction WRITE_REG, une variante du présent bloc fonction, qui fonctionne sur demande et non en mode continu.

Description des paramètres

SLAVEREG

Début de la zone de l'abonné cible dans laquelle les données sont écrites. La zone cible réside toujours dans la zone d'adresse %MW.

NOTE : Pour les esclaves d'un automate **non-Unity Pro** :

La zone cible réside toujours dans la zone de registre 4x. SLAVEREG voit l'adresse cible comme un décalage à l'intérieur de la zone 4x. Le "4" de tête doit être omis (p. ex. 59 (contenu de la variable ou valeur du littéral) = 40059).

Ce paramètre peut être indiqué comme adresse, variable localisée, variable non localisée ou littéral.

NO_REG

Nombre de registres à écrire dans le processeur esclave (1 ... 100). Ce paramètre peut être indiqué comme adresse, variable localisée, variable non localisée ou littéral.

STATUS

Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie.

Code d'erreur, voir :

- *Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137*
- *Codes d'erreur spécifiques SY/MAX, page 142*
- *Codes d'erreur Ethernet TCP/IP, page 144*

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

REG_WRIT

Pour ce paramètre un `ARRAY` de la taille de la transmission projetée doit être spécifié (\geq `NO_REG`). Le nom de ce tableau est transmis comme paramètre. Si le tableau est défini sur une taille trop réduite, la quantité de données transmise sera limitée par la place proposée dans le tableau.

Ce paramètre doit être indiqué comme variable localisée.

DATA_EXCH : échange de données entre applications

9

Objet de ce chapitre

Ce chapitre décrit la fonction DATA_EXCH.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	70
Ecran de saisie assistée	75
Exemple d'utilisation d'un réseau Fipway	77

Description

Description de la fonction

La fonction DATA_EXCH permet d'effectuer des transferts de données entre équipements avec des automates M340 et Premium :

- émission de données,
- réception de données,
- émission suivi d'une réception de données.

NOTE :

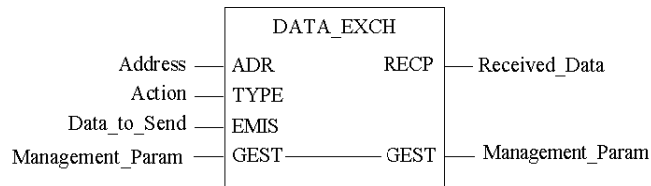
le type de requête susceptible d'être envoyée dépend du type d'automate :

- Sur les automates Modicom M340, cette fonction peut être utilisée pour envoyer des requêtes Modbus à un autre équipement.
- Sur les automates Premium, cette fonction peut être utilisée pour envoyer des requêtes UNI-TE ou Modbus à un autre équipement. A cette fin, il est nécessaire de s'assurer que les données envoyées constituent la totalité de la trame requise par le protocole qui est utilisé.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

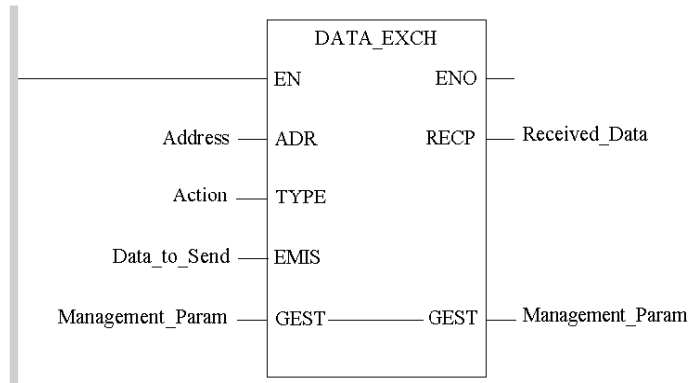
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

DATA_EXCH Action, Data_to_Send, Management_Param,
Received_Data

Représentation en ST

Représentation :

DATA_EXCH(Address, Action, Data_to_Send, Management_Param,
Received_Data);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètres	Type	Commentaire
Adresse	ARRAY [0.. 5] OF INT pour Premium ARRAY [0.. 7] OF INT pour Modicon M340	Adresse de l'entité destinataire de l'échange. Si le paramètre <i>Action</i> est de type émission/réception, les adresses en diffusion sont interdites. Pour les automates Premium : <ul style="list-style-type: none"> ● la fonction ADDR doit être utilisée. Pour les automates Modicon M340 : <ul style="list-style-type: none"> ● la fonction ADDM doit être utilisée. ● l'entité de destination de l'échange est un tableau de huit mots : %MWx:8. %MWx:8 - Initialisé par le bloc de conversion ADDM. %MW0:8:=ADDM('0.3.0.0')
Action	INT	Type d'action à réaliser. Pour les automates Premium, les valeurs possibles sont : <ul style="list-style-type: none"> ● 1: émission suivie d'une mise en attente pour réception (cette action n'est pas possible en Uni-Telway esclave). ● 2: émission simple. ● 3: mise en réception. Pour les automates Modicon M340, les valeurs possibles sont : <ul style="list-style-type: none"> ● 1: émission suivie d'une mise en attente.
Data_to_Send	ARRAY [n... m] OF INT	Tableau d'entiers qui sera émis à l'équipement destinataire de la requête. Attention : il doit avoir une longueur minimale de 1 élément même s'il n'y a pas de données à envoyer (code 3 pour l'action par exemple). Remarque : la longueur des données à émettre (en nombre d'octets) doit impérativement être affectée au quatrième mot de la table de gestion avant le lancement de la fonction pour que celle-ci s'exécute correctement.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0... 3] OF INT	Table de gestion des échanges (<i>voir page 33</i>). Sur les automates Modicon M340, un bit d'annulation est disponible dans le mot de rang 1 de la table de gestion des échanges. Le mot de rang 1 se compose de deux octets : <ul style="list-style-type: none"> ● Octet de poids fort : numéro d'échange. ● Octet de poids faible : bit d'activité (rang 0) et bit d'annulation (rang 1). La fonction élémentaire (EF) DATA_EXCH peut être annulée par la fonction élémentaire CANCEL ou en réglant le bit d'annulation de la table de gestion sur 1 (<i>voir Modicon M340 avec Unity Pro, Liaison série, Manuel de l'utilisateur</i>).

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Received_Data	ARRAY [n... m] OF INT	Tableau d'entiers qui contient les données reçues lors d'une mise en réception ou d'une action émission/réception. Attention : même si l'action est une simple émission, le tableau Received_Data doit exister et avoir une taille minimale de 1. Remarque : la taille des données reçues (en nombre d'octets) est écrite automatiquement par le système dans le quatrième mot de la table de gestion (<i>voir page 37</i>).

Services Modicon M340

Sur les automates Modicon M340, l'utilisateur peut coder un protocole privé et libre pour envoyer des requêtes Modbus. Cela offre la possibilité aux supports Modbus d'envoyer ou de recevoir une chaîne libre de type Byte.

La fonction élémentaire (EF) DATA_EXCH peut être utilisée sur tous les ports de communication à l'exception du port CANopen. Les limites de taille SendBuffer sont fournies par les caractéristiques du port de destination.

Le tableau ci-après fournit la longueur pour tous les ports de communication.

Port	Longueur
Ethernet (NOE ou port intégré)	1 Ko
Modbus	256 octets

La longueur du message à envoyer ou à recevoir est indiquée dans le quatrième mot de la table de gestion (*voir page 37*).

Le bloc de conversion ADDM permet d'indiquer la destination de la requête.

Voir la fonction ADDM.

NOTE : le mot clé TCP.MBS doit être indiqué lorsque la fonction élémentaire DATA_EXCH est utilisée pour les commandes Modbus par Ethernet.

Les autres entités Ethernet ne sont pas prises en charge. La fonction élémentaire DATA_EXCH ne gère pas directement les connexions par l'intermédiaire du protocole TCP.

Pour le protocole utilisateur ouvert sur Modbus, la syntaxe suivante est acceptée :

- **rack.module** - serveur de module
- **rack.module.voie.dispositif.MBS** - protocole Modbus
- **rack.module.voie.dispositif** - protocole utilisateur

Exemple de service Modicon M340

Objectif : écrire dans un seul registre %MW100, longueur := 5

(* REQUEST WRITE SINGLE REGISTER %MW100 Length := 5 *)

(* Data_to_send = encodage de la requête Modbus *)

(* Octet 1 = adresse de registre (poids fort) = 0 ; Octet 0 = code fonction = 06 *)

Data_to_Send[0] := 6;

(* Octet 3 = Valeur de registre (poids fort) ; Octet 2 = Adresse de registre (poids faible) = 100 *)

Data_to_Send[1] := (RegisterValue & 16#FF00) + 100;

(* Octet 5 = inutilisé ; Octet 4 = valeur de registre (poids faible) *)

Data_to_Send[2] := RegisterValue & 16#FF;

IF ((Management_Param[ACTIVITY] & 1) = 0) THEN

Management_Param[LENGTH] := 5; (* LENGTH RQ WRITE *)

DATA_EXCH (ADDM('0.0.0.1'), 1, Data_To_Send, Management_Param, Received_Data);

END_IF;

NOTE : le bus Modbus est gros-boutiste, alors que les mots P-UNIT sont petit-boutistes. Pour certaines requêtes, il est nécessaire d'effectuer une conversion.

Il est possible d'utiliser l'instruction ROL suivante :

Value_read := ROL(Received_Data[1], 8); (* Conversion de gros- en petit-boutiste *)

Ecran de saisie assistée

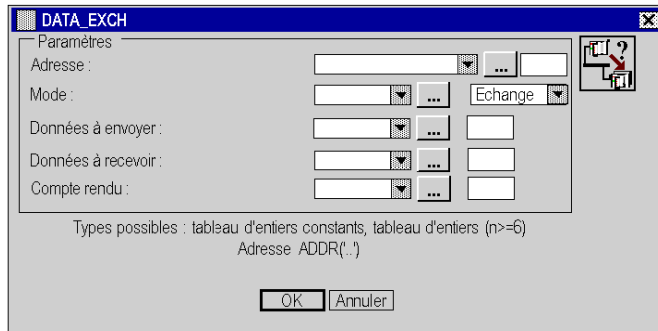
Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que l'écran de saisie assistée n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptées dans les différents champs de l'écran.

Illustration

La capture suivante présente un écran de saisie assistée de la fonction :



Adresse

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING) ,
- ARRAY [0..5] OF INT.

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

Mode

Choix possibles :

- 1 : échange,
- 2 : émission,
- 3 : réception.

NOTE : si vous utilisez le champ de saisie à la place du menu, vous pouvez saisir une variable de type INT, affectée ou non.

Données à envoyer

La variable en émission est un tableau d'entiers. Ce tableau peut être affecté ou non.

Données à recevoir

La zone de réception est un tableau d'entiers pouvant être affecté ou non. La taille de ce tableau dépend du nombre d'objets à recevoir.

Compte rendu

Le compte rendu est un tableau de 4 entiers pouvant être affecté ou non.

NOTE : veillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

Exemple d'utilisation d'un réseau Fipway

Présentation

Supposons une communication entre deux stations d'automate sur un réseau Fipway. La station 1 doit envoyer des données (20 octets situés dans %MW70:10) à la station 2 qui les reçoit dans une table située dans %MW80:10. Chaque automate dispose d'une table de gestion commençant à l'adresse %MW90.

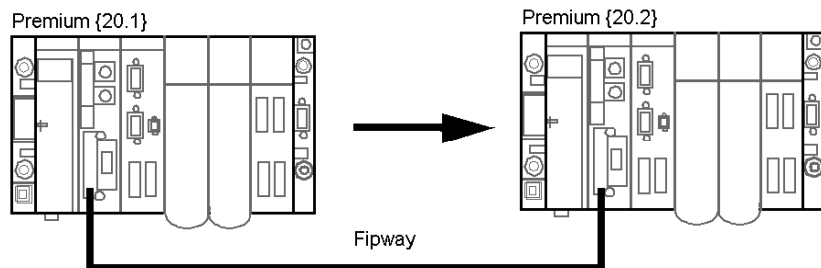
Pour la station 1, le mot %MW93 doit contenir la longueur des données à envoyer ou 20 (10 mots à envoyer).

Pour la station 2, le mot %MW93 est défini sur 0 avant l'échange afin de connaître le nombre de caractères reçus à la fin de l'échange.

La fonction DATA_EXCH requiert la programmation des deux automates, l'un pour l'envoi et l'autre pour la réception.

Illustration

Les deux stations sont connectées via un réseau Fipway :



Emission programmée dans la station 1

Programmation en ST :

```
IF RE(%I0.3.2) AND NOT %MW90.0 THEN
  (* initialisation des données à envoyer *)
  %MW93:= 20;
  (* fonction de communication *)
  DATA_EXCH(ADDR(' {20.2}APP' ), 2, %MW70:10, %MW90:4, %MW80:1);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.2}APP')	<ul style="list-style-type: none"> ● 20 : réseau ● 2 : station ● APP : application de la station 2
2	Type de communication : émission
%MW70:10	Données à envoyer
%MW90:4	Table de gestion
%MW80:1	Zone de réception des réponses. Dans le cas présent, il n'existe aucune donnée à recevoir mais un mot doit cependant être réservé (obligatoire).

Réception programmée dans la station 2

```

IF RE(%I0.3.4) AND NOT %MW90.0 THEN
  (* initialisation des données à recevoir *)
  %MW93:= 0;
  (* fonction de communication *)
  DATA_EXCH(ADDR( '{20.1}APP' ), 3, %MW70:1, %MW90:4, %MW80:10);
END_IF;

```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.1}APP')	<ul style="list-style-type: none"> ● 20 : réseau ● 1 : station ● APP : application de la station 1
3	Type de communication : réception
%MW70:1	Données à envoyer. Dans le cas présent, il n'existe aucune donnée à envoyer mais au moins un mot doit être réservé.
%MW90:4	Table de gestion
%MW80:10	Zone de réception : 10 mots sont fournis à partir de la station 1.

INPUT_BYTE : réception de chaînes de caractères

10

Description

Description de la fonction

La fonction `INPUT_BYTE` est utilisée pour envoyer une requête de lecture de tableau d'octets vers un module de communication en mode caractère. Le message reçu est enregistré dans un tableau d'octets.

NOTE : En général, `INPUT_BYTE` offre une fonctionnalité similaire à `INPUT_CHAR`, mais elle permet de transmettre un tableau d'octets à la place d'une chaîne en tant que paramètre de sortie. Pour cette raison, il est possible de lire une valeur d'octet 0 (NULL) dans un flux d'octets à partir d'un port série.

Pour les automates Premium, cette fonction permet de recevoir jusqu'à 4 Ko (120 octets au niveau du port terminal).

Pour les automates Modicon M340, cette fonction permet de recevoir jusqu'à 1 024 octets.

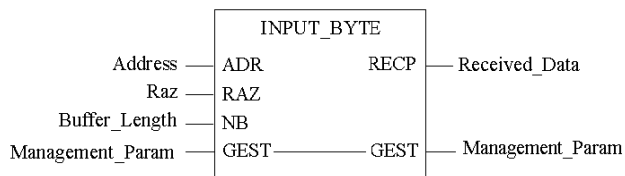
Deux possibilités s'offrent à l'utilisateur :

- lecture d'un nombre d'octets : aucune condition ne doit être définie ;
- lecture d'un message : une condition d'arrêt doit être définie dans l'écran de configuration.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

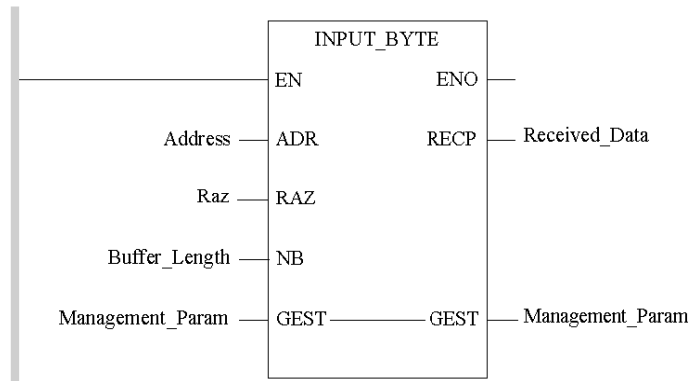
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

Adresse LD

INPUT_BYTE Reset, Buffer_Length, Management_Param,
Received_Data

Représentation en ST

Représentation :

INPUT_BYTE (Address, Reset, Buffer_Length, Management_Param,
Received_Data)

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
ADR	ARRAY [0.. 5] OF INT pour les automates Premium ARRAY [0.. 7] OF INT pour les automates Modicon M340	<p>Pour les automates Premium :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDR. ● Seules les adresses se terminant par SYS sont valides (par exemple, port terminal du processeur 0.0.0.SYS). <p>Pour les automates Modicon M340 :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDM. ● La syntaxe de l'adresse est de type ADDM ('r.m.c.node'). Le champ Node est facultatif. Il peut être de type SYS ou vide (par exemple ADDM('0.0.0.SYS') est égal à ADDM('0.0.0').
RAZ	INT	<p>Réinitialisation. Ce paramètre permet de réinitialiser la mémoire de réception du coupleur.</p> <ul style="list-style-type: none"> ● valeur = 0 : aucune réinitialisation de la mémoire ; ● valeur = 1 : réinitialisation de la mémoire. <p>Remarque : La valeur doit être égale à 1 pour les communications au niveau du port terminal des automates Premium.</p> <p>Remarque : Sur les automates Modicon M340, l'EF INPUT_CHAR peut être programmée avec ou sans ce paramètre.</p>
NB	INT	<p>Longueur du tampon ou nombre d'octets à recevoir.</p> <ul style="list-style-type: none"> ● valeur = 0 : le message est lu dès qu'il est disponible sur la voie. Une condition d'arrêt doit être définie dans l'écran de configuration ; ● valeur supérieure à 0 : spécifie le nombre d'octets à lire. <p>Remarque : 0 est la seule valeur autorisée pour les communications au niveau du port terminal des automates Premium. Le caractère de fin de message par défaut est un retour chariot (CR).</p>

Le tableau suivant décrit le paramètre d'entrée/sortie :

Paramètre	Type	Commentaire
GEST	ARRAY [0.. 3] OF INT	<p>Table de gestion des échanges (<i>voir page 33</i>).</p> <p>Un bit d'annulation est disponible seulement sur les automates Modicon M340.</p> <p>Le bit d'annulation est situé au mot de rang 1 dans la table de gestion des échanges et comprend 2 octets :</p> <ul style="list-style-type: none"> ● Octet de poids fort : numéro d'échange, ● Octet de poids faible : bit d'activité (rang 0) et bit d'annulation (rang 1). <p>Le bloc fonction <code>INPUT_CHAR</code> peut être annulé par le bloc fonction <code>CANCEL</code> ou en définissant sur 1 le bit d'annulation de la table de gestion. Reportez-vous aussi à la section Annulation d'un échange (<i>voir Modicon M340 avec Unity Pro, Liaison série, Manuel de l'utilisateur</i>).</p>

Le tableau suivant décrit le paramètre de sortie :

Paramètre	Type	Commentaire
RECP	ARRAY OF BYTE	Octets reçus. Le résultat est alors enregistré dans un tableau <code>BYTE</code> .

Règles de programmation

Si plusieurs fonctions `INPUT_BYTE` sont lancées simultanément, le paramètre `Reset` doit être défini sur 0 (mémoire de réception du module non réinitialisée).

Une requête peut être envoyée pour réinitialiser la mémoire du module pour le message suivant afin de ne pas recevoir d'anciennes données.

Si le paramètre `Reset` est défini sur 1, la fonction `INPUT_BYTE` doit être lancée avant d'envoyer des données.

Sur les automates Premium, plusieurs cycles d'automate sont nécessaires pour recevoir une chaîne d'octets de plus de 240 octets (la chaîne est fragmentée). Il est donc important de s'assurer que les données de gestion n'ont pas été modifiées au cours du traitement de la fonction. Le système reçoit la chaîne de manière cohérente sur plusieurs fragments.

Sur les automates Modicon M340, un cycle d'automate est nécessaire pour recevoir une chaîne d'octets de 1 024 octets maximum. Il est important de s'assurer que les données de gestion n'ont pas été modifiées au cours du traitement de la fonction.

Le port série de l'automate Modicon M340 est en full duplex. Par conséquent, une fonction `PRINT_CHAR` peut être envoyée même si une fonction `INPUT_CHAR` est en attente d'envoi.

INPUT_CHAR : réception de chaînes de caractères

11

Objet de ce chapitre

Ce chapitre décrit la fonction `INPUT_CHAR`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	84
Ecran de saisie assistée	89
Exemple de lecture de chaînes de caractères via un réseau Fipway	91
Exemple de lecture de chaînes de caractères via une liaison série de processeurs Modicon M340	93

Description

Description de la fonction

La fonction `INPUT_CHAR` est utilisée pour envoyer une requête de lecture de chaîne de caractères vers un module de communication en mode caractère (attente de réception des chaînes de caractères). Le message reçu est enregistré dans une chaîne de caractères.

Pour les automates Premium, cette fonction permet de recevoir jusqu'à 4 Ko (120 octets au niveau du port terminal).

Pour les automates Modicon M340, cette fonction permet de recevoir jusqu'à 1 024 octets.

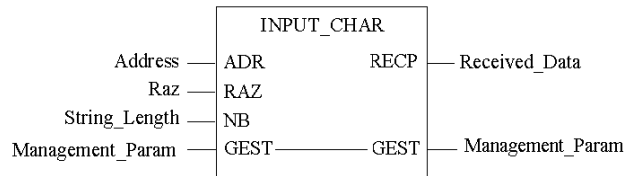
Deux possibilités s'offrent à l'utilisateur :

- lecture d'un nombre de caractères : aucune condition ne doit être définie.
- lecture d'un message : une condition d'arrêt doit être définie dans l'écran de configuration

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

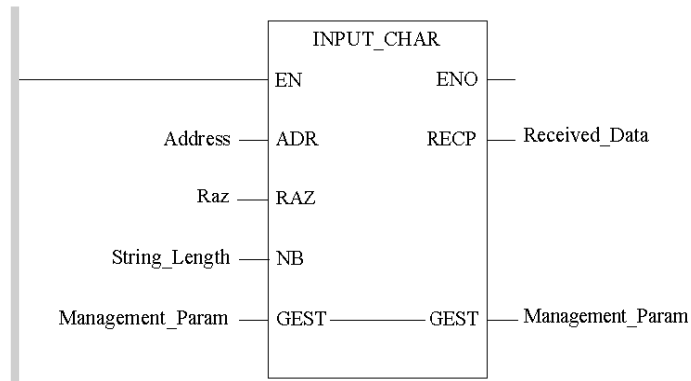
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

Adresse LD

```
INPUT_CHAR Reset, String_Length, Management_Param,
Received_Data
```

Représentation en ST

Représentation :

```
INPUT_CHAR(Address, Reset, String_Length, Management_Param,
Received_Data);
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
Adresse	ARRAY [0... 5] OF INT pour les automates Premium ARRAY [0... 7] OF INT pour les automates Modicon M340	<p>Les instructions suivantes s'appliquent uniquement aux automates Premium :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDR. ● Seules les adresses se terminant par SYS sont acceptées (exemple : port terminal du processeur 0.0.0.SYS). <p>Les instructions suivantes s'appliquent uniquement aux automates Modicon M340 :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDM. ● La syntaxe de l'adresse est de type ADDM ('r.m.c.node'). Le champ Node est facultatif. Il peut être de type SYS ou vide (par exemple ADDM('0.0.0.SYS') est égal à ADDM('0.0.0')).
RAZ	INT	<p>RAZ. Ce paramètre permet de réinitialiser la mémoire de réception du coupleur.</p> <ul style="list-style-type: none"> ● valeur = 0 : aucune réinitialisation de la mémoire ● valeur = 1 : réinitialisation de la mémoire <p>REMARQUE : la valeur doit être égale à 1 pour les communications au niveau du port terminal des automates Premium.</p> <p>REMARQUE : sur les automates Modicon M340, la fonction élémentaire INPUT_CHAR peut être programmée avec ou sans ce paramètre.</p>
String_Length	INT	<p>Longueur de la chaîne de caractères ou nombre de caractères à recevoir.</p> <ul style="list-style-type: none"> ● Valeur = 0 : le message est lu dès qu'il est disponible sur la voie. Une condition d'arrêt doit être définie dans l'écran de configuration. ● Valeur supérieure à 0 : définit le nombre de caractères à lire. <p>REMARQUE : 0 est la seule valeur autorisée pour les communications au niveau du port terminal des automates Premium. Le caractère de fin de message par défaut est un retour chariot (CR).</p>

Le tableau suivant décrit les paramètres d'entrée/de sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0... 3] OF INT	Table de gestion des échanges (<i>voir page 33</i>). Sur les automates Modicon M340 uniquement, un nouveau bit d'annulation est disponible au mot de rang 1 dans la table de gestion des échanges. Ce bit d'annulation est situé au niveau du mot de rang 1 qui se compose de deux octets : <ul style="list-style-type: none"> ● octet de poids fort : numéro d'échange ● octet de poids faible : bit d'activité (rang 0) et bit d'annulation (rang 1) L'EF INPUT_CHAR peut être annulée par l'EF CANCEL ou en définissant sur 1 le bit d'annulation de la table de gestion (<i>voir Modicon M340 avec Unity Pro, Liaison série, Manuel de l'utilisateur</i>).

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Received_Data	STRING	Chaîne reçue. Cette chaîne est ensuite stockée dans une chaîne de caractères.

Règles de programmation

Lorsque plusieurs fonctions INPUT_CHAR sont lancées simultanément, le paramètre `Reset` doit être défini sur 0 (mémoire de réception du module non réinitialisée).

Une requête peut être envoyée afin de réinitialiser la mémoire du module pour le message suivant et ce, pour éviter de recevoir d'anciennes données.

Lorsque le paramètre `Reset` est défini sur 1, la fonction INPUT_CHAR doit être lancée avant d'envoyer des données.

Sur les automates Premium, plusieurs cycles d'automate sont nécessaires pour recevoir une chaîne de caractères de plus de 240 octets (la chaîne est fragmentée). Il est donc important de s'assurer que les données de gestion n'ont pas été modifiées au cours du traitement de la fonction. Le système reçoit la chaîne de manière cohérente sur plusieurs fragments.

Sur les automates Modicon M340, un seul cycle d'automate est nécessaire pour recevoir une chaîne de caractères de 1 024 octets maximum. Il est important de s'assurer que les données de gestion n'ont pas été modifiées au cours du traitement de la fonction.

Le port série de l'automate Modicon M340 est en full duplex. Par conséquent, une fonction `PRINT_CHAR` peut être envoyée même si une fonction `INPUT_CHAR` est en attente d'envoi.

Il est possible de lancer la fonction `INPUT_CHAR` avant de finaliser les caractères sur l'automate.

Si les caractères finaux sont utilisés et que nombre d'entre eux se trouvent dans le buffer mais que celui-ci n'a pas été réinitialisé, chaque fonction `INPUT_CHAR` reçoit la chaîne de début du buffer jusqu'à ce qu'elle atteigne le premier caractère final. Le buffer est alors supprimé des caractères lus.

La lecture d'un nombre de caractères fonctionne de la même manière.

Si les caractères finaux sont configurés, il est possible d'utiliser la fonction de nombre de caractères.

NOTE : gestion RTS/CTS : dans une communication en mode caractère, lors de l'utilisation de la fonction élémentaire `INPUT_CHAR`, les caractères reçus sur la liaison série sont stockés dans un buffer en anneau. La taille de ce buffer est de 1 024. Lorsque le buffer est plein, les autres caractères sont perdus. Pour empêcher cette perte de caractères, il est possible de sélectionner la gestion RTS/CTS. Dans ce cas, lorsque le buffer en anneau est presque plein, l'UC réinitialise son signal RTS pour interrompre l'envoi des caractères.

AVERTISSEMENT

PERTES DE DONNEES

Lors de l'utilisation de la gestion RTS/CTS, le nombre maximum de caractères dans une chaîne autorisés pour l'émission par l'équipement distant s'élève à 1 000 (caractère de fin inclus). Si une chaîne de plus de 1 000 caractères est émise (par exemple 1 013), le RTS ne permettra de recevoir que les 1 000 premiers caractères, et la fonction `INPUT_CHAR` attendra le caractère de fin indéfiniment (si aucun délai d'expiration n'a été défini pour cette fonction).

Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

NOTE : si vous souhaitez envoyer une chaîne de plus de 1 000 caractères (1 013 par exemple), vous devez envoyer deux chaînes, une envoyant les 1 000 premiers caractères et l'autre les 13 caractères restants par exemple.

Ecran de saisie assistée

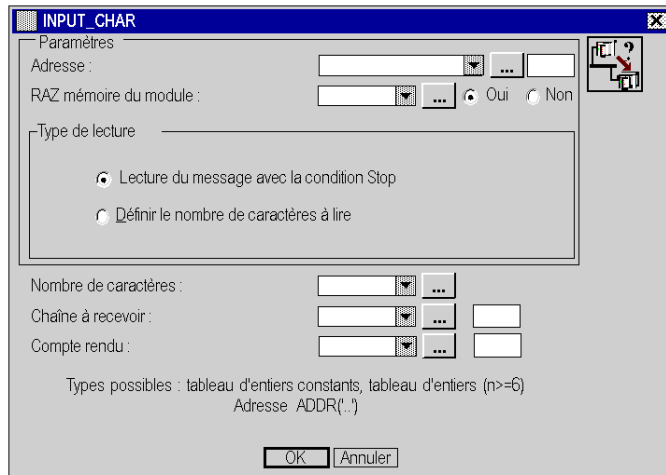
Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que l'écran de saisie assistée n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptées dans les différents champs de l'écran.

Illustration

La capture suivante est un exemple d'écran de saisie assistée de la fonction :



Adresse

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING),
- ARRAY [0..5] OF INT.

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

RAZ mémoire du module

Choix possible de type INT :

- Oui
- Non.

NOTE : le fait de sélectionner les boutons Oui/Non affiche immédiatement la valeur 1 ou 0.

Type de lecture

Les deux boutons radio permettent de sélectionner le mode de fonctionnement. La sélection doit être effectuée en fonction de la configuration de la voie utilisée.

Nombre de caractères

Vous pouvez entrer un entier sous forme de variable ou une valeur immédiate.

Chaîne à recevoir

La zone de réception est une variable de type `STRING`. La taille de cette variable dépend du nombre de caractères à recevoir. La variable doit être déclarée avant d'être utilisée dans cet écran.

Compte rendu

Le compte rendu est un tableau de 4 entiers.

NOTE : veillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

Exemple de lecture de chaînes de caractères via un réseau Fipway

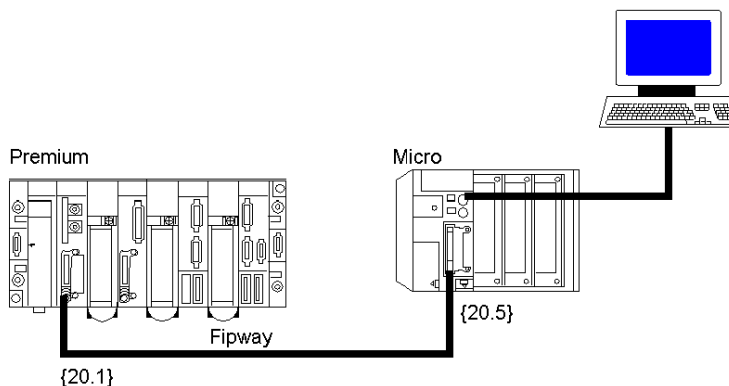
Présentation

Sur une station avec l'adresse 1 sur le réseau Fipway 20, nous devons lire une chaîne de caractères envoyée par un terminal vidéo (écran et clavier) connecté au port TER de l'automate avec l'adresse réseau 20, station 5.

Cette chaîne de caractères est stockée dans la variable `Str_1`, la table de gestion de la fonction de communication étant `%MW110:4`.

Illustration

Les deux stations sont connectées via un réseau Fipway.



Programmation

Programmation en ST :

```
IF RE(%I0.3.6) AND NOT %MW110.0 THEN
    INPUT_CHAR(ADDR('{20.5}0.0.SYS'), 1, 0, %MW110:4, Str_1);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.5}0.0.0.SYS')	<ul style="list-style-type: none">● {20.5} : réseau 20, station 5● 0 : rack● 0 : module● 0 : voie 0● SYS : adresse système (port terminal)
1	Réinitialisation
0	Lecture de la totalité de la chaîne de caractères
%MW110:4	Table de gestion
Str_1	Variable de type STRING devant recevoir le message

Exemple de lecture de chaînes de caractères via une liaison série de processeurs Modicon M340

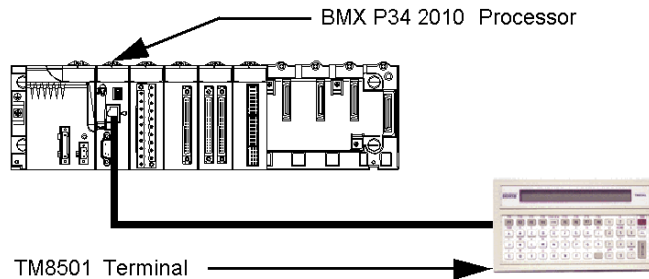
Présentation

Nous voulons lire une chaîne de caractères envoyée par un terminal de saisie/d'affichage de données compact raccordé au port série d'un processeur Modicon M340.

Cette chaîne de caractères est stockée dans la variable `Str`, la table de gestion de la fonction de communication étant `gestion`.

Illustration

Un automate Modicon M340 est relié à un terminal de saisie/d'affichage de données TM8501 :



Programmation

Programmation en ST :

```
IF (%M15) THEN
    INPUT_CHAR(ADDM('0.0.0'), 1, 0, gestion, Str);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDM('0.0.0')	<ul style="list-style-type: none">● 0 : rack● 0 : module● 0 : voie 0● SYS : adresse système (optionnelle sur les automates Modicon M340)
1	RAZ
0	Lecture de la totalité de la chaîne de caractères
gestion	Table de gestion
Str	Variable de type STRING devant recevoir le message

Introduction

Ce chapitre décrit le bloc MBP_MSTR.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description du bloc	97
Codes fonction des opérations	100
Structures du bloc de commande de réseau	101
Lecture de données	104
Ecriture de données	106
Extraction de statistiques locales	108
Suppression de statistiques locales	110
Ecriture de données globales	111
Lecture de données globales	112
Lire statistiques distantes	113
Effacer statistiques distantes	115
Validité de Peer Cop	116
Réinitialisation du module optionnel	117
Lecture de la CTE	118
Ecriture de la CTE	120
Envoi de message électronique	122
Lire/écrire données	124
Etat de validité des communications Peer Cop	126
Statistiques du réseau Modbus Plus	128
Statistiques de réseau Ethernet TCP/IP	134
Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP	137

Sujet	Page
Codes d'erreur spécifiques SY/MAX	142
Codes d'erreur Ethernet TCP/IP	144
Codes d'erreur CTE pour Ethernet SY/MAX et TCP/IP	148
Codes d'erreur du service de messagerie	149

Description du bloc

Description de la fonction

Vous pouvez sélectionner l'une des 14 opérations de communication réseau disponibles (*voir page 100*) à l'aide du bloc fonction MBP_MSTR.

Selon le protocole de communication que vous utilisez, vous pouvez avoir jusqu'à 16 blocs fonction MBP_MSTR actifs simultanément.

- Modbus Plus prend en charge 4 blocs simultanément.
- Ethernet TCP/IP prend en charge 16 blocs simultanément.

Les blocs fonction utilisent un chemin de transaction de données et plusieurs cycles leur sont nécessaires pour réaliser une opération.

EN et ENO peuvent être configurés en tant que paramètres supplémentaires.

NOTE : pour programmer un bloc fonction MBP_MSTR, vous devez bien connaître les procédures de routage de votre réseau. Les structures de chemin de routage Modbus Plus sont détaillées dans le *Guide de planification et d'installation du réseau Modbus Plus*. Si le routage Ethernet TCP/IP ou SY/MAX est implémenté, vous devez utiliser les routeurs IP Ethernet standard. Vous trouverez une description complète du routage TCP/IP dans le *Guide utilisateur de configuration du système TCP/IP Quantum avec Unity Pro*.

NOTE : dans les sections FBD et LD, ce bloc fonction est utilisable au niveau programme et avec des blocs fonction dérivés (DFB). En cas d'utilisation des DFB, les paramètres CONTROL et DATABUF doivent être directement associés aux broches E/S des DFB.

NOTE : pour qu'une communication TCP/IP entre un automate Quantum et un automate Momentum puisse avoir lieu, il faut qu'une seule tâche de lecture ou d'écriture soit réalisée au cours de chaque cycle. Si plusieurs tâches sont envoyées par cycle d'automate, la communication est interrompue sans générer de message d'erreur dans le registre d'état du bloc fonction.

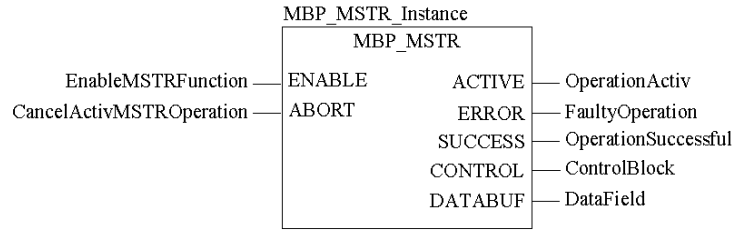
Exemple :

- Vous pouvez envoyer MBP_MSTR.Enable:=(HSBY_NOEPLCMSTR_ON) AND (%SW61.1) AND NOT (%SW61.0)
ou
- Vous pouvez aussi créer une variable booléenne, primary_state:=(%SW61.1) AND NOT (%SW61.0) et l'insérer pour l'exécution de la section

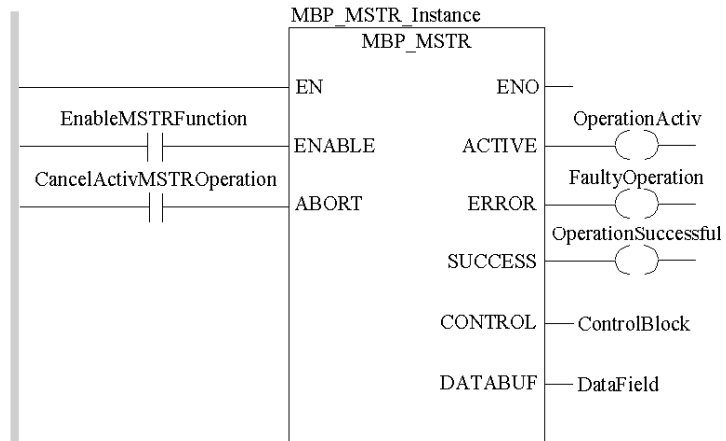
NOTE : pour empêcher l'ancienne UC redondante (dont l'état a été commuté pour une exécution locale) d'exécuter des fonctions de communication, vous devez ajouter une condition aux bits d'état afin de désactiver la fonction, à condition que l'UC soit hors ligne.

NOTE : il est possible d'utiliser plusieurs copies de ce bloc fonction dans le programme. En revanche, l'instanciation multiple de ces copies n'est pas possible.

Représentation dans FBD



Représentation en LD



Paramètres d'entrée

Paramètre	Type de données	Description
ENABLE	BOOL	Sur ON, l'opération spécifiée dans le premier élément du registre <code>COMMANDE</code> est activée.
ABORT	BOOL	Sur ON, l'opération active (<i>voir page 100</i>) est abandonnée.

Paramètres de sortie

Paramètre	Type de données	Description
ACTIVE	BOOL	ON lorsque l'opération est active.
ERROR	BOOL	ON lorsque l'abandon de l'opération a échoué.
SUCCESS	BOOL	ON lorsque l'opération s'est déroulée correctement.
CONTROL	WORD	Ce champ contient le bloc de commande. Le premier élément, COMMANDE[1], contient le numéro du code de l'opération à réaliser (<i>voir page 100</i>). Le contenu du registre de séquences est déterminé par l'opération. Le champ de données doit être déclaré en tant que variable affectée. La structure du bloc de commande varie selon le réseau utilisé (<i>voir page 101</i>).
DATABUF	WORD	Pour les opérations fournissant des données, par exemple une opération d'écriture, le champ de données est la source des données. Pour les opérations recevant des données, par exemple une opération de lecture, le champ de données est la destination des données. Avec les opérations de lecture et d'écriture de la CTE Ethernet, le contenu de la table d'extension de configuration Ethernet se trouve dans les champs de données. DATABUF doit être défini en tant que tableau d'au moins 10 éléments dans ce cas. Le champ de données doit être déclaré en tant que variable affectée.

Erreur d'exécution

Si une erreur se produit au cours d'une opération MBP_MSTR, un code d'erreur hexadécimal s'affiche dans le registre COMMANDE[2] du bloc de commande pendant un cycle.

Les codes d'erreur des fonctions varient selon les réseaux :

- Codes d'erreur Modbus Plus et Ethernet SY/MAX (*voir page 137*)
- Codes d'erreur spécifiques SY/MAX (*voir page 142*)
- Codes d'erreur Ethernet TCP/IP (*voir page 144*)
- Codes d'erreur CTE pour Ethernet SY/MAX et TCP/IP (*voir page 148*)
- Codes d'erreur d'envoi de message électronique (*voir page 149*)

NOTE : pour obtenir la liste de tous les codes et valeurs d'erreur du bloc, reportez-vous aux tableaux des codes d'erreur pour la bibliothèque de communication.

Codes fonction des opérations

Codes fonction MBP_MSTR valides

Il est possible de déclencher l'une des 14 opérations de communication réseau disponibles via le réseau à l'aide du bloc MBP_MSTR. Un code fonction est affecté à chaque opération. La disponibilité des opérations dépend du type de réseau et de module utilisés.

Code fonction	Opération	Modbus Plus	Ethernet TCP/IP	Ethernet SY/MAX
1	Ecriture de données	X	X	X
2	Lecture de données	X	X	X
3	Extraction de statistiques locales	X	X	-
4	Suppression de statistiques locales	X	X	-
7	Obtention de statistiques distantes	X	X	-
8	Suppression de statistiques distantes (voir page 115)	X	X	-
10	Réinitialisation du module optionnel	-	X	X
11	Lecture de la CTE (extension de configuration)	-	X	X
12	Ecriture de la CTE (extension de configuration)	-	X	X
13	Envoi de message électronique (voir page 122)	-	X	-
23	Lecture/écriture de données (voir page 124)	-	X	-

où :

- X signifie *Oui*
- - signifie *Non*

Structures du bloc de commande de réseau

Récapitulatif

La structure du bloc de commande MBP_MSTR varie selon le type de réseau que vous utilisez. Les structures pour Modbus Plus, Ethernet TCP/IP et Ethernet SyMax sont décrites ci-dessous.

Bloc de commande pour Modbus Plus

Registre	Sommaire
COMMANDE[1]	Indique une opération valide pour Modbus Plus.
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Indique la longueur, c'est-à-dire le nombre d'unités de données transférées (max. 100).
COMMANDE[4]	Indique des informations liées à l'opération MSTR.
COMMANDE[5]	<p>Registre de routage 1 : sert à indiquer un abonné cible pendant un transfert réseau (adresses du chemin de routage - 1 sur 5). Octet de poids fort : adresse de l'abonné source, soit l'emplacement pour le module NOM (Network Options Module) Modbus Plus. Si vous utilisez le port Modbus Plus de l'UC, cet octet doit être réglé sur 0 (pour tous les emplacements de l'UC). Octet de poids faible : adresse de l'abonné cible, soit une valeur qui représente une adresse directe ou de pont. S'il n'y a pas de pont, cette valeur contient l'adresse de l'abonné cible. S'il y a un pont, cette valeur contient son adresse. Si le NOM est inséré dans l'emplacement 7 du rack du module, l'octet de poids fort du registre de routage 1 se présente comme suit (valeur 0x0706) :</p> <div style="text-align: center;"> <pre> octet de poids fort octet de poids faible <-----> <-----> 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ </pre> </div> <p>Octet de poids fort Emplacements 1 à 16 Octet de poids faible Adresse cible (valeur binaire entre 1 et 64 (normale) ou entre 65 et 255 (étendue))</p>

Registre	Sommaire
COMMANDE[6]	Registre de routage 2, adresse de l'abonné cible (pont ou modules Modbus Plus supplémentaires). Si l'adressage du registre de routage précédent est terminé, la valeur est réglée sur 0.
COMMANDE[7]	Registre de routage 3, identique au 2.
COMMANDE[8]	Registre de routage 4, identique au registre de routage 2 (voir registre de routage 2)
COMMANDE[9]	Registre de routage 5, identique au registre de routage 2 (voir registre de routage 2)

Bloc de commande pour Ethernet TCP/IP

Registre	Sommaire
COMMANDE[1]	Indique une opération valide pour TCP/IP.
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Indique la longueur, c'est-à-dire le nombre d'unités de données transférées (max. 100).
COMMANDE[4]	Indique des informations liées à l'opération MSTR.
COMMANDE[5]	<p>Registre de routage : sert à indiquer un abonné cible pendant un transfert réseau.</p> <p>Octet de poids fort : adresse de l'abonné source, soit l'emplacement NOE pour le module NOE.</p> <p>Si vous utilisez un Ethernet intégré dans l'UC, cet octet doit être réglé sur 254 (FE hex) pour tous les emplacements de l'UC.</p> <p>Octet de poids faible : adresse de l'abonné cible, soit une valeur qui représente une adresse directe ou de pont. S'il n'y a pas de pont, la valeur de l'octet de poids faible est réglée sur 0. S'il y a un pont, cette valeur contient le MBP pour l'index de mappage sur Ethernet (MET).</p> <p>Si le NOM est inséré dans l'emplacement 7 du rack du module et que l'index de mappage sur Ethernet (MET) est 6, le registre de routage se présente comme suit (valeur 0x0706) :</p> <div style="text-align: center;"> </div> <p>Octet de poids fort Emplacements 1 à 16 Octet de poids faible Index de mappage MBP sur Ethernet Transporter (MET)</p>
COMMANDE[6]	Octet 4, MSB de l'adresse IP cible à 32 bits
COMMANDE[7]	Octet 3 de l'adresse IP cible à 32 bits

Registre	Sommaire
COMMANDE[8]	Octet 2 de l'adresse IP cible à 32 bits
COMMANDE[9]	Octet 1, LSB de l'adresse IP cible à 32 bits
COMMANDE[10]	Indique des informations liées à l'opération MSTR.
COMMANDE[11]	Indique des informations liées à l'opération MSTR.

NOTE : CONTROL[10] et CONTROL[11] sont utilisés lors de la configuration du bloc MBP_MSTR pour une opération Lecture/Ecriture (code de fonction 23).

Bloc de commande pour Ethernet SY/MAX

Registre	Sommaire
COMMANDE[1]	Indique une opération valide pour SY/MAX.
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Indique la longueur, c'est-à-dire le nombre de registres transférés (max. 100).
COMMANDE[4]	Indique des informations liées à l'opération MSTR.
COMMANDE[5]	<p>Registre de routage : sert à indiquer un abonné cible pendant un transfert réseau.</p> <p>Octet de poids fort : adresse de l'abonné source, soit l'emplacement pour le module NOE.</p> <p>Octet de poids faible : adresse de l'abonné cible, soit une valeur qui représente une adresse directe ou de pont. S'il n'y a pas de pont, la valeur de l'octet de poids faible est réglée sur 0. S'il y a un pont, cette valeur contient le MBP pour l'index de mappage sur Ethernet (MET).</p> <p>Si le NOM est inséré dans l'emplacement 7 du rack du module et que l'index de mappage sur Ethernet (MET) est 6, le registre de routage se présente comme suit (valeur 0x0706) :</p> <div style="text-align: center;"> <pre> octet de poids fort octet de poids faible <-----> <-----> 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 </pre> </div> <p>Octet de poids fort Emplacements 1 à 16</p> <p>Octet de poids faible index de mappage MBP sur Ethernet Transporter (MET).</p>
COMMANDE[6]	Numéro de station cible (ou réglé sur FF hex)
COMMANDE[7]	Terminaison (réglée sur FF hex)

Lecture de données

Description

L'opération de lecture transfère des données d'un équipement esclave source spécifique vers un équipement maître cible du réseau. Elle utilise un chemin de transaction maître et sa réalisation peut nécessiter plusieurs cycles. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération d'écriture, utilisez le code fonction 2 (*voir page 100*).

NOTE : n'essayez pas de programmer un MBP_MSTR pour lire sur sa propre adresse station. Par cette action, le bloc fonction génère une erreur dans le registre COMMANDE[2] du bloc de commande (*voir page 101*).

Vous pouvez effectuer une opération de lecture sur un registre esclave inexistant. L'esclave détecte l'état et l'enregistre. Cela peut durer plusieurs cycles.

Implémentation réseau

L'opération de lecture peut être réalisée sur des réseaux Modbus Plus, Ethernet TCP/IP et Ethernet SY/MAX.

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	2 = lecture de données
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre de registres à lire depuis l'esclave
COMMANDE[4]	Détermine le registre de départ %MW dans l'esclave à partir duquel les données sont lues, par exemple 1 = %MW1, 49 = %MW49.
COMMANDE[5] ...	Le registre de routage 1 sert à indiquer l'adresse (adresse du chemin de routage 1 sur 5) de l'abonné pendant un transfert réseau.
COMMANDE[9]	Le dernier octet du chemin de routage qui n'est pas réglé sur 0 est l'abonné cible.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	2 = lecture de données
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses à lire depuis l'esclave
COMMANDE[4]	Détermine le registre de départ %MW dans l'esclave à partir duquel les données sont lues, par exemple 1 = %MW1, 49 = %MW49.
COMMANDE[5]	Registre de routage : Octet de poids fort : emplacement du module de la carte réseau Octet de poids faible : index de mappage MBP sur Ethernet Transporter (MET)
COMMANDE[6] ... COMMANDE[9]	Chaque adresse contient 1 octet de l'adresse IP à 32 bits, où le MSB est dans COMMANDE[6] et le LSB dans COMMANDE[9].

Utilisation du bloc de commande pour Ethernet SY/MAX

Registre	Signification
COMMANDE[1]	2 = lecture de données
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses à lire depuis l'esclave
COMMANDE[4]	Détermine le registre de départ %MW dans l'esclave dans lequel les données sont écrites, par exemple 1 = %MW1, 49 = %MW49.
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau. Octet de poids faible : numéro de station cible
COMMANDE[6] ... COMMANDE[9]	Terminaison : FF hex

Ecriture de données

Description

L'opération d'écriture transfère des données d'un équipement source maître vers un équipement esclave cible spécifique du réseau. Elle utilise un chemin de transaction maître et sa réalisation peut nécessiter plusieurs cycles. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération d'écriture, utilisez le code fonction 1 (voir page 100).

NOTE : n'essayez pas de programmer un MBP_MSTR pour écrire dans sa propre adresse de station. Par cette action, le bloc fonction génère une erreur dans le registre COMMANDE[2] du bloc de commande (voir page 101).

Vous pouvez effectuer une opération d'écriture dans un registre esclave inexistant. L'esclave détecte l'état et l'enregistre. Cela peut durer plusieurs cycles.

Implémentation réseau

L'opération d'écriture peut être réalisée sur des réseaux Modbus Plus, Ethernet TCP/IP et Ethernet SY/MAX.

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	1= écriture de données
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses envoyées à l'esclave
COMMANDE[4]	Détermine le registre de départ %MW dans l'esclave dans lequel les données sont écrites, par exemple 1 = %MW1, 49 = %MW49
COMMANDE[5] ...	Le registre de routage 1 sert à indiquer l'adresse (adresse du chemin de routage 1 sur 5) de l'abonné pendant un transfert réseau.
COMMANDE[9]	Le dernier octet du chemin de routage qui n'est pas réglé sur 0 est l'abonné cible.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	1= écriture de données
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses envoyées à l'esclave
COMMANDE[4]	Détermine l'adresse de départ de COMMANDE[] de l'esclave dans lequel les données sont écrites.
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau Octet de poids faible : index de mappage MBP sur Ethernet Transporter (MET)
COMMANDE[6]	Chaque adresse contient 1 octet de l'adresse IP à 32 bits.
...	
COMMANDE[7]	

Utilisation du bloc de commande pour Ethernet SY/MAX

Registre	Signification
COMMANDE[1]	1= écriture de données
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses envoyées à l'esclave
COMMANDE[4]	Détermine le registre de départ %MW dans l'esclave dans lequel les données sont écrites, par exemple 1 = %MW1, 49 = %MW49
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau. Octet de poids faible : numéro de station cible
COMMANDE[6]	Terminaison : FF hex
...	
COMMANDE[9]	

Extraction de statistiques locales

Description

L'opération d'extraction de statistiques locales lit les données depuis l'abonné local en un cycle et ne nécessite pas de chemin de transaction maître. Pour programmer un bloc MBP_MSTR en vue d'extraire des statistiques locales, utilisez le code fonction 3 (*voir page 100*).

Implémentation réseau

Une opération d'extraction de statistiques locales peut s'effectuer sur les réseaux Modbus Plus et Ethernet TCP/IP (*voir page 134*).

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	3 = extraction de statistiques locales
COMMANDE[2]	Indique l'état d'erreur
COMMANDE[3]	Nombre d'adresses à lire depuis les statistiques locales (0 à 37) Remarque : la taille du tampon de données doit être égale ou supérieure à celle de cette entrée.
COMMANDE[4]	Première adresse à partir de laquelle la table des statistiques doit être lue (Reg1=0)
COMMANDE[5]	Le registre de routage 1 sert à indiquer l'adresse (adresse du chemin de routage 1 sur 5) de l'abonné pendant un transfert réseau. Le dernier octet du chemin de routage qui n'est pas réglé sur 0 est l'abonné cible.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	3 = extraction de statistiques locales
COMMANDE[2]	Indique l'état d'erreur
COMMANDE[3]	Nombre d'adresses à lire depuis les statistiques locales (0 à 37) Remarque : la taille du tampon de données doit être égale à celle de cette entrée.
COMMANDE[4]	Première adresse à partir de laquelle la table des statistiques doit être lue (Reg1=0)
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau
COMMANDE[6] ... COMMANDE[9]	Non utilisé

Suppression de statistiques locales

Description

L'opération de suppression de statistiques locales supprime les valeurs des mots 13 à 22 dans la table des statistiques de l'abonné local. Cette opération s'effectue en un cycle et ne nécessite pas de chemin de transaction maître. Pour programmer un bloc MBP_MSTR en vue de supprimer des statistiques locales, utilisez le code fonction 4 (voir page 100).

Implémentation réseau

L'opération de suppression de statistiques locales peut être réalisée sur les réseaux Modbus Plus et Ethernet TCP/IP (voir page 134).

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	4 = suppression de statistiques locales
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Réservé
COMMANDE[4]	Réservé
COMMANDE[5]	Le registre de routage 1 sert à indiquer l'adresse (adresse du chemin de routage 1 sur 5) de l'abonné pendant un transfert réseau. Le dernier octet du chemin de routage qui n'est pas réglé sur 0 est l'abonné cible.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	4 = suppression de statistiques locales
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Réservé
COMMANDE[4]	Réservé
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau
COMMANDE[6]	Réservé
...	
COMMANDE[9]	

Ecriture de données globales

Description

L'opération d'écriture de données globales transfère des données au processeur de communication de l'abonné actuel ; les données peuvent être transmises sur le réseau dès que l'abonné reçoit le jeton, puis être lues par tous les abonnés connectés au réseau local (*voir page 112*).

L'opération d'écriture de données globales s'effectue en un cycle et ne nécessite pas de chemin de transaction maître. Pour programmer un bloc MBP_MSTR en vue d'écrire des données globales, utilisez le code fonction 5 (*voir page 100*).

Implémentation réseau

L'opération d'écriture de données globales ne peut être réalisée que sur les réseaux Modbus Plus. Les opérations de lecture et d'écriture de données globales comprennent une fonctionnalité Modbus Plus appelée *Peer Cop*.

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	5 = écriture de données globales
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses à écrire à partir de la RAM d'état dans la mémoire de données globales (processeur de communication) (1 à 32)
COMMANDE[4]	Réservé
COMMANDE[5]	Si les données globales sont envoyées via un NOM, entrez l'emplacement de ce module dans l'octet de poids fort de ce registre.

Lecture de données globales

Description

L'opération de lecture de données globales lit des données à partir du processeur de communication d'un abonné du réseau qui dispose de données globales écrites (voir page 111). Cette opération ne nécessite pas de chemin de transaction maître.

L'opération de lecture de données globales peut durer plusieurs cycles si les données ne sont pas disponibles sur les abonnés appelés. Si les données globales sont disponibles, l'opération s'exécute en un cycle. Pour programmer un bloc MBP_MSTR en vue d'écrire des données globales, utilisez le code fonction 6 (voir page 100).

Implémentation réseau

L'opération de lecture de données globales ne peut être réalisée que sur des réseaux Modbus Plus. Les opérations de lecture et d'écriture de données globales comprennent une fonctionnalité Modbus Plus appelée *Peer Cop*.

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	6 = lecture de données globales
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre d'adresses à envoyer depuis la mémoire de données globales (processeur de communication) (1 à 32)
COMMANDE[4]	Affiche les adresses disponibles dans l'abonné scruté. Ce registre est automatiquement mis à jour.
COMMANDE[5]	L'octet de poids faible contient l'adresse de l'abonné dont les données globales doivent être lues. Sa valeur peut être comprise dans la plage 1 à 64. Si les données globales sont reçues via un NOM, entrez l'emplacement de ce module dans l'octet de poids fort de cette adresse.

Lire statistiques distantes

Description

L'opération d'obtention de statistiques distantes permet de lire des données à partir d'abonnés distants du réseau. A chaque requête, le processeur de communication distant fournit une table complète de statistiques, même si la requête ne fait pas référence à la totalité de la table. Il copie ensuite uniquement les mots que vous avez interrogés dans les adresses \$MW identifiées.

La réalisation d'une opération peut prendre plusieurs cycles et elle ne nécessite pas de chemin de transaction maître. Pour programmer un bloc MBP_MSTR en vue d'extraire des statistiques distantes, utilisez le bloc fonction 7 (*voir page 100*).

Implémentation réseau

L'opération d'extraction de statistiques distantes peut être réalisée sur des réseaux Modbus Plus et Ethernet TCP/IP.

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	7 = obtention de statistiques distantes
COMMANDE[2]	Indique l'état d'erreur
COMMANDE[3]	Nombre d'adresses à lire depuis le champ des données de statistiques (0 à 37). Remarque : la taille du tampon de données doit être égale ou supérieure à celle de cette entrée.
COMMANDE[4]	Première adresse à partir de laquelle les statistiques de l'abonné doivent être lues. Il est impossible de dépasser le nombre de registres de statistiques disponibles.
COMMANDE[5] ... COMMANDE[9]	Adresse de routage 1 à 5 de l'abonné. Le dernier octet du chemin de routage qui n'est pas réglé sur 0 est l'abonné cible.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	7 = obtention de statistiques distantes
COMMANDE[2]	Indique l'état d'erreur
COMMANDE[3]	Nombre d'adresses à lire depuis le champ des données de statistiques (0 à 37). Remarque : la taille du tampon de données doit être égale ou supérieure à celle de cette entrée.
COMMANDE[4]	Première adresse à partir de laquelle les statistiques de l'abonné doivent être lues. Il est impossible de dépasser le nombre de registres de statistiques disponibles.
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau
COMMANDE[6] ... COMMANDE[9]	Chaque adresse contient 1 octet de l'adresse IP à 32 bits, où la valeur de COMMANDE[6] est le MSB et celle de COMMANDE[9] est le LSB.

Effacer statistiques distantes

Description

L'opération de suppression de statistiques distantes supprime les valeurs de l'abonné distant des mots 13 à 22 dans la table des statistiques de l'abonné local. Elle utilise un chemin de transaction maître et sa réalisation peut nécessiter plusieurs cycles. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération de suppression de statistiques distantes, utilisez le code fonction 8 (voir page 100).

Implémentation réseau

L'opération de suppression de statistiques distantes peut être réalisée sur les réseaux Modbus Plus et Ethernet TCP/IP (voir page 134).

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	8 = suppression de statistiques distantes
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Réservé
COMMANDE[4]	Réservé
COMMANDE[5] ...	Le registre de routage 1 sert à indiquer l'adresse (adresse du chemin de routage 1 sur 5) de l'abonné cible pendant un transfert réseau.
COMMANDE[9]	Le dernier octet du chemin de routage qui n'est pas réglé sur 0 est l'abonné cible.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	8 = suppression de statistiques distantes
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Réservé
COMMANDE[4]	Réservé
COMMANDE[5]	Registre de routage Octet de poids fort : emplacement du module de la carte réseau
COMMANDE[6] ...	Chaque adresse contient un octet de l'adresse IP à 32 bits, où le MSB est dans COMMANDE[6] et le LSB dans COMMANDE[9].
COMMANDE[9]	

Validité de Peer Cop

Description

L'opération de validité de Peer Cop lit les données sélectionnées dans la table de validité des communications Peer Cop et les télécharge à l'adresse %MW spécifiée dans la RAM d'état. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération de suppression de statistiques distantes, utilisez le code fonction 9 (voir page 100).

NOTE : la validité Peer Cop est opérationnelle uniquement si un scrutateur d'E/S à diffusion des E/S est configuré.

La table de validité des communications Peer Cop a une longueur de 12 mots ; MBP_MSTR indexe ces mots avec les numéros 0 à 11.

Implémentation réseau

L'opération de validité Peer Cop ne peut être réalisée que sur les réseaux Modbus Plus.

Utilisation du bloc de commande pour Modbus Plus

Registre	Signification
COMMANDE[1]	9 = validité de Peer Cop
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre de mots requis par la table Peer Cop (1 à 12)
COMMANDE[4]	Premier mot à lire à partir de la table Peer Cop, où 0 est le premier mot et 11 le dernier mot
COMMANDE[5]	Adresse de routage 1 S'il s'agit du second de deux abonnés locaux, réglez la valeur de l'octet de poids fort sur 1.

Réinitialisation du module optionnel

Description

Suite à une opération de réinitialisation du module optionnel, le module de communication Ethernet Quantum NOE ou le port Ethernet sur un module d'UC 140 CPU 65150/60 entre dans un cycle de réinitialisation de son environnement de travail. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération de réinitialisation du module optionnel, utilisez le code fonction 10 (voir page 100).

Implémentation réseau

L'opération de réinitialisation du module optionnel peut être réalisée sur des réseaux Ethernet TCP/IP (voir page 134) et Ethernet SY/MAX.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	10 = réinitialisation du module optionnel
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Aucune signification
COMMANDE[4]	Aucune signification
COMMANDE[5]	Registre de routage La valeur apparaissant dans l'octet de poids fort dans la zone entre 1 et 16 indique l'emplacement du module NOE dans l'embase Quantum.
COMMANDE[6]	Aucune signification
...	
COMMANDE[9]	

Utilisation du bloc de commande pour Ethernet SY/MAX (COMMANDE)

Registre	Signification
COMMANDE[1]	10 = réinitialisation du module optionnel
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Aucune signification
COMMANDE[4]	Aucune signification
COMMANDE[5]	Registre de routage MSB : emplacement du module de la carte réseau
COMMANDE[6]	Aucune signification
...	
COMMANDE[9]	

Lecture de la CTE

Description

L'opération de lecture de la CTE lit un nombre spécifique d'octets de la table d'extension de configuration Ethernet dans le tampon indiqué de la mémoire de l'automate. Les octets à lire commencent par un décalage d'octet au début de la table CTE. Le contenu de la table CTE s'affiche dans le paramètre de sortie `DATABUF` (paramètre de sortie (*voir page 98*)). Pour programmer un bloc `MBP_MSTR` en vue d'effectuer une opération de suppression de statistiques distantes, utilisez le code fonction 11 (*voir page 100*).

Implémentation réseau

L'opération de lecture de la CTE peut être réalisée sur des réseaux Ethernet TCP/IP et Ethernet SY/MAX.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	11 = lecture de la CTE
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Le paramètre de longueur : une valeur comprise entre 12 et 37.
COMMANDE[4]	Aucune signification
COMMANDE[5]	Registre de routage Octet de poids faible = index de mappage Soit une valeur affichée dans l'octet du registre, soit non utilisé. ou Octet de poids fort = emplacement du module de la carte réseau
COMMANDE[6] ... COMMANDE[9]	Le nombre apparaissant dans l'octet de poids faible dans la zone de 1 à 16 indique l'emplacement du module optionnel.

Utilisation du bloc de commande pour Ethernet SY/MAX

Registre	Signification
COMMANDE[1]	11 = lecture de la CTE
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre de mots transférés
COMMANDE[4]	Décalage d'octet dans la structure du registre de l'automate, qui indique à partir d'où les octets CTE sont lus.
COMMANDE[5]	Registre de routage MSB : emplacement du module NOE
COMMANDE[6]	Terminaison : FF hex
...	
COMMANDE[9]	

Implémentation de l'indicateur CTE (DATABUF)

Les valeurs de la table CTE apparaissent dans la sortie `DATABUF` lorsqu'une opération de lecture de la CTE est implémentée. Le registre affiche les données CTE suivantes :

Implémentation de l'indicateur CTE (`DATABUF`) :

Paramètres	Registre	Contenu
Type de trame	DATABUF[0]	1 = 802.3 2 = Ethernet
Adresse IP	DATABUF[1]	Premier octet de l'adresse IP
	DATABUF[2]	Deuxième octet de l'adresse IP
	DATABUF[3]	Troisième octet de l'adresse IP
	DATABUF[4]	Quatrième octet de l'adresse IP
Masque réseau inférieur	DATABUF[5]	Mot de poids fort
	DATABUF[6]	Mot de poids faible
Passerelle	DATABUF[7]	Premier octet de la passerelle
	DATABUF[8]	Deuxième octet de la passerelle
	DATABUF[9]	Troisième octet de la passerelle
	DATABUF[10]	Quatrième octet de la passerelle

Ecriture de la CTE

Description

L'opération d'écriture de la CTE écrit la table de configuration CTE à partir des données spécifiées (*DATABUF*) dans une table d'extension de configuration Ethernet spécifiée ou dans un emplacement spécifique. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération d'écriture de la CTE, utilisez le code fonction 12 (*voir page 100*).

Implémentation réseau

L'opération d'écriture de la CTE peut être réalisée sur des réseaux Ethernet TCP/IP (*voir page 134*) et Ethernet SY/MAX.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	12 = écriture de la CTE
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Le paramètre de longueur : une valeur comprise entre 12 et 37.
COMMANDE[4]	Aucune signification
COMMANDE[5]	Registre de routage Octet de poids faible = index de mappage Soit une valeur affichée dans l'octet de l'adresse, soit non utilisé. ou Octet de poids fort = emplacement du module de la carte réseau
COMMANDE[6] ... COMMANDE[9]	Le nombre apparaissant dans l'octet de poids faible dans la zone de 1 à 16 indique l'emplacement du module optionnel.

Utilisation du bloc de commande pour Ethernet SY/MAX

Registre	Signification
COMMANDE[1]	12 = écriture de la CTE (table d'extension de configuration)
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre de mots transférés
COMMANDE[4]	Décalage d'octet dans la structure de l'adresse de l'automate, qui indique où les octets CTE sont écrits.
COMMANDE[5]	Registre de routage Octet de poids fort = emplacement du module NOE Octet de poids faible = numéro de station cible
COMMANDE[6]	Terminaison : FF hex
COMMANDE[7]	Aucune signification
...	
COMMANDE[9]	

Implémentation de l'indicateur CTE (DATABUF)

Les valeurs de la table d'extension de configuration Ethernet apparaissent dans le champ de sortie `DATABUF` lorsqu'une opération d'écriture de la CTE est implémentée. Les registres servent à transférer les données CTE suivantes :

Implémentation de l'indicateur CTE (`DATABUF`) :

Paramètres	Registre	Contenu
Type de trame	DATABUF[0]	1 = 802.3 2 = Ethernet
Adresse IP	DATABUF[1]	Premier octet de l'adresse IP
	DATABUF[2]	Deuxième octet de l'adresse IP
	DATABUF[3]	Troisième octet de l'adresse IP
	DATABUF[4]	Quatrième octet de l'adresse IP
Masque réseau inférieur	DATABUF[5]	Mot de poids fort
	DATABUF[6]	Mot de poids faible
Passerelle	DATABUF[7]	Premier octet de la passerelle
	DATABUF[8]	Deuxième octet de la passerelle
	DATABUF[9]	Troisième octet de la passerelle
	DATABUF[10]	Quatrième octet de la passerelle

Envoi de message électronique

Description

Avec le service de notification par message électronique, les projets faisant intervenir des automates peuvent rapporter des alarmes ou événements. L'automate surveille le système et crée dynamiquement un message électronique pour alerter les utilisateurs locaux ou distants.

Un événement ou une condition définis par l'utilisateur déclenche la création d'un message par le bloc MSTR. Chaque message utilise l'un des trois en-têtes de message définis par l'utilisateur. Chaque message envoyé depuis l'automate peut contenir des informations textuelles et de variable (238 octets maximum).

Le projet sélectionne l'en-tête approprié. Chaque en-tête contient les éléments suivants :

- nom de l'expéditeur ;
- liste des destinataires ;
- objet.

Pour programmer un bloc MBP_MSTR pour l'envoi d'un message électronique, utilisez le code fonction 13 (*voir page 100*).

Implémentation réseau

Une opération d'envoi de message électronique peut s'effectuer sur un réseau Ethernet TCP/IP.

Utilisation du bloc de commande pour Ethernet TCP/IP

Registre	Signification
COMMANDE[1]	13 = envoi de message électronique
COMMANDE[2]	Indique les codes d'erreur spécifiques du message électronique (<i>voir page 149</i>).
COMMANDE[3]	Nombre de mots transférés
COMMANDE[4]	Non utilisé
COMMANDE[5]	Octet de poids fort : emplacement du module NOE ou 0xFE pour le 140 CPU 651 60.
	Octet de poids faible : toujours 0
COMMANDE[6]	Non utilisé
...	
COMMANDE[9]	

Description du paramètre DATABUF

Registre	Sommaire
DATABUF 1	L'en-tête de message électronique est l'octet de poids faible de valeur 1, 2 ou 3. L'octet de poids fort contient le nombre (n) de caractères de l'objet, une valeur comprise entre 0 et 238.
DATABUF 2 ... DATABUF 119	Données (au format ASCII) qui vont être copiées dans le message électronique. Les n premiers caractères sont ajoutés à l'objet du message configuré. Les caractères restants ($2 * N - 2 - n$) font partie du corps du message, où N correspond au nombre de mots transférés.

Lire/écrire données

Introduction

Au cours d'une transaction unique, l'opération MSTR de lecture/écriture permet de transférer des données d'un équipement source maître vers un équipement esclave cible donné, puis de transférer des données depuis cet équipement esclave source vers le maître. L'opération utilise un chemin de transaction maître et sa réalisation peut nécessiter plusieurs cycles. Pour programmer un bloc MBP_MSTR en vue d'exécuter une opération combinée de lecture/écriture, utilisez le code fonction 23 (voir page 100).

L'opération combinée de lecture/écriture peut être utilisée avec ces deux modèles Quantum :

- NOE 771 01 (version 3.0 ou ultérieure)
- NOE 771 11 (version 3.0 ou ultérieure)

Utilisation du bloc de commande

Registre	Contenu
COMMANDE[1]	23 = lecture/écriture des données.
COMMANDE[2]	Indique l'état d'erreur.
COMMANDE[3]	Nombre de registres à envoyer à l'esclave.
COMMANDE[4]	Indique l'adresse %MW de départ dans l'esclave, dans laquelle il faut écrire (ex. : 1 = %MW1, 49 = %MW49).
COMMANDE[5]	Registre de routage : Octet de poids fort : emplacement du module de la carte réseau. Octet de poids faible : index de mappage MBP sur Ethernet Transporter (MET).
COMMANDE[6] ... COMMANDE[9]	Chaque adresse contient 1 octet de l'adresse IP à 32 bits, où le MSB est dans COMMANDE[6] et le LSB dans COMMANDE[9].
COMMANDE[10]	Nombre de registres à lire depuis l'esclave.
COMMANDE[11]	Indique le registre de départ %MW dans l'esclave à partir duquel les données sont lues, par exemple 1 = %MW1, 49 = %MW49.

NOTE :

lorsque vous configurez le bloc MBP_MSTR pour une opération de lecture/écriture de données, notez que :

- le paramètre de sortie DATABUF est utilisé pour stocker, dans l'ordre suivant :
 - 1 les données à écrire,
 - 2 les données à lire,

- la taille du paramètre de sortie `DATABUF` doit être égale à la somme des tailles des données à écrire et des données à lire ; si sa taille est inférieure, des données sont écrasées et peuvent être perdues,
- les paramètres `CONTROL` et `DATABUF` doivent tous deux être stockés à des adresses affectées telles que les adresses `%MW`.

Etat de validité des communications Peer Cop

Etat de validité des communications Peer Cop

La table contenant les informations sur l'état de Peer Cop remplit 12 registres contigus indexés avec les nombres 0 ... 11 dans une opération MBP_MSTR. Chaque octet des mots de la table sert à présenter un aspect de la validité des communications pour un abonné spécifique du réseau Modbus Plus. Pour programmer un bloc MBP_MSTR pour obtenir l'état de validité de Peer Cop, utilisez le code fonction 9 (voir page 100).

Implémentation réseau

Une opération d'état de validité des communications Peer Cop ne peut s'effectuer que sur les réseaux Modbus Plus.

Relation entre les bits des abonnés du réseau

Les bits des mots 0 à 3 représentent la validité à l'entrée des communications globales des abonnés 1 à 64. Les bits des mots 4 ... 7 représentent la validité de la sortie d'un abonné spécifique.

Les bits des mots 8 à 11 représentent la validité de l'entrée d'un abonné spécifique.

Type d'état	Index des mots	Relation entre les bits des abonnés du réseau
Réception globale	0	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	2	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	3	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Envoi direct	4	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	5	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	6	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	7	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49

Type d'état	Index des mots	Relation entre les bits des abonnés du réseau
Réception directe	8	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	9	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	10	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	11	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49

Etat du bit de validité

L'état du bit de validité de Peer Cop indique l'état actuel des communications de l'abonné qui lui est affecté. Un bit de validité est défini lorsque l'abonné associé accepte l'entrée pour son bloc de données Peer Cop ou lorsqu'il reçoit un signal lui indiquant qu'un autre abonné a accepté des données de sortie spécifiques depuis son bloc de données de sortie Peer Cop. Un bit de validité est supprimé si le bloc de données associé n'accepte pas de communications dans la période de timeout de validité Peer Cop.

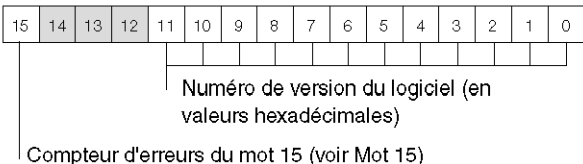
Tous les bits de validité sont supprimés lorsque la commande d'interface **Placer Peer Cop** est exécutée pendant le démarrage de l'automate. Les valeurs de la table deviennent valides lorsque le jeton est entièrement contourné, après que la commande **Placer Peer Cop** a été exécutée. Le bit de validité d'un abonné spécifique est toujours 0 lorsque l'entrée Peer Cop affectée est 0.

Statistiques du réseau Modbus Plus

Statistiques du réseau Modbus Plus

Le tableau suivant contient les statistiques disponibles sur Modbus Plus. Pour obtenir ces données, exécutez l'opération MBP_MSTR correspondante (code fonction 8 Modbus).

NOTE : Si vous modifiez l'opération de suppression de statistiques locales (voir page 110) ou de suppression de statistiques distantes (voir page 115), seuls les mots 13 à 22 de la table des statistiques sont supprimés.

Mot	données	Description
00	ID du type d'abonné	
	0	Type d'abonné inconnu
	1	Abonné automate
	2	Abonné pont Modbus
	3	Abonné ordinateur hôte
	4	Abonné routeur Plus
	5	Abonné E/S Peer
	6 ... 15	Réservé
01	0 ... 11	Numéro de version du logiciel sous forme de valeur hexadécimale (pour lire cette valeur, isoler les bits 12-15 du mot)
	12 ... 14	Réservé
	15	Définit les compteurs d'erreurs du mot 15. Le bit de poids fort définit l'utilisation des compteurs d'erreurs du mot 15. La moitié ayant la valeur la plus basse de l'octet de poids fort avec l'octet de poids faible contient la version du logiciel.
		
02		Adresse réseau de cette station

Mot	données	Description
03	Variable de l'état MAC :	
	0	Etat de démarrage
	1	Signaux de l'indicateur de l'état hors ligne
	2	Etat hors ligne dupliqué
	3	Etat de repos
	4	Etat d'utilisation du jeton
	5	Etat de la réponse du travail
	6	Etat du transfert du jeton
	7	Etat de la requête de réponse
	8	Contrôle de l'état du transfert
	9	Etat de la requête de jeton
10	Etat de la requête de réponse	
04	Etat de Peer (code LED) ; indique l'état de cet équipement par rapport au réseau :	
	0	Opération de connexion du moniteur
	32	Opération de connexion normale
	64	Jeton non reçu
	96	Station unique
128	Station en double	
05	Compteur de transfert du jeton (augmente chaque fois que la station reçoit le jeton)	
06		Temps de cycle du jeton en ms
07	BAS	Echec du maître de données pour la structure de bits pendant la propriété du jeton
	HAUT	Echec du programme maître pour la structure de bits (bitmap) pendant la propriété du jeton
08	BAS	Propriété du jeton pour l'activité bitmap du maître de données
	HAUT	Propriété du jeton pour l'activité bitmap du programme maître
09	BAS	Propriété du jeton pour l'activité bitmap des données esclaves
	HAUT	Propriété du jeton pour l'activité bitmap du programme esclave
10	BAS	
	HAUT	Commande de requête de transfert bitmap - interrogation données esclaves/esclaves
11	BAS	Requête de transfert de réponse bitmap - interrogation programme maître/maître
	HAUT	Commande de requête de transfert bitmap - interrogation programme esclave/esclave

Mot	données	Description
12	BAS	Etat de connexion bitmap du programme maître
	HAUT	Fermeture de session bitmap du programme esclave
13	BAS	Compteur d'erreurs de retard de prétransfert
	HAUT	Compteur d'erreurs de débordement du tampon de réception
14	BAS	Commande de répétition du compteur de réceptions
	HAUT	Compteur d'erreurs de taille de bloc de données
15	Si le bit 15 du mot 1 n'est pas défini, le mot 15 a la signification suivante :	
	BAS	Compteur d'erreurs d'abandon sur collision du destinataire
	HAUT	Compteur d'erreurs d'alignement du destinataire
	Si le bit 15 du mot 1 est défini, le mot 15 a la signification suivante :	
	BAS	Erreur de bloc de données sur le câble A
	HAUT	Erreur de bloc de données sur le câble B
16	BAS	Compteur d'erreurs CRC du destinataire
	HAUT	Compteur d'erreurs de longueur de paquet incorrecte
17	BAS	Compteur d'erreurs d'adresse de liaison incorrecte
	HAUT	Compteur d'erreurs de stockage de tampon de transfert de dépassement par valeur inférieure DMA
18	BAS	Compteur d'erreurs de longueur de paquet interne incorrecte
	HAUT	Compteur d'erreurs de code fonction MAC incorrect
19	BAS	Compteur de nouvelles tentatives de communication
	HAUT	Compteur d'erreurs d'échec de communication
20	BAS	Compteur de réception de paquets correcte
	HAUT	Compteur d'erreurs d'absence de réponse
21	BAS	Compteur d'erreurs de réception de réponse inattendue
	HAUT	Compteur d'erreurs de chemin inattendu
22	BAS	Compteur d'erreurs de réponse inattendue
	HAUT	Compteur d'erreurs de transaction ignorée
23	BAS	Table des stations actives bitmap, abonnés 1 à 8
	HAUT	Table des stations actives bitmap, abonnés 9 à 16
24	BAS	Table des stations actives bitmap, abonnés 17 à 24
	HAUT	Table des stations actives bitmap, abonnés 25 à 32
25	BAS	Table des stations actives bitmap, abonnés 33 à 40
	HAUT	Table des stations actives bitmap, abonnés 41 à 48

Mot	données	Description
26	BAS	Table des stations actives bitmap, abonnés 49 à 56
	HAUT	Table des stations actives bitmap, abonnés 57 à 64
27	BAS	Table des stations à jeton bitmap, abonnés 1 à 8
	HAUT	Table des stations à jeton bitmap, abonnés 9 à 16
28	BAS	Table des stations à jeton bitmap, abonnés 17 à 24
	HAUT	Table des stations à jeton bitmap, abonnés 25 à 32
29	BAS	Table des stations à jeton bitmap, abonnés 33 à 40
	HAUT	Table des stations à jeton bitmap, abonnés 41 à 48
30	BAS	Table des stations à jeton bitmap, abonnés 49 à 56
	HAUT	Table des stations à jeton bitmap, abonnés 57 à 64
31	BAS	Table bitmap concernant l'existence de données globales, abonnés 1 à 8
	HAUT	Table bitmap concernant l'existence de données globales, abonnés 9 à 16
32	BAS	Table bitmap concernant l'existence de données globales, abonnés 17 à 24
	HAUT	Table bitmap concernant l'existence de données globales, abonnés 25 à 32
33	BAS	Table bitmap concernant l'existence de données globales, abonnés 33 à 40
	HAUT	Table bitmap concernant l'existence de données globales, abonnés 41 à 48
34	BAS	Table bitmap concernant l'existence de données globales, abonnés 49 à 56
	HAUT	Table bitmap concernant l'existence de données globales, abonnés 57 à 64
35	BAS	Tampon de réception bitmap utilisé, tampons 1 à 8
	HAUT	Tampon de réception bitmap utilisé, tampons 9 à 16
36	BAS	Tampon de réception bitmap utilisé, tampons 17 à 24
	HAUT	Tampon de réception bitmap utilisé, tampons 25 à 32
37	BAS	Tampon de réception bitmap utilisé, tampons 33 à 40
	HAUT	Compteur des commandes traitées activées pour l'administration de la station

Mot	données	Description
38	BAS	Chemin de sortie 1 de la commande d'activation du compteur du maître de données
	HAUT	Chemin de sortie 2 de la commande d'activation du compteur du maître de données
39	BAS	Chemin de sortie 3 de la commande d'activation du compteur du maître de données
	HAUT	Chemin de sortie 4 de la commande d'activation du compteur du maître de données
40	BAS	Chemin de sortie 5 de la commande d'activation du compteur du maître de données
	HAUT	Chemin de sortie 6 de la commande d'activation du compteur du maître de données
41	BAS	Chemin de sortie 7 de la commande d'activation du compteur du maître de données
	HAUT	Chemin de sortie 8 de la commande d'activation du compteur du maître de données
42	BAS	Chemin d'entrée 41 du traitement de la commande de compteur des données esclaves
	HAUT	Chemin d'entrée 42 du traitement de la commande de compteur des données esclaves
43	BAS	Chemin d'entrée 43 du traitement de la commande de compteur des données esclaves
	HAUT	Chemin d'entrée 44 du traitement de la commande de compteur des données esclaves
44	BAS	Chemin d'entrée 45 du traitement de la commande de compteur des données esclaves
	HAUT	Chemin d'entrée 46 du traitement de la commande de compteur des données esclaves
45	BAS	Chemin d'entrée 47 du traitement de la commande de compteur des données esclaves
	HAUT	Chemin d'entrée 48 du traitement de la commande de compteur des données esclaves
46	BAS	Chemin de sortie 81 de l'activation de la commande de compteur du programme maître
	HAUT	Chemin de sortie 82 de l'activation de la commande de compteur du programme maître

Mot	données	Description
47	BAS	Chemin de sortie 83 de l'activation de la commande de compteur du programme maître
	HAUT	Chemin de sortie 84 de l'activation de la commande de compteur du programme maître
48	BAS	Chemin de sortie 85 de l'activation de la commande de compteur du programme maître
	HAUT	Chemin de sortie 86 de l'activation de la commande de compteur du programme maître
49	BAS	Chemin de sortie 87 de l'activation de la commande de compteur du programme maître
	HAUT	Chemin de sortie 88 de l'activation de la commande de compteur du programme maître
50	BAS	Chemin d'entrée C1 du traitement de la commande de compteur du programme esclave
	HAUT	Chemin d'entrée C2 du traitement de la commande de compteur du programme esclave
51	BAS	Chemin d'entrée C3 du traitement de la commande de compteur du programme esclave
	HAUT	Chemin d'entrée C4 du traitement de la commande de compteur du programme esclave
52	BAS	Chemin d'entrée C5 du traitement de la commande de compteur du programme esclave
	HAUT	Chemin d'entrée C6 du traitement de la commande de compteur du programme esclave
53	BAS	Chemin d'entrée C7 du traitement de la commande de compteur du programme esclave
	HAUT	Chemin d'entrée C8 du traitement de la commande de compteur du programme esclave

Statistiques de réseau Ethernet TCP/IP

Statistiques de réseau Ethernet TCP/IP

Un module Ethernet TCP/IP répond aux commandes de statistiques locales et distantes provenant du bloc MBP_MSTR par le contenu du tableau tampon de données `databuf` (voir les informations dans la table ci-après) :

Mot	Signification	Mot	Contenu
00 à 02	Adresse MAC Par exemple, l'adresse MAC 00 00 54 00 12 34 s'affiche comme suit :	00 01 02	00 54 00 34 00 12
03	Etat de la carte (consultez la table ci-après)		
04 et 05	Nombre d'interruptions récepteur		
06 et 07	Nombre d'interruptions transfert		
08 et 09	Compte d'erreur de timeout de transfert		
10 et 11	Compte d'erreur détection de collisions		
12 et 13	Paquets omis		
14 et 15	Compte d'erreur mémoire		
16 et 17	Nombre de redémarrages effectués par le driver		
18 et 19	Compte d'erreurs de trame de réception		
20 et 21	Compte d'erreurs de débordement du récepteur		
22 et 23	Compte d'erreurs CRC de réception		
24 et 25	Compte d'erreurs du tampon de réception		
26 et 27	Compte d'erreurs du tampon de transfert		
28 et 29	Compteur d'erreurs de dépassement par valeur inférieure corbeille de transfert		
30 et 31	Compteur de collisions tardives		
32 et 33	Compteur de pertes de porteuse		
34 et 35	Nombre de réitérations		
36 et 37	Adresse IP Par exemple, l'adresse IP 198.202.137.113 (ou c6 CA 89 71) s'affiche comme suit :	36 37	89 C6 71 CA

Définition des bits du mot d'état de la carte

NOTE : Il est préférable d'afficher le mot d'état de la carte au format binaire.

Le tableau suivant donne la signification des bits du mot d'état de la carte.

- 140 NOE 771 x1, versions 2.0, 3.0, 3.1, 3.3 et 3.6 ou supérieures
- 140 NOE 771x0, versions 3.0, 3.3 et 3.4 ou supérieures

N° de bit	Définition
15	0 = DEL Link éteinte 1 = DEL Link allumée
14	0 = DEL Appl éteinte 1 = DEL Appl allumée
13	0 = paire torsadée 1 = fibre optique
12	0 = 10 Mbits 1 = 100 Mbits
11 ... 8	(Réservé)
7 ... 4	Type de module (voir le tableau ci-dessous)
3	(Réservé)
2	0 = semi-duplex 1 = full duplex
1	0 = non configuré 1 = configuré
0	0 = ne fonctionne pas 1 = fonctionne

NOTE : Les bits sont comptés de la droite vers la gauche, en commençant par le bit 0 (bit faible). Par exemple, **Automate en fonctionnement** = 0000 0000 0000 0001 et **Connexion de la DEL** = 1000 0000 0000 0000.

Le tableau ci-après décrit les définitions du bit de mot concernant l'état de la carte pour :

- 140 NOE 771 x1, version 3.5
- 140 NOE 771 x0, versions 1.02 et 2.0
- 140 CPU 651 x0

N° de bit	Définition
15 ... 12	Type de module (voir le tableau ci-dessous)
11	(Réservé)
10	0 = semi-duplex 1 = full duplex
9	0 = non configuré 1 = configuré
8	0 = l'automate ne fonctionne pas 1 = l'automate/NOE fonctionne
7	0 = DEL Link éteinte 1 = DEL Link allumée

N° de bit	Définition
6	0 = DEL Appl éteinte 1 = DEL Appl allumée
5	0 = paire torsadée 1 = fibre optique
4	0 = 10 Mbits 1 = 100 Mbits
3 ... 0	(Réservé)

NOTE : Les bits sont comptés de la droite vers la gauche, en commençant par le bit 0 (bit faible). Par exemple, **automate fonctionne** = 0x0100, **DEL Application** = 0x0040 et **connexion DEL** = 0x0080.

Définition des bits du mot d'état de la carte par type de module

Le tableau ci-après décrit les valeurs des types de module :

Valeur des bits 7 à 4 ou 15 à 12 Remarque : Consultez les tables précédentes pour la plage de bits qui s'applique à la version logicielle de votre module.	Type de module
0	NOE 2x1
1	ENT
2	M1E
3	NOE 771 00
4	ETY
5	CIP
6	(réservé)
7	140 CPU 651 x0
8	(réservé)
9	(réservé)
10	NOE 771 10
11	NOE 771 01
12	NOE 771 11
13 ... 15	(réservé)

Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP

Forme du code d'erreur fonction

Les codes d'erreur fonction pour les transactions Modbus Plus et Ethernet SY/MAX apparaissent sous la forme **Mmss**, où :

- **M** correspond au code supérieur,
- **m** correspond au code inférieur,
- **ss** correspond à un sous-code.

Erreurs réseau Modbus Plus et Ethernet SY/MAX

Codes d'erreur hexadécimaux pour Modbus Plus et Ethernet SY/MAX :

Code d'erreur hex.	Description
1001	Abandon par l'utilisateur.
2001	Un type d'opération non pris en charge a été spécifié dans le bloc de commande.
2002	Un ou plusieurs paramètres du bloc de commande ont été modifiés pendant que l'élément MSTR était actif (cela s'applique uniquement aux opérations qui nécessitent plusieurs cycles d'exécution). Les paramètres du bloc de commande ne peuvent être modifiés que lorsque les composants MSTR sont inactifs.
2003	Valeur incorrecte dans le champ de longueur du bloc de commande.
2004	Valeur incorrecte dans le champ d'offset du bloc de commande.
2005	Valeur incorrecte dans les champs de longueur et d'offset du bloc de commande.
2006	Champ de données non autorisé sur l'esclave.
2007	Champ de réseau non autorisé sur l'esclave.
2008	Chemin de routage réseau non autorisé sur l'esclave.
2009	Chemins de routage équivalent à leur propre adresse.
200A	Tentative d'obtenir plus de mots global data que ceux qui sont disponibles.
200C	Motif incorrect pour la requête de changement d'adresse.
200D	Adresse incorrecte pour la requête de changement d'adresse.
200E	Le bloc de commande n'est pas affecté, ou des éléments du bloc de commande sont situés en dehors de la plage %MW (4x).
30ss	Réponse exceptionnelle de l'esclave Modbus. (voir page 140)

Code d'erreur hex.	Description
4001	Réponse incohérente de l'esclave Modbus.
5001	Réponse incohérente du réseau.
6007	ID d'emplacement incorrect.
6mss	Erreur de chemin de routage. (voir page 140) Le sous-champ m indique l'emplacement de l'erreur (0 indique le nœud local, 2 correspond au deuxième équipement du chemin, etc.).

Erreurs réseau Ethernet TCP/IP

Codes d'erreur hexadécimaux pour Ethernet TCP/IP :

Code d'erreur hex.	Signification
5004	Appel système interrompu
5005	Erreur d'E/S
5006	Adresse inexistante
5009	Descripteur de socket incorrect
500C	Mémoire insuffisante
500D	Autorisation refusée
5011	L'entrée existe
5016	Argument incorrect
5017	Espace insuffisant dans la table interne
5020	Connexion rompue
5028	Adresse de destination requise
5029	Protocole de type incorrect pour socket
502A	Protocole non disponible
502B	Protocole non géré
502C	Type socket non géré
502D	Opération non gérée sur un socket
502E	Famille de protocoles non gérée
502F	Famille d'adresses non gérée
5030	Adresse déjà utilisée
5031	Impossible d'affecter l'adresse demandée
5032	Opération socket sur un non-socket
5033	Réseau inaccessible

Code d'erreur hex.	Signification
5034	Connexion dérivée réseau sur réinitialisation
5035	Abandon de la connexion provoqué par le réseau
5036	Réinitialisation connexion par pair
5037	Pas d'espace buffer disponible
5038	Socket déjà relié
5039	Socket non relié
503A	Impossible d'émettre après arrêt socket
503B	Trop de références : impossible de lier
503C	Expiration connexion (voir remarque ci-dessous)
503D	Connexion refusée
503E	Arrêt réseau
503F	Fichier texte occupé
5040	Trop de niveaux de liaisons
5041	Pas de route vers l'hôte
5042	Equipement de bloc requis
5043	Hôte arrêté
5044	Opération en cours
5045	Opération déjà en cours
5046	Blocage possible de l'opération
5047	Fonctionnalité non implémentée
5048	Longueur de matériel incorrecte
5049	Chemin indiqué introuvable
504A	Collision dans l'appel de sélection : ces conditions ont déjà été sélectionnées par une autre tâche
504B	ID de tâche incorrect
5050	Aucune ressource réseau
5051	Erreur de longueur
5052	Erreur d'adressage
5053	Erreur d'application

Code d'erreur hex.	Signification
5054	Client associé à un état incorrect pour la requête
5055	Pas de ressource distante -- peut indiquer qu'il n'existe pas de chemin d'accès à l'équipement distant (voir remarque ci-dessous)
5056	Connexion TCP non opérationnelle
5057	Configuration incohérente

NOTE :

- Le code d'erreur 5055 peut être généré avant une erreur 503C.
- Aucun équipement distant n'a priorité sur un timeout.

Valeur hexadécimale ss dans le code d'erreur 30ss

Valeur hexadécimale ss dans le code d'erreur 30ss :

Valeur hex. ss	Description
01	L'esclave ne prend pas en charge l'opération demandée.
02	Des registres inexistant de l'esclave ont été demandés.
03	Une valeur de données non autorisée a été demandée.
05	L'esclave a accepté une commande de programme longue.
06	La fonction ne peut pas être exécutée actuellement : une commande longue est en cours d'exécution.
07	L'esclave a rejeté une commande de programme longue.

Valeur hexadécimale ss dans le code d'erreur 6mss

NOTE : le sous-champ m du code d'erreur 6mss est un `Index` dans les informations de routage qui indiquent l'emplacement de l'erreur (0 indique le nœud local, 2 correspond au deuxième équipement du chemin, etc.).

Le sous-champ ss du code d'erreur 6mss est le suivant :

Valeur hexadécimale ss	Description
01	Pas de réception réponse.
02	Accès au programme refusé.
03	Nœud hors service et incapable de communiquer.
04	Réponse reçue inhabituelle.
05	Chemin de données du nœud du routeur occupé.
06	Esclave hors service.
07	Adresse cible incorrecte.

Valeur hexadécimale ss	Description
08	Type de nœud non autorisé dans le chemin de routage.
10	L'esclave a rejeté la commande.
20	L'esclave a perdu une transaction active.
40	Chemin de sortie maître non attendu reçu.
80	Réponse reçue non attendue.
F001	Nœud cible erroné indiqué pour l'opération MSTR.

Codes d'erreur spécifiques SY/MAX

Codes d'erreur spécifiques SY/MAX

Si vous employez Ethernet SY/MAX, il est possible de déclarer trois types d'erreur supplémentaires dans le registre `CONTROL[1]` du bloc de commande ().

Les codes d'erreur ont la signification suivante :

- Erreur 71xx : Erreur détectée par l'équipement distant SY/MAX
- Erreur 72xx : Erreur détectée par le serveur
- Erreur 73xx : Erreur détectée par le compilateur Quantum

Code d'erreur HEX spécifique SY/MAX

Code d'erreur HEX spécifique SY/MAX :

Code Code d'erreur	Signification
7101	Code opérande invalide détecté par l'équipement distant SY/MAX
7103	Adresse invalide détectée par l'équipement distant SY/MAX
7109	Essai d'écriture d'un registre protégé en écriture détecté par l'équipement distant SY/MAX
F710	Débordement récepteur détecté par l'équipement distant SY/MAX
7110	Longueur invalide détectée par l'équipement distant SY/MAX
7111	Equipement distant non actif, pas de liaison (se produit lorsque toutes les tentatives et temporisations ont été utilisées), détecté par l'équipement distant SY/MAX
7113	Paramètre invalide détecté par l'équipement distant SY/MAX lors d'une commande de lecture
711D	Itinéraire invalide détecté par l'équipement distant SY/MAX
7149	Paramètre invalide détecté par l'équipement distant SY/MAX lors d'une commande d'écriture
714B	Numéro de station invalide détecté par l'équipement distant SY/MAX
7101	Code opérande invalide détecté par le serveur SY/MAX
7203	Adresse invalide détectée par le serveur SY/MAX
7209	Essai d'écriture dans un registre protégé en écriture détecté par le serveur SY/MAX
F720	Débordement récepteur détecté par le serveur SY/MAX
7210	Longueur invalide détectée par le serveur SY/MAX

Code Code d'erreur	Signification
7211	Equipement distant non actif, pas de liaison (se produit lorsque toutes les tentatives et temporisations ont été utilisées), détecté par le serveur SY/MAX
7213	Paramètre invalide détecté par le serveur SY/MAX lors d'une commande de lecture
721D	Itinéraire invalide détecté par le serveur SY/MAX
7249	Paramètre invalide détecté par le serveur SY/MAX lors d'une commande d'écriture
724B	Numéro de station invalide détecté par le serveur SY/MAX
7301	Code opérande invalide dans une requête de bloc MSTR par le compilateur Quantum
7303	Etat du module QSE Lecture/Ecriture (adresse de routage 200 hors limites)
7309	Essai d'écriture dans un registre protégé en écriture, lorsqu'une écriture d'état est en cours d'exécution (Routage 200)
731D	Itinéraire incorrect détecté par le compilateur Quantum. Itinéraires valides : <ul style="list-style-type: none"> ● dest_drop, 0xFF ● 200, dest_drop, 0xFF ● 100+drop, dest_drop, 0xFF ● Toutes les autres valeurs de routage entraînent une erreur.
734B	L'une des erreurs suivantes est survenue : <ul style="list-style-type: none"> ● Aucun tableau (de configuration) CTE n'a été configuré ● Aucune entrée de tableau CTE n'a été créée pour le numéro d'emplacement de modèle QSE ● Aucune station valide n'a été définie ● Le module QSE n'a pas été réinitialisé après la création de CTE. Rappel : Après écriture et configuration du CTE et chargement sur le module QSE, vous devez réinitialiser le module QSE pour que les modifications soient effectives. ● Lors de l'utilisation d'une instruction MSTR, aucun emplacement ou station valide n'a été défini(e)

Codes d'erreur Ethernet TCP/IP

Codes d'erreur Ethernet TCP/IP

Une erreur dans une routine MSTR via Ethernet TCP/IP peut générer l'une des erreurs suivantes dans le bloc de commande MSTR :

Le code d'erreur apparaît sous la forme **Mmss**, où :

- **M** correspond au code supérieur ;
- **m** correspond au code inférieur ;
- **ss** correspond à un sous-code.

Codes d'erreur HEX Ethernet TCP/IP

Codes d'erreur HEX Ethernet TCP/IP :

Code d'erreur hex.	Signification
1001	Abandon par l'utilisateur
2001	Un type d'opération non pris en charge a été spécifié dans le bloc de commande.
2002	Un ou plusieurs paramètres du bloc de commande ont été modifiés pendant que l'élément MSTR était actif (cela s'applique uniquement aux opérations qui nécessitent plusieurs cycles d'exécution). Les paramètres du bloc de commande ne peuvent être modifiés que lorsque les composants MSTR sont inactifs.
2003	Valeur incorrecte dans le champ de longueur du bloc de commande
2004	Valeur incorrecte dans le champ d'offset du bloc de commande
2005	Valeur incorrecte dans les champs de longueur et d'offset du bloc de commande
2006	Champ de données non autorisé sur l'esclave
2008	Chemin de routage réseau non autorisé sur l'esclave
200E	Le bloc de commande n'est pas affecté, ou des éléments du bloc de commande sont situés en dehors de la plage %MW (4x).
3000	Code d'échec Modbus générique
30ss	Réponse exceptionnelle de l'esclave Modbus (<i>voir page 145</i>)
4001	Réponse incohérente de l'esclave Modbus

Valeur hexadécimale ss dans le code d'erreur 30ss

Valeur hexadécimale ss dans le code d'erreur 30ss :

Valeur hex. ss	Signification
01	L'esclave ne prend pas en charge l'opération requise.
02	Des registres inexistant de l'esclave ont été requis.
03	Une valeur de données non autorisée a été requise.
05	L'esclave a accepté une commande de programme longue.
06	La fonction ne peut actuellement pas être exécutée : une commande longue est en cours d'exécution.
07	L'esclave a rejeté une commande de programme longue.

Codes d'erreur HEX du réseau Ethernet TCP/IP

Une erreur sur le réseau Ethernet TCP/IP peut générer l'une des erreurs suivantes dans le registre du bloc de commande `CONTROL[1]`.

Codes d'erreur HEX du réseau Ethernet TCP/IP :

Code d'erreur hex.	Signification
5004	Appel système interrompu
5005	Erreur E/S
5006	Adresse inexistante
5009	Descripteur de socket non valide
500C	Espace de stockage insuffisant
500D	Autorisation refusée
5011	Entrée existante
5016	Argument non valide
5017	Table interne saturée
5020	Interférence sur la connexion
5023	Cette opération a été bloquée et le socket n'est pas bloquant.
5024	Le socket n'est pas bloquant et la connexion ne peut pas être interrompue.
5025	Le socket n'est pas bloquant et la tentative de connexion précédente n'a pas abouti.
5026	Opération socket sur non socket
5027	Adresse cible non valide
5028	Message trop long

Code d'erreur hex.	Signification
5029	Type de protocole incorrect pour le socket
502A	Protocole non disponible
502B	Protocole non pris en charge
502C	Type socket non pris en charge
502D	Opération non prise en charge sur socket
502E	Famille de protocoles non prise en charge
F502	Famille d'adresses non prise en charge
5030	Adresse est utilisée
5031	Adresse non disponible
5032	Réseau hors service
5033	Réseau injoignable
5034	Le réseau a interrompu la connexion pendant la réinitialisation.
5035	La connexion a été interrompue par l'homologue.
5036	La connexion a été réinitialisée par l'homologue.
5037	Une mémoire tampon interne est requise, mais elle ne peut pas être affectée.
5038	Socket déjà connecté
5039	Socket non connecté
503A	Transmission impossible après l'arrêt du socket
503B	Trop de références : jonction impossible
503C	Connexion expirée
503D	Tentative de connexion refusée
5040	Hôte hors service
5041	Impossible d'accéder à l'hôte cible depuis cet abonné
5042	Répertoire non vide
5046	NI_INIT a renvoyé la valeur -1.
5047	MTU non valide
5048	Longueur matérielle non valide
5049	Chemin spécifié introuvable
504A	Collision lors de l'appel de la commande Sélectionner : ces conditions ont déjà été sélectionnées par un autre job.
504B	ID de job non valide

Code d'erreur hex.	Signification
5050	Aucune ressource réseau
5051	Erreur de longueur
5052	Erreur d'adressage
5053	Erreur d'application
5054	Le client ne peut pas traiter la requête.
5055	Aucune ressource réseau
5056	Connexion TCP non opérationnelle
5057	Configuration incohérente
6003	FIN ou RST inattendu
F001	En mode de réinitialisation
F002	Le composant n'est pas totalement initialisé.

Codes d'erreur CTE pour Ethernet SY/MAX et TCP/IP

Codes d'erreur CTE pour Ethernet SY/MAX et TCP/IP

Les codes d'erreur suivants sont indiqués dans le registre `CONTROL[1]` du bloc de commande si la configuration de votre programme est à l'origine d'un problème avec le tableau d'extension de configuration (CTE).

Codes d'erreur CTE pour Ethernet SY/MAX et TCP/IP :

Code Code d'erreur	Signification
7001	Il n'existe pas d'extension de configuration Ethernet
7002	Le CTE n'est pas disponible en accès
7003	Le décalage est non valide
7004	Décalage + longueur incorrects
7005	Tableau de données défectueux dans le CTE

Codes d'erreur du service de messagerie

Codes d'erreur

Le service de notification par message électronique prend en charge les codes d'erreur suivants :

Code d'erreur hex.	Description
5100	Erreur interne
5101	Composant SMTP non opérationnel
5102	En-tête de message non configuré
5103	Valeur d'en-tête de message incorrecte (valeurs correctes : 1, 2 ou 3)
5104	Connexion au serveur SMTP impossible
5105	Erreur d'émission du contenu du message électronique vers le serveur SMTP
5106	Erreur de fermeture de connexion SMTP avec le serveur
5107	Echec de requête HELO SMTP
5108	Echec de requête MAIL SMTP. Le serveur SMTP nécessite peut-être une authentification.
5109	Echec de requête RCPT SMTP
510A	Aucun destinataire accepté par le serveur SMTP
510B	Echec de requête DATA SMTP
510C	Longueur incorrecte de la requête d'envoi de message électronique
510D	Echec d'authentification
510E	Réception d'une requête de réinitialisation de composant alors que la connexion était ouverte

ModbusP_ADDR : adresse Modbus Plus

13

Introduction

Ce chapitre décrit le bloc ModbusP_ADDR.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	152
Description détaillée	155

Description

Description de la fonction

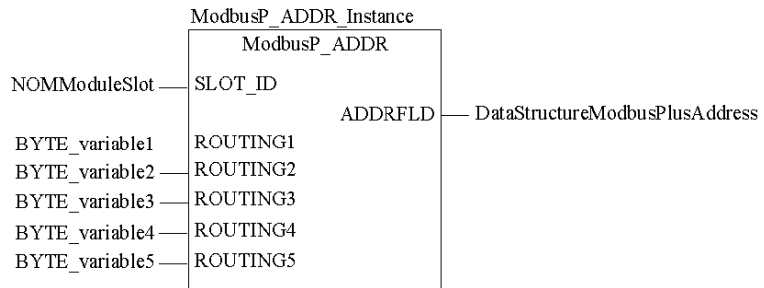
Ce bloc fonction permet d'indiquer l'adresse Modbus Plus pour les blocs fonction REAG_REG, CREAD_REG, WRITE_REG et CWRITE_REG. L'adresse est transmise sous forme de structure de données.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

NOTE : Lorsque vous programmez le bloc fonction ModbusP_ADDR, il vous faut connaître le réseau que vous utilisez. Les structures des itinéraires de routage Modbus Plus sont décrites en détail dans le "Guide de planification et d'installation du réseau Modbus Plus".

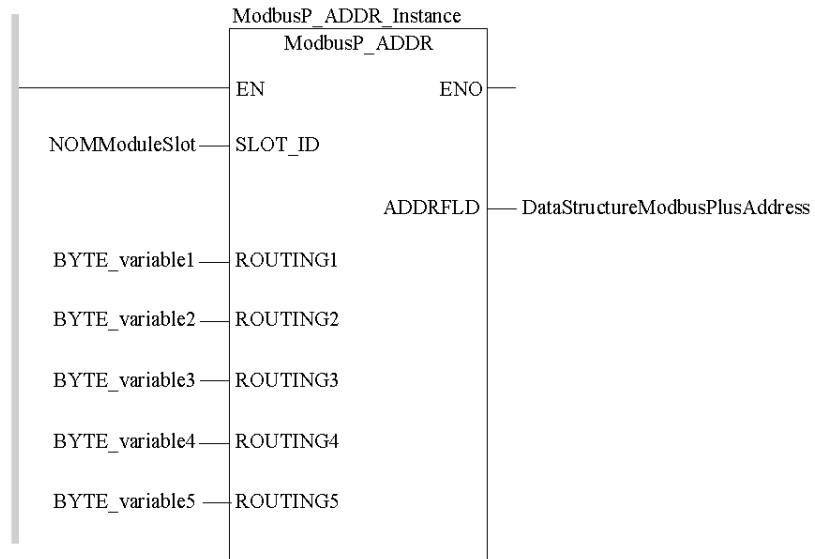
Représentation dans FBD

Représentation :



Représentation dans LD

Représentation :



Représentation dans IL

Représentation :

```

CAL ModbusP_ADDR_Instance (SLOT_ID:=NOMModuleSlot,
    ROUTING1:=BYTE_variable1, ROUTING2:=BYTE_variable2,
    ROUTING3:=BYTE_variable3, ROUTING4:=BYTE_variable4,
    ROUTING5:=BYTE_variable5,
    ADDRFLD=>DataStructureModbusPlusAddress)
  
```

Représentation dans ST

Représentation :

```

ModbusP_ADDR_Instance (SLOT_ID:=NOMModuleSlot,
    ROUTING1:=BYTE_variable1, ROUTING2:=BYTE_variable2,
    ROUTING3:=BYTE_variable3, ROUTING4:=BYTE_variable4,
    ROUTING5:=BYTE_variable5,
    ADDRFLD=>DataStructureModbusPlusAddress) ;
  
```

Description des paramètres

Description des paramètres d'entrée :

Paramètre	Type de données	Signification
SLOT_ID	BYTE	ID d'emplacement Emplacement du module NOM
ROUTING1	BYTE	Routage 1, sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau. Le dernier octet différent de zéro de l'itinéraire de routage est l'abonné cible.
ROUTING2	BYTE	Routage 2
ROUTING3	BYTE	Routage 3
ROUTING4	BYTE	Routage 4
ROUTING5	BYTE	Routage 5

Description des paramètres de sortie :

Paramètre	Type de données	Signification
ADDRFLD	WordArr5	Structure de données pour la transmission de l'adresse Modbus Plus

Description détaillée

Types de données dérivés

Description des éléments de `WordArr5` :

Élément	Type de données	Signification
<code>WordArr5[1]</code>	WORD	Registre de routage 1 Octet de poids faible : sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau. Octet de poids fort : Emplacement du module réseau (NOM), si disponible.
<code>WordArr5[2]</code>	WORD	Registre 2 de routage
<code>WordArr5[3]</code>	WORD	Registre 3 de routage
<code>WordArr5[4]</code>	WORD	Registre 4 de routage
<code>WordArr5[5]</code>	WORD	Registre 5 de routage

Slot_ID

Lorsqu'un module réseau optionnel (NOM) Modbus Plus est interrogé et réagit comme abonné cible dans le châssis d'un automate Quantum, la valeur de l'entrée `Slot_ID` représente l'emplacement physique du NOM, c.-à-d. que lorsque le NOM est enfiché sur l'emplacement 7 du châssis, la valeur a l'aspect suivant :

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

Routage x

L'entrée `Routage x` sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau. Le dernier octet différent de zéro de l'itinéraire de routage est l'abonné cible.

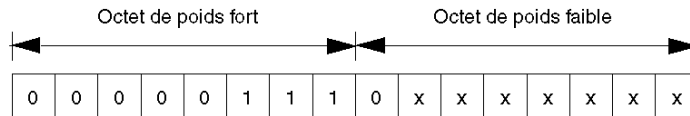
0	x	x	x	x	x	x	x
---	---	---	---	---	---	---	---

Adresse cible (valeur binaire entre 1 et 64 (normal) ou entre 65 et 249 (étendu))

Registre 1 de routage

Lorsqu'un module réseau optionnel (NOM) Modbus Plus dans le châssis d'un automate Quantum est interrogé comme abonné cible, la valeur de l'octet de poids fort représente l'emplacement physique du NOM. Si l'abonné cible est une UC, l'octet de poids fort est configuré sur "0" (quelque soit l'emplacement de l'UC).

Lorsque le NOM est enfiché sur l'emplacement 7 du châssis, l'octet de poids fort du registre 1 de routage est le suivant :



Octet de poids fort Emplacement 1 à 16

Octet de poids faible Adresse cible (valeur binaire entre 1 et 64 (normal) ou entre 65 et 255 (étendu))

OUT_IN_CHAR : envoi/réception de chaînes de caractères

14

Objet de ce chapitre

Ce chapitre décrit la fonction de communication `OUT_IN_CHAR`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	158
Ecran de saisie assistée	162
Exemple d'envoi/réception d'une chaîne de caractères	164

Description

Description de la fonction

La fonction `OUT_IN_CHAR` permet l'émission d'une chaîne de 210 octets maximum (120 pour la prise terminal) suivie d'une demande de réception de message (l'émission seule ou la réception seule est également possible).

La chaîne de caractères peut être contenue dans une variable statique (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) ou être stipulée sous forme de valeur immédiate (suite d'octets entre apostrophes, exemple : 'Message à envoyer').

Ces chaînes de caractère peuvent contenir des caractères spéciaux, ils doivent débiter par le caractère `$` suivi par la valeur hexadécimale du caractère à transmettre, exemple `$0D`.

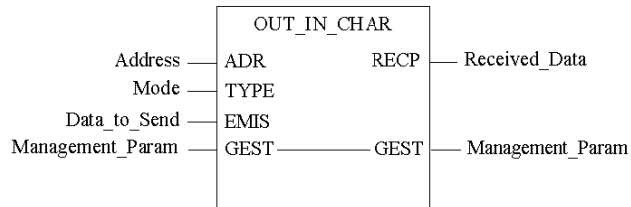
Certains caractères spéciaux (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) peuvent être utilisés comme :

`$R` = CR (retour chariot), `$L` = LF (retour à la ligne), `$N` = CR+LF.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

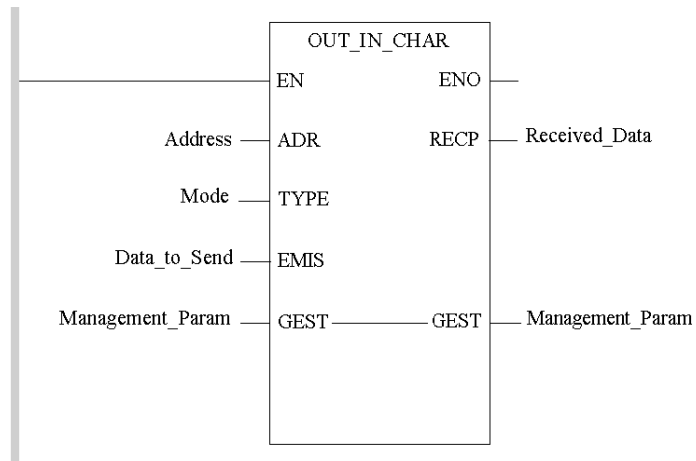
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

OUT_IN_CHAR Mode, Data_to_Send, Management_Param,
Received_Data

Représentation en ST

Représentation :

OUT_IN_CHAR(Address, Mode, Data_to_Send, Management_Param,
Received_Data);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0..5] OF INT	Adresse de l'entité destinataire de l'échange Seules les adresses qui se terminent par SYS sont possibles (exemple : {Réseau.Station}SYS).
Mode	INT	Mode d'opération : <ul style="list-style-type: none"> ● 1 : émission suivie d'une mise en réception, ● 2 : émission d'un message, ● 3 : mise en réception de message.
Data_to_Send	STRING	Chaîne de caractères à envoyer. Cette chaîne de caractères doit obligatoirement exister lors de l'appel de la fonction même s'il n'y a aucune donnée à émettre (mode réception simple par exemple).

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0..3] OF INT	Table de gestion de l'échange (<i>voir page 33</i>).

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Received_Data	STRING	Chaîne de caractère reçue. Cette chaîne de caractère doit obligatoirement exister lors de l'appel de la fonction même s'il n'y a aucune donnée à recevoir (mode émission simple par exemple).

NOTE : lors d'une émission ou d'une émission suivie d'une réception, il est conseillé d'initialiser le quatrième mot de la table de gestion (longueur) avant l'exécution de la fonction. D'autre part, la longueur de la chaîne de caractères reçue est mémorisée dans ce même mot à la fin de l'échange. Toutefois, il faut au préalable lors de la configuration définir une condition d'arrêt (*voir Premium et Atrium sous Unity Pro, Liaison série asynchrone, Manuel de l'utilisateur*).

Pour envoyer des chaînes de caractères contenant des caractères de fin de chaîne (NULL). Vous devez :

- utiliser des STRING localisées,
- initialiser le dernier mot du tableau de gestion des échanges avec le nombre de caractères à émettre. Si vous initialisez ce mot avec la valeur 0 la chaîne envoyée s'arrêtera au premier caractère NULL rencontré. Si vous l'initialisez avec une valeur, la longueur de la chaîne de caractère envoyée sera égale à cette valeur.

Ecran de saisie assistée

Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que l'écran de saisie assistée n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptés dans les différents champs de l'écran.

Illustration

La capture suivante est un exemple d'écran de saisie assistée de la fonction :

The screenshot shows a dialog box titled "OUT_IN_CHAR" with a "Paramètres" section. It contains several input fields and buttons:

- Adresse :** A text field with a dropdown arrow and a button with three dots.
- Mode :** A dropdown menu with an "Echange" button.
- Chaîne à émettre :** A section containing:
 - Variable :** A text field with a dropdown arrow and a button with three dots.
 - Valeur :** A large text area with a button with three dots on the right.
- Chaîne à recevoir :** A text field with a dropdown arrow and a button with three dots.
- Compte rendu :** A text field with a dropdown arrow and a button with three dots.

At the bottom, there is a note: "Types possibles : tableau d'entiers constants, tableau d'entiers (n>=6) Adresse ADDR(.'.)" and two buttons: "OK" and "Annuler".

Adresse

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING),
- ARRAY [0..5] OF INT.

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

Mode

Les choix dans la liste déroulante :

- Echange,
- Emission,
- Réception.

affichent directement la valeur immédiate 1, 2 ou 3.

NOTE : sélectionnez une solution parmi celles proposées dans le menu déroulant.

NOTE : si vous utilisez le champ de saisie à la place du menu, vous pouvez saisir une variable de type INT, affectée ou non.

Chaîne à émettre

La chaîne à émettre est une variable de type `STRING`. Cette variable doit être déclarée avant d'être utilisée.

Si une variable est choisie (par exemple, `String_0`), la valeur immédiate du champ de saisie disparaît.

Chaîne à recevoir

La zone de réception est une variable de type `STRING`. La taille de cette variable dépend du nombre de caractères à recevoir. La variable doit être déclarée avant d'être utilisée dans cet écran.

Compte rendu

Le compte rendu est un tableau de 4 entiers.

NOTE : veillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

Exemple d'envoi/réception d'une chaîne de caractères

Présentation

Considérons que vous souhaitez envoyer une chaîne de caractères `Str_1` de la station 1 du réseau 20 au port terminal de la station 5, puis recevoir une chaîne de caractères `Str_2` dans le port terminal de la station 5 du même réseau.

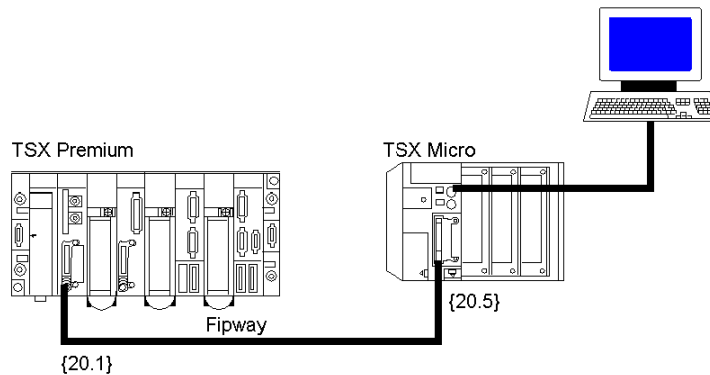
Un terminal vidéo est connecté au port terminal de la station 5 du réseau 20.

La chaîne de caractères à émettre contient 10 caractères.

La table de gestion de la fonction est composée d'une table de mots `%MW170:4`.

Illustration

Les deux stations sont connectées via un réseau Fipway.



Programmation

Programmation en ST :

```
IF RE(%I0.3.8) AND NOT %MW170.0 THEN
  (* initialisation des données à envoyer *)
  %MW173 := 10;
  (* fonction de communication *)

  OUT_IN_CHAR(ADDR('{20.5}0.0.0.SYS'),1,Str_1,%MW170:4,Str_2);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.5}0.0.0.SYS')	<ul style="list-style-type: none"> ● {20.5}: réseau 20, station 5 ● 0: rack ● 0: module ● 0: voie 0 ● SYS : adresse système
1	Emission puis réception
Str_1	Variable de type <code>STRING</code> contenant le message à émettre
%MW170:4	Table de gestion
Str_2	Variable de type <code>STRING</code> devant contenir le message reçu

NOTE : Avant le lancement de la fonction, initialisez le paramètre de longueur (dans l'exemple : %MW173) à l'aide de la valeur correspondant au nombre de caractères (en octets) à envoyer à Str_1).

A la fin de l'échange, %MW173 contient la longueur des données reçues (en octets).

OUT_IN_MBUS : fonction de communication Modbus

15

Objet de ce chapitre

Ce chapitre décrit la fonction de communication OUT_IN_MBUS.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre	Sujet	Page
15.1	Présentation générale du bloc de communication OUT_IN_MBUS	168
15.2	Description du bloc de communication OUT_IN_MBUS	173
15.3	Mise en oeuvre du bloc de communication OUT_IN_MBUS	185
15.4	Exemple d'utilisation du bloc de communication OUT_IN_MBUS	193

15.1 **Présentation générale du bloc de communication OUT_IN_MBUS**

Objet de ce sous-chapitre

Ce sous-chapitre présente une description sommaire du bloc de communication OUT_IN_MBUS.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Description de la fonction	169
Cas d'utilisation	170
Fonctionnalités	172

Description de la fonction

Introduction

La fonction `OUT_IN_MBUS` permet d'émuler une communication Modbus maître à partir d'une liaison série configurée en mode caractères.

Combinée à la possibilité de transmettre une configuration Modbus esclave en temps réel à une configuration en mode caractères via la fonction `WRITE_CMD`, cette fonction permet à l'automate de fonctionner comme maître ou comme esclave Modbus sur la même liaison.

NOTE : cette fonction n'est utile que lorsque les deux modes Modbus maître et esclave fonctionnent simultanément. Si tel n'est pas le cas, les normes EF `READ_VAR` et `WRITE_VAR` sont recommandées pour la gestion de la fonction de maître Modbus (l'esclave Modbus est implicitement géré par le système). Dans ce cas, consultez la documentation Modbus.

NOTE : Veillez à ce que deux maîtres (sur le même bus) n'envoient pas des requêtes simultanément, autrement les demandes risquent d'être perdues et les différents rapports sont susceptibles de générer un résultat incorrect 16#0100 (en cas d'impossibilité de traiter la requête) ou 16#ODFF (en cas d'absence de l'esclave).

AVERTISSEMENT

COMPORTEMENT INATTENDU DE L'APPLICATION - ECRITURE DE VARIABLES SUR DES ESCLAVES

Avant d'écrire des variables d'un maître vers un esclave, vérifiez que le comportement de l'application résultant reste acceptable.

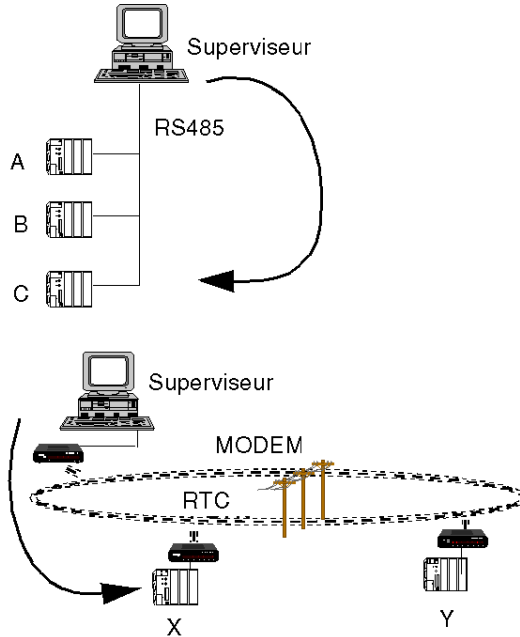
Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

Cas d'utilisation

Le mode nominal

La plupart des applications sont constituées d'un PC supportant des applications de supervision. Le superviseur est un maître Modbus et communique avec différents esclaves. Ce mode de fonctionnement est appelé mode nominal.

La figure ci dessous illustre le mode nominal (superviseur vers esclave) :

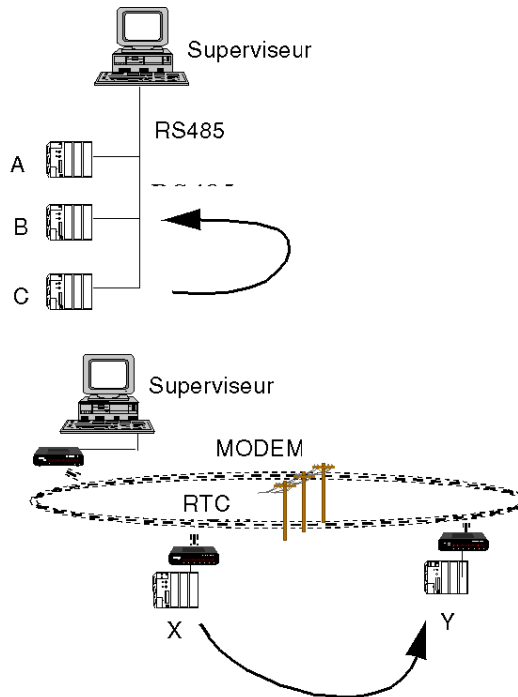


Le mode d'exception

Sur demande, un automate doit être capable de basculer de l'état esclave Modbus vers l'état maître Modbus afin d'envoyer des requêtes aux autres automates ou équipements. Ce basculement est appelé mode d'exception.

Quand le mode d'exception est terminé, le système doit être capable de revenir à l'état initial (au mode nominal).

La figure ci-dessous illustre le mode d'exception (maître émulé vers esclave) :



Fonctionnalités

Présentation

La fonction `OUT_IN_MBUS` prend en charge :

- Les codes Modbus 1, 2, 3, 4, 5, 6, 15, 16 (*voir page 177*),
- Les modes ASCII et RTU (*voir page 181*),
- L'adressage complet (*voir page 177*).

La fonction `OUT_IN_MBUS` est présente sous forme de DFB et doit être redémarrée pour chaque cycle de l'automate tant que le bit d'activité est réglé sur 1.

Cette fonction est disponible sur des automates Premium utilisant une carte de communication TSXSCP111 ou TSXSCP114 installée sur le processeur ou un module hôte (TSXSCY21601 ou TSXSCY11601).

Restrictions

Sur chacun des ports, vous ne devez pas activer simultanément plus d'un DFB `OUT_IN_MBUS`. De même, vous ne devez pas utiliser simultanément les blocs `PRINT_CHAR`, `INPUT_CHAR` ou `OUT_IN_CHAR`.

La fonction `OUT_IN_MBUS` ne prend pas en charge :

- La modification des paramètres des couches physiques ou des couches de liaison : Débit en bauds, format de caractère, passage du RTU en ASCII ou vice versa, gestion des signaux RS232,
- Les conflits et erreurs de communication pouvant résulter de la présence de deux maîtres Modbus actifs simultanément sur la même liaison,
- Tous les modes opératoires pouvant résulter d'un défaut ou d'une connexion temporaire à une carte de communication,
- La configuration et la gestion de modems.

15.2 Description du bloc de communication OUT_IN_MBUS

Objet de ce sous-chapitre

Ce sous-chapitre présente une description détaillée du bloc de communication OUT_IN_MBUS.

Contenu de ce sous-chapitre

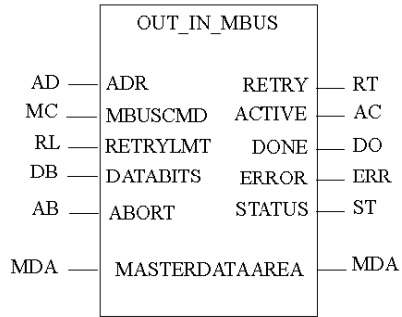
Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Représentations et paramètres	174
Le paramètre <code>MbusCmd</code>	177
Le paramètre <code>RetryLmt</code>	180
Le paramètre <code>DataBits</code>	181
Le paramètre <code>RespTout</code>	182
Le paramètre <code>MasterDataArea</code>	183
Le paramètre <code>Status</code>	184

Représentations et paramètres

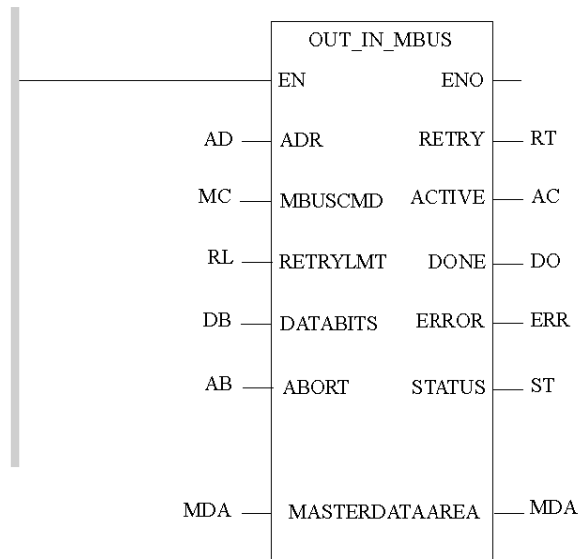
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

Adresse LD

OUT_IN_MBUS AD, MC, RL, DB, AB, MDA, RT, AC, DO, ERR, ST

Représentation en ST

Représentation :

OUT_IN_MBUS (AD, MC, RL, DB, AB, MDA, RT, AC, DO, ERR, ST)

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
ADDR	ADDR_TYPE	Informations sur l'adresse du port de communications de l'équipement esclave. Cette adresse doit être de la forme : ADDR('r.m.c.SYS'). Abréviations utilisées : r = rack, m = emplacement de module, c = voie (channel).
MBUSCMD	Tableau [1.. 4] de INT	Définition (<i>voir page 177</i>) du tableau Modbus
RETRYLMT	INT	Nombre de tentatives (<i>voir page 180</i>) d'envoi d'un message effectuées par le bloc
DATABITS	BOOL	Messages (<i>voir page 181</i>) Modbus à envoyer en mode ASCII (DATABITS =0) ou en mode RTU (DATABITS=1).
RESPTOUT	INT	Délai d'attente (<i>voir page 182</i>) du bloc.
ABORT	BOOL	Bit d'annulation DFB

Le tableau suivant décrit le paramètre d'E/S :

Paramètre	Type	Commentaire
MASTERDATAAREA	Tableau [x.. y] de INT	Zone de données (<i>voir page 183</i>) de l'automate maître.

NOTE : Pour pouvoir lire l'un des 8 derniers bits de la mémoire, vous devez lire l'intégralité des 8 derniers bits. Dans le cas contraire, la fonction renvoie un compte rendu d'erreurs. Vous pouvez ensuite extraire le bit souhaité.

NOTE : Lorsque le paramètre d'E/S utilise un tableau dynamique, vous devez sélectionner l'option Autoriser les tableaux dynamiques Unity Pro [ANY_ARRAY_XXX], située sous **Outil** → **Options du projet** → **Extensions de langage**.

Le tableau suivant décrit les paramètres de sortie (lecture seule) :

Paramètre	Type	Commentaire
RETRY	INT	La valeur affichée indique le nombre de tentatives en cours effectuées par le bloc.
ACTIVE	BOOL	La valeur 1 indique qu'une opération est en cours.
DONE	BOOL	La valeur 1 indique que l'opération est terminée.
ERROR	BOOL	La valeur 1 indique qu'une erreur s'est produite.
STATUS	INT	La valeur affiche un code de statut (<i>voir page 184</i>) généré par le bloc.

Le paramètre MbusCmd

Définition du paramètre MbusCmd

Le paramètre MbusCmd représente la commande Modbus.

Le paramètre MbusCmd est constitué d'un tableau de 4 registres tel que présenté ci-dessous :

Contenu	Description	Description
MbusCmd[1]	Adresse esclave	<p>Ce mot contient l'adresse de l'automate Modbus esclave.</p> <p>La plage des adresses admissibles est de 0 à 248. L'adresse 0 est réservée pour envoyer un message Modbus à plusieurs automates. Ce type de transmission est appelé mode diffusion.</p> <p>Le mode diffusion prend uniquement en charge les codes de fonction Modbus écrivant des données de l'automate maître vers des automates esclaves. Il ne prend pas en charge les codes de fonction Modbus lisant des données des automates esclaves.</p> <p>L'adresse 248 est réservée à la communication point à point quand l'adresse de l'esclave n'est pas connue. Cette adresse n'est pas supportée par tous les équipements.</p>
MbusCmd[2]	Code fonction Modbus	<p>Le bloc OUT_IN_MBUS prend en charge les codes de fonction suivants :</p> <ul style="list-style-type: none"> ● 01 = lecture de plusieurs bits de sortie (0x), ● 02 = lecture de plusieurs bits d'entrée (1x), ● 03 = lecture de plusieurs registres de sortie (4x), ● 04 = lecture de plusieurs registres d'entrée (3x), ● 05 = écriture d'un seul bit de sortie (0x), ● 06 = écriture d'un seul registre de sortie (4x), ● 15 = écriture de plusieurs bits de sortie (0x), ● 16 = écriture de plusieurs registres de sortie (4x). <p>Note : quand l'automate esclave est de type Premium, tous les bits deviennent des %M et les registres deviennent des %MW.</p>

Contenu	Description	Description
MbusCmd[3]	Zone de données de l'automate esclave	<p>Pour une commande de lecture, la zone de données de l'automate esclave est la source des données. Pour une commande d'écriture, la zone de données de l'automate esclave est la destination des données.</p> <p>Par exemple :</p> <ul style="list-style-type: none"> • pour lire les bits de sortie 300 à 500 d'un automate esclave, entrez 300 dans ce champ, • pour écrire des données d'un automate maître dans le registre 100 de type 4x d'un automate esclave, entrez 100 dans ce champ. <p>Selon le type de commande Modbus (lecture ou écriture), les zones de données source et cible doivent être conformes à celles du tableau ci-après (voir page 178).</p>
MbusCmd[4]	Quantité	<p>Ce registre contient la quantité de données à écrire ou à lire dans l'automate esclave.</p> <p>Par exemple, entrez 100 pour lire 100 registres de sortie dans l'automate esclave ou entrez 32 pour écrire 32 bits de sortie dans un automate esclave.</p> <p>Il existe une taille limite, qui dépend du code fonction Modbus utilisé et du mode de transmission (RTU ou ASCII). Ces valeurs limites de MbusCmd[4] sont détaillées dans le tableau ci-après (voir page 179). Cette taille est non significative pour les codes fonctions 5 et 6.</p> <p>Note : la zone mémoire est limitée en fonction de l'équipement et du paramétrage de l'esclave</p>

MbusCmd[3]

Le tableau ci-dessous présente la zone de données de l'automate esclave pour **MbusCmd[3]**. Cette zone de données dépend du code fonction Modbus utilisé et du type d'automate esclave :

Code fonction	Zone de données pour équipement standard Modbus	Zone de données pour automate Premium
01 (lecture de plusieurs bits de sortie (0x))	0x (source)	%M (source)
02 (lecture de plusieurs bits d'entrée (1x))	1x (source)	%M (source)
03 (lecture de plusieurs registres de sortie (4x))	4x (source)	%MW (source)
04 (lecture de plusieurs registres d'entrée (3x))	3x (source)	%MW (source)

Code fonction	Zone de données pour équipement standard Modbus	Zone de données pour automate Premium
05 (écriture d'un seul bit de sortie (0x))	0x (destination)	%M (destination)
06 (écriture d'un seul registre de sortie (4x))	4x (destination)	%MW (destination)
15 (écriture de plusieurs bits de sortie (0x))	0x (destination)	%M (destination)
16 (écriture de plusieurs registres de sortie (4x)).	4x (destination)	%MW (destination)

MbusCmd[4]

Le tableau ci-dessous présente la valeur limite de **MbusCmd[4]**. Cette valeur dépend du code fonction Modbus utilisé et du mode de transmission :

Code fonction	Mode RTU (8 bits)	Mode ASCII (7 bits)
01 (lecture de plusieurs bits de sortie (0x))	1000	500
02 (lecture de plusieurs bits d'entrée (1x))	1000	500
03 (lecture de plusieurs registres de sortie (4x))	100	50
04 (lecture de plusieurs registres d'entrée (3x))	100	50
05 (écriture d'un seul bit de sortie (0x))	1	1
06 (écriture d'un seul registre de sortie (4x))	1	1
15 (écriture de plusieurs bits de sortie (0x))	1000	500
16 (écriture de plusieurs registres de sortie (4x)).	100	50

Le paramètre `RetryLmt`

Définition

Ce paramètre correspond au nombre de tentatives d'envoi d'un message effectuées par le bloc `OUT_IN_MBUS` avant réception d'une réponse correcte de la part d'un équipement esclave (automate, modem, etc.).

Lorsque la réponse n'est pas complètement structurée dans le délai imparti, le bloc génère une erreur et un code d'erreur.

Le nombre de nouvelles tentatives doit être compris entre 0 et 32767.

Ce champ est utilisé conjointement avec `RespTout`.

Le paramètre DataBits

Définition

Les messages Modbus peuvent être envoyés en mode ASCII ou RTU.

Le mode ASCII utilise 7 bits de données tandis que le mode RTU en utilise 8. Pour l'envoi d'un message en caractères RTU, Databits doit être à 1.

La valeur doit être conforme à la configuration de la carte de communication.

NOTE : pour éviter tout problème en cas de modification de configuration, utilisez les constantes systèmes de type `%KW r.m.c.1.8` pour initialiser ce paramètre.

Le paramètre `RespTout`

Définition

Ce paramètre correspond au délai de temps d'attente du bloc `OUT_IN_MBUS` avant réception d'une réponse correcte de la part d'un équipement esclave (automate, modem, etc.).

Lorsque la réponse n'est pas complètement structurée dans le délai imparti, le bloc génère une erreur. Le système n'admet aucune réponse après ce délai.

La base de temps est de 100ms. Les valeurs valides sont comprises entre 0 (attente infinie) et 32767.

Le timeout commence à s'écouler après l'envoi du dernier caractère du message.

Le paramètre `MasterDataArea`

Définition

Pour une commande de lecture, la zone de données de l'automate maître est la destination des données renvoyées par l'esclave.

Pour une commande d'écriture, la zone de données de l'automate maître est la source des données.

Pour les codes commandes Modbus 1, 2, 5 et 15, le codage des bits s'effectue de la manière suivante :

- Les bits 1 à 16 sont stockés dans le premier élément du tableau d'INT passé en argument, le premier bit étant dans le bit de poids faible de l'élément.
- Les bits 17 à 32 sont stockés dans le second élément du tableau, le bit 17 correspondant au bit de poids faible de l'élément.
- Etc.

Ainsi, pour échanger 1000 bits, il convient de déclarer un tableau de 63 INT (1000/16 + 8).

NOTE : une zone de données de type entier `%MW` peut être utilisée directement dans le paramètre `MasterDataArea` (exemple : `%MW100:50` désigne un tableau de 50 entiers commençant à l'adresse 100).

Une suite d'éléments d'un tableau de bit de type `%M` doit au préalable être convertie et recopiée dans un tableau d'entiers (INT) selon le codage décrit ci-dessus.

Le paramètre Status

Définition

Ce paramètre affiche un code d'état généré par le bloc OUT_IN_MBUS.

Le tableau ci-dessous présente les différents codes d'état.

Code d'état	Description de l'état
1	Exception Modbus - Fonction incorrecte
2	Exception Modbus - Adresse de données incorrecte
3	Exception Modbus - Valeur de données incorrecte
4	Exception Modbus - Erreur abonné esclave
5	Exception Modbus - Confirmation
6	Exception Modbus - Abonné esclave occupé
7	Exception Modbus - Confirmation négative
8	Exception Modbus - Erreur de parité mémoire
104	La longueur de données ne peut être égale à zéro
108	Erreur non définie
113	Checksum LRC de l'automate esclave non valide
114	Checksum CRC de l'automate esclave non valide
115	Code fonction Modbus invalide
116	Timeout message de réponse Modbus
124	Etat interne indéfini
125	Mode diffusion interdit pour cette fonction Modbus
128	Réponse inattendue de la part de l'esclave Modbus
130	Mot de commande modifié en cours d'activité
131	Nombre de caractères incorrect
200	Adresse esclave hors limites
201	Erreur de communication avec le port série
202	Nombre binaire invalide
203	Quantité de données trop grande
204	Zone de données du maître trop petite
205	Le timeout doit être positif
206	Exception Modbus inconnue
207	Action annulée par l'utilisateur
208	RetryLmt doit être positif

15.3 Mise en œuvre du bloc de communication

OUT_IN_MBUS

Objet de ce sous-chapitre

Ce sous-chapitre décrit la mise en œuvre du bloc de communication OUT_IN_MBUS.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Configuration de la liaison série	186
Marche à suivre pour la programmation	189
Utilisation d'un modem	191

Configuration de la liaison série

Introduction

L'utilisation du bloc `OUT_IN_MBUS` nécessite une configuration préalable correcte de la liaison série.

On distingue les paramètres :

- liés à la transmission et configurés à partir de Unity Pro,
- liés à l'application et passés en argument à la fonction.

Rappel des paramètres de transmission

La liaison série des cartes est configurée à partir de Unity Pro. Ces paramètres sont :

- la vitesse de transmission,
- le délai inter-caractères,
- le bit de données,
- le bit de stop,
- la parité.

Pour permettre l'échange de données entre tous les équipements connectés sur le bus, le paramétrage de la liaison série doit être identique pour tous.

L'écran de configuration permettant de saisir les paramètres dépend de la configuration choisie en mode nominal.

Paramètres de transmission du mode Modbus

La figure ci-dessous représente un écran de configuration quand le mode nominal est en mode Modbus.

The screenshot shows a configuration window titled 'CARTE PCMCIA RS485 MP'. On the left, there is a sidebar with 'TSX SCP 114/1114' and 'Voie 1'. Below this, 'Fonction:' is set to 'LIAISON_BUS MODBUS' and 'Tâche:' is set to 'MAST'. The main 'Config' window contains the following settings:

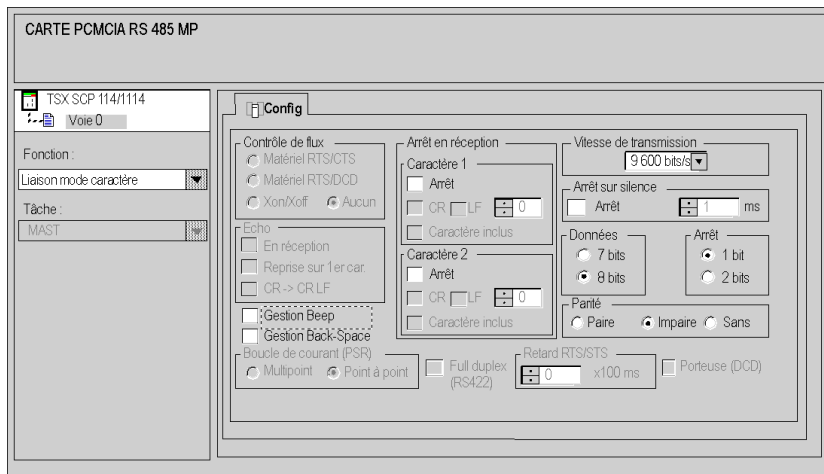
- Type:** Esclave
- Vitesse de transmission:** 9600 bits/s
- Délai inter-caractères:** Par défaut (4 ms), Stop
- Données:** ASCII (7 bits), RTU (8 bits), 1 bit, 2 bits
- Parité:** Paire, Impaire, Sans
- Relard RTS/CTS:** 0 x100 ms, Porteuse (DCD)
- Maitre:**
 - Nombre de répétitions: 0
 - Délai de réponse: 1 x 10 ms
- Esclave:** Numéro d'esclave: 1
- Boucle de courant (PSR):** Multipoint, Point à point

NOTE : la valeur du paramètre d'entrée Databit (*voir page 181*) du bloc OUT_IN_MBUS doit être conforme à la valeur cochée dans la fenêtre 'Données' de l'écran de configuration.

L'utilisation de OUT_IN_MBUS requiert une commutation dynamique en mode caractère. Dans ce mode, les conditions d'arrêt (sur caractères ou sur silence) sont désactivées et non modifiables par l'application. La fonction OUT_IN_CHAR n'est alors plus utilisable en mode réception (arrêt sur timeout), seules les fonctions INPUT_CHAR et INPUT_BYTES peuvent être utilisées en réception en précisant un nombre de caractères à recevoir.

Paramètres de transmission du mode Caractère

La figure ci-dessous représente un écran de configuration quand le mode nominal est en mode caractère.



NOTE : la valeur du paramètre d'entrée Databit (voir page 181) du bloc OUT_IN_MBUS doit être conforme à la valeur cochée dans la fenêtre 'Données' de l'écran de configuration.

Lorsque la configuration de la carte est le mode caractère :

- les conditions d'arrêt (sur caractères ou sur silence) configurables dans ce mode doivent être invalidées pour un fonctionnement correct de OUT_IN_MBUS. La fonction OUT_IN_CHAR n'est alors plus utilisable en mode réception (arrêt sur timeout), seules les fonctions INPUT_CHAR et INPUT_BYTES peuvent être utilisées en réception en précisant un nombre de caractères à recevoir,
- le délai inter-caractères n'est pas configurable. Il convient donc de s'assurer que la valeur de ce paramètre dans les équipements Modbus distants est compatible avec la configuration du mode caractère.

Paramètres liés à l'application

Deux paramètres liés à l'application sont transmis en tant qu'argument à la fonction OUT_IN_MBUS.

Ces paramètres sont :

- le nombre de répétitions (voir page 180),
- le délai de réponse (voir page 182).

Marche à suivre pour la programmation

Marche à suivre

Le tableau ci-dessous présente la marche à suivre pour programmer le bloc OUT_IN_MBUS :

Etape	Action	détails
1	Préparation du port de communication.	<ul style="list-style-type: none"> ● Si le port série n'est pas configuré en mode caractères, changez le mode Modbus du port en mode caractères en envoyant sur le port série la commande WRITE_CMD (voir page 190), ● dans le cas d'une transmission modem, envoyez la commande HAYES en utilisant le bloc PRINT_CHAR ou OUT_IN_CHAR afin de configurer le modem (voir page 191), ● dans le cas d'une transmission modem, utilisez la commande HAYES pour envoyer un message de numérotation au modem en utilisant le bloc PRINT_CHAR ou OUT_IN_CHAR. Le message de numérotation est utilisé pour envoyer un numéro de téléphone au modem (voir page 191).
2	Initialisation des paramètres.	Initialisez les paramètres d'entrées du bloc DFB. Il n'est pas utile de répéter cette opération à chaque cycle automate.
3	Appel du bloc OUT_IN_MBUS.	<ul style="list-style-type: none"> ● OUT_IN_MBUS doit être appelé à chaque cycle automate jusqu'à ce que le bit d'activité soit à zéro, ● dès que le bit d'activité est à zéro, forcez un bit dans la condition d'appel du bloc pour éviter un nouvel appel, ● vérifiez le bit d'erreur (en cas d'erreur, le mot de statut précise la cause d'erreur).
4	Réinitialisation du port de communication.	<ul style="list-style-type: none"> ● Dans le cas d'une transmission modem, envoyez la commande HAYES pour envoyer un message de déconnexion au modem (voir page 191) en utilisant le bloc PRINT_CHAR ou OUT_IN_CHAR. ● Si le port a été commuté en mode caractères (dans l'étape 1), retournez dans le mode d'origine du port série par la commande WRITE_CMD (voir page 190).

Ecrire les mots de commande sur un port de communication

Les étapes qui suivent doivent être exécutées pour envoyer un `WRITE_CMD` à un port de communication :

Etape	Action
1	Testez si aucune commande n'est en attente. <ul style="list-style-type: none">Avant d'exécutez un <code>WRITE_CMD</code>, testez si un échange est en cours en utilisant l'objet langage <code>%MWr.m.c.0</code>. Pour rafraîchir ce mot, il convient d'utiliser le bloc <code>READ_STS</code>.
2	Affectez le mot de commande <ul style="list-style-type: none">Vous devez ensuite modifier la valeur de l'objet langage commande pour réaliser la commande requise. Pour un lien Modbus, l'objet langage est le mot interne <code>%MWr.m.c.15</code>. Par exemple, pour commuter du mode Modbus au mode caractères, <code>%MWr.m.c.15</code> est mis à <code>16#4000</code> (<code>%MWr.m.c.15.14=1</code>). Note : un seul bit de commande doit être commuté de 0 à 1 avant de transmettre le <code>WRITE_CMD</code>.
3	Envoyez la commande <ul style="list-style-type: none">Finalement, un <code>WRITE_CMD</code> doit être exécuté pour acquitter la commande.

Dans l'exemple (*voir page 193*) qui suit, nous utilisons l'interface IODDT correspondante pour communiquer avec le canal du port série de communication.

Utilisation d'un modem

Description

Il est nécessaire de se familiariser avec trois commandes pour interfacier des modems téléphoniques à des automates. Ces commandes sont :

- initialiser le modem,
- numérotéer,
- déconnecter le modem.

Il est impératif d'envoyer au modem un message d'initialisation puis de numérotation avant de lui envoyer un message ASCII ou Modbus.

Lorsque la connexion a réussi entre les deux modems, vous pouvez envoyer un nombre illimité de messages ASCII ou de messages Modbus.

Quand tous les messages ont été envoyés, vous devez envoyer la chaîne de déconnexion au modem.

Initialiser le modem

Le message d'initialisation est un message ASCII comportant 512 caractères maximum, bien que 50 caractères suffisent généralement pour initialiser un modem.

Vous pouvez utiliser n'importe quelle commande Hayes AT comme composant de la chaîne d'initialisation.

Exemple : un message Hayes typique d'initialisation :

- AT&F&K0&Q0&D0V1X0Q0 <CR><LF>

NOTE : pour simplifier la programmation, vous pouvez initialiser le modem par un terminal (exemple : Windows hypertexte) et ne pas utiliser la fonction OUT_IN_CHAR. Une fois les paramètres chargés dans le modem, ils peuvent être sauvegardés en mémoire non volatile avec une commande AT, habituellement &W.

Numérotéer le modem

Le message de numérotation est utilisé pour envoyer le numéro de téléphone au modem.

Seules les commandes AT relatives à la numérotation doivent être incluses dans le message.

Exemple 1 : numérotéer par fréquence :

- AT DT 6800326 <CR><LF>

Exemple 2 : numérotéer par impulsion :

- AT DP 6800326 <CR><LF>

Exemple 3 : numéroté par fréquence avec attente de la tonalité :

- AT DT W,6800326 <CR><LF>

NOTE : la valeur du Timeout doit être grande car la connexion entre deux modems prend du temps (par exemple, mettez le timeout à 30000ms). Un code d'état (voir page 184) 116 est généré par le bloc OUT_IN_MBUS si la valeur est trop courte. Plusieurs essais peuvent être nécessaires avant de trouver le temps optimal.

Déconnecter le modem

Le message de déconnexion est utilisé pour déconnecter le modem.

Exemple 1 : message Hayes typique de déconnexion :

- +++AT H0 <CR><LF>

NOTE : la valeur du Timeout doit être grande car la déconnexion d'un modem prend du temps (par exemple, mettez le timeout à 30000ms). Un code d'état (voir page 184) 116 est généré par le bloc OUT_IN_MBUS si la valeur est trop courte. Plusieurs essais peuvent être nécessaires avant de trouver le temps optimal.

15.4 Exemple d'utilisation du bloc de communication OUT_IN_MBUS

Objet de ce sous-chapitre

Ce sous-chapitre présente un exemple d'utilisation du bloc de communication OUT_IN_MBUS.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

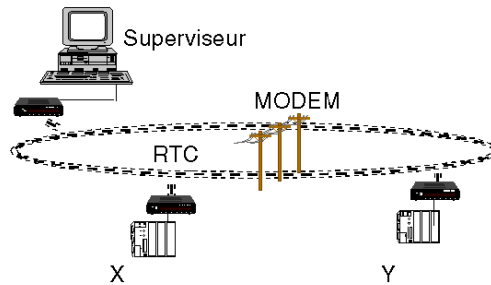
Sujet	Page
Description de l'exemple	194
Structure de la programmation	195
Déclaration des variables	197
Programmation	198

Description de l'exemple

Présentation

L'exemple choisi est une application de communication Modbus au travers de modems.

La figure ci-dessous illustre l'exemple :



Les équipements communiquent entre eux par modem. Le superviseur est maître Modbus alors que les automates X et Y sont esclaves.

Le but de l'exemple est d'écrire les valeurs d'une zone de données de l'automate X dans Y.

L'automate X doit écrire une zone de données de 41 entiers commençant à l'adresse %MW100 dans l'automate Y à partir de %MW100.

Pour ce faire, l'automate X doit devenir maître Modbus. L'adresse Modbus de l'automate X est 9, Y est 10.

Pour simplifier la programmation, les modems ont été initialisés avec les bons paramètres via un terminal de programmation. Ces paramètres sont stockés dans la mémoire non volatile par les commandes AT&W.

Structure de la programmation

Commentaires d'étapes

Le tableau ci-dessous récapitule les étapes de programmation de l'exemple :

Numéro d'étape	Description de l'étape
0	Etat initial de la fonction. Attente du passage à 1 du bit <code>Start_4</code> pour passer à l'étape 5.
5	Si aucune commande n'est en cours sur le port série, une commande est envoyée pour passer le port série du mode Modbus en mode caractère. Passage à l'étape 10.
10	Lecture du status du port série. <ul style="list-style-type: none"> ● S'il y a une erreur sur le port série alors <ul style="list-style-type: none"> ● <code>Error_4</code> est à -2, ● passage à l'étape 65. ● Si aucune erreur sur le port série, <ul style="list-style-type: none"> ● et mode caractère actif, alors passage à l'étape 15, ● et pas de mode caractère actif, alors test de l'état de passage en mode caractère sur 1000 cycles. Si, au bout des 1000 cycles, le mode n'a pas changé, alors <code>Error_4</code> est à -1, et passage à l'étape 65.
15	Envoi d'une commande de numérotation au modem par le bloc <code>PRINT_CHAR</code> et attente de la réponse via <code>INPUT_CHAR</code> . Passage à l'étape 20.
20	Si le résultat de <code>PRINT_CHAR</code> est bien concluant alors passage à l'étape 25 sinon passage à l'étape 65 avec <code>Error_4</code> à -3.
25	Si le résultat de <code>INPUT_CHAR</code> est bien concluant alors passage à l'étape 30 sinon passage à l'étape 65 avec <code>Error_4</code> à -4.
30	Si le modem répond alors passage à l'étape 35 sinon passage à l'étape 65 avec <code>Error_4</code> à -5.
35	Initialisation du paramétrage du bloc <code>OUT_IN_MBUS</code> . Passage en étape 40.
40	<ul style="list-style-type: none"> ● Appel du bloc <code>OUT_IN_MBUS</code>. ● Si le bit <code>Active_4</code> est à 0, <ul style="list-style-type: none"> ● et si le bit <code>Flag_Error_4</code> est à 0, alors passage à l'étape 45, ● et si le bit <code>Flag_Error_4</code> est à 1, alors passage à l'étape 45 avec <code>Error_4</code> à -6.
45	Envoi d'une commande de déconnexion au modem par le bloc <code>PRINT_CHAR</code> . Passage à l'étape 50.

Numéro d'étape	Description de l'étape
50	Si le résultat de <code>PRINT_CHAR</code> est concluant alors passage à l'étape 55 sinon passage à l'étape 65 avec <code>Error_4</code> à la valeur 1.
55	Si aucune commande n'est en cours sur le port série, une commande est envoyée pour passer le port série du mode caractère au mode Modbus. Passage à l'étape 60.
60	Lecture du status du port série. <ul style="list-style-type: none">● S'il y a une erreur sur le port série alors<ul style="list-style-type: none">● <code>Error_4</code> est à 3,● passage à l'étape 65.● Si aucune erreur sur le port série,<ul style="list-style-type: none">● et passage en mode Modbus, alors passage à l'étape 65,● et pas en mode Modbus, alors test de l'état de passage en mode Modbus sur 1000 cycles. Si, au bout des 1000 cycles, le mode n'a pas changé, alors <code>Error_4</code> est à 2, et passage à l'étape 65.
65	Retour à l'étape 0 (l'état initial de la fonction) et bit <code>Start_4</code> à 0.

Déclaration des variables

Présentation

Le tableau ci-dessous recense le détail des variables utilisées :

Variable	Type	Définition
Start_4	BOOL	Bit de démarrage de la fonction
Step_4	INT	Etape de la fonction
Error_4	INT	Code d'erreur de la fonction
MngtPrint_4	ARRAY[0..4] of INT	Tableau des paramètres de communication pour le bloc INPUT_CHAR
MngtInput_4	ARRAY[0..4] of INT	Tableau des paramètres de communication pour le bloc PRINT_CHAR
ReqString_4	STRING	Commande d'une chaîne de caractères au modem
AnsString_4	STRING	Réponse d'une chaîne de caractères du modem
Out_In_Mbus_4	OUT_IN_MBUS	Instance du bloc OUT_IN_MBUS
Adr_4	INT	Port de communication de l'automate esclave
MbusCmd_4[1]	INT	Code fonction Modbus
MbusCmd_4[2]	INT	Quantité de données à lire
MbusCmd_4[3]	INT	Adresse Modbus de l'automate esclave
MbusCmd_4[4]	INT	Début de la zone de données à lire de l'esclave
RetryLmt_4	INT	Nombre de tentatives
DataBits_4	INT	Mode de transmission (1 pour RTU et 0 pour ASCII)
RespTout_4	INT	Timeout
Retry_4	INT	Nombre de tentatives effectuées par le bloc
Active_4	BOOL	La valeur 1 indique que l'opération est en cours
Done_4	BOOL	La valeur 1 indique que l'opération a réussie
Flag_Error_4	BOOL	La valeur 1 indique qu'une erreur est survenue ou que l'opération courante est terminée
Status_4	INT	Code d'état généré par le bloc

Programmation

Programmation en langage ST

L'exemple est programmé en langage littéral structuré ST. La section dédiée est sous la tâche maître (MAST).

```
(* Function to write %MW100 to %MW140 in slave Y *)
(* ----- *)

CASE Step_4 OF

0:
    (* Initialisation *)
    IF (Start_4) THEN (* trigger flag *)
        Error_4 := 0;
        Step_4 := 5; (* next step *)
    END_IF;

5:
    (* Send command to switch serial port from modbus to
character mode *)

    READ_STS(Ioddt_Pcmcia_0_3_1); (* read serial port status *)
    IF (Ioddt_Pcmcia_0_3_1.EXCH_STS = 0) THEN (* no active
command *)
        Ioddt_Pcmcia_0_3_1.CONTROL := 16#00; (* reset control
word *)
        SET(Ioddt_Pcmcia_0_3_1.MB_TO_CHAR); (* set MB_TO_CHAR
command bit *)
        WRITE_CMD (Ioddt_Pcmcia_0_3_1); (* send command *)
        i := 0; (* initialize retry counter *)
        Step_4 := 10; (* next step *)
    END_IF;
```

```
10:
    (* Test result of switch command *)
    READ_STS(Ioddt_Pcmcia_0_3_1); (* read serial port status *)
    IF (Ioddt_Pcmcia_0_3_1.EXCH_STS = 0) THEN (* command
completed *)
        RESET(Ioddt_Pcmcia_0_3_1.MB_TO_CHAR); (* reset
MB_TO_CHAR command bit *)
        IF (Ioddt_Pcmcia_0_3_1.EXCH_RPT = 0) THEN (* no error *)
            IF (AND(Ioddt_Pcmcia_0_3_1.PROTOCOL, 16#0F) = 03)
THEN (* character mode OK *)
                Step_4 := 15; (* next step *)
            ELSE
                i := i + 1;
                IF (i > 1000) THEN
                    Error_4 := -1; (* error *)
                    Step_4 := 65; (* next step = end *)
                END_IF;
            END_IF;
        ELSE (* error in sending command to port *)
            Error_4 := -2; (* error *)
            Step_4 := 65; (* next step = end *)
        END_IF;
    END_IF;

15:
    (* Send dial command to modem *)

    Adr_4 := ADDR('0.3.1.SYS'); (* communication port *)
    MngtPrint_4[3] := 50; (* timeout *)
    MngtPrint_4[4] := 16; (* number of bytes to send *)
    ReqString_4 := 'ATDT0102030405$N'; (* dial message *)
```

```
    PRINT_CHAR(Adr_4, ReqString_4, MngtPrint_4);
    MngtInput_4[3] := 300; (* timeout *)
    MngtInput_4[4] := 0; (* number of bytes to send *)
    INPUT_CHAR(Adr_4, 0, 12, MngtPrint_4, AnsString_4); (* wait
modem reply *)
    Step_4 := 20; (* next step *)
```

20:

```
    (* Test PRINT_CHAR function result *)
    IF (NOT MngtPrint_4[1].1) THEN
        IF (MngtPrint_4[2] = 0) THEN
            Step_4 := 25; (* success : next step *)
        ELSE
            Error_4 := -3; (* error *)
            Step_4 := 65; (* next step = end *)
        END_IF;
    END_IF;
```

25:

```
    (* Test INPUT_CHAR function result *)
    IF (NOT MngtInput_4[1].1) THEN
        IF (MngtInput_4[2] = 0) THEN
            Step_4 := 30; (* success : next step *)
        ELSE
            Error_4 := -4; (* error *)
            Step_4 := 65; (* next step = end *)
        END_IF;
    END_IF;
```



```
30:
    (* Test Modem reply *)
    IF (AnsString_4 = 'CONNECT 9600') THEN
        Step_4 := 35; (* success : next step *)
    ELSE
        Error_4 := -5; (* error *)
        Step_4 := 65; (* next step = end *)
    END_IF;

35:
    (* Initialize OUT_IN_MBUS parameters *)
    MbusCmd_4[1] := 10; (* slave PLC address *)
    MbusCmd_4[2] := 16#06; (* Modbus function 16#06 *)
    MbusCmd_4[3] := 100; (* slave PLC area = %MW100 *)
    MbusCmd_4[4] := 41; (* quantity of data *)
    RetryLmt_4 := 2; (* number of retry *)
    DataBits_4 := %KW0.3.1.1.8; (* 1 = 8 bits -> RTU mode, 0
= 7 bits -> ASCII mode *)
    RespTout_4 := 300; (* timeout = 30s *)
    Flag_Error_4 := 0;
    Step_4 := 40; (* next step *)

40:
    (* Call OUT_IN_MBUS *)
    Out_In_Mbus_4 (Adr_4, MbusCmd_4, RetryLmt_4, DataBits_4,
RespTout_4, Abort_4,
                %MW100:41, Retry_4, Active_4, Done_4,
Flag_Error_4, Status_4);
```

```
IF (NOT Active_4) THEN (* request completed *)
  IF (NOT Flag_Error_4) THEN (* no error *)
    Step_4 := 45; (* next step *)
  ELSE (* error *)
    Error_4 := -6; (* error *)
    Step_4 := 45; (* next step *)
  END_IF;
END_IF;
```

45:

```
(* Hangup modem *)
MngtPrint_4[3] := 50; (* timeout *)
MngtPrint_4[4] := 9; (* number of bytes to send *)
ReqString_4 := '+++ATH0$N'; (* hangup message *)
PRINT_CHAR(Adr_4, ReqString_4, MngtPrint_4);
Step_4 := 50; (* next step *)
```

50:

```
(* Test PRINT_CHAR function result *)
IF (NOT MngtPrint_4[1].1) THEN
  IF (MngtPrint_4[2] = 0) THEN
    (* Success : next step *)
    Step_4 := 55;
  ELSE
    (* End on error *)
    Error_4 := 1;
    Step_4 := 65;
  END_IF;
END_IF;
```

```
55:
    (* Send command to switch serial port from Modbus to
character mode *)
    READ_STS(Ioddt_Pcmcia_0_3_1); (* read serial port status *)
    IF (Ioddt_Pcmcia_0_3_1.EXCH_STS = 0) THEN (* no active
command *)
        Ioddt_Pcmcia_0_3_1.CONTROL := 16#00; (* reset control
word *)
        SET(Ioddt_Pcmcia_0_3_1.CHAR_TO_MB); (* set MB_TO_CHAR
command bit *)
        WRITE_CMD (Ioddt_Pcmcia_0_3_1); (* send command *)
        i := 0; (* initialize retry counter *)
        Step_4 := 60; (* next step *)
    END_IF;
```

```
60:
    (* Test result of switch command *)
    READ_STS(Ioddt_Pcmcia_0_3_1); (* read serial port status *)
    IF (Ioddt_Pcmcia_0_3_1.EXCH_STS = 0) THEN (* command
completed *)
        RESET(Ioddt_Pcmcia_0_3_1.CHAR_TO_MB); (* reset
CHAR_TO_MB command bit *)
        IF (Ioddt_Pcmcia_0_3_1.EXCH_RPT = 0) THEN (* no error *)
            IF (AND(Ioddt_Pcmcia_0_3_1.PROTOCOL, 16#0F) = 07)
THEN (* Modbus mode OK *)
                Step_4 := 65; (* next step *)
            ELSE
                i := i + 1;
                IF (i > 1000) THEN
                    Error_4 := 2; (* error *)
                    Step_4 := 65; (* next step *)
                END_IF;
```

```
        END_IF;  
    ELSE (* error in sending command to port *)  
        Error_4 := 3; (* error *)  
        Step_4 := 65; (* next step *)  
    END_IF;  
END_IF;  
  
65:  
    (* End *)  
    Start_4 := 0; (* allow new demand *)  
    Step_4 := 0; (* goto waiting state *)  
  
END_CASE;
```

PRINT_CHAR : envoi de chaînes de caractères

16

Objet de ce chapitre

Ce chapitre décrit la fonction `PRINT_CHAR`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	206
Ecran de saisie assistée	210
Exemple d'envoi de chaînes de caractères via un réseau Fipway	212
Exemple d'envoi de chaînes de caractères via une liaison série de processeurs Modicon M340	214

Description

Description de la fonction

Pour les automates Premium, la fonction `PRINT_CHAR` est utilisée pour envoyer une chaîne de caractères de 4 Ko maximum (120 octets sur le port terminal) à transmettre sur une liaison mode caractère.

Pour les automates Modicon M340, la fonction `PRINT_CHAR` est utilisée pour envoyer une chaîne de caractères de 1 024 octets maximum.

La chaîne de caractères peut être contenue dans une variable statique (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) ou définie sous forme de valeur immédiate (série d'octets entre apostrophes, par exemple 'Message à envoyer').

Les chaînes de caractères peuvent contenir des caractères spéciaux et doivent commencer par le caractère `$`, suivi de la valeur hexadécimale du caractère à envoyer, par exemple `$0D`.

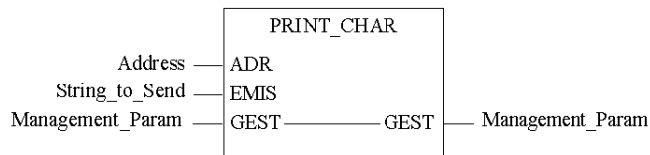
Certains caractères spéciaux (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) peuvent être utilisés, notamment :

`$R` = CR (retour chariot), `$L` = LF (retour à la ligne), `$N` = CR+LF.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

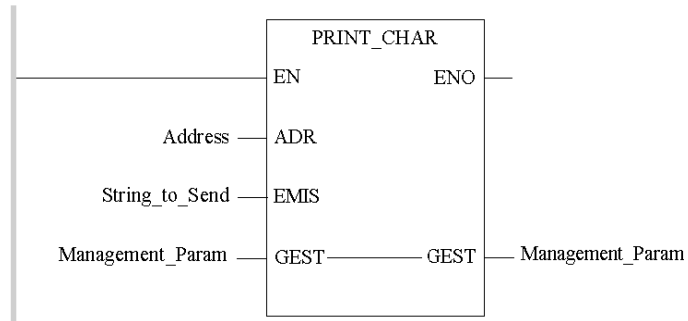
Représentation en FBD

Représentation :



Représentation en LD

Représentation :

**Représentation en IL**

Représentation :

Adresse LD

PRINT_CHAR String_to_Send, Management_Param

Représentation en ST

Représentation :

PRINT_CHAR(Address, String_to_Send, Management_Param);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Adresse	ARRAY [0...5] OF INT pour les automates Premium ARRAY [0...7] OF INT pour les automates Modicon M340	<p>Les instructions suivantes s'appliquent uniquement aux automates Premium :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDR. ● Seules les adresses se terminant par SYS sont acceptées (exemple : port terminal du processeur 0.0.0.SYS). <p>Les instructions suivantes s'appliquent uniquement aux automates Modicon M340 :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDM. ● La syntaxe de l'adresse est de type ADDM ("r.m.c.node"). Le champ Node est facultatif. Il peut être de type SYS ou vide (par exemple ADDM('0.0.0.SYS') est égal à ADDM('0.0.0').
String_to_Send	STRING	<p>Chaîne de caractères à envoyer. Intégrée dans une chaîne de caractères ou indiquée sous forme de valeur immédiate.</p> <p>Remarque : la chaîne de caractères doit être définie, même en l'absence de données à envoyer.</p>

Le tableau suivant décrit les paramètres d'entrée/de sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0...3] OF INT	<p>Table de mots utilisée pour gérer les échanges (voir page 33).</p> <p>Sur les automates Modicon M340, un nouveau bit d'annulation est disponible au mot de rang 1, dans la table de gestion des échanges. Ce bit d'annulation se situe au mot de rang 1. Il est constitué de 2 octets :</p> <ul style="list-style-type: none"> ● octet de poids fort : numéro d'échange ● octet de poids faible : bit d'activité (rang 0) et bit d'annulation (rang 1) <p>L'EF PRINT_CHAR peut être annulée par l'EF CANCEL ou en définissant sur 1 le bit d'annulation de la table de gestion (voir Modicon M340 avec Unity Pro, Liaison série, Manuel de l'utilisateur).</p>

Règles de programmation

Les caractères spéciaux doivent être précédés du caractère `§` dans la chaîne à transmettre. Les caractères `§` ne sont pas transmis par l'émetteur et ne doivent donc pas être comptés lors de l'initialisation du paramètre de longueur.

Les espaces entre deux caractères sont comptés dans un octet.

Par conséquent, dans l'exemple `PRINTING IN PROGRESS§L§R`, la longueur des données à transmettre est de 22 octets.

Sur les automates Premium, plusieurs cycles d'automate sont nécessaires pour envoyer une chaîne de caractères de plus de 240 octets (la chaîne est fragmentée). Il est donc important de s'assurer que les données de gestion n'ont pas été modifiées au cours du traitement de la fonction. Le système transmet la chaîne de manière cohérente sur plusieurs fragments, mais n'empêche pas la transmission d'une autre chaîne de caractères entre deux fragments.

Sur les automates Modicon M340, un cycle d'automate est nécessaire pour envoyer une chaîne de caractères de 1 024 octets maximum. Il est important de s'assurer que les données de gestion n'ont pas été modifiées au cours du traitement de la fonction.

Le port série de l'automate Modicon M340 est en full duplex. Par conséquent, une fonction `PRINT_CHAR` peut être envoyée même si une fonction `INPUT_CHAR` est en attente d'envoi.

NOTE : Pour envoyer des chaînes de caractères contenant des caractères de fin de chaîne (ZERO), vous devez :

- utiliser des chaînes affectées
- initialiser le dernier mot de la table de gestion des échanges avec le nombre de caractères à envoyer Si vous initialisez le mot avec la valeur 0, la chaîne envoyée s'arrête au premier caractère ZERO qu'elle rencontre. Si vous initialisez le mot avec une valeur, la longueur de la chaîne de caractères envoyée est égale à cette valeur.

Ecran de saisie assistée

Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que l'écran de saisie assistée n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptés dans les différents champs de l'écran.

Illustration

La capture suivante est un exemple d'écran de saisie assistée de la fonction :

Types possibles : tableau d'entiers constants; tableau d'entiers (n>=6)
Adresse: ADDR(".")

Adresse

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING) ,
- ARRAY [0..5] OF INT.

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

Chaîne à émettre

La chaîne à émettre est une variable de type STRING ou une valeur immédiate. Lorsqu'une variable de type STRING est sélectionnée, le champ de valeur disparaît.

Compte rendu

Le compte rendu est un tableau de 4 entiers pouvant être affecté ou non.

NOTE : veuillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

Exemple d'envoi de chaînes de caractères via un réseau Fipway

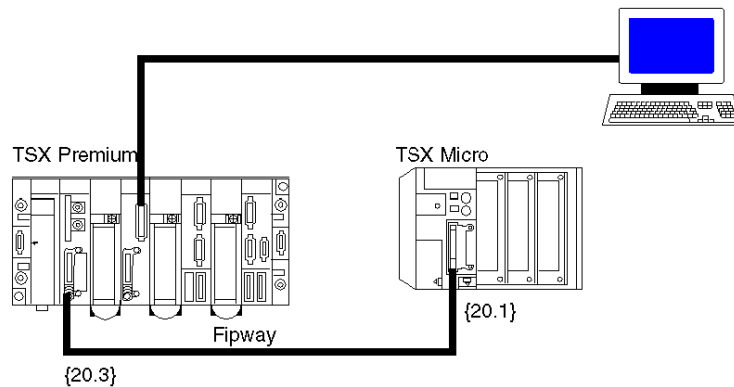
Présentation

Considérons que vous souhaitez envoyer une chaîne de caractères vers un terminal vidéo connecté à la liaison intégrée du module TSX SCY 21601 d'un automate, avec l'adresse réseau 20, station 3. Le module TSX SCY 21601 se trouve à l'emplacement 2 du rack de base.

La table de gestion est située dans `%MW110:4` et la chaîne à envoyer est située dans la variable `Str_1`.

Illustration

Les deux stations sont connectées via un réseau Fipway.



Emission

Programmation en ST :

```
IF RE(%I0.3.6) AND NOT %MW110.0 THEN
    PRINT_CHAR(ADDR('{20.3}0.2.0.SYS'), Str_1, %MW110:4);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.3}0.2.0.SYS')	<ul style="list-style-type: none">● {20.3}: réseau 20, station 3● 0: rack● 2: module● 0: voie 0● SYS : adresse système
Str_1	La chaîne de caractères à envoyer, la variable Str_1, est de type <code>STRING</code> .
%MW110:4	Table de gestion

NOTE : A chaque lancement de la fonction, initialisez le paramètre de longueur (dans l'exemple : %MW113) à l'aide de la valeur correspondant au nombre de caractères (en octets) à envoyer à Str_1).

Exemple d'envoi de chaînes de caractères via une liaison série de processeurs Modicon M340

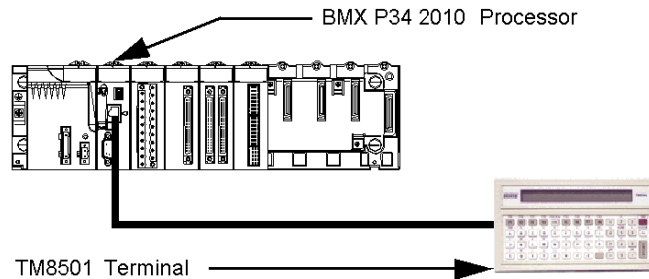
Présentation

Supposons que nous voulions envoyer une chaîne de caractères vers un terminal d'affichage/saisie de données compact raccordé au port série d'un processeur Modicon M340.

La table de gestion est située dans `%MW110:4` et la chaîne à envoyer est située dans la variable `caract`.

Illustration

Un automate Modicon M340 est relié à un terminal de saisie/d'affichage de données TM8501 :



Emission

Programmation en ST :

```
IF (%M16) THEN
    PRINT_CHAR(ADDM('0.0.0'), caract, gestion);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDM('0.0.0')	<ul style="list-style-type: none">● 0 : rack● 0 : module● 0 : voie 0● SYS : adresse système (optionnelle sur les automates Modicon M340)
caract	Chaîne de caractères à émettre, variable <code>caract</code> .
gestion	Table de gestion

NOTE : A chaque lancement de la fonction, initialisez le paramètre de longueur (par exemple : %MW113 si la table de gestion se situe entre %MW110 et %MW113) avec la valeur correspondant au nombre de caractères (en octets) à envoyer à `caract`.

RCV_TLG : réception de télégrammes

17

Objet de ce chapitre

Ce chapitre décrit la fonction RCV_TLG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	218
Exemple de réception d'un télégramme	221

Description

Description de la fonction

La fonction `RCV_TLG` permet de lire des données de type télégramme à partir d'une application distante.

Les données reçues ne peuvent excéder la longueur maximale de 16 octets. Contrairement aux autres fonctions de communication, cette fonction est traitée immédiatement (synchrone) : il n'y a donc aucun bit d'activité ou de paramètres timeout.

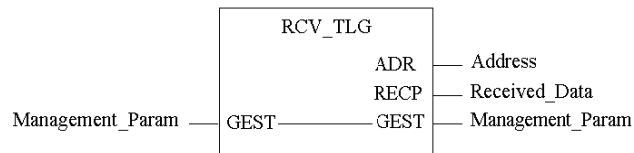
Par conséquent, la table d'entiers affectée aux paramètres de gestion utilise seulement deux mots au lieu de quatre (le nombre d'échanges et de timeout n'est pas requis).

NOTE : Cette fonction peut uniquement être utilisée sur Fipway dans le processeur et pour les stations de 0 à 15.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

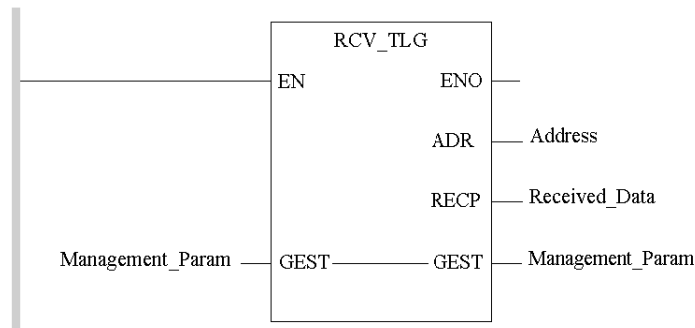
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Management_Param
```

```
RCV_TLG Address, Received_Data
```

Représentation en ST

Représentation :

```
RCV_TLG(Management_Param, Address, Received_Data);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée/de sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 1] OF INT	<p>Table de mots utilisée pour gérer les échanges. La table comporte deux mots : le mot de compte rendu et le mot indiquant la quantité de données reçues (en nombre d'octets). Le compte rendu de l'échange comporte :</p> <ul style="list-style-type: none"> ● le compte rendu d'opération (octet de poids fort du premier mot) ; ● le compte rendu de communication (octet de poids faible du premier mot). <p>Le compte rendu de communication prend l'une des valeurs suivantes :</p> <ul style="list-style-type: none"> ● 16#00 : échange correct ; ● 16#05 : paramètres de gestion incorrect ; ● 16#06 : paramètres spécifiques incorrects ; ● 16#09 : taille de tampon de réception insuffisante ; ● 16#0B : pas de ressources système ; ● 16#0D : pas de télégramme reçu ; ● 16#10 : module réseau absent ; ● 16#0F : service de télégrammes non configuré.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Address	INT	Adresse de l'entité émettant le télégramme à la fin de l'échange. <ul style="list-style-type: none">● L'octet de poids faible correspond au numéro de réseau en hexadécimal.● L'octet de poids fort correspond au numéro de station en hexadécimal.
Received_Data	ARRAY [n... m] OF INT	Tampon de réception. Table d'entiers contenant les données reçues. Cette table ne peut excéder la longueur maximale de 8 entiers (16 octets).

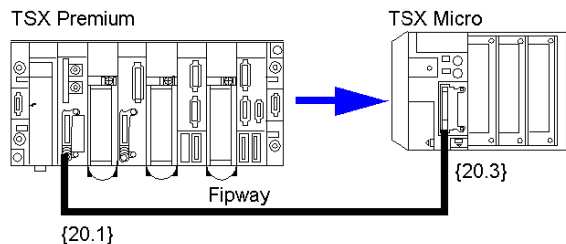
Exemple de réception d'un télégramme

Présentation

Considérons que vous souhaitez recevoir un télégramme à 8 mots (16 octets) d'une application distante sur un réseau Fipway.

Illustration

Les deux stations sont connectées via un réseau Fipway.



Programmation

Programmation en ST :

```
IF RE(%I0.3.11) THEN
  RCV_TLG(%MW200:2, %MW300,%MW310:8);
END_IF;
```

Paramètres de la requête :

Paramètre	Description
%MW200:2	Table de gestion
%MW300	Contient l'adresse de l'expéditeur à la fin de l'échange.
%MW310:8	Contenu du télégramme reçu

NOTE : Lorsqu'une fonction `RCV_TLG` est programmée dans une tâche événement, elle ne peut pas être utilisée dans une tâche MAST ou FAST.

Pour exécuter cette fonction de manière synchrone, il est nécessaire de tester le compte-rendu de l'opération immédiatement après la ligne de programme activant l'exécution de la fonction.

READ_ASYNC : lecture de données de façon asynchrone

18

Description

Description de la fonction

La fonction `READ_ASYNC` permet de lire 1 Koctet de données par le canal de messagerie asynchrone des coupleurs TSX ETY en mode TCP/IP.

Les données accessibles en lecture sont les suivantes :

- bits internes,
- mots internes.

La lecture asynchrone ne peut s'effectuer qu'entre deux stations d'un même segment de réseau Ethernet TCP/IP.

La fonction `READ_ASYNC` est émise à la fin de la tâche MAST seulement si celle-ci est configurée en mode périodique. Il est possible d'activer 8 fonctions simultanément.

Le principe de fonctionnement est identique à celui de la fonction `WRITE_ASYNC` (voir page 304).

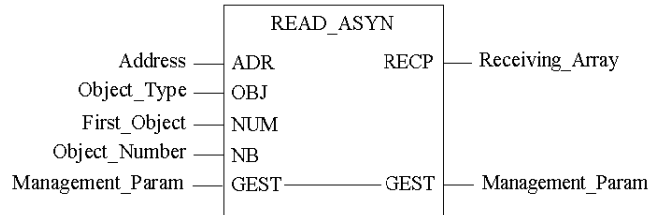
La taille des buffers d'émission et de réception est exprimée en mots. Elle est de 512 mots soit 1024 octets.

NOTE : La fonction serveur asynchrone supporte les protocoles UNI-TE V2.0 ou V1. La fonction `READ_ASYNC` utilise le protocole UNI-TE V2.0.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

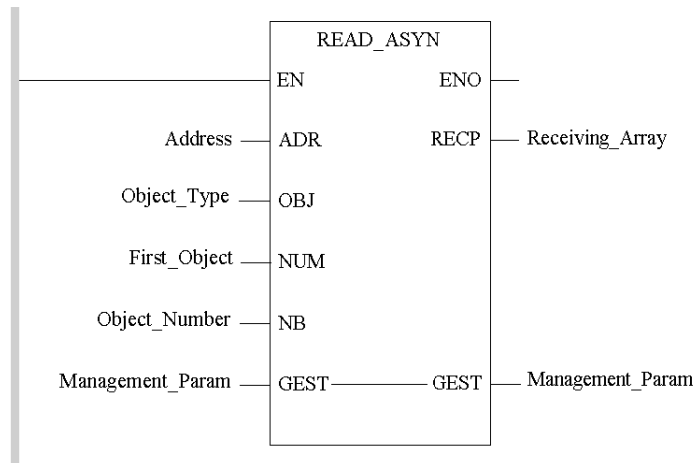
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

READ_ASYN Object_Type, First_Object, Object_Number,
Management_Param, Receiving_Array

Représentation en ST

Représentation :

READ_ASYN(Address, Object_Type, First_Object, Object_Number,
Management_Param, Receiving_Array);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0.. 5] OF INT	Adresse de l'entité destinataire de l'échange. Les adresses sont du type : ADDR(' {Réseau.Station}SYS.
Object_Type	STRING	Type des objets à lire : <ul style="list-style-type: none"> ● '%M' : bits internes, ● '%MW' : mots internes, ● '%S' : bits système, ● '%SW' : mots système.
First_Object	DINT	Indice du premier objet à lire.
Object_Number	INT	Nombre d'objets à lire.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 3] OF INT	Table de gestion de l'échange (<i>voir page 33</i>)

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Receiving_Array	ARRAY [n... m] OF INT	Tableau de mots contenant la valeur des objets lus.

READ_GDATA : lecture des données globales Modbus Plus

19

Description

Description de la fonction

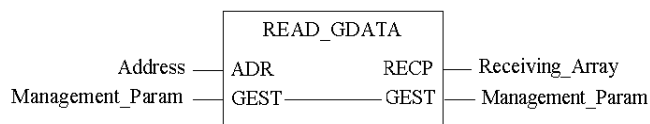
La fonction `READ_GDATA` permet de lire les données partagées appelées aussi **Global Data** sur un réseau **Modbus Plus**.

Les Global Data sont partagées entre 64 stations maximum d'un même réseau Modbus Plus. Chaque station peut écrire jusqu'à 32 entiers qui sont utilisables par toutes les stations du réseau. Réciproquement, chaque station peut lire les 32 (maximum) entiers de chaque station du réseau.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

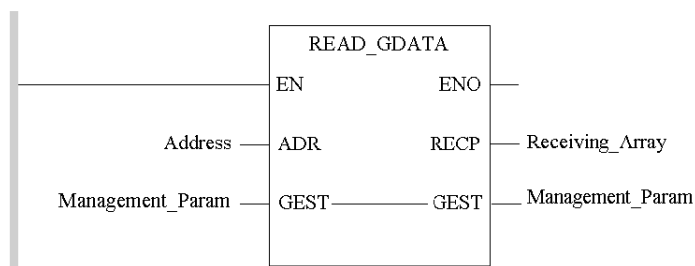
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

READ_GDATA Management_Param, Receiving_Array

Représentation en ST

Représentation :

READ_GDATA(Address, Management_Param, Receiving_Array);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0.. 5] OF INT	Adresse de l'entité destinataire de l'échange. Cette adresse est initialisée, avant l'échange, avec la valeur du noeud sur lequel est raccordé la station dont on veut connaître les Global Data. Exemple d'adresse : ADDR('0.0.1.10') correspond à la station connectée sur le noeud 10 du réseau. Les trois premiers chiffres, 0.0.1 correspondent à l'adresse de la voie, carte PCMCIA Modbus Plus du Premium.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 3] OF INT	Table de gestion de l'échange (<i>voir page 33</i>). Il n'est pas nécessaire d'initialiser le paramètre de longueur avant de lancer l'échange. A la fin de cet échange, ce paramètre de longueur (le quatrième mot) contient le nombre d'octets qui compose les données produites par la station spécifiée dans l'adresse.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Receiving_Array	ARRAY [n... m] OF INT	Tableau de mots contenant la valeur des objets lus.

READ_REG : lecture de registres

20

Introduction

Ce chapitre décrit le bloc READ_REG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	230
Types de données dérivés	233
Mode de fonctionnement	235
Description des paramètres	236

Description

Description de la fonction

En cas de front montant sur l'entrée `REQ` ce bloc fonction lit une zone de registre d'un esclave adressé via Modbus Plus, Ethernet TCP/IP ou Ethernet SY/MAX.

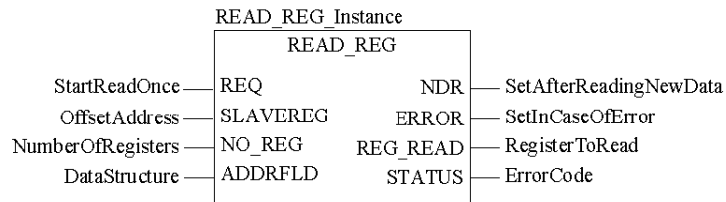
NOTE : Lorsque vous programmez une fonction `READ_REG`, il vous faut connaître les procédures de routage utilisées par votre réseau. Les structures des itinéraires de routage Modbus Plus sont décrites en détail dans le *Guide de planification et d'installation du réseau Modbus Plus*. Si un routage TCP/IP ou Ethernet SY/MAX est implémenté, vous devez obligatoirement utiliser des produits routeurs standard Ethernet IP. Vous trouverez une description détaillée du routage TCP/IP dans le *Guide utilisateur de Quantum avec la configuration Unity Pro TCP/IP*.

NOTE : Plusieurs exemplaires de ce bloc fonction peuvent être utilisés dans le programme. Il n'est cependant pas possible de procéder à une instantiation multiple de ces exemplaires.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

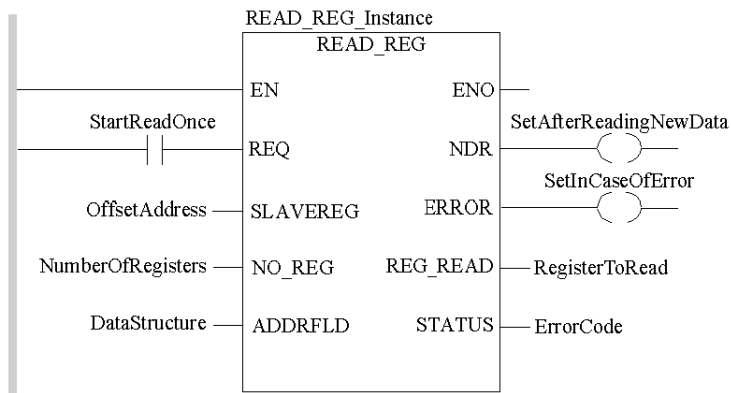
Représentation dans FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL READ_REG_Instance (REQ:=StartReadOnce,
  SLAVEREG:=OffsetAddress, NO_REG:=NumberOfRegisters,
  ADDRFLD:=DataStructure, NDR=>SetAfterReadingNewData,
  ERROR=>SetInCaseOfError, REG_READ=>RegisterToRead,
  STATUS=>ErrorCode)
```

Représentation en ST

Représentation :

```
READ_REG_Instance (REQ:=StartReadOnce,
  SLAVEREG:=OffsetAddress, NO_REG:=NumberOfRegisters,
  ADDRFLD:=DataStructure, NDR=>SetAfterReadingNewData,
  ERROR=>SetInCaseOfError, REG_READ=>RegisterToRead,
  STATUS=>ErrorCode) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Signification
REQ	BOOL	En cas de front montant sur l'entrée REQ ce bloc fonction lit une zone de registre d'un esclave adressé via Modbus Plus, Ethernet TCP/IP ou Ethernet SY/MAX.
SLAVEREG	DINT	Adresse de la première adresse %MW devant être lue dans l'esclave
NO_REG	INT	Nombre d'adresses à lire par l'esclave
ADDRFLD	WordArr5	Structure de données décrivant l'adresse Modbus Plus, l'adresse TCP/IP ou l'adresse SY/MAX-IP.

Description des paramètres de sortie :

Paramètres	Type de données	Signification
NDR	BOOL	Mis à "1" pendant un cycle après la lecture de nouvelles données
ERROR	BOOL	Mis à "1" pendant un cycle si une erreur apparaît
STATUS	WORD	Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie. Code d'erreur, voir : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137 ● Codes d'erreur spécifiques SY/MAX, page 142 ● Codes d'erreur Ethernet TCP/IP, page 144
REG_READ	ANY	Données à lire (Une structure de données doit être déclarée en tant que variable localisée pour les données à lire.)

Erreur d'exécution

Pour obtenir une liste de tous les codes et valeurs d'erreur du bloc, voir *Booléen étendu*, page 416.

Types de données dérivés

Description de WordArr5 sur Modbus Plus

Description de WordArr5 sur Modbus Plus :

Elément	Type de données	Description
WordArr5[1]	WORD	Octet de poids faible : Registre 1 de routage, sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau. Le dernier octet différent de zéro de l'itinéraire de routage est l'abonné cible. Octet de poids fort : Adresse de l'abonné source. <ul style="list-style-type: none"> ● Position de l'emplacement du module lors de l'utilisation du port Modbus Plus sur le module NOM. ● Si vous utilisez le port Modbus Plus de l'UC, cet octet doit être réglé sur 0 (pour tous les emplacements de l'UC).
WordArr5[2]	WORD	Registre 2 de routage
WordArr5[3]	WORD	Registre 3 de routage
WordArr5[4]	WORD	Registre 4 de routage
WordArr5[5]	WORD	Registre 5 de routage

Description de WordArr5 sur Ethernet TCP/IP

Description de WordArr5 sur Ethernet TCP/IP :

Elément	Type de données	Description
WordArr5[1]	WORD	Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET) Octet de poids fort : Emplacement du module NOE (16#FE si Ethernet est intégré à l'UC)
WordArr5[2]	WORD	Octet 4 (octet de poids fort) de l'adresse IP cible 32 bits
WordArr5[3]	WORD	Octet 3 de l'adresse IP cible 32 bits
WordArr5[4]	WORD	Octet 2 de l'adresse IP cible 32 bits
WordArr5[5]	WORD	Octet 1 (octet de poids faible) de l'adresse IP cible 32 bits

Description de wordArr5 sur Ethernet SY/MAX

Description de WordArr5 sur Ethernet SY/MAX :

Élément	Type de données	Description
WordArr5 [1]	WORD	Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET) Octet de poids fort : Emplacement du module NOE
WordArr5 [2]	WORD	Numéro de station cible (ou mettre FF en hexadécimal)
WordArr5 [3]	WORD	Terminaison (ou mettre FF en hexadécimal)
WordArr5 [4]	WORD	Réservé
WordArr5 [5]	WORD	Réservé

Mode de fonctionnement

Mode de fonctionnement des blocs READ_REG

Un grand nombre de blocs fonction READ_REG peut être programmé, mais seuls quatre opérations de lecture peuvent être actives en même temps, que celles-ci soient déclenchées par ce bloc fonction ou par d'autres (p. ex. MBP_MSTR, CREAD_REG) n'est pas significatif. Tous les blocs fonction utilisent la même session de transaction de données et nécessitent plusieurs cycles de programme pour achever une commande.

NOTE : Une communication TCP/IP entre un API Quantum (NOE 211 00) et un API Momentum (toutes les UC TCP/IP et tous les modules d'E/S TCP/IP) n'est possible que si **une** seule tâche de lecture ou d'écriture est effectuée dans chaque cycle d'API. Si plusieurs tâches par cycle sont envoyées, la communication est stoppée, sans qu'un message d'erreur soit généré dans le registre d'état.

L'information complète de routage est contenue dans la structure de données WordArr5 de l'entrée ADDRFLD. Le type du bloc fonction lié à cette entrée est fonction du réseau utilisé.

Veuillez utiliser pour :

- Modbus Plus le bloc fonction ModbusP_ADDR
- Ethernet TCP/IP le bloc fonction TCP_IP_ADDR
- Ethernet SY/MAX le bloc fonction SYMAX_IP_ADDR

NOTE : Vous pouvez également utiliser la structure de données WordArr5 avec des constantes.

Description des paramètres

REQ

Un front montant déclenche la transaction de lecture.

Ce paramètre peut être une adresse directe, une variable localisée, une variable non localisée ou une valeur littérale.

SLAVEREG

Début de la zone de l'esclave adressé dans laquelle les données sont lues. La zone source réside toujours dans la zone d'adresse %MW.

NOTE : Pour les esclaves d'un automate **non-Unity Pro** :

La zone source réside toujours dans la zone de registre 4x. SLAVEREG voit l'adresse source comme un décalage à l'intérieur de la zone 4x. Le "4" de tête doit être omis (p. ex. 59 (contenu de la variable ou valeur du littéral) = 40059).

Ce paramètre peut être une adresse directe, une variable localisée, une variable non localisée ou une valeur littérale.

NO_REG

Nombre d'adresses à lire dans l'équipement esclave adressé (1 ... 100).

Ce paramètre peut être une adresse directe, une variable localisée, une variable non localisée ou une valeur littérale.

NDR

Le passage à l'état ON sur un cycle programme signale la réception de nouvelles données prêtes au traitement.

Ce paramètre peut être une adresse directe, une variable localisée ou une variable non localisée.

ERROR

Le passage à l'état ON sur un cycle programme signifie la détection d'une nouvelle erreur.

Ce paramètre peut être une adresse directe, une variable localisée ou une variable non localisée.

REG_READ

Pour ce paramètre un `ARRAY` de la taille de la transmission demandée doit être spécifié (\geq `NO_REG`). Le nom de ce tableau est transmis comme paramètre. Si le tableau est défini sur une taille trop réduite, la quantité de données transmise sera limitée par la place proposée dans le tableau.

Ce paramètre doit être indiqué comme variable localisée.

STATUS

Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie.

Code d'erreur, voir :

- *Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137*
- *Codes d'erreur spécifiques SY/MAX, page 142*
- *Codes d'erreur Ethernet TCP/IP, page 144*

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

READ_VAR : lecture de variables

21

Objet de ce chapitre

Ce chapitre décrit la fonction de communication READ_VAR.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	240
Ecran de saisie assistée	245
Exemple d'utilisation sur un bus Uni-Telway	247
Exemple de lecture de bits	249
Exemple d'utilisation au sein d'un réseau	251
Exemple de lecture de mots via la liaison série des processeurs Modicon M340	253
Exemple de vérification d'exécution	255

Description

Description de la fonction

La fonction `READ_VAR` permet de lire la valeur d'un ou de plusieurs objets langage :

- bits internes,
- mots internes.

Les objets qui sont lus doivent toujours être consécutifs. Ils peuvent se trouver dans une UC distante ou sur un équipement connecté à une voie de communication.

Sur les automates Modicon M340, la fonction `READ_VAR` peut lire jusqu'à 2 000 bits consécutifs sur un équipement distant.

Sur les automates Premium, la fonction `READ_VAR` peut lire jusqu'à 1 000 bits consécutifs sur un équipement distant, quels que soient l'équipement et le protocole utilisés (Uni-Telway ou Modbus/Jbus).

NOTE : pour les automates Premium uniquement, la lecture de plus de 1 000 bits nécessite le recours à fonction `SEND_REQ`.

⚠ AVERTISSEMENT

INCOMPATIBILITE DES DONNEES ECHANGEES

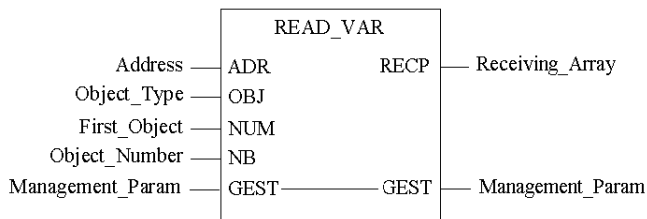
Les alignements de structure de données ne sont pas identiques pour les modules Premium/Quantum et M340, et il est donc nécessaire de vérifier que les données sont compatibles. Voir la page DDT : règles d'affectation (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) pour obtenir les règles d'alignement.

Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

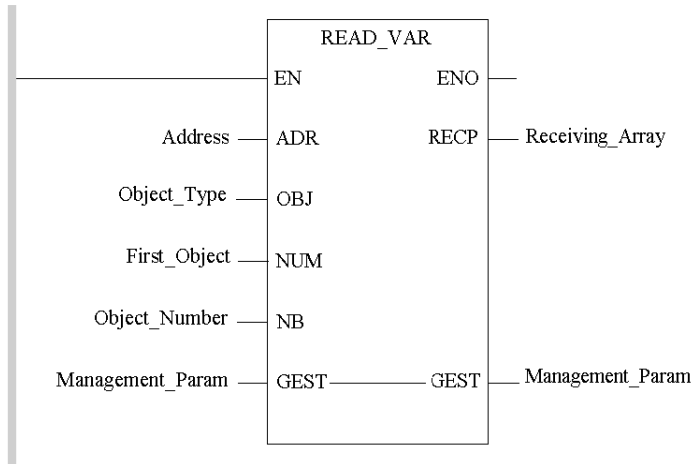
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

READ_VAR Object_Type, First_Object, Object_Number,
Management_Param, Receiving_Array

Représentation en ST

Représentation :

READ_VAR(Address, Object_Type, First_Object, Object_Number,
Management_Param, Receiving_Array);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètres	Type	Commentaire
Address	ARRAY [0 à 5] OF INT pour Premium ARRAY [0 à 7] OF INT pour Modicon M340	<p>Les instructions suivantes sont correctes uniquement pour les automates Premium :</p> <ul style="list-style-type: none"> ● L'adresse de la voie du mode caractère de réception du message est fournie par la fonction ADDR. ● Adresse de l'entité destinataire de l'échange. Les adresses suivantes sont interdites : <ul style="list-style-type: none"> ● {Réseau.Station}APP, ● {Réseau.Station}APP.num. <p>Les instructions suivantes sont correctes uniquement pour les automates Modicon M340 :</p> <ul style="list-style-type: none"> ● L'adresse de la voie du mode caractère de réception du message est fournie par la fonction ADDM. ● La syntaxe de l'adresse se présente de la façon suivante : ADDM ('r.m.c.node').
Object_Type	STRING	<p>Type des objets à lire pour les automates Premium :</p> <ul style="list-style-type: none"> ● '%M': bits internes, ● '%MW': mots internes, ● '%S': bits système, ● '%SW': mots système. ● '%I': bits d'entrée. ● '%IW': mots d'entrée. <p>Type des objets à lire pour les automates Modicon M340 :</p> <ul style="list-style-type: none"> ● '%M': bits internes, ● '%MW': mots internes,
First_Object	DINT	Indice du premier objet à lire.
Object_Number	INT	Nombre d'objets à lire.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0 à 3] OF INT	Table de gestion des échanges (<i>voir page 33</i>) Sur les automates Modicon M340, un nouveau bit d'annulation est disponible dans le mot de rang 1 de la table de gestion des échanges. Ce bit d'annulation est situé au niveau du mot de rang 1 qui se compose de deux octets : <ul style="list-style-type: none"> ● octet de poids fort : numéro d'échange, ● octet de poids faible : bit d'activité (rang 0) et bit d'annulation (rang 1). La fonction élémentaire (EF) <code>READ_VAR</code> peut être annulée par la fonction élémentaire <code>CANCEL</code> ou en réglant le bit d'annulation de la table de gestion sur 1 (<i>voir Modicon M340 avec Unity Pro, Liaison série, Manuel de l'utilisateur</i>).

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Receiving_Array	ARRAY [n à m] OF INT	Table de mots contenant la valeur des objets lus.

Règles d'utilisation

Les types d'objet doivent être saisis de manière cohérente. Les entrées doivent être écrites toutes en minuscules ou toutes en majuscules. Sinon, la fonction renvoie un rapport égal à 16#06 (paramètres spécifiques incorrects).

Pour les automates Micro, Premium ou Atrium recevant la fonction `READ_VAR`, l'accès en lecture seule aux bits internes présente les particularités suivantes :

- La valeur de forçage des bits est renvoyée dans la réponse,
- La lecture d'un bit comprend par conséquent deux octets de réponse :
 - Le premier contient la valeur des huit bits de celui demandé.
 - Le deuxième indique si ces bits ont été forcés.
- Pour lire l'un des huit derniers bits de la mémoire, il est nécessaire de lire les huit derniers. Sinon, la fonction renvoie un rapport d'opération 16#01. Cette dernière particularité s'applique également aux automates Modicon M340.

NOTE : il est nécessaire de prévoir de l'espace pour les octets d'indication du forçage dans la table de réception. Sinon, le code de défaut 16#03 est renvoyé dans le rapport.

NOTE : pour les automates Modicon M340, les bits forcés ne peuvent pas être utilisés par les fonctions `READ_VAR` et `WRITE_VAR`, car le protocole Modbus ne les prend pas en charge.

Transactions simultanées

Le tableau ci-après fournit les capacités de chaque voie de communication pour traiter simultanément les transactions en fonction des diverses configurations sur les automates Micro et Premium.

Configuration	Micro	TSX 57 10	TSX 57 20	TSX 57 23/30/40/45/55, PCX 57, PMX 57	TSX 57 46/56
Prise terminal maître Uni-Telway	4	4	4	4	8
Liaison PCMCIA ou SCY maître Uni-Telway	1	8	8	8	8
Prise terminal esclave client Uni-Telway	4	1	1	1	8
Liaison PCMCIA ou SCY esclave client Uni-Telway	1	1	1	1	1
Prise terminal esclave serveur Uni-Telway	4	4	4	4	4
Liaison PCMCIA ou SCY esclave serveur Uni-Telway	4	6	6	6	6
Prise terminal Modbus	4	-	-	-	-
Liaison PCMCIA ou SCY Modbus	4	8	8	8	8
Bloc terminal mode caractère	1	1	1	1	1
Liaison PCMCIA ou SCY mode caractère	4	8	8	8	8
PCMCIA CANopen	-	10	10	10	10
Liaison PCMCIA ou SCY Fipway	4	8	8	8	8
Modbus Plus	4	4	4	4	4
Ethernet	-	16	16	16	16
Ethernet intégré	-	-	-	-	64

Le tableau ci-après fournit les capacités de chaque voie de communication pour traiter simultanément les transactions en fonction des diverses configurations sur les automates Modicon M340.

Configuration,	BMX P34 1000	BMX P34 2000	BMX P34 2010/ 20102	BMX P34 2020	BMX P34 2030/20302
CANopen intégré	-	-	16	-	16
Ethernet intégré	-	-	-	16	16
Port série maître Modbus	8	16	16	16	-

Ecran de saisie assistée

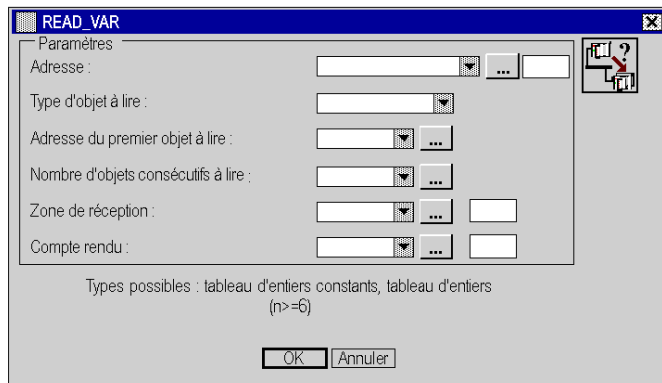
Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que cet écran n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptés dans les différents champs de l'écran.

Illustration

La capture suivante est un exemple d'écran de saisie assistée de la fonction :



Address

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING),
- ARRAY [0..5] OF INT

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

Type d'objet à lire

Pour les automates Premium, les choix possibles sont les suivants :

- '%M' pour lire des bits internes.
- '%MW' pour lire des mots internes.
- '%S' pour lire des bits système.
- '%SW' pour lire des mots système.
- '%I' pour lire des bits d'entrée.
- '%W' pour lire des mots d'entrée.

Pour les automates Modicon M340, les choix possibles sont les suivants :

- '%M' pour lire des bits internes.
- '%MW' pour lire des mots internes.

NOTE : faites votre choix parmi les solutions proposées dans le menu déroulant.

Adresse du premier objet à lire

Les objets possibles sont de type DINT :

- Variables
- Constantes
- Valeur immédiate

NOTE : lorsque vous saisissez une constante, un champ de saisie correspondant apparaît. Lorsque vous saisissez une variable, elle peut être affectée ou non.

Nombre d'objets consécutifs à lire

Les objets possibles sont de type INT :

- Variables
- Constantes
- Valeur immédiate

NOTE : lorsque vous saisissez une constante, un champ de saisie correspondant apparaît. Lorsque vous saisissez une variable, elle peut être affectée ou non.

Zone de réception

La zone de réception est une table d'entiers. La taille de cette table dépend du nombre d'objets à lire. Le tableau d'entiers peut être affecté ou non.

Compte rendu

Le compte rendu est un tableau de 4 entiers.

NOTE : veillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

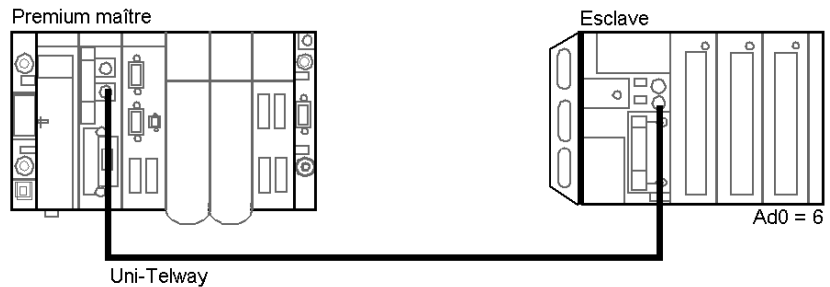
Exemple d'utilisation sur un bus Uni-Telway

Présentation

L'automate maître doit lire les mots internes %MW100 à %MW109 de la station située à l'adresse 6 sur un bus Uni-Telway. Les valeurs des mots lus sont triées par rapport au mot interne %MW10. Les paramètres de gestion sont stockés par rapport à %MW40.

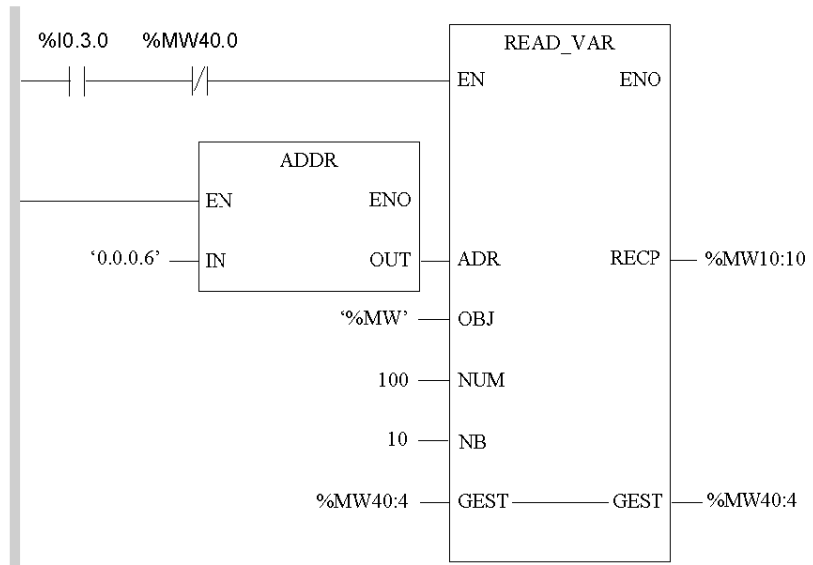
Illustration

Les deux stations sont reliées par un bus Uni-Telway.



Programmation

Programmation en LD :



Paramètres de la requête :

Paramètres	Description
'0.0.0.6'	<ul style="list-style-type: none"> ● 0 : rack ● 0 : module ● 0 : voie 0 ● 6 : adresse cible
'%MW'	Type d'objet (mot interne)
100	Adresse du premier objet
10	Nombre d'objets consécutifs
%MW40:4	Compte rendu
%MW10:10	Contenu de la réponse

Exemple de lecture de bits

Présentation

Sur les automates Modicon M340, les bits forcés ne sont pas accessibles par les fonctions `READ_VAR` et `WRITE_VAR`, car le protocole Modbus ne les prend pas en charge. Par conséquent, les instructions suivantes concernant les bits forcés ne peuvent pas être appliquées aux automates Modicon M340.

Les exemples suivants illustrent la fonction de communication `READ_VAR` pour la lecture de bits. La table de réception contient de manière consécutive les valeurs des bits, ainsi que l'indication de forçage.

Lecture de 32 bits internes

Dans ST, la syntaxe de la fonction de lecture des bits internes est la suivante :

```
READ_VAR (ADDR(`{20.1}0.5.1.3'),`%M', 0, 32, %MW100:4,
%M50:4);
```

La table de réception doit contenir 8 octets (4 mots), 4 octets pour la valeur et 4 octets pour l'indication de forçage.

	Mot	Octet 3	Octet 2	Octet 1	Octet 0
Valeur	%MW100	0000	0000	1100	1100
	%MW101	1111	1111	0000	1111
Forcer	%MW102	0000	0000	0101	0101
	%MW103	0000	0000	0000	1111

Le forçage peut être effectué pour chaque bit `%MW102` ou `%MW103` défini sur 1. La valeur de forçage est celle du bit lu correspondant.

Exemple :

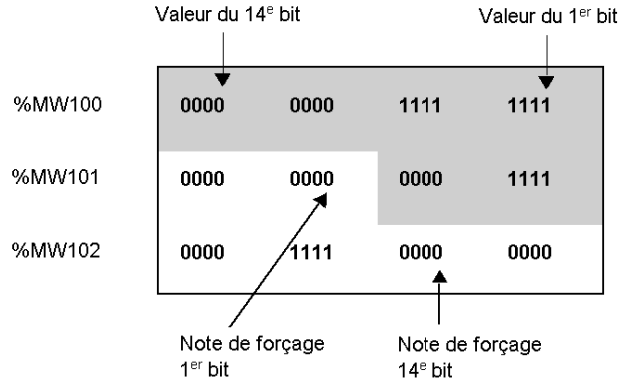
Valeur des 4 premiers bits (octet 0 du mot %MW100)	Forçage des 4 premiers bits (octet 0 du mot %MW102)	Description
0	1	Le bit est forcé sur 0
0	0	Le bit n'est pas forcé
1	1	Le bit est forcé sur 1
1	0	Le bit n'est pas forcé

Lecture de 18 bits internes

Dans ST, la syntaxe de la fonction de lecture des bits internes est la suivante :

```
READ_VAR (ADDR( '{20.1}0.5.1.3' ), '%M', 0, 18, %MW100:3,
%MW50:4);
```

La table de réception doit contenir 3 mots (ou 6 octets). Pour obtenir la valeur de 18 bits, 3 octets, plus 3 octets supplémentaires sont nécessaires pour contenir la valeur de forçage de 18 bits.



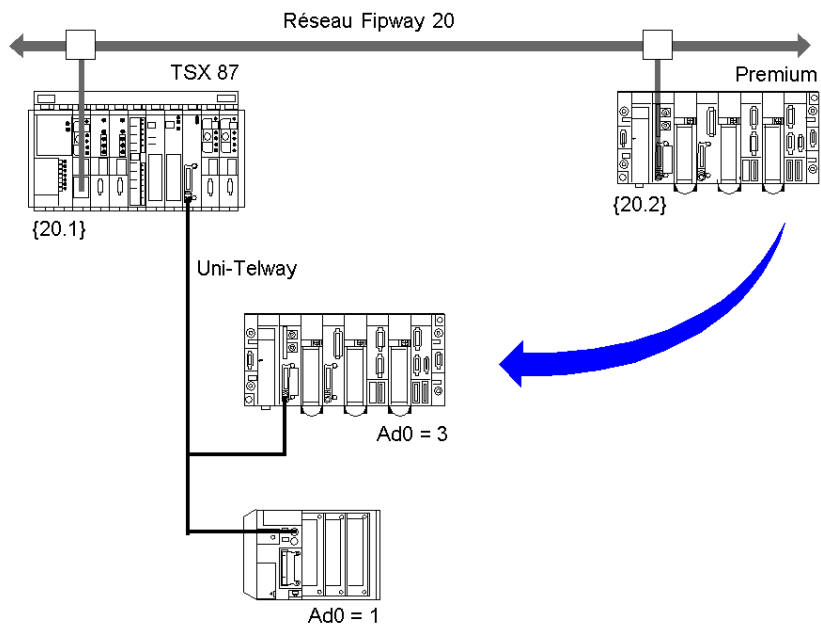
Exemple d'utilisation au sein d'un réseau

Présentation

La station 2 d'un réseau 20 doit lire une table de 5 mots $\%MW0$ à $\%MW4$ de l'esclave Uni-Telway, l'adresse réseau 20, la station 1, le module de communication TSX SCM 2116 dans l'emplacement 5, la voie 1 dans le module de communication, l'adresse de serveur Ad0 = 3.

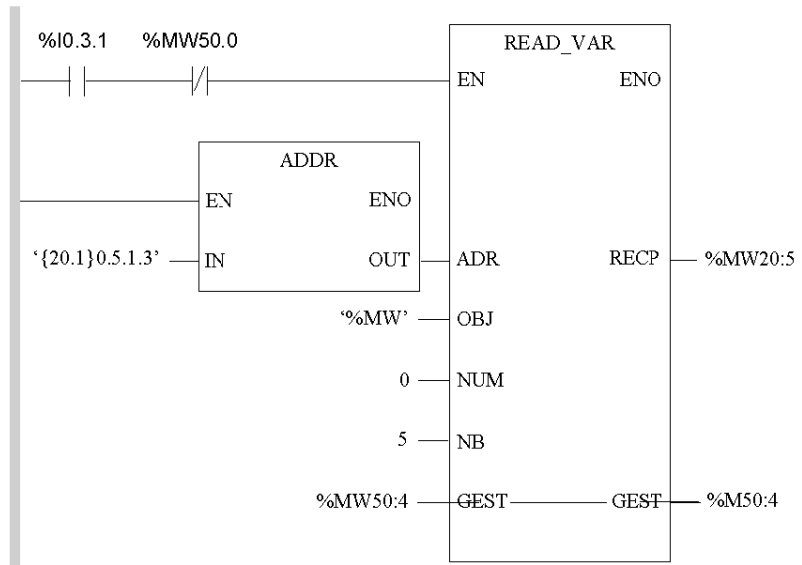
Illustration

Les deux stations sont connectées via un réseau Fipway.



Programmation

Programmation en LD :



Paramètres de la requête :

Paramètre	Description
ADDR('{20.1}0.5.1.3')	<ul style="list-style-type: none"> ● {20.1} : réseau 20, station 1 ● 0 : rack ● 5 : module ● 1 : voie 1 ● 3 : lecture de l'adresse esclave cible
%MW	Type d'objet (mot interne)
0	Adresse du premier objet
5	Nombre d'objets consécutifs
%MW50:4	Table de gestion
%MW20:5	Contenu de la réponse

Exemple de lecture de mots via la liaison série des processeurs Modicon M340

Présentation

Cet exemple utilise deux processeurs Modicon M340 qui communiquent via une liaison série Modbus. L'automate maître Modbus doit lire les mots internes %MW100 à %MW109 de l'automate esclave Modbus.

Description de l'exemple

Les valeurs des mots lus sont triées par rapport au mot interne %MW10.

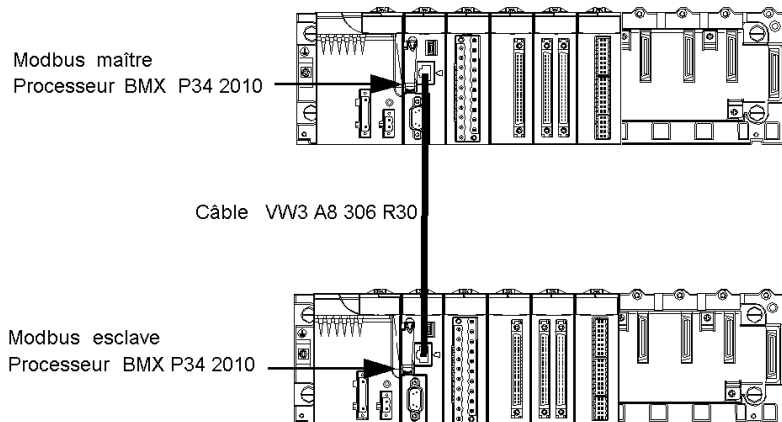
L'exemple ci-après utilise des variables non affectées et illustre la réception de données dans une table de 10 mots non affectée nommée `Tab_recp` (déclarée comme `ARRAY [0..9] OF INT`). Les paramètres de gestion se trouvent dans une table de 4 entiers nommée `Management_Parameter` (déclarée comme `ARRAY [0..3] OF INT`).

Dans cet exemple, le numéro de l'esclave Modbus est 7. Le paramètre ADDM d'entrée est donc '0.0.0.7'.

- 0: numéro de rack du processeur égal à 0
- 0: numéro de l'emplacement du processeur dans le rack égal à 0 (le numéro d'emplacement d'un processeur Modicon M340 est toujours 0)
- 0: numéro de voie égal à 0 (la liaison série d'un processeur Modicon M340 correspond toujours à la voie 0)
- 7: le numéro d'esclave configuré est 7

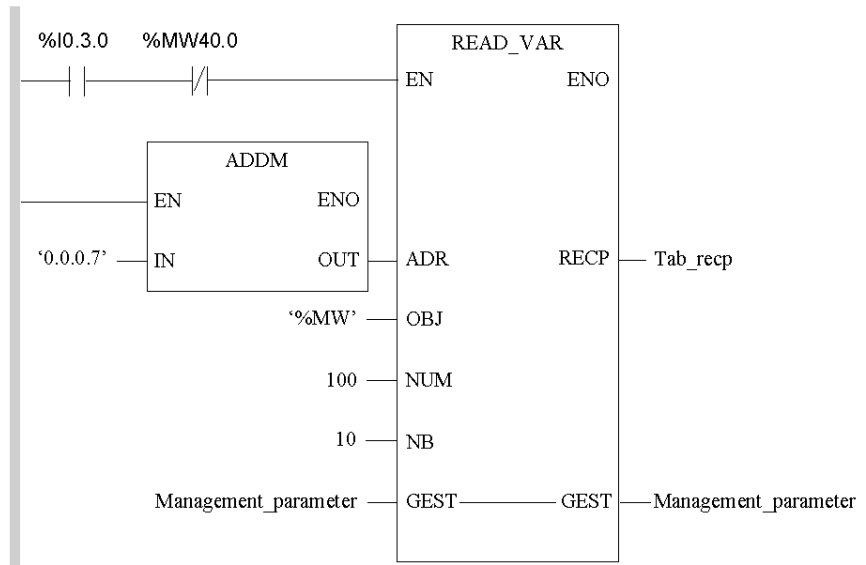
Illustration

Les deux processeurs Modicon 340 sont connectés via une liaison Modbus :



Programmation

Programmation en LD :



Les paramètres de requête sont les suivants :

Paramètres	Description
ADDM('0.0.0.7')	<ul style="list-style-type: none"> ● 0 : numéro du rack du processeur esclave ● 0 : numéro d'emplacement du processeur esclave ● 0 : numéro de voie (numéro de port série) ● 7 : numéro d'esclave configuré
%MW	Type d'objet (mot interne)
100	Adresse du premier objet à lire
10	Nombre d'objets consécutifs à lire
Tab_recp	Contenu de la réponse
Management_Parameter	Table de gestion

Exemple de vérification d'exécution

Présentation

L'exemple ci-après illustre la fonction READ_VAR avec la vérification des paramètres de gestion.

Programmation de la fonction

Programmation en ST :

```
IF NOT %M21 AND %I0.1.2 THEN
    %MW210:4 := 0;
    %MW212 := 50;
    READ_VAR(ADDR('0.3.1.7'), '%MW', 20, 1, %MW210:4, %MW1701:1);
    SET %M21;
END_IF;
```

- le bit d'entrée %I0.1.2 contrôle la fonction,
- le bit interne %M21 permet de tester l'activité de la fonction
- %MW210:4 := 0; définit la table de gestion sur la valeur 0
- MW212 := 50; initialise la valeur timeout sur 5 secondes

NOTE : La syntaxe

READ_VAR(ADDR('0.3.1.7'), '%MW', 20, 1, %MW210:4, %MW1701:1); doit être utilisée pour les automates Modicon M340, car la fonction ADDR n'est pas compatible avec ces automates.

Programmation de la vérification de l'échange

Programmation en ST :

```
IF %M21 AND NOT %M210.0 THEN
  INC %MW214;
  IF %MW211 = 0 THEN
    INC %MW215;
  ELSE
    SET %Q0.2.2;
    INC %MW216;
    %MW217 := %MW211;
  END_IF;
END_IF;
```

- %MW214 compte le nombre d'échanges,
- %MW215 compte le nombre d'échanges corrects,
- %MW216 compte le nombre d'échanges sources d'erreurs,
- %MW217 stocke le message d'erreur,
- %Q0.2.2 indique l'échec d'un échange.

SEND_EMAIL : messagerie électronique

22

Send Email

Description de la fonction

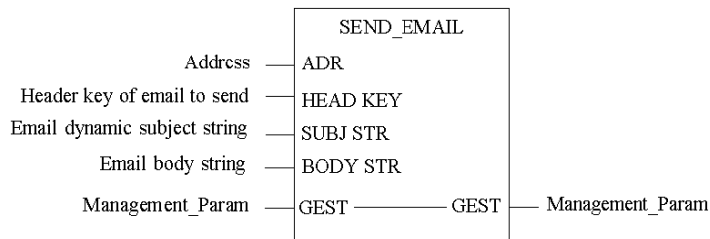
La fonction `SEND_EMAIL` permet d'envoyer un message électronique par l'intermédiaire du port Ethernet de l'UC d'un automate M340.

Le service `SEND_EMAIL` est limité au port Ethernet intégré. Il n'est pas pris en charge par l'intermédiaire des modules NOE.

NOTE : pour que le service `SEND_EMAIL` fonctionne correctement, l'adresse IP du serveur SMTP et les destinations utilisables des messages doivent tout d'abord être configurées dans le logiciel Unity Pro.

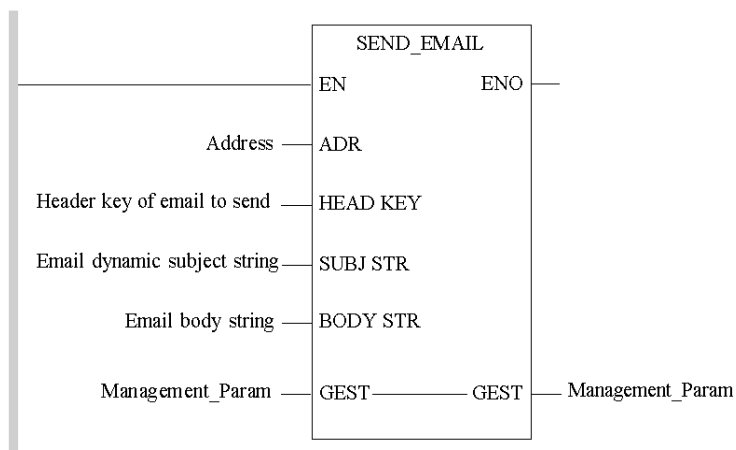
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en ST

```
SEND_EMAIL (ADR, HEAD_KEY, SUBJ_STR, BODY_STR, ManagWords);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètres	Type	Description
ADR	ARRAY [0.. 7] OF INT	Adresse ou résultat de la fonction ADDM. Utilise la fonction élémentaire ADDM pour composer ce champ. (voir page 43) Le seul moyen d'envoyer des messages électroniques est de passer par le port Ethernet de l'UC (0.0.3).
HEAD_KEY	INT	Correspond aux adresses de messagerie gérées avec le logiciel Unity Pro (seules les versions 1, 2 et 3 sont acceptées).
SUBJ_STR	STRING	Représente la partie dynamique de l'objet qui est ajoutée à la fin de la chaîne d'objet statique.
BODY_STR	STRING	Représente le corps du message électronique.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètres	Type	Commentaire
Management_Param	ARRAY [0... 3] OF INT	Table de gestion des échanges (<i>voir page 33</i>). Table de 4 mots utilisée pour définir les paramètres d'exécution. Le paramètre de longueur (4e mot de la table de gestion) est un paramètre de sortie. Ce mot est écrit par le système et indique la longueur totale du message électronique (en-tête + corps). La longueur totale maximale est de 1024 caractères.

Règles d'envoi de messages électroniques (Send Email)

Après le lancement d'une fonction élémentaire SEND_EMAIL, le bit d'activité est réglé jusqu'à ce que le message soit envoyé. Cependant, il n'y a pas d'accusé de réception. Si un timeout est programmé (le troisième mot de gestion est différent de 0), le message est annulé s'il n'a pas été envoyé dans ce délai. Dans ce cas, le deuxième mot de gestion reçoit une réponse négative **échange arrêté sur timeout** (0x01).

Caractéristiques d'exécution : le système peut gérer quatre requêtes d'envoi simultanées, à partir de quatre fonctions élémentaires. Si une cinquième fonction élémentaire tente d'envoyer un message, elle reçoit un message d'erreur **pas de ressources système du processeur** (0x0B) jusqu'à ce que l'une des ressources soit libre.

Exemple de service d'envoi de message électronique

```
IF (default_id = 0) THEN
(* PUMP IS OK *)
SEND_EMAIL(ADDM('0.0.3'), 1, 'Pump n° 3 is OK', ' ', Mng_send_email);
ELSE
(* PUMP IS FAULTY *)
str_default := INT_TO_STRING(default_id);
str_email_body := CONCAT_STR(' Default = ', str_default);
SEND_EMAIL(ADDM('0.0.3'), 1, 'Pump n° 3 is faulty', str_email_body,
Mng_send_email);
END_IF;
```

SEND_REQ : envoi de requêtes

23

Objet de ce chapitre

Ce chapitre décrit la fonction SEND_REQ.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	262
Liste de requêtes UNI-TE	266
Ecran de saisie assistée	273
Exemple d'envoi d'une requête UNI-TE	275
Modification des paramètres IP avec SEND_REQ (exemple)	277
Utilisation de la fonction SEND_REQ	278

Description

Description de la fonction

La fonction `SEND_REQ` sert à coder et à envoyer toutes les requêtes UNI-TE et Modbus/Jbus et à recevoir les réponses correspondantes.

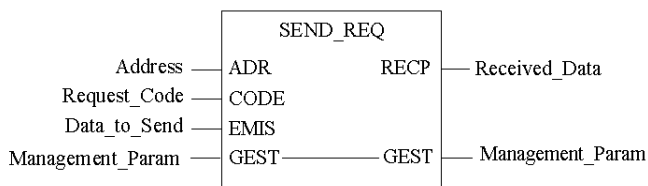
Les détails de codage des requêtes UNI-TE sont fournis dans le manuel de référence : Référence Communication TSX DR NET.

Quant au codage des requêtes Modbus/Jbus, il est expliqué dans le manuel TSX DG MDB.

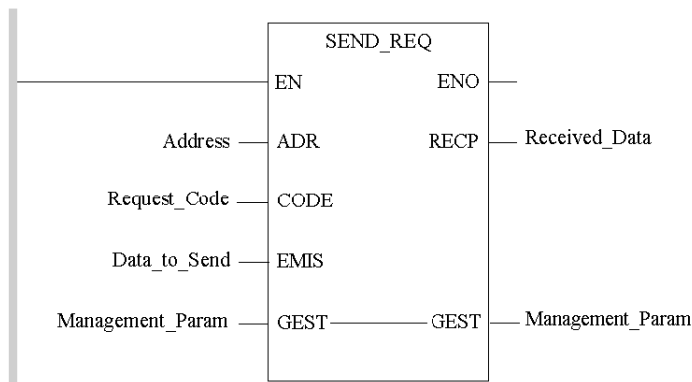
Les requêtes Modbus communes à tous les équipements Schneider sont présentées dans le manuel **Architecture et services de communication** (voir *Premium, Atrium et Quantum sous Unity Pro, Architectures et services de communication, Manuel de référence*).

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

Représentation en FBD



Représentation en LD



Représentation en IL

Adresse LD

```
SEND_REQ Request_Code, Data_to_Send, Management_Param,
Received_Data
```

Représentation en ST

```
SEND_REQ(Address, Request_Code, Data_to_Send,
Management_Param, Received_Data);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0.. 5] OF INT	Adresse de l'entité destinataire de l'échange. Ce type d'adresse dépend de la requête envoyée. Il est donc possible, par exemple, de diffuser la requête RUN (ALL, 0 pour le module TSX SCY 11601), alors qu'il est impossible d'envoyer une requête d'identification simultanément à plusieurs équipements.
Request_Code	INT	Requête à envoyer à l'équipement destinataire, également appelé serveur . Il peut s'agir de requêtes UNI-TE (<i>voir page 266</i>) ou de requêtes Modbus.
Data_to_Send	ARRAY [n... m] OF INT	Table d'entiers à envoyer à l'équipement destinataire de la requête. Cette table dépend de la requête envoyée. Sa longueur minimum doit être d'au moins 1 élément, même si la requête utilisée ne nécessite pas d'envoi de données (par exemple, requête d'identification). Remarque : Il est impératif d'affecter la longueur des données à envoyer (en octets) au quatrième mot de la table de gestion avant de lancer la fonction, pour que celle-ci soit exécutée correctement.

Le tableau suivant décrit les paramètres d'entrée/de sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 3] OF INT	Table de gestion d'échange (<i>voir page 33</i>)

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Received_Data	ARRAY [n... m] OF INT	Table d'entiers contenant les données renvoyées par l'équipement serveur destinataire de la requête. Même si certaines requêtes ne nécessitent pas de réponse (comme dans le cas d'une requête Run), il est toutefois nécessaire de réserver une table comportant au moins 1 entier chaque fois que la fonction SEND_REQ est utilisée. Remarque : La taille des données reçues est automatiquement écrite par le système dans le quatrième mot de la table de gestion (voir page 37). Remarque : Dans certains cas (lors de la lecture de tables de mots, par exemple), il est nécessaire de reséquencer les objets reçus à l'aide de la fonction ROR1_ARB (voir Unity Pro, Obsolète, Bibliothèque de blocs) (décalage d'un octet dans une table).

Transactions simultanées

Le tableau ci-après indique les capacités de chaque voie de communication pour traiter simultanément les transactions selon diverses configurations des automates Micro et Premium. (SEND_REQ n'est pas disponible sur les automates Modicon M340).

Configuration	Micro	TSX 57 10	TSX 57 20	TSX 57 23/30/40/45/55, PCX 57, PMX 57	TSX 57 46/56
Port terminal principal Uni-Telway	4	4	4	4	8
Liaison PCMCIA ou SCY principale Uni-Telway	1	8	8	8	8
Port terminal esclave de client Uni-Telway	4	1	1	1	8
Liaison PCMCIA ou SCY esclave de client Uni-Telway	1	1	1	1	1
Port terminal esclave de serveur Uni-Telway	4	4	4	4	4
Liaison PCMCIA ou SCY esclave de serveur Uni-Telway	4	6	6	6	6
Port terminal Modbus	4	-	-	-	-

Configuration	Micro	TSX 57 10	TSX 57 20	TSX 57 23/30/40/45/55, PCX 57, PMX 57	TSX 57 46/56
Liaison PCMCIA ou SCY Modbus	4	8	8	8	8
Bornier mode caractère	1	1	1	1	1
Liaison PCMCIA ou SCY mode caractère	4	8	8	8	8
PCMCIA CANopen	-	10	10	10	10
Liaison PCMCIA ou SCY Fipway	4	8	8	8	8
Modbus Plus	4	4	4	4	4
Ethernet	-	16	16	16	16
Ethernet intégré	-	-	-	-	64

Liste de requêtes UNI-TE

Vue d'ensemble

Le protocole UNI-TE est utilisé pour :

- identifier et diagnostiquer tous les équipements disposant d'un serveur UNI-TE ;
- fournir un ensemble de services donnant l'accès en lecture/écriture aux données de type ;
- télécharger les données d'un équipement à un autre ;
- protéger un serveur contre les connexions concurrentes en période critique.

Ces différents services sont disponibles grâce à la fonction `SEND_REQ`, moyennant le codage de la requête UNI-TE à envoyer.

NOTE : pour obtenir des informations détaillées et la liste des requêtes reconnues par chaque équipement, reportez-vous au manuel de référence TSX DR NET.

Les tableaux suivants fournissent une liste non exhaustive des requêtes reconnues par les équipements Premium.

Requêtes d'utilisation générale

Ces requêtes sont utilisées pour identifier et diagnostiquer tous les types d'équipement disposant d'un serveur UNI-TE.

Nom de requête	Code de requête	Code de compte rendu	Commentaire
IDENTIFICATION	16#0F	16#3F	Fournit des informations sur: <ul style="list-style-type: none"> ● la gamme de produits, ● le type d'application spécifique, ● le type de produit, ● la référence catalogue.
READ_CPU	16#4F	16#7F	Réalise un diagnostic système sur n'importe quel équipement.
PROTOCOL_VERSION	16#30	16#60	Permet d'adapter la version du protocole entre deux équipements communicants.
MIRROR	16#FA	16#FB	Teste le routage correct des données entre deux équipements communicants.

Modification dynamique des paramètres IP

Modification des paramètres IP

Nom de requête	Code de requête	Commentaire
REQUEST CODE	16#37	Fonction de codage d'une requête.
CHANGE IP PARAMETERS	16#13	Modifie la sous-fonction des paramètres IP.

Pour garantir que le module ETY est prêt à fonctionner, laissez l'automate fonctionner (en mode Run) pendant 15 secondes après le dernier arrêt avant de lancer la fonction SEND_REQ. Après l'émission par l'utilisateur de la commande CHANGE IP PARAMETERS et l'acceptation par le module ETY des nouveaux paramètres, le module ETY est réinitialisé et lance les opérations en fonction des nouveaux paramètres.

NOTE : les clients FDR qui utilisent le module ETY comme serveur doivent être redémarrés après toute modification de l'adresse IP du module, Faute de quoi ils ne pourraient pas mettre à jour leurs fichiers de paramètres sur le serveur FDR (le module ETY).

NOTE : les mots constants contiennent les paramètres de configuration d'origine et non ceux ayant été mis à jour après modification de l'adresse IP.

NOTE : vous pouvez consulter la nouvelle configuration (paramètres IP, masque de sous-réseau et adresse de la passerelle) sur l'écran de mise au point de Unity Pro ETY (voir *Premium et Atrium avec Unity Pro, Modules réseau Ethernet, Manuel utilisateur*). Vous pouvez également consulter l'adresse IP nouvellement affectée sur la page Web des statistiques du module Ethernet (voir *Premium et Atrium avec Unity Pro, Modules réseau Ethernet, Manuel utilisateur*). Notez toutefois que les données d'adresse IP, de masque de sous-réseau et de passerelle de cette page reflètent l'ancienne configuration.

Utilisation des paramètres en cas de modification de l'adresse IP

Ce tableau est basé sur un exemple utilisant l'adresse IP 139.158.10.7, le masque de sous-réseau 255.255.248.0 et l'adresse de passerelle 139.158.8.1.

Paramètre	Type	Valeur	Commentaire
ADDRESS	array [0...5] of INT	ADDR ('rack.slot.channel.SYS')	Exemple : ADDR (0.x.0.SYS') x = emplacement dans lequel le module ETY est installé.
REQUEST_CODE	INT	16#37	
Data_to_Send	array [0...8] of INT	Octet 1 : sous-fonction (13h)	octet de poids fort
		Octet 2 : sous-fonction (96h)	octet de poids faible
		Octet 3 : 0	La valeur est ignorée.
		Octet 4 : 0	La valeur est ignorée.
		Octet 5 : adresse IP 2 (158)	1-239 (octet de poids fort)
		Octet 6 : adresse IP 1 (139)	0-255 (octet de poids faible)
		Octet 7 : adresse IP 4 (7)	0-255 (octet de poids fort)
		Octet 8 : adresse IP 3 (10)	0-255 (octet de poids faible)
		Octet 9 : masque de sous-réseau 2 (255)	255
		Octet 10 : masque de sous-réseau 1 (255)	0-255
		Octet 11 : masque de sous-réseau 4 (0)	0-255
		Octet 12 : masque de sous-réseau 3 (248)	0-255
		Octet 13 : passerelle 2 (158)	1-239 (l'adresse de la passerelle doit se trouver sur le même sous-réseau que l'adresse IP.)
		octet 14 : passerelle 1 (139)	0-255
octet 15 : passerelle 4 (1)	0-255		
octet 16 : passerelle 3 (8)	0-255		
Manage_Param	<i>numéro du mot</i>	<i>octet de poids fort</i>	<i>octet de poids faible</i>
	1	compte rendu d'activité	00
	2	compte rendu d'opération (voir remarque)	compte rendu de communication (voir remarque)
	3	temporisation (ms)	
	4	longueur : 18 (INT)	
NOTE : le tableau suivant fournit des informations détaillées sur les codes d'adresse IP (corrects et incorrects).			

Reportez-vous à l'exemple de la section Modification des paramètres IP avec SEND_REQ (voir page 277).

Modification des codes d'adresse IP

Compte rendu d'opération	Compte rendu de communication	Signification
<i>code correct</i>		
FE (hex)	00 (hex)	(SEND_REQ) a permis de modifier correctement l'adresse IP.
<i>codes d'erreur</i>		
01 (hex)	FF (hex)	valeur de code de requête incorrecte (par exemple, différente de 16#37)
00 (hex)	03 (hex)	SEND_REQ envoyée à l'adresse IP du module ETY au lieu d'être envoyée sur l'embase
00 (hex)	07 (hex)	mappage d'adresse vers le module ETY incorrect
16 (hex)	FF (hex)	adresse IP incorrecte
17 (hex)	FF (hex)	sous-réseau incorrect
18 (hex)	FF (hex)	adresse de passerelle incorrecte
19 (hex)	FF (hex)	adresse réseau incorrecte
1A (hex)	FF (hex)	adresse IP du module ETY déjà réglée pour envoyer une requête (SEND_REQ)
FD (hex)	00 (hex)	(SEND_REQ) n'a pas permis de modifier correctement l'adresse IP.

Utilisation des paramètres de commande de réinitialisation de module

L'opération de réinitialisation de module force le module Premium ETY et le module de communication ETY PORT à amorcer un cycle de réinitialisation de leur environnement de travail. Pour programmer un bloc fonction SEND_REQ qui exécute cette commande de réinitialisation, utilisez le code fonction 37 et le code sous-fonction 10.

Paramètre	Type	Valeur	Commentaire
ADDRESS	array [0...5] of INT	ADDR ('rack.slot.channel.SYS')	Exemple : ADDR (0.x.0.SYS) x = emplacement dans lequel le module ETY est installé.
REQUEST_CODE	INT	16#37	
Data_to_send	array [0...1] of INT	Octet 1 : sous-fonction (10h)	octet de poids fort
		Octet 2 : sous-fonction (96h)	octet de poids faible
		Octets 3, 4 : 0	réservés
Manage_Param	numéro du mot	octet de poids fort	octet de poids faible
	1	compte rendu d'activité	00
	2	compte rendu d'opération (voir remarque)	compte rendu de communication (voir remarque)
	3	temporisation (ms)	
	4	longueur : 4 (INT) (dans cet exemple)	
	NOTE : le tableau suivant fournit des informations détaillées sur les codes de réinitialisation de module (corrects et incorrects).		

Codes de commande de réinitialisation de module

Compte rendu d'opération	Compte rendu de communication	Signification
<i>code correct</i>		
FE (hex)	00 (hex)	SEND_REQ a correctement réinitialisé le module
<i>codes d'erreur</i>		
01 (hex)	FF (hex)	valeur de code de requête incorrecte (par exemple, différente de 16#37)
00 (hex)	07 (hex)	mappage d'adresse vers le module ETY incorrect
FD (hex)	00 (hex)	SEND_REQ n'a pas réinitialisé le module

Accès aux objets

Ces requêtes fournissent un ensemble de services qui permettent l'accès en lecture/écriture aux données internes de type bit et mot, aux données système de type bit et mot, aux données de virgule flottante, aux données constantes et aux données SFC.

Nom de requête	Code de requête	Code de compte rendu	Commentaire
READ_OBJECT	16#36	16#66	Permet de lire un ou plusieurs objets consécutifs du même type.
WRITE_OBJECT	16#37	16#FE	Permet d'écrire un ou plusieurs objets consécutifs du même type.
READ_INTERNAL_BIT	16#00	16#30	Permet de lire la valeur d'un bit interne.
WRITE_INTERNAL_BIT	16#10	16#FE	Permet d'écrire la valeur d'un bit interne.
READ_INTERNAL_WORD	16#04	16#34	Permet de lire la valeur d'un mot interne.
WRITE_INTERNAL_WORD	16#14	16#FE	Permet d'écrire la valeur d'un mot interne.

Gestion des modes de marche

Ces requêtes fournissent un ensemble de services servant à gérer les modes de marche d'un processeur.

Nom de requête	Code de requête	Code de compte rendu	Commentaire
RUN	16#24	16#FE	Permet de lancer l'exécution des tâches d'un processeur.
STOP	16#25	16#FE	Permet d'arrêter l'exécution des tâches d'un processeur.
INIT	16#33	16#63	Permet de déclencher un redémarrage à chaud ou à froid.

Gestion des réservations

Ces requêtes fournissent un mécanisme de réservation servant à protéger un serveur contre des connexions concurrentes en période critique.

Nom de requête	Code de requête	Code de compte rendu	Commentaire
RESERVE	16#1D	16#FE	Permet à un client de réserver tout ou partie des fonctions d'un serveur.
RELEASE	16#1E	16#FE	Permet à un client de libérer le serveur réservé.
I_AM_ALIVE	16#2D	16#FE	Permet de maintenir la réservation.

Ecran de saisie assistée

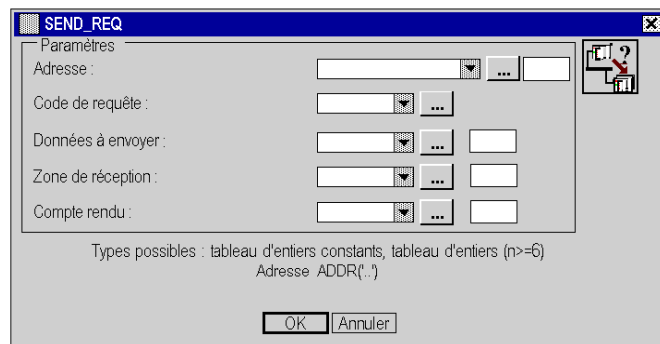
Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que l'écran de saisie assistée n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptés dans les différents champs de l'écran.

Illustration

La capture suivante est un exemple d'écran de saisie assistée de la fonction :



Adresse

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING),
- ARRAY [0..5] OF INT.

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

Code de requête

Les objets possibles sont de type INT :

- des variables,
- des constantes
- des valeurs immédiates.

NOTE : lorsque vous saisissez une constante, un champ de saisie correspondant apparaît. Lorsque vous saisissez une variable, elle peut être affectée ou non.

Données à envoyer

Les données à envoyer sont stockées sous la forme d'un tableau d'entiers. Ce tableau peut être affecté ou non.

Zone de réception

La zone de réception est un tableau d'entiers. Celui-ci peut être affecté ou non et sa taille dépend du code de requête utilisé.

Compte rendu

Le compte rendu est un tableau de 4 entiers.

NOTE : veillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

Exemple d'envoi d'une requête UNI-TE

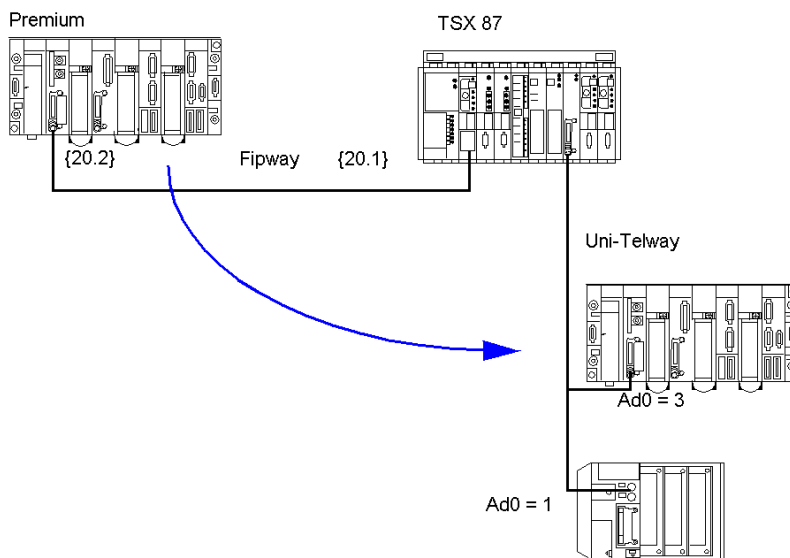
Présentation

La station 2 du réseau 20 doit envoyer une requête d'identification à l'équipement portant l'adresse Ad0=3 sur le bus Uni-Telway de la station 1 du même réseau. La requête d'identification porte le code décimal 15 (ou 16#0F).

La table de gestion se trouve à l'adresse %MW10:4.

Illustration

Les deux stations sont connectées via un réseau Fipway :



Programmation

Programmation sous ST :

```
IF RE(%I0.3.2) AND NOT %MW10.0 THEN
    SEND_REQ(ADDR(' {20.1}0.5.1.3'),15,%MW0:1,
    %MW10:4,%MW100:24);
END_IF;
```

Paramètres de la requête :

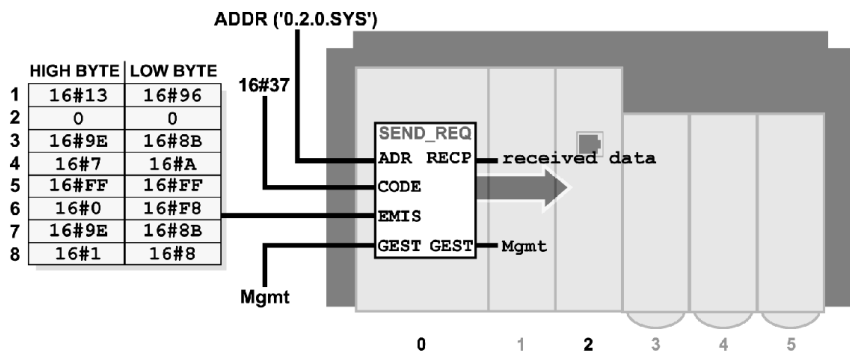
Paramètres	Description
ADDR('{20.1}0.5.1.3')	<ul style="list-style-type: none">● {20.1}: réseau 20, station 1● 0: rack● 5: module● 1: voie 1● 3: Adresse cible
15	Requête 15 (ou 16#0F si le codage est hexadécimal)
%MW0:1	Données envoyées (par exemple : aucune donnée à envoyer)
%MW10:4	Table de gestion
%MW100:24	Contenu de la réponse (réception de 24 mots)

NOTE : A chaque lancement de la fonction, initialisez le paramètre de longueur (dans l'exemple : %MW13 = 0).

Modification des paramètres IP avec SEND_REQ (exemple)

Illustration

Le graphique ci-dessous vous indique comment régler les paramètres IP du module ETY dans l'emplacement 2 avec le bloc SEND_REQ :



Remarque :

- ADR : indique la position du module ETY dans l'emplacement 2.
- CODE : indique la valeur de REQUEST_CODE.
- EMIS : contient les paramètres IP dans Data_to_Send :
 - Adresse (139.158.10.7)
 - Masque de sous-réseau (255.255.248.0)
 - Passerelle (139.158.8.1)
- GEST : indique Management_Param (paramètres de gestion). Vous devez attribuer une durée au troisième mot de Management_Param. Le quatrième mot doit avoir la valeur INT 18.
- RECP : ce paramètre requiert une valeur INT minimum de 1, même lorsque aucun message de réponse n'est renvoyé, comme dans le cas d'une requête de modification IP.

Utilisation de la fonction SEND_REQ

Présentation

La fonction SEND_REQ sert à coder et à envoyer toutes les requêtes UNI-TE et Modbus/Jbus et à recevoir les réponses correspondantes.

Dans certains cas (lors de la lecture de tables de mots, par exemple), il est nécessaire de reséquencer les objets reçus à l'aide de la fonction ROR1_ARB (décalage d'un octet dans une table). .

Exemple

Objets à lire :

16#0201
16#0403
16#0605
16#0807
16#0A09

Table de réception après exécution d'une fonction SEND_REQ (lecture d'objet) :

%MW100=16#0107
%MW101=16#0302
%MW102=16#0504
%MW103=16#0706
%MW104=16#0908
%MW105=16#000A

Table de réception après ROR1_ARB(%MW100:6) :

%MW100=16#0201
%MW101=16#0403
%MW102=16#0605
%MW103=16#0807
%MW104=16#0A09
%MW105=16#0700

SEND_TLG : envoi de télégrammes

24

Objet de ce chapitre

Ce chapitre décrit la fonction `SEND_TLG`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	280
Exemple d'envoi d'un télégramme	283

Description

Description de la fonction

La fonction `SEND_TLG` permet d'envoyer des données de type télégramme à une application distante.

Les données à envoyer doivent avoir une longueur maximum de 16 octets. Contrairement aux autres fonctions de communication, cette fonction est traitée immédiatement (synchrone) : il n'y a donc aucun bit d'activité ni paramètre de timeout.

Par conséquent, la table d'entiers affectée aux paramètres de gestion n'utilise que deux mots au lieu de quatre (le nombre d'échanges et de timeouts n'est pas requis).

⚠ AVERTISSEMENT

FUNCTION NON OPERATIONNELLE - CONFIGURATION NON VALIDE

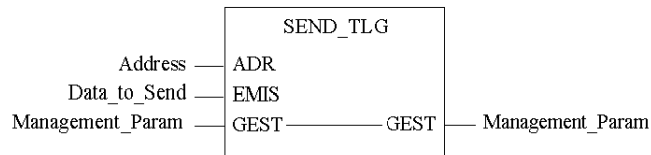
Pour utiliser la fonction `SEND_TLG` sur Fipway : installez la carte `TSX FPP20` uniquement sur les stations 0 à 15.

Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

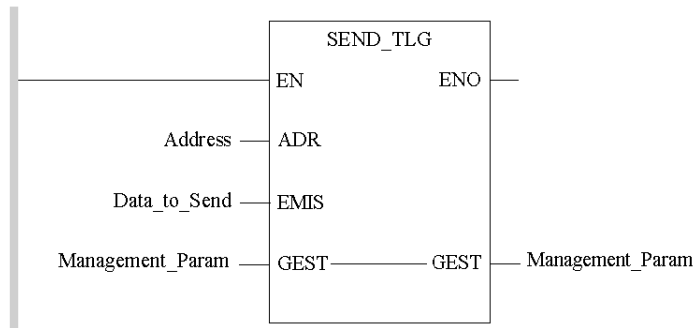
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

Adresse LD

```
SEND_TLG Data_to_Send, Management_Param
```

Représentation en ST

Représentation :

```
SEND_TLG(Address, Data_to_Send, Management_Param);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0... 5] OF INT	Adresse de l'entité destinataire de l'échange. Seules les adresses {Réseau.Station}APP ou {Réseau.Station}APP.num sont autorisées.
Data_to_Send	ARRAY [n... m] OF INT	Table d'entiers à envoyer à l'équipement destinataire de la requête. Elle doit avoir une longueur maximum de 8 entiers (16 octets). Remarque : il est primordial que le nombre d'octets à envoyer soit placé dans le second mot de la table de gestion avant de lancer l'échange.

Le tableau suivant décrit le paramètre d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0... 1] OF INT	<p>Table de mots utilisée pour gérer les échanges. La table comporte deux mots : le mot de compte rendu et le mot de longueur des données à envoyer. Le compte rendu comporte :</p> <ul style="list-style-type: none">● le compte rendu d'opération (octet de poids fort du premier mot) ;● le compte rendu de communication (octet de poids faible du premier mot). <p>Le compte rendu d'opération prend l'une des valeurs suivantes :</p> <ul style="list-style-type: none">● 16#00 : échange correcte,● 16#03 : format d'adresse incorrecte,● 16#04 : adresse cible incorrecte,● 16#05 : paramètres de gestion incorrects (longueur, par exemple)● 16#06 : paramètres spécifiques incorrects,● 16#07 : module en défaut,● 16#0A : taille insuffisante du buffer d'émission,● 16#0B : pas de ressources système,● 16#0F : service de télégramme non configuré.

Exemple d'envoi d'un télégramme

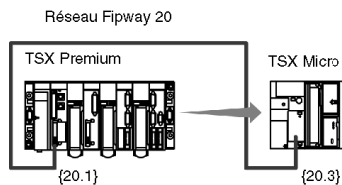
Présentation

Nous souhaitons envoyer un télégramme de 8 mots depuis la station 1 vers la station distante 3 sur le réseau Fipway 20.

La table %MW190:8 contiendra les mots à envoyer et la table %MW200:2 inclura la table de gestion de l'échange.

Illustration

Les deux stations sont connectées via un réseau Fipway.



Programmation

Programmation en ST :

```
IF RE(%I0.3.10) THEN
    SEND_TLG(ADDR('{20.3}APP'), %MW190:8, %MW200:2);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.3}APP')	<ul style="list-style-type: none"> ● {20.2} : réseau 20, station 3 ● APP : application
%MW190:8	Contenu du télégramme à envoyer
%MW200:2	Table de gestion

NOTE : Le mot %MW 201 doit être initialisé sur 16 (8 mots) avant l'envoi de la requête.

Pour exécuter cette fonction de manière synchrone, il est nécessaire de tester le compte rendu d'opération immédiatement après la ligne de programme activant l'exécution de la fonction.

SYMAX_IP_ADDR : adresse IP SY/MAX

25

Introduction

Ce chapitre décrit le bloc SYMAX_IP_ADDR.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	286
Description détaillée	288

Description

Description de la fonction

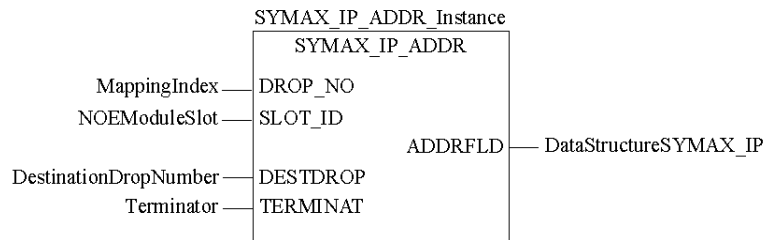
Ce bloc fonction permet d'indiquer l'adresse IP SY/MAX pour les blocs fonction REAG_REG, CREAD_REG, WRITE_REG et CWRITE_REG. L'adresse est transmise sous forme de structure de données.

EN et ENO peuvent être configurés comme paramètres supplémentaires.

NOTE : Lorsque vous programmez le bloc fonction SYMAX_IP_ADDR, il vous faut connaître le réseau que vous utilisez.

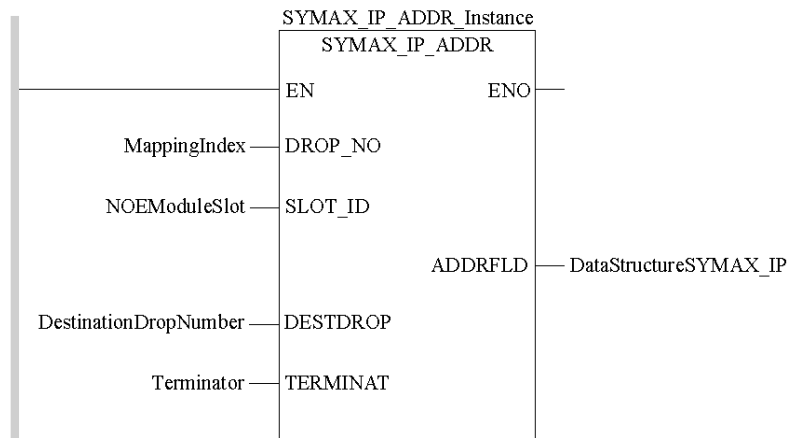
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL SYMAX_IP_ADDR_Instance (DROP_NO:=MappingIndex,
    SLOT_ID:=NOEModuleSlot, DESTDROP:=DestinationDropNumber,
    TERMINAT:=Terminator, ADDRFLD=>DataStructureSYMAX_IP)
```

Représentation en ST

Représentation :

```
SYMAX_IP_ADDR_Instance (DROP_NO:=MappingIndex,
    SLOT_ID:=NOEModuleSlot, DESTDROP:=DestinationDropNumber,
    TERMINAT:=Terminator, ADDRFLD=>DataStructureSYMAX_IP) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètre	Type de données	Signification
DROP_NR	BYTE	Index de mapping MBP-EtherNet (MET)
SLOT_ID	BYTE	Emplacement du module NOE
DESTDROP	WORD	Numéro de station cible (ou mettre FF hex)
TERMINAT	WORD	Terminaison (mettre FF hex)

Description des paramètres de sortie :

Paramètre	Type de données	Signification
ADDRFLD	WordArr5	Structure de données pour la transmission de l'adresse IP SY/MAX

Description détaillée

Description des éléments de `WordArr5`

Description des éléments de `WordArr5` :

Élément	Type de données	Signification
<code>WordArr5[1]</code>	WORD	Octet de poids fort : Numéro d'emplacement du module NOE Octet de poids faible : Index de mapping MBP-EtherNet (MET)
<code>WordArr5[2]</code>	WORD	Numéro de station cible (ou mettre FF hex)
<code>WordArr5[3]</code>	WORD	Terminaison (mettre FF hex)
<code>WordArr5[4]</code>	WORD	Réservé
<code>WordArr5[5]</code>	WORD	Réservé

DROP_NR

L'entrée `DROP_NR` indique l'index de mapping MBP-EtherNet (MET), c'est-à-dire que lorsque le MET est égal à 6, la valeur est la suivante :

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

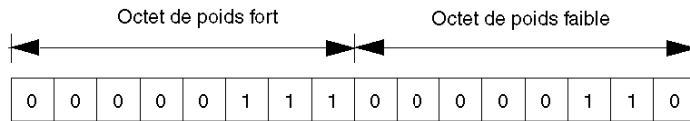
SLOT_ID

Lorsqu'un module NOE est interrogé, la valeur de l'entrée `SLOT_ID` représente l'emplacement physique du module NOE, c.-à-d. que lorsque le NOE est enfiché à l'emplacement 7 du châssis, la valeur est la suivante :

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

ADDRFLD

Lorsqu'un module NOE enfiché dans le châssis d'un automate Quantum est interrogé, la valeur de l'octet de poids fort représente l'emplacement physique du NOE et l'octet de poids faible représente l'index de mapping MBP-Ethernet (MET), à savoir que lorsque le NOE est enfiché à l'emplacement 7 du châssis et que l'index de mapping MET vaut 6, le premier élément de la structure de données se représente comme suit :



Octet de poids fort Emplacement 1 à 16

Octet de poids faible Index de mapping MBP-Ethernet (MET)

Introduction

Ce chapitre décrit le bloc TCP_IP_ADDR.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	292
Description détaillée	295

Description

Description de la fonction

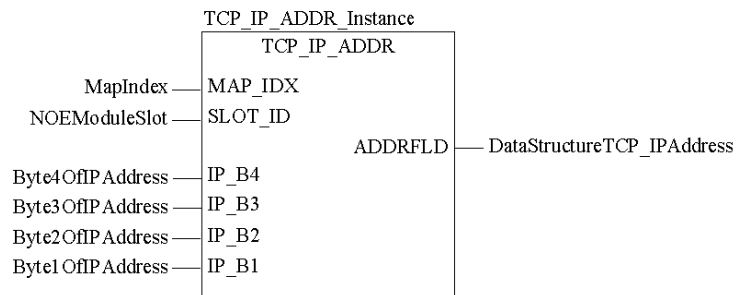
Ce bloc fonction autorise la saisie de l'adresse TCP/IP des bloc fonction `READ_REG`, `CREAD_REG`, `WRITE_REG` et `CWRITE_REG`. L'adresse est transférée sous forme de structure de données.

`EN` et `ENO` peuvent être configurés en tant que paramètres supplémentaires.

NOTE : Pour programmer le bloc fonction `TCP_IP_ADDR`, vous devez être parfaitement familiarisé avec votre réseau. L'ouvrage "Modicon Quantum - Module Ethernet TCP/IP - Guide utilisateur" fournit une description complète du routage TCP/IP.

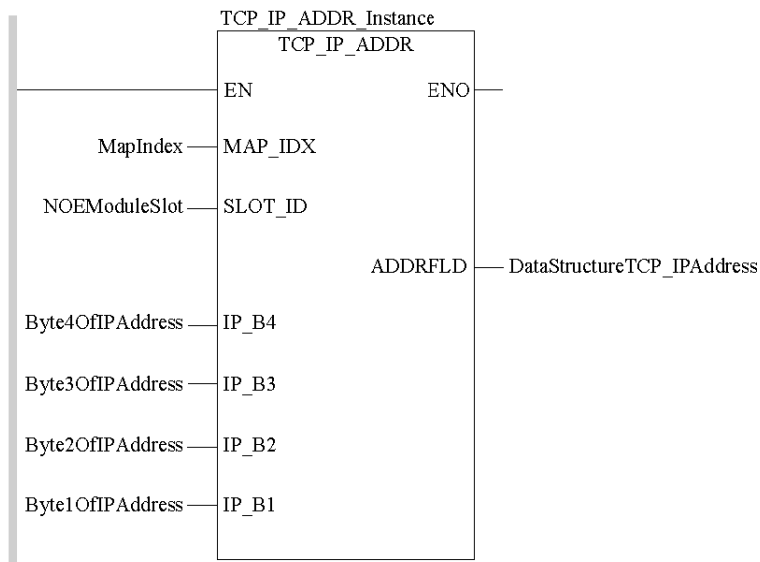
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL TCP_IP_ADDR_Instance (MAP_IDX:=MapIndex,
  SLOT_ID:=NOEModuleSlot, IP_B4:=Byte4OfIPAddress,
  IP_B3:=Byte3OfIPAddress, IP_B2:=Byte2OfIPAddress,
  IP_B1:=Byte1OfIPAddress,
  ADDRFLD=>DataStructureTCP_IPAddress)
```

Représentation en ST

Représentation :

```
TCP_IP_ADDR_Instance (MAP_IDX:=MapIndex,
  SLOT_ID:=NOEModuleSlot, IP_B4:=Byte4OfIPAddress,
  IP_B3:=Byte3OfIPAddress, IP_B2:=Byte2OfIPAddress,
  IP_B1:=Byte1OfIPAddress,
  ADDRFLD=>DataStructureTCP_IPAddress) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Description
MAP_IDX	BYTE	Index de mappage Index de mappage MBP sur Ethernet Transporter (MET)
Slot_ID	BYTE	ID d'emplacement Emplacement du module NOE
IP_B4	BYTE	Octet 4 (octet de poids fort) de l'adresse IP cible 32 bits
IP_B3	BYTE	Octet 3 de l'adresse IP cible 32 bits
IP_B2	BYTE	Octet 2 de l'adresse IP cible 32 bits
IP_B1	BYTE	Octet 1 (octet de poids faible) de l'adresse IP cible 32 bits

NOTE : Pour le paramètre Slot_ID : lorsque vous utilisez un module d'UC Ethernet intégré comme le module 140 CPU 651 x0, l'ID d'emplacement doit être 254 (FE hex) quel que soit l'emplacement de l'UC.

Description du paramètre de sortie :

Paramètre	Type de données	Description
ADDRFLD	WordArr5	Structure de données utilisée pour transférer l'adresse IP

Description détaillée

Description de l'élément `WordArr5`

Description de l'élément `WordArr5` :

Élément	Type de données	Description
<code>WordArr5[1]</code>	WORD	Octet de poids fort : Emplacement du module NOE Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET)
<code>WordArr5[2]</code>	WORD	Octet 4 de l'adresse IP cible 32 bits
<code>WordArr5[3]</code>	WORD	Octet 3 de l'adresse IP cible 32 bits
<code>WordArr5[4]</code>	WORD	Octet 2 de l'adresse IP cible 32 bits
<code>WordArr5[5]</code>	WORD	Octet 1 de l'adresse IP cible 32 bits

MAP_IDX

L'index de mappage MBP sur Ethernet Transporter (MET) est fourni à l'entrée `Map_Idx+`. Si MET est 6, la valeur apparaît comme suit :

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

SLOT_ID

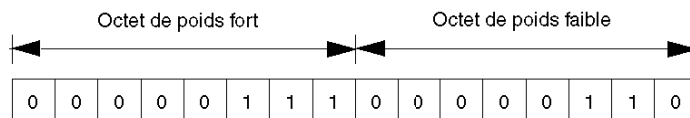
Si un module NOE du rack d'un automate Quantum est désigné comme abonné cible, la valeur à l'entrée `SLOT_ID` représente l'emplacement physique du module. Exemple : Si le module NOE est branché sur l'emplacement 7 du rack, la valeur apparaît comme suit :

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

NOTE : Lorsque vous utilisez un module d'UC Ethernet intégré comme le module 140 CPU 651 x0, l'ID d'emplacement doit être 254 (FE hex) quel que soit l'emplacement de l'UC.

ADDRFLD

Si un module NOE du rack d'un automate Quantum est désigné comme abonné cible, la valeur de l'octet de poids fort représente l'emplacement physique du module et l'octet de poids faible l'index de mappage MBP sur Ethernet Transporter (MET). Donc si le module NOE est inséré dans l'emplacement 7 du rack et si l'index de mappage MET est 6, le premier élément de la structure de données prend la forme qui suit :



Octet de poids fort Emplacements 1 à 16

Octet de poids faible Index de mappage MBP sur Ethernet Transporter (MET)

UNITE_SERVER : serveur immédiat

27

Objet de ce chapitre

Ce chapitre décrit la fonction de communication UNITE_SERVER.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	298
Exemple de serveur immédiat	301

Description

Description de la fonction

La fonction `UNITE_SERVER` permet de traiter les requêtes UNI-TE immédiatement à partir du programme d'application.

Cette fonction n'est pas activée dans la tâche `MAST` ou `FAST`.

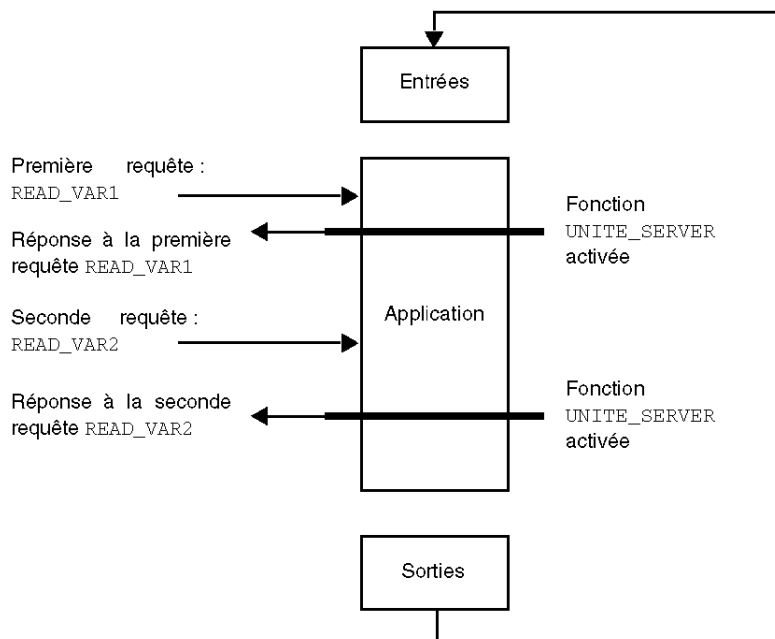
NOTE : Quel que soit le moment, seule une fonction `UNITE_SERVER` peut être activée par l'application.

NOTE : La fonction `UNITE_SERVER` permet de traiter des requêtes à partir d'une liaison Modbus (carte PCMCIA TSX SCP 114 dans un module TSX SCY 21601 configuré en tant qu'esclave Modbus avec serveur immédiat (*voir Premium et Atrium sous Unity Pro, Liaison série asynchrone, Manuel de l'utilisateur*)).

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

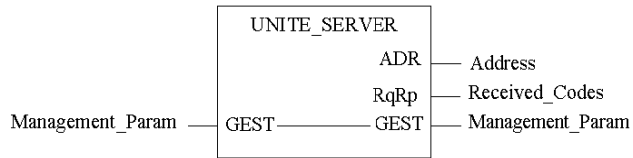
Principe d'un échange

Le diagramme illustre les échanges effectués lors de l'utilisation de la fonction de communication `UNITE_SERVER`.



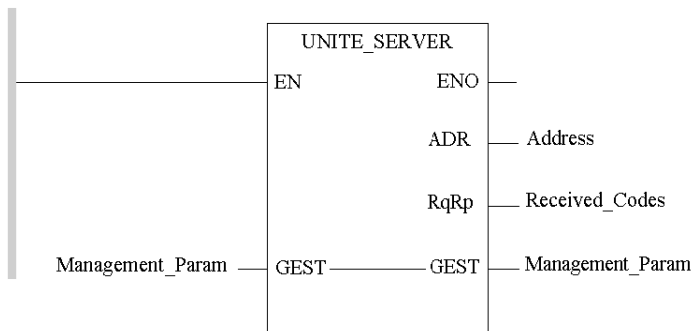
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Management_Param
UNITE_SERVER Address, Received_Codes
```

Représentation en ST

Représentation :

```
UNITE_SERVER(Management_Param, Address, Received_Codes);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée/de sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 1] OF INT	<p>Table de mots utilisée pour gérer les échanges. La table comporte deux mots : le premier contient le numéro d'échange et le bit d'activité, le deuxième contient le compte rendu. Le compte rendu comporte :</p> <ul style="list-style-type: none"> ● le compte rendu d'opération (octet de poids fort) ; ● le compte rendu de communication (octet de poids faible). <p>Le compte rendu de communication prend l'une des valeurs suivantes :</p> <ul style="list-style-type: none"> ● 16#00 : échange correct ; ● 16#01 : arrêt au timeout, la réponse n'a pas pu être envoyée en moins de 2 secondes ; ● 16#02 : arrêt à la demande de l'utilisateur (STOP, %SO, INIT, redémarrage à chaud ou à froid) ; ● 16#03 : format d'adresse inconnu ; ● 16#05 : paramètres de gestion incorrect ; ● 16#07 : problème d'envoi vers la destination ; ● 16#11 : aucune requête reçue ; ● 16#12 : fonction UNITE_SERVER appelée par une autre tâche ; ● 16#FF : message refusé. <p>Note : Lorsqu'un message est refusé (code 16#FF), le compte rendu d'opération peut alors prendre la valeur 16#14 (serveur arrêté).</p>

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Address	ARRAY [0.. 2] OF INT	Adresse de l'entité destinataire de l'échange. Cette adresse correspond à la voie associée à l'émetteur de la requête.
Received_Codes	INT	<p>A la fin de l'échange, ce mot comporte :</p> <ul style="list-style-type: none"> ● le code de la requête reçue (octet de poids faible) ; ● le code de la réponse renvoyée (octet de poids fort).

Exemple de serveur immédiat

Présentation

Cet exemple présente l'implémentation d'une fonction UNITE_SERVER en tant que serveur immédiat pour une fonction de communication READ_VAR. La liaison Modbus concernée est connectée à la carte PCMCIA d'un module TSX SCY 21601 situé sur l'emplacement 2 du rack de base.

Programmation

Programmation en ST :

```
IF NOT %MW100:X0 THEN
    UNITE_SERVER(%MW100:2, %MW110:3, %MW10);
END_IF;
```

Paramètres de la requête :

Paramètres	Description
%MW100:2	Table de gestion
%MW110:3	Exemple : le serveur immédiat est SCP114 configuré en tant qu'esclave 49 dans un module SCY21601 (emplacement 4, rack 0). <ul style="list-style-type: none"> ● Mot 1 : 16#FE00 correspond au rack ; ● Mot 2 : 16#0405 correspond à l'emplacement ; ● Mot 3 : 16#0095 correspond à la voie. Pour plus d'informations, voir modes d'adressage X-WAY (voir <i>Pilotes de communication, Manuel d'installation</i> ,).
%MW10	Réponse : <ul style="list-style-type: none"> ● Octet de poids faible : 16#03, code reçu de la fonction UNITE de n mots lus. ● Octet de poids fort : 16#03, code réponse de la fonction UNITE de n mots lus.

WRITE_ASYNC : écriture de données de façon asynchrone

28

Description

Description de la fonction

La fonction `WRITE_ASYNC` permet d'écrire 1 Koctet de données par le canal de messagerie asynchrone des coupleurs TSX ETY en mode TCP/IP.

Les données accessibles en écriture sont les suivantes :

- bits internes,
- mots internes.

L'écriture asynchrone ne peut s'effectuer qu'entre deux stations d'un même segment de réseau Ethernet TCP/IP.

La fonction `WRITE_ASYNC` est émise à la fin de la tâche MAST seulement si celle-ci est configurée en mode périodique. Il est possible d'activer 8 fonctions simultanément.

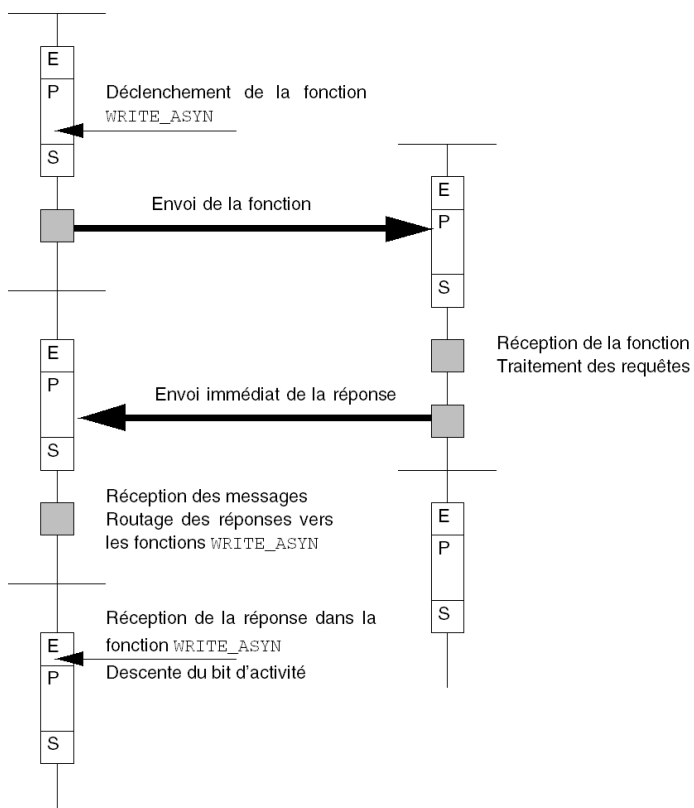
La taille des buffers d'émission et de réception est exprimée en mots. Elle est de 512 mots soit 1024 octets.

NOTE : La fonction serveur asynchrone supporte les protocoles UNI-TE V2.0 ou V1. La fonction `WRITE_ASYNC` utilise le protocole UNI-TE V2.0.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

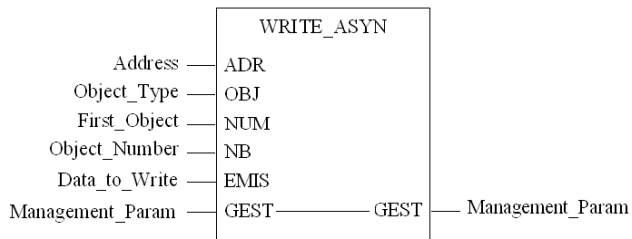
Principe des échanges

La figure suivante illustre les échanges entre deux stations pour une fonction WRITE_ASYNC :



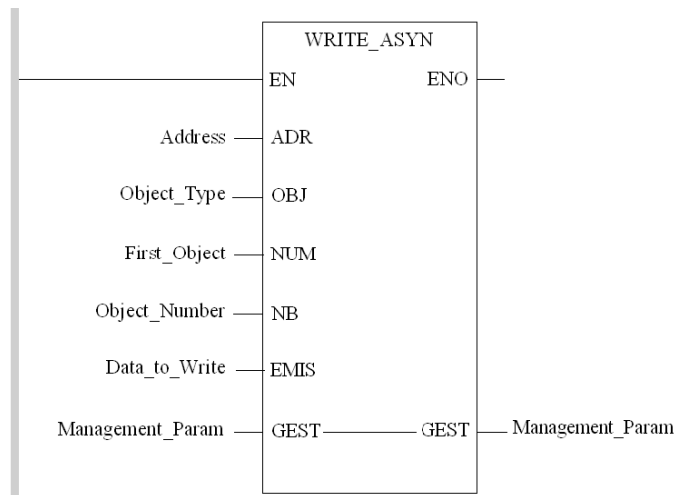
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

WRITE_ASYN Object_Type, First_Object, Object_Number,
Data_to_Write, Management_Param

Représentation en ST

Représentation :

WRITE_ASYN(Address, Object_Type, First_Object, Object_Number,
Data_to_Write, Management_Param);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0.. 5] OF INT	Adresse de l'entité destinataire de l'échange. Les adresses sont du type : ADDR (' {Réseau.Station}SYS.
Object_Type	STRING	Type des objets à écrire : <ul style="list-style-type: none"> ● '%M' : bits internes, ● '%MW' : mots internes, ● '%S' : bits système, ● '%SW' : mots système.
First_Object	DINT	Indice du premier objet à écrire dans l'équipement destinataire.
Object_Number	INT	Nombre d'objets à écrire.
Data_to_Write	ARRAY [n... m] OF INT	Tableau de mots contenant la valeur des objets à écrire.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 3] OF INT	Table de gestion de l'échange (<i>voir page 33</i>). Le compte rendu opération prend l'une des valeurs suivantes : <ul style="list-style-type: none"> ● 16#00 : échange correcte, ● 16#01 : arrêt sur timeout, la réponse n'a pas pu être émise en moins de 2 secondes, ● 16#02 : arrêt sur demande utilisateur (STOP, S0, INIT, reprise à chaud ou à froid), ● 16#03 : format d'adresse incorrecte, ● 16#05 : paramètres de gestion incorrects, ● 16#07 : destinataire absent, ● 16#09 : taille du buffer de réception insuffisante, ● 16#10 : taille du buffer d'émission insuffisante, ● 16#11 : absence de ressource système (déjà 8 fonctions actives), ● 16#19 : numéro d'échange incorrecte, ● 16#FF : message refusé. <p>Note : ne pas oublier de programmer une valeur de Timeout pour arrêter un échange en cours lorsque la réponse ne revient pas à l'émetteur.</p>

WRITE_GDATA : écriture des données globales Modbus Plus

29

Description

Description de la fonction

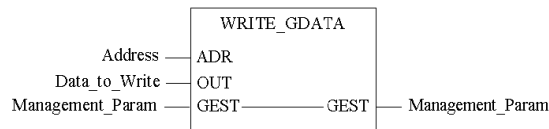
La fonction `WRITE_GDATA` permet d'écrire les données partagées appelées aussi **Global Data** sur un réseau **Modbus Plus**.

Les Global Data sont partagées entre 64 stations maximum d'un même réseau Modbus Plus. Chaque station peut écrire jusqu'à 32 entiers qui sont utilisables par toutes les stations du réseau. Réciproquement, chaque station peut lire les 32 (maximum) entiers de chaque station du réseau.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

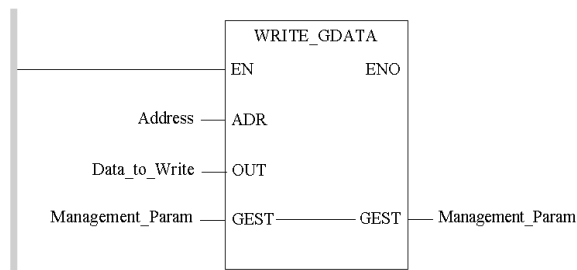
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

LD Address

WRITE_GDATA Data_to_Write, Management_Param

Représentation en ST

Représentation :

WRITE_GDATA(Address, Data_to_Write, Management_Param);

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0.. 5] OF INT	Adresse de la carte PCMCIA qui connecte le Premium sur le réseau Modbus Plus. Cette adresse vaut : ADDR('0.0.1.SYS'). Note : l'écriture des données s'effectue dans la carte PCMCIA qui se charge ensuite de partager les données.
Data_to_Write	ARRAY [n... m] OF INT	Tableau de mots contenant la valeur des objets à écrire. La taille de ce tableau doit toujours être de 32 entiers de 16 bits, taille maximum des Global Data sur un réseau Modbus Plus.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0.. 3] OF INT	Table de gestion de l'échange (<i>voir page 33</i>). Il n'est pas nécessaire d'initialiser le paramètre de longueur avant de lancer l'échange.

WRITE_REG : écriture de registre

30

Introduction

Ce chapitre décrit le bloc WRITE_REG.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	310
Types de données dérivés	313
Mode de fonctionnement	315
Description des paramètres	316

Description

Description de la fonction

En cas de front montant sur l'entrée `REQ` ce bloc fonction écrit une zone de registre sur un esclave adressé depuis l'automate via Modbus Plus, Ethernet TCP/IP ou Ethernet SY/MAX.

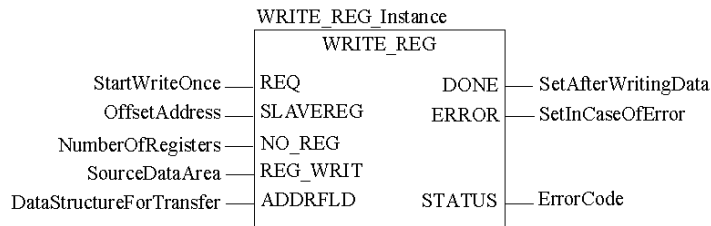
Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

NOTE : Lorsque vous programmez une fonction `WRITE_REG`, il vous faut connaître les procédures de routage utilisées par votre réseau. Les structures des itinéraires de routage Modbus Plus sont décrites en détail dans le *Guide de planification et d'installation du réseau Modbus Plus*. Si un routage TCP/IP ou Ethernet SY/MAX est implémenté, vous devez obligatoirement utiliser des produits routeurs standard Ethernet IP. Vous trouverez une description détaillée du routage TCP/IP dans le *Guide utilisateur de Quantum avec la configuration Unity Pro TCP/IP*.

NOTE : Plusieurs exemplaires de ce bloc fonction peuvent être utilisés dans le programme. Il n'est cependant pas possible de procéder à une instantiation multiple de ces exemplaires.

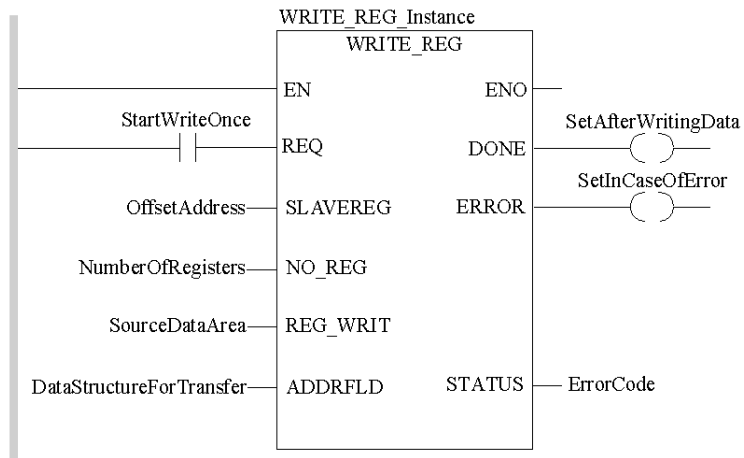
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL WRITE_REG_Instance (REQ:=StartWriteOnce,
  SLAVEREG:=OffsetAddress, NO_REG:=NumberOfRegisters,
  REG_WRIT:=SourceDataArea,
  ADDRFLD:=DataStructureForTransfer,
  DONE=>SetAfterWritingData, ERROR=>SetInCaseOfError,
  STATUS=>ErrorCode)
```

Représentation en ST

Représentation :

```
WRITE_REG_Instance (REQ:=StartWriteOnce,
  SLAVEREG:=OffsetAddress, NO_REG:=NumberOfRegisters,
  REG_WRIT:=SourceDataArea,
  ADDRFLD:=DataStructureForTransfer,
  DONE=>SetAfterWritingData, ERROR=>SetInCaseOfError,
  STATUS=>ErrorCode) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Signification
REQ	BOOL	En cas de front montant sur l'entrée REQ ce bloc fonction écrit une zone de registre sur un esclave adressé depuis l'automate via Modbus Plus, Ethernet TCP/IP ou Ethernet SY/MAX.
SLAVEREG	DINT	Adresse de la première adresse %MW devant être écrite dans l'esclave.
NO_REG	INT	Nombre d'adresses à écrire dans l'esclave
REG_WRIT	ANY	Champ de données source (Une structure de données doit être déclarée en tant que variable localisée pour les données source.)
ADDRFLD	WordArr5	Structure de données pour la transmission de l'adresse Modbus Plus, de l'adresse TCP/IP ou de l'adresse SY/MAX-IP.

Description des paramètres de sortie :

Paramètres	Type de données	Signification
DONE	BOOL	Mis à "1" pendant un cycle quand les données ont été écrites.
ERROR	BOOL	Mis à "1" pendant un cycle si une erreur apparaît.
STATUS	WORD	Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie. Code d'erreur, voir : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137 ● Codes d'erreur spécifiques SY/MAX, page 142 ● Codes d'erreur Ethernet TCP/IP, page 144

Erreur d'exécution

Pour obtenir une liste de tous les codes et valeurs d'erreur du bloc, voir *Booléen étendu*, page 416.

Types de données dérivés

Description de WordArr5 sur Modbus Plus

Description de WordArr5 sur Modbus Plus :

Élément	Type de données	Description
WordArr5[1]	WORD	Octet de poids faible : Registre 1 de routage, sert à déterminer l'adresse de l'abonné cible (l'une des cinq adresses de l'itinéraire de routage) lors d'une transmission par réseau. Le dernier octet différent de zéro de l'itinéraire de routage est l'abonné cible. Octet de poids fort : Adresse de l'abonné source. <ul style="list-style-type: none"> ● Position de l'emplacement du module lors de l'utilisation du port Modbus Plus sur le module NOM. ● Si vous utilisez le port Modbus Plus de l'UC, cet octet doit être réglé sur 0 (pour tous les emplacements de l'UC).
WordArr5[2]	WORD	Registre 2 de routage
WordArr5[3]	WORD	Registre 3 de routage
WordArr5[4]	WORD	Registre 4 de routage
WordArr5[5]	WORD	Registre 5 de routage

Description de WordArr5 sur Ethernet TCP/IP

Description de WordArr5 sur Ethernet TCP/IP :

Élément	Type de données	Description
WordArr5[1]	WORD	Octet de poids fort : Emplacement du module NOE Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET)
WordArr5[2]	WORD	Octet 4 (octet de poids fort) de l'adresse IP cible 32 bits
WordArr5[3]	WORD	Octet 3 de l'adresse IP cible 32 bits
WordArr5[4]	WORD	Octet 2 de l'adresse IP cible 32 bits
WordArr5[5]	WORD	Octet 1 (octet de poids faible) de l'adresse IP cible 32 bits

Description de wordArr5 sur Ethernet SY/MAX

Description de WordArr5 sur Ethernet SY/MAX :

Élément	Type de données	Description
WordArr5 [1]	WORD	Octet de poids fort : Emplacement du module NOE Octet de poids faible : Index de mappage MBP sur Ethernet Transporter (MET)
WordArr5 [2]	WORD	Numéro de station cible (ou mettre FF en hexadécimal)
WordArr5 [3]	WORD	Terminaison (ou mettre FF en hexadécimal)
WordArr5 [4]	WORD	Réservé
WordArr5 [5]	WORD	Réservé

Mode de fonctionnement

Mode de fonctionnement du bloc WRITE_REG

Un grand nombre de blocs fonction WRITE_REG peut être programmé, mais seules quatre commandes d'écriture peuvent être actives en même temps, que celles-ci soient déclenchées par ce bloc fonction ou par d'autres (p. ex. MBP_MSTR, CWRITE_REG). Tous les blocs fonction utilisent la même session de transaction de données et nécessitent plusieurs cycles de programme pour achever une commande.

Si plusieurs blocs fonction WRITE_REG sont utilisés dans une application, ils doivent se différencier entre eux au moins par les paramètres NO_REG ou REG_WRIT.

NOTE : Une communication TCP/IP entre un API Quantum (NOE 211 00) et un API Momentum (toutes les UC TCP/IP et tous les modules d'E/S TCP/IP) n'est possible que si **une** seule tâche de lecture ou d'écriture est effectuée dans chaque cycle d'API. Si plusieurs tâches par cycle sont envoyées, la communication est stoppée, sans qu'un message d'erreur soit généré dans le registre d'état.

Les signaux d'état DONE et ERROR signalent l'état du bloc fonction au programme utilisateur.

L'information complète de routage est contenue dans la structure de données WordArr5 de l'entrée ADDRFLD. Le type du bloc fonction connecté à cette entrée est fonction du réseau utilisé.

Veuillez utiliser pour :

- Modbus Plus le bloc fonction ModbusP_ADDR (voir page 151)
- Ethernet TCP/IP le bloc fonction TCP_IP_ADDR (voir page 291)
- Ethernet SY/MAX le bloc fonction SYMAX_IP_ADDR (voir page 285)

NOTE : Vous pouvez également utiliser la structure de données WordArr5 avec des constantes.

Description des paramètres

REQ

Un front montant déclenche la transaction d'écriture.

Ce paramètre peut être indiqué comme adresse, variable localisée, variable non localisée ou littéral.

SLAVEREG

Début de la zone de l'abonné cible dans laquelle les données sont écrites. La zone cible réside toujours dans la zone d'adresse %MW.

NOTE : Pour les esclaves d'un automate **non-Unity Pro** :

La zone cible réside toujours dans la zone de registre 4x. SLAVEREG voit l'adresse cible comme un décalage à l'intérieur de la zone 4x. Le "4" de tête doit être omis (p. ex. 59 (contenu de la variable ou valeur du littéral) = 40059).

Ce paramètre peut être indiqué comme adresse, variable localisée, variable non localisée ou littéral.

NO_REG

Nombre d'adresses à écrire dans le processeur esclave (1 ... 100).

Ce paramètre peut être indiqué comme adresse, variable localisée, variable non localisée ou littéral.

REG_WRIT

Pour ce paramètre un `ARRAY` de la taille de la transmission à effectuer doit être spécifié (\geq NO_REG). Le nom de ce tableau est transmis comme paramètre. Si le tableau est défini sur une taille trop réduite, la quantité de données transmise sera limitée par la place proposée dans le tableau.

Ce paramètre doit être indiqué comme variable localisée.

DONE

Le passage à l'état ON sur un cycle de programme signifie la fin du transfert des données.

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

ERROR

Le passage à l'état ON sur un cycle programme signifie la détection d'une nouvelle erreur.

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

STATUS

Si une erreur se produit lors de l'exécution de la fonction, le code d'erreur apparaît pendant un cycle au niveau de cette sortie.

Code d'erreur, voir :

- *Codes d'erreur Modbus Plus, SY/MAX et Ethernet TCP/IP, page 137*
- *Codes d'erreur spécifiques SY/MAX, page 142*
- *Codes d'erreur Ethernet TCP/IP, page 144*

Ce paramètre peut être indiqué comme adresse, variable localisée ou variable non localisée.

WRITE_VAR : écriture de variables

31

Objet de ce chapitre

Ce chapitre décrit la fonction de communication `WRITE_VAR`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Description	320
Ecran de saisie assistée	325
Exemple d'écriture de mots sur un réseau	327
Exemple d'écriture de mots via la liaison série des processeurs Modicon M340	329
Exemple de vérification d'exécution	331

Description

Description de la fonction

La fonction `WRITE_VAR` permet d'écrire un ou plusieurs objets langage du même type :

- bits internes,
- mots internes.

Les objets à écrire doivent toujours être consécutifs. Ils se trouvent dans une UC distante ou un équipement connecté à une voie de communication.

⚠ AVERTISSEMENT

INCOMPATIBILITE DES DONNEES ECHANGEES

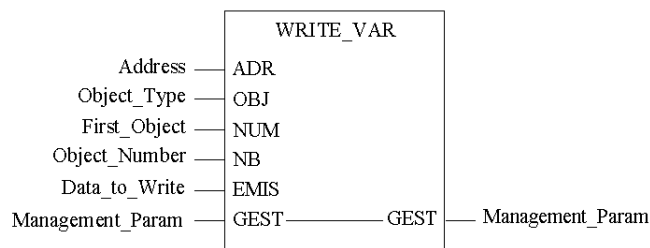
Dans la mesure où les alignements de structure de données sont différents pour Premium/Quantum et M340, vous devez vérifier la compatibilité des données échangées. Voir la page DDT : règles d'affectation (*voir Unity Pro, Langages de programmation et structure, Manuel de référence*) pour obtenir les règles d'alignement.

Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

Les paramètres supplémentaires `EN` et `ENO` peuvent être configurés.

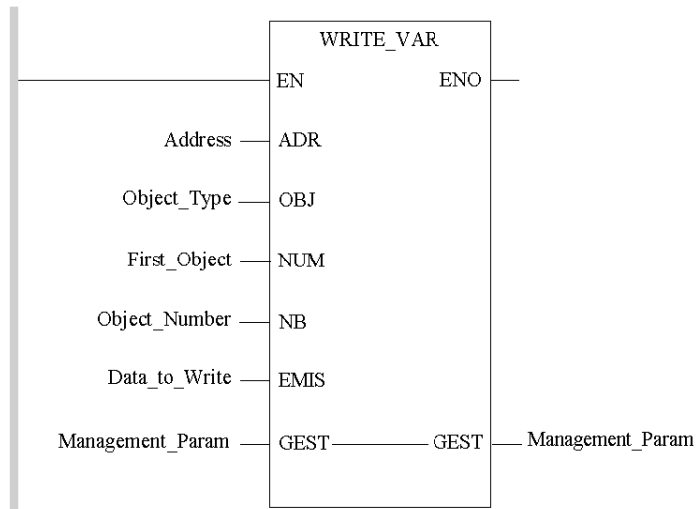
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

Adresse LD

```
WRITE_VAR Object_Type, First_Object, Object_Number,
Data_to_Write, Management_Param
```

Représentation en ST

Représentation :

```
WRITE_VAR(Address, Object_Type, First_Object, Object_Number,
Data_to_Write, Management_Param);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Address	ARRAY [0 à 5] OF INT pour Premium ARRAY [0 à 7] OF INT pour Modicon M340	<p>Les instructions suivantes s'appliquent uniquement aux automates Premium :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDR. ● Adresse de l'entité destinataire de l'échange. Les adresses suivantes sont interdites : <ul style="list-style-type: none"> ● {Réseau.Station}APP ● {Réseau.Station}APP.num ● adresses de diffusion (ALL, 0 pour le module TSX SCY 11601) <p>Les instructions suivantes s'appliquent uniquement aux automates Modicon M340 :</p> <ul style="list-style-type: none"> ● L'adresse de la voie en mode caractère de réception du message est indiquée par la fonction ADDM. ● La syntaxe de l'adresse est de type ADDM ('r.m.c.node').
Object_Type	STRING	<p>Type d'objets à écrire pour les automates Premium :</p> <ul style="list-style-type: none"> ● '%M' : bits internes, ● '%MW' : mots internes, ● '%S' : bits système, ● '%SW' : mots système. <p>Types d'objets à écrire pour les automates Modicon M340 :</p> <ul style="list-style-type: none"> ● '%M' : bits internes, ● '%MW' : mots internes.
First_Object	DINT	Indice du premier objet à écrire dans l'équipement destinataire.
Object_Number	INT	Nombre d'objets à écrire.
Data_to_Write	ARRAY [n à m] OF INT	Tableau de mots contenant la valeur des objets à écrire.

Le tableau suivant décrit les paramètres d'entrée/sortie :

Paramètre	Type	Commentaire
Management_Param	ARRAY [0 à 3] OF INT	<p>Table de gestion des échanges (<i>voir page 33</i>) Sur les automates Modicon M340, un nouveau bit d'annulation est disponible dans le mot de rang 1 de la table de gestion des échanges. Ce bit d'annulation est situé au niveau du mot de rang 1 qui se compose de deux octets :</p> <ul style="list-style-type: none">● Octet de poids fort : numéro d'échange,● Octet de poids faible : bit d'activité (rang 0) et bit d'annulation (rang 1). <p>La fonction élémentaire (EF) <code>WRITE_VAR</code> peut être annulée par la fonction élémentaire <code>CANCEL</code> ou en réglant le bit d'annulation de la table de gestion sur 1 (<i>voir Modicon M340 avec Unity Pro, Liaison série, Manuel de l'utilisateur</i>).</p>

NOTE : le paramètre de longueur ne doit pas être initialisé avant le lancement de la fonction.

Transactions simultanées

Le tableau ci-après indique les capacités de chaque voie de communication pour traiter simultanément les transactions selon diverses configurations des automates Micro et Premium.

Configuration	Micro	TSX 57 10	TSX 57 20	TSX 57 23/30/40/45/55, PCX 57, PMX 57	TSX 57 46/56
Port terminal principal Uni-Telway	4	4	4	4	8
Liaison PCMCIA ou SCY principale Uni-Telway	1	8	8	8	8
Port terminal esclave de client Uni-Telway	4	1	1	1	8
Liaison PCMCIA ou SCY esclave de client Uni-Telway	1	1	1	1	1
Port terminal esclave de serveur Uni-Telway	4	4	4	4	4
Liaison PCMCIA ou SCY esclave de serveur Uni-Telway	4	6	6	6	6
Port terminal Modbus	4	-	-	-	-
Liaison PCMCIA ou SCY Modbus	4	8	8	8	8
Bornier mode caractère	1	1	1	1	1
Liaison PCMCIA ou SCY mode caractère	4	8	8	8	8
PCMCIA CANopen	-	10	10	10	10
Liaison PCMCIA ou SCY Fipway	4	8	8	8	8
Modbus Plus	4	4	4	4	4
Ethernet	-	16	16	16	16
Ethernet intégré	-	-	-	-	64

Le tableau ci-après fournit les capacités de chaque voie de communication pour traiter simultanément les transactions en fonction des diverses configurations sur les automates Modicon M340.

Configuration	BMX P34 1000	BMX P34 2000	BMX P34 2010/ 20102	BMX P34 2020	BMX P34 2030/ 20302
CANopen intégré	-	-	16	-	16
Ethernet intégré	-	-	-	16	16
Port série principal ModBus	8	16	16	16	-

Ecran de saisie assistée

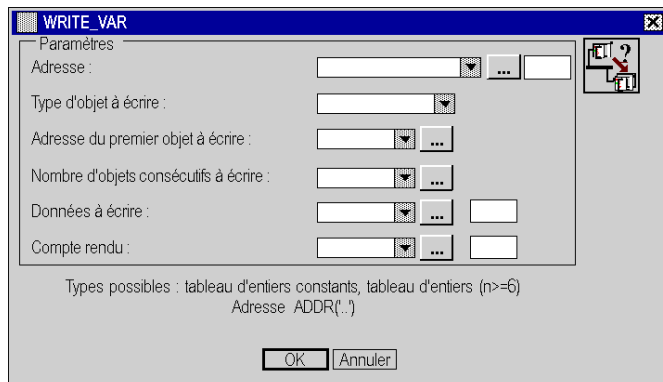
Vue d'ensemble

Pour cette fonction de communication, vous pouvez avoir recours à l'écran de saisie assistée. Notez que l'écran de saisie assistée n'est pas disponible pour le Modicon M340.

NOTE : les symboles de variable sont acceptées dans les différents champs de l'écran.

Illustration

La capture suivante est un exemple d'écran de saisie assistée de la fonction :



Adresse

Pour les automates Premium, les types d'objets possibles sont les suivants :

- ADDR (STRING),
- ARRAY [0..5] OF INT.

NOTE : si vous saisissez une valeur directement dans le champ, le bouton de saisie d'adresse assistée est grisé.

Type d'objet à écrire

Pour les automates Premium, les choix possibles sont les suivants :

- `'%M'` pour écrire des bits internes.
- `'%MW'` pour écrire des mots internes.
- `'%S'` pour écrire des bits système.
- `'%SW'` pour écrire des mots système.

Pour les automates Modicon M340, les choix possibles sont les suivants :

- `'%M'` pour écrire des bits internes.
- `'%MW'` pour écrire des mots internes.

NOTE : sélectionnez une solution parmi celles proposées dans le menu déroulant.

Adresse du premier objet à écrire

Les objets possibles sont de type `DINT` :

- des variables,
- des constantes,
- des valeurs immédiates.

NOTE : lorsque vous saisissez une constante, un champ de saisie correspondant apparaît. Lorsque vous saisissez une variable, elle peut être affectée (*voir page 452*) ou non. En revanche, les objets à écrire sont des variables affectées obligatoires.

Nombre d'objets consécutifs à écrire

Les objets possibles sont de type `INT` :

- des variables,
- des constantes,
- des valeurs immédiates.

NOTE : lorsque vous saisissez une constante, un champ de saisie correspondant apparaît. Si vous saisissez une variable, elle peut être affectée ou non.

Données à écrire

La zone des données à écrire est un tableau d'entiers. La taille de ce tableau dépend du nombre d'objets à écrire. Le tableau d'entiers peut être affecté ou non.

Compte rendu

Le compte rendu est un tableau de 4 entiers.

NOTE : veillez à ne pas utiliser plusieurs zones de mémoire identiques pour les tables de comptes rendus, car la fonction de lecture de variables risque de ne pas fonctionner.

Exemple d'écriture de mots sur un réseau

Présentation

Jusqu'à présent, les exemples ont été écrits en utilisant l'adressage direct (utilisation de %MWi), mais il est également possible de créer ces exemples à l'aide de variables non affectées.

L'exemple ci-après utilise des variables non affectées et illustre l'écriture d'une table de 50 mots nommée `Tab_1` (déclarée comme `ARRAY [0..49] OF INT`) dans l'esclave Uni-Telway, avec l'adresse :

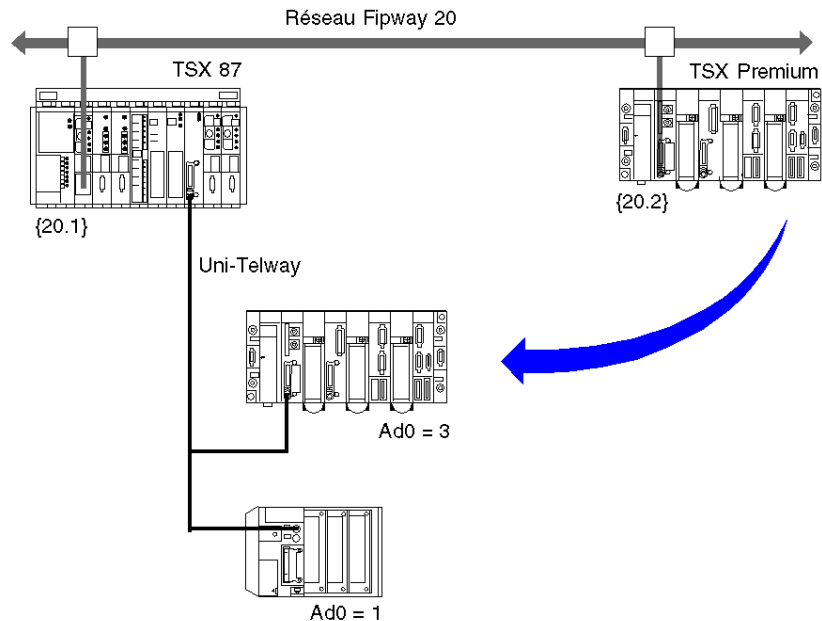
- Réseau 20
- Station 1
- Module SCM2116 dans un TSX 87 dans l'emplacement 5 du rack de base
- Voie 1
- Adresse serveur de l'esclave `Ad0 = 3`.

Les valeurs à écrire se trouvent dans la variable `Tab_1` de l'expéditeur.

Les paramètres de gestion se trouvent dans une table de 4 entiers nommée `Management_Parameter` (déclarée comme `ARRAY [0..3] OF INT`).

Illustration

Les deux stations sont connectées via un réseau Fipway.



Programmation

Programmation en ST :

```
IF RE(%I0.3.1) AND NOT Management_Parameter[0].0 THEN

WRITE_VAR(ADDR('{20.1}0.5.1.3'), '%MW', 0, 50, Tab_1, Management_
Parameter);

END_IF;
```

Paramètres de la requête :

Paramètres	Description
ADDR('{20.1}5.1.3')	<ul style="list-style-type: none">● {20.1} : réseau 20, station 1● 0 : rack● 5 : module● 1 : voie 1● 3 : adresse cible
%MW	Type d'objet (mot interne)
0	Adresse du premier objet à écrire
50	Nombre d'objets consécutifs à écrire
Tab_1	Données à écrire
Management_Parameter	Table de gestion

Exemple d'écriture de mots via la liaison série des processeurs Modicon M340

Présentation

Cet exemple utilise deux processeurs Modicon M340 qui communiquent via une liaison série Modbus.

Description de l'exemple

L'exemple ci-après utilise des variables non affectées et illustre l'écriture d'une table de 50 mots non affectée nommée `Tab_1` (déclarée comme `ARRAY [0..49] OF INT`) dans l'esclave Modbus. Les paramètres de gestion se trouvent dans une table de 4 entiers nommée `Management_Parameter` (déclarée comme `ARRAY [0..3] OF INT`).

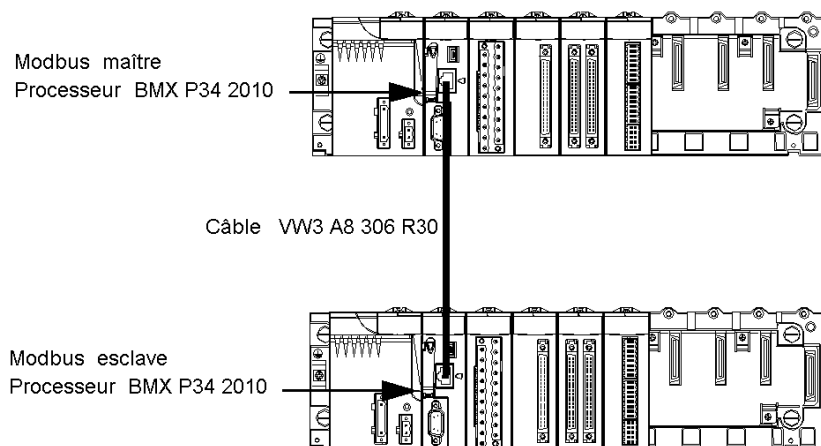
Dans cet exemple, le numéro de l'esclave Modbus est 7, et le paramètre `ADDM` d'entrée est '0.0.0.7'.

- 0: numéro du rack du processeur égal à 0
- 0: numéro d'emplacement du processeur dans le rack égal à 0 (le numéro d'emplacement d'un processeur Modicon M340 est toujours 0).
- 0: numéro de voie égal à 0 (la liaison série d'un processeur Modicon M340 est toujours la voie 0).
- 7: le numéro d'esclave configuré est 7.

Les valeurs à écrire se trouvent dans la variable `Tab_1` de l'expéditeur.

Illustration

Les deux processeurs Modicon 340 sont connectés via une liaison Modbus :



Programmation

Programmation en ST :

```
IF RE(%I0.3.1) AND NOT Management_Parameter[0].0 THEN

WRITE_VAR(ADDM('0.0.0.7'), '%MW', 0, 50, Tab_1, Management_Parameter);

END_IF;
```

Les paramètres de requête sont les suivants :

Paramètres	Description
ADDM('0.0.0.7')	<ul style="list-style-type: none">● 0: numéro du rack du processeur esclave● 0: numéro d'emplacement du processeur esclave● 0: numéro de voie (numéro de port série)● 7: numéro d'esclave configuré
%MW	Type d'objet (mot interne)
0	Adresse du premier objet à écrire
50	Nombre d'objets consécutifs à écrire
Tab_1	Données à écrire
Management_Parameter	Table de gestion

Exemple de vérification d'exécution

Présentation

L'exemple ci-après illustre la fonction `WRITE_VAR` avec la vérification des paramètres de gestion.

Programmation de la fonction

Programmation en ST :

```
IF NOT %M20 AND %I0.1.2 THEN
    %MW200:4:= 0;
    INC %MW1700
    %MW202:= 50;
    WRITE_VAR(ADDR('0.3.1.7'), '%MW', 20, 1, %MW1700:1, %MW200:4);
    SET %M20;
END_IF;
```

- le bit d'entrée %I0.1.2 contrôle la fonction,
- le bit interne %M20 permet de tester l'activité de la fonction,
- %MW200:4:= 0; définit la table de gestion sur la valeur 0,
- INC %MW1700; incrémente le mot %MW1700,
- MW202:= 50; initialise la valeur timeout sur 5 secondes.

NOTE : La syntaxe

`WRITE_VAR(ADDR('0.3.1.7'), '%MW', 20, 1, %MW1700:4, %MW200:4);` doit être utilisée pour les automates Modicon M340, car la fonction `ADDR` n'est pas compatible avec ces automates.

Programmation de la vérification de l'échange

Programmation en ST :

```
IF %M20 AND NOT %M200.0 THEN
  INC %MW204;
  IF %MW201 = 0 THEN
    INC %MW205;
  ELSE
    SET %Q0.2.2;
    INC %MW206;
    %MW207 := %MW201;
  END_IF;
END_IF;
```

- %MW204 compte le nombre d'échanges,
- %MW205 compte le nombre d'échanges corrects,
- %MW206 compte le nombre d'échanges sources d'erreurs,
- %MW207 stocke le message d'erreur,
- %Q0.2.2 indique l'échec d'un échange.

Introduction

Ce chapitre décrit le bloc XXMIT.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre	Sujet	Page
32.1	Introduction au bloc XXMIT	334
32.2	Fonctions XXMIT	335
32.3	XXMIT : Règles de programmation	383
32.4	Références techniques XXMIT	385
32.5	Informations sur le câblage	396

32.1 Introduction au bloc XXMIT

Fonctionnalités du bloc XXMIT

Présentation de la fonction

Le bloc XXMIT permet d'utiliser le port série de l'automate pour les communications, sous le contrôle du programme d'application.

Les types de communication suivants sont pris en charge :

- maître Modbus ;
- entrée/sortie ASCII simple ;
- entrée ASCII avec un ou deux caractères de fin ;
- communication par modem.

Description de la fonction

Le bloc XXMIT envoie des messages Modbus depuis un automate "maître" vers plusieurs automates esclaves ou envoie des chaînes de caractères ASCII depuis le port Modbus 1 des automates esclaves vers des imprimantes et des terminaux ASCII. Ces messages sont envoyés à l'aide de modems à numérotation automatique, de modems radio ou simplement via des connexions directes. En mode de communication, le bloc XXMIT exécute les fonctions d'entrée ASCII générales, y compris les fonctions d'entrée ASCII simple et terminée. Vous pouvez importer/exporter des données ASCII ou binaires vers/à partir de votre automate. Le bloc XXMIT intègre des fonctions de diagnostic permettant de vérifier qu'aucun autre bloc XXMIT n'est actif dans l'automate sur le même port. Dans le bloc XXMIT, des entrées de contrôle vous permettent de surveiller la liaison de communication entre l'automate et les équipements DCE (Data Communication Equipment, matériel de transmission de données) connectés au port Modbus 1 de l'automate. Le bloc XXMIT active le voyant du port lors de l'émission de données.

ATTENTION

DONNEES INVALIDES

Dans le cas de messages ASCII, aucun contrôle (tel un CRC ou une checksum) ne garantit la validité des données reçues.

Pour éviter de recevoir des données invalides en raison de perturbations électriques, il est recommandé d'utiliser des messages Modbus contenant un mécanisme de contrôle de redondance cyclique (CRC).

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

32.2 Fonctions XXMIT

Vue d'ensemble

Cette section décrit XXMIT.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Description sommaire	336
Représentation	337
Description détaillée des paramètres	341
Fonctions de communication du bloc XXMIT	352
Fonctions ASCII du bloc XXMIT	353
Fonctions de modem du bloc XXMIT	359
Fonctions Modbus du bloc XXMIT	361
Tampon FIFO et contrôle de flux	368
Exemples d'application	372

Description sommaire

Description de la fonction

Le bloc fonction XXMIT envoie des messages Modbus depuis un automate « maître » vers plusieurs automates esclaves ou envoie des chaînes de caractères ASCII depuis le port Modbus 1 des automates esclaves vers des imprimantes et des terminaux ASCII. Ces messages sont envoyés à l'aide de modems à numérotation automatique, de modems radio ou simplement via des connexions directes.

NOTE : le bloc fonction XXMIT peut être utilisé uniquement pour une tâche MAST. Un code d'erreur (127) apparaît immédiatement si le bloc est activé dans les tâches FAST / AUX ou EVENT (Unity Pro n'effectue aucun contrôle lors de la compilation).

NOTE : EN et ENO ne doivent PAS être utilisés avec le bloc XXMIT, car ils risquent de figer les paramètres de sortie.

NOTE : la communication Modbus par modem radio avec un automate Quantum n'est possible que si le bloc fonction XXMIT est configuré comme maître Modbus.

NOTE : notez que deux maîtres (sur le même bus) n'envoient pas de requêtes simultanément ; sinon, les requêtes seraient perdues et chaque rapport obtiendrait un résultat erroné qui pourrait être 16#0100 (impossible de traiter la requête) ou 16#ODFF (esclave non présent).

Logiciels et matériel requis

Logiciels

Le bloc XXMIT nécessite les logiciels suivants :

- Unity Pro 2.3 ou ultérieur
- Automate Quantum avec système d'exploitation 2.3 ou version supérieure

NOTE : pour importer dans Unity Pro une section de programme contenant un bloc fonction XXMIT, il est nécessaire de configurer l'automate Quantum avec un système d'exploitation 2.30 ou ultérieur.

Matériel

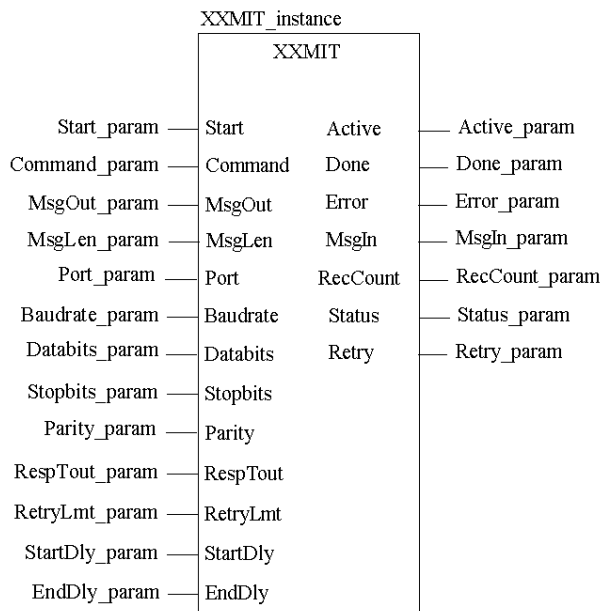
Le matériel suivant n'est pas pris en charge par le bloc fonction XXMIT :

- Automates Quantum ne prenant pas en charge les langages CEI Unity Pro
- Automate de sécurité Quantum
- Automate Premium
- Automate Modicon M340
- Simulateur d'automate

Représentation

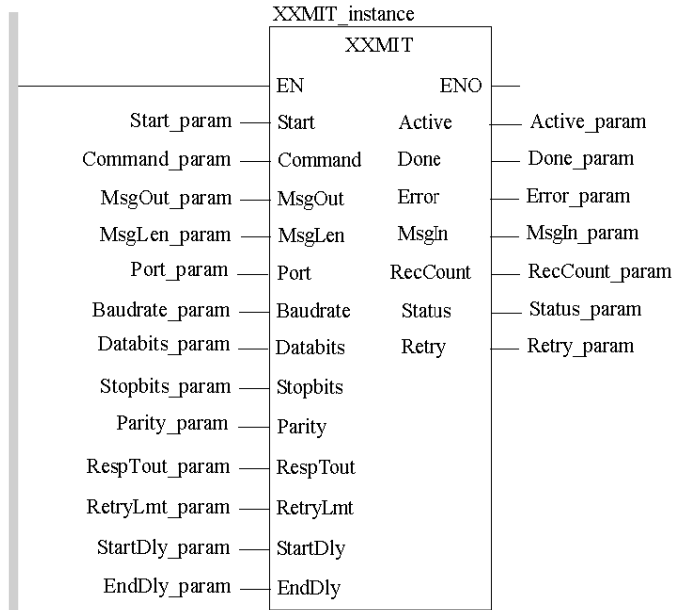
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL XXMIT_instance(Start:=Start_param,
  Command:=Command_param, MsgOut:=MsgOut_param,
  MsgLen:=Msglen_param, Port:=Port_param,
  Baudrate:=Baudrate_param, Databits:=Databits_param,
  Stopbits:=Stopbits_param, Parity:=Parity_param,
  RespTout:=RespTout_param, RetryLmt:=RetryLmt_param,
  StartDly:=StartDly_param, EndDly:=EndDly_param,
  Error=>Error_param, MsgIn=>MsgIn_param,
  RecCount=>RecCount_param, Status=>Status_param,
  Retry=>Retry_param)
```

Représentation en ST

Représentation :

```
XXMIT_instance (Start:=Start_param, Command:=Command_param,
MsgOut:=MsgOut_param, MsgLen:=MsgLen_param,
Port:=Port_param, Baudrate:=Baudrate_param,
Databits:=Databits_param, Stopbits:=Stopbits_param,
Parity:=Parity_param, RespTout:=RespTout_param,
RetryLmt:=RetryLmt_param, StartDly:=StartDly_param,
EndDly:=EndDly_param, Error=>Error_param,
MsgIn=>MsgIn_param, RecCount=>RecCount_param,
Status=>Status_param, Retry=>Retry_param);
```

Description des paramètres

Description du paramètre de bloc

Paramètres	Type de donnée	Signification
Start <i>(voir page 341)</i>	BOOL	La valeur 1 lance l'opération XXMIT <i>(voir page 341)</i>
Command <i>(voir page 341)</i>	WORD	Indique la commande à exécuter <i>(voir page 341)</i>
MsgOut <i>(voir page 344)</i>	ANY	Message à envoyer <i>(voir page 344)</i>
MsgLen <i>(voir page 345)</i>	INT	Longueur du message de sortie <i>(voir page 345)</i>
Port <i>(voir page 345)</i>	BYTE	Sélection de l'interface de communication (port 1 uniquement) <i>(voir page 345)</i>
Baudrate <i>(voir page 345)</i>	INT	Débit <i>(voir page 345)</i>
Databits <i>(voir page 345)</i>	BYTE	Bits de données <i>(voir page 345)</i>
Stopbits <i>(voir page 346)</i>	BYTE	Bits d'arrêt <i>(voir page 346)</i>
Parity <i>(voir page 346)</i>	BYTE	Parité <i>(voir page 346)</i>
RespTout <i>(voir page 346)</i>	INT	Délai d'attente avant réception d'une réponse valide <i>(voir page 346)</i>
RetryLmt <i>(voir page 346)</i>	INT	Nombre de tentatives jusqu'à réception d'une réponse valide <i>(voir page 346)</i>
StartDly <i>(voir page 346)</i>	INT	Délai d'attente avant émission du message <i>(voir page 346)</i>

Paramètres	Type de donnée	Signification
EndDly <i>(voir page 347)</i>	INT	Délai d'attente après émission du message <i>(voir page 347)</i>
Active <i>(voir page 347)</i>	BOOL	La valeur 1 indique qu'une opération XXMIT est en cours <i>(voir page 347)</i>
Done <i>(voir page 347)</i>	BOOL	La valeur 1 indique que l'opération XXMIT a réussi. <i>(voir page 347)</i>
Error <i>(voir page 347)</i>	BOOL	La valeur 1 indique qu'une erreur est survenue ou que l'opération XXMIT en cours est terminée <i>(voir page 347)</i>
MsgIn <i>(voir page 348)</i>	ANY	Message entrant <i>(voir page 348)</i>
RecCount <i>(voir page 348)</i>	INT	Affiche le nombre de caractères reçus <i>(voir page 348)</i>
Status <i>(voir page 348)</i>	INT	Affiche le code d'erreur généré par le bloc XXMIT <i>(voir page 348)</i>
Retry <i>(voir page 351)</i>	INT	Indique le nombre de tentatives en cours effectuées par le bloc XXMIT <i>(voir page 351)</i>

Description détaillée des paramètres

Start

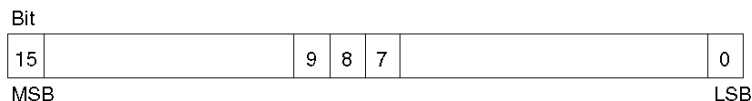
La valeur 1 du signal Start déclenche l'opération XXMIT. La valeur 1 doit être appliquée tant que l'opération n'est pas terminée ou qu'aucune erreur n'est survenue.

NOTE : Les paramètres d'entrée du bloc XXMIT doivent être initialisés avant de configurer l'entrée START. Ils ne doivent pas être modifiés lorsque le bloc fonction est en cours d'exécution.

Command

Le bloc XXMIT interprète chaque bit du mot de commande comme une fonction à exécuter. Si les bits 9 et 8 sont activés simultanément ou si plusieurs bits parmi les bits 3, 2, 1 ou 0 sont activés simultanément ou si le bit 9 n'est pas activé alors que l'un des bits 3, 2, 1 ou 0 l'est, le système génère l'erreur 129. Pour plus de détails, reportez-vous à la section *Fonctions de communication du bloc XXMIT*, page 352. La définition de chaque bit est présentée dans le tableau ci-dessous.

Disposition du mot de commande



Définition des bits du mot de commande du bloc XXMIT

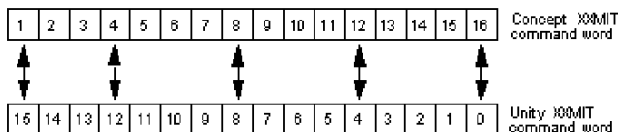
Bit	Définition
Bit 15 (bit de poids fort)	Réservé
Bit 14 Validation du contrôle modem RTS/CTS	Mis à 1 lorsqu'un équipement DCE (Data Communication Equipment, matériel de transmission de données) connecté à l'automate nécessite un protocole matériel utilisant le contrôle RTS/CTS. Ce bit peut être utilisé conjointement avec les valeurs contenues dans StartDly et EndDly. Le retard de début de transmission garde le signal RTS activé pendant le temps défini dans StartDly (en ms) avant l'envoi d'un message par le bloc XXMIT à partir du port de l'automate. De même, le retard de fin de transmission garde le signal RTS activé pendant le temps défini dans EndDly (en ms) après l'envoi d'un message par le bloc XXMIT à partir du port de l'automate. Après l'expiration du retard de fin de transmission, le bloc XXMIT désactive le signal RTS.

Bit	Définition
Bit 13 Validation du mode RS485	Mis à 1 lorsque le port sélectionné doit fonctionner en mode RS485. Sinon, il est défini par défaut sur 0, valeur correspondant au mode RS232. Lors de l'utilisation du port Modbus en mode RS485 avec la messagerie Modbus, assurez-vous d'utiliser pour le bloc XXMIT exactement les mêmes paramètres (vitesse, bits de données, bits d'arrêt, parité) que ceux configurés pour le port.
bit 12	Réservé
Bit 11 Entrée ASCII terminée	Mis à 1 pour supprimer et ignorer tous les caractères du tampon FIFO jusqu'à ce que la chaîne de départ soit trouvée. Ensuite, les caractères de départ et les caractères suivants sont écrits dans MsgIn jusqu'à ce que la séquence de fin soit trouvée. La chaîne de fin est également écrite dans MsgIn. Reportez-vous à la section <i>Fonction d'entrée ASCII terminée</i> , page 354 pour plus de détails.
Bit 10 Entrée ASCII simple	Mis à 1 pour supprimer les caractères ASCII du tampon FIFO pour l'écriture dans le tableau MsgIn. Reportez-vous à la section <i>Fonction d'entrée ASCII simple</i> , page 357 pour plus de détails.
Bit 9 Validation des messages sous forme de chaînes ASCII	Mis à 1 lorsque vous souhaitez envoyer des messages ASCII à partir de l'automate. Le bloc XXMIT envoie des chaînes ASCII d'une longueur maximale de 1 024 caractères. Programmez le message ASCII dans MsgOut. Utilisez uniquement le bit 9 OU le bit 8 ; n'essayez pas d'utiliser les deux.
Bit 8 Validation des messages Modbus	Mis à 1 lorsque vous souhaitez envoyer des messages Modbus à partir de l'automate. Les messages Modbus peuvent être envoyés au format RTU ou ASCII. Avec 8 bits de données, le bloc XXMIT utilise le format Modbus RTU. Avec 7 bits de données, il utilise le format Modbus ASCII. Utilisez uniquement le bit 9 OU le bit 8 ; n'essayez pas d'utiliser les deux.
Bit 7 Validation du tampon FIFO de réception ASCII	Mis à 1 pour permettre au bloc XXMIT de prendre le contrôle du port sélectionné (1) à partir de l'automate. Le bloc commence à recevoir des caractères ASCII dans un tampon FIFO circulaire vide de 512 octets. Reportez-vous à la section <i>Tampon FIFO de réception ASCII</i> , page 368 pour plus de détails.
Bit 6 Validation du caractère de retour arrière	Mis à 1 pour permettre une utilisation particulière du caractère de retour arrière ASCII (BS, 8Hex) lors de l'utilisation d'une entrée ASCII simple (bit 10) ou d'une entrée ASCII terminée (bit 11). Si le bit 6 est à 1, aucun caractère de retour arrière n'est stocké dans MsgIn. Reportez-vous à la section <i>Validation du caractère de retour arrière</i> , page 368 pour plus de détails.

Bit	Définition
Bit 5 Validation du contrôle de flux RTS/CTS	Mis à 1 pour autoriser le contrôle de flux matériel Full Duplex utilisant les signaux de synchronisation RTS et CTS pour les messages ASCII. Le contrôle de flux RTS/CTS fonctionne en mode d'entrée et en mode de sortie. Reportez-vous à la section <i>Validation du contrôle de flux RTS/CTS</i> , page 369 pour plus de détails.
Bit 4 Validation du contrôle de flux Xon/Xoff	Mis à 1 pour permettre un contrôle de flux logiciel Full Duplex utilisant les caractères ASCII Xon (DC1, 11 Hex) et Xoff (DC3, 13 Hex). Le contrôle de flux Xon/Xoff fonctionne en mode d'entrée et en mode de sortie. Reportez-vous à la section <i>Validation du contrôle de flux Xon/Xoff</i> , page 370 pour plus de détails.
Bit 3 Modem à numérotation par impulsion	Mis à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez composer un numéro de téléphone en utilisant la numérotation par impulsion. Programmez le numéro de téléphone dans MsgOut. La longueur du message doit être définie dans MsgLen. Les numéros composés par impulsion sont envoyés au modem, automatiquement précédés d'ATDT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le message de composition étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du numéro à composer.
Bit 2 Raccrochage du modem	Mis à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez raccrocher le modem. Vous devez utiliser le programme utilisateur pour mettre ce bit à 1. Le message de raccrochage étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du message. Les messages raccrochés sont envoyés au modem, automatiquement précédés de +++AT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le bloc XXMIT recherche une réponse de déconnexion correcte de la part du modem avant d'ACTIVER le signal de sortie Done qui indique que l'exécution a réussi.
Bit 1 Modem à numérotation à tonalité	Mis à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez composer un numéro de téléphone au clavier. Programmez le numéro de téléphone dans MsgOut. La longueur du message doit être définie dans MsgLen. Les numéros composés au clavier sont envoyés au modem, automatiquement précédés d'ATDT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le message de composition étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du numéro à composer.
Bit 0 Initialisation du modem	Mis à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez initialiser le modem. Programmez le message d'initialisation dans MsgOut et la longueur du message dans MsgLen. Tous les messages sont envoyés au modem, automatiquement précédés d'AT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le message d'initialisation étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du message.

NOTE : La numérotation des bits du mot de commande Unity a changé par rapport à celle des bits du mot de commande du bloc XXMIT Concept :

Conversion du mot de commande



Lorsque l'application doit être convertie de Concept à Unity, la nouvelle numérotation des bits doit être prise en compte si l'accès au mot de commande s'effectue à l'aide des bits. Cette règle ne s'applique **PAS** lorsque vous accédez au mot de communication à l'aide de mots.

MsgOut

MsgOut contient les données du message à transférer, par exemple des caractères ASCII pour un transfert ASCII, la définition de caractères de fin pour une entrée ASCII terminée ou des modèles Modbus pour des messages Modbus maîtres.

Le type de données à affecter au paramètre doit respecter les exigences de la fonction à exécuter. Dans le cas d'une opération Modbus, MsgOut et MsgIn doivent utiliser le même type de données.

NOTE : Pour tous les types de communications (Modbus maître, entrée/sortie ASCII), les paramètres MsgOut et MsgIn doivent être affectés à une variable. Les deux variables peuvent être soit un tableau de mots, soit un tableau d'octets. Les autres types génèrent des résultats imprévisibles.

NOTE : Pour les messages Modbus, MsgOut doit être un champ de mots. La taille minimale du tableau est WordArr9.

MsgLen

Vous devez entrer la longueur du message courant selon la fonction XXMIT sélectionnée.

Le tableau suivant présente les fonctions Modbus et ASCII :

Fonction XXMIT	Sous-fonction	Longueur du message
Messagerie Modbus	01, 02, 03, 04, 05, 06, 08, 15, 16	5
Entrée ASCII terminée		5
Entrée ASCII simple		1...1024.
Messages sous forme de chaînes ASCII		1...1024. La longueur sélectionnée doit correspondre à la taille du tableau affecté à MsgOut. Dans le cas contraire, le système génère l'erreur 129.

NOTE : Pour les codes fonction Modbus, la valeur de MsgLen peut être supérieure à cinq mots, mais seuls les cinq premiers mots du tableau de définition Modbus seront pris en compte au moment de l'exécution.

Port

Le paramètre Port définit l'interface de communication. La seule valeur autorisée est 1.

Baudrate

Le bloc XXMIT prend en charge les vitesses de données suivantes : 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200. Pour configurer une vitesse de données, entrez un nombre décimal. Si vous entrez une vitesse de données incorrecte, le bloc affiche une erreur de configuration incorrecte (code d'erreur 127) dans l'élément Status du bloc XXMIT.

Datubits

Le bloc XXMIT prend en charge les bits de données suivants : 7 et 8. Pour configurer une taille de bit de données, entrez un nombre décimal dans cet élément. Les messages Modbus peuvent être envoyés en mode ASCII ou RTU. Le mode ASCII utilise 7 bits de données tandis que le mode RTU en utilise 8. Lors de l'envoi d'un message en caractères ASCII, vous pouvez utiliser 7 ou 8 bits de données. Si vous entrez un bit de données incorrect, le bloc affiche une erreur de configuration incorrecte (code d'erreur 127) dans l'élément Status du bloc XXMIT. Reportez-vous au document *Modicon Modbus Protocol Reference Guide* (www.modbus.org) pour plus de détails sur les formats des messages Modbus.

Stopbits

Le bloc XXMIT prend en charge un ou deux bits d'arrêt. Entrez l'un des nombres décimaux suivants : 1 = un bit d'arrêt ou = deux bits d'arrêt. Si vous entrez un bit d'arrêt incorrect, le bloc affiche une erreur de configuration incorrecte (code d'erreur 127) dans l'élément Status du bloc XXMIT.

Parité

Le bloc XXMIT prend en charge la parité suivante : aucune, impaire et paire. Entrez l'un des nombres décimaux suivants : 0 = aucune parité, 1 = parité impaire ou 2 = partie paire. Si vous entrez une parité incorrecte, le bloc affiche une erreur de configuration incorrecte (code d'erreur 127) dans l'élément Status du bloc XXMIT.

RespTout

Entrez la valeur du délai en millisecondes (ms) correspondant au temps d'attente du bloc XXMIT avant réception d'une réponse correcte de la part d'un équipement esclave (automate, modem, etc.). Ce retard s'applique également aux transmissions ASCII et aux opérations de contrôle de flux. Lorsque la réponse n'est pas complètement structurée dans le délai imparti, le bloc XXMIT génère une erreur. Les valeurs correctes sont comprises entre 0 et 32 767 ms. Le timeout commence à s'écouler après l'envoi du dernier caractère du message.

RetryLmt

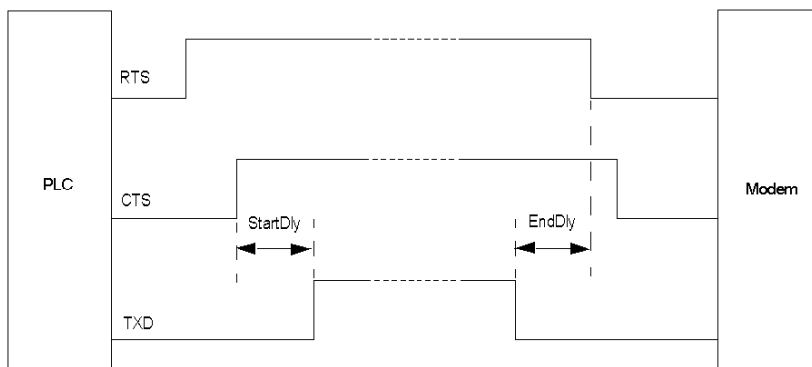
Entrez le nombre de tentatives d'envoi d'un message effectuées par le bloc XXMIT avant réception d'une réponse correcte de la part d'un équipement esclave (automate, modem, etc.). Lorsque la réponse n'est pas complètement structurée dans le délai imparti, le bloc XXMIT génère une erreur et un code d'erreur. Les valeurs correctes sont comprises entre 0 et 32 767 répétitions. Ce champ est utilisé conjointement avec RespTout.

StartDly

Lorsque le contrôle RTS/CTS est activé, entrez la valeur du délai en millisecondes (ms) déterminant le temps d'attente du bloc XXMIT entre la réception du signal CTS et l'envoi d'un message à partir du port de l'automate. Vous pouvez également utiliser ce registre lorsque le contrôle RTS/CTS n'est PAS activé. Dans ce cas, la valeur du délai entré détermine le temps d'attente du bloc XXMIT avant l'envoi d'un message à partir du port de l'automate. Vous pouvez l'utiliser comme un temporisateur avant message. Les valeurs correctes sont comprises entre 0 et 32 767 ms.

EndDly

Lorsque le contrôle RTS/CTS est activé, entrez la valeur du délai en millisecondes (ms) déterminant le temps pendant lequel le bloc XXMIT garde le signal RTS activé après l'envoi du message à partir du port de l'automate. Après l'expiration du délai, le bloc XXMIT désactive le signal RTS. Vous pouvez également utiliser ce registre lorsque le contrôle RTS/CTS n'est PAS activé. Dans ce cas, la valeur du délai entré détermine le temps d'attente du bloc XXMIT après l'envoi d'un message à partir du port de l'automate. Vous pouvez l'utiliser comme un temporisateur après message. Les valeurs correctes sont comprises entre 0 et 32 767 ms.



NOTE : Lors de la communication RS 485, le signal de transmission est maintenu sur "1" tout au long du délai EndDly. Lors des connexions à deux fils, tout caractère provenant du partenaire sera perdu. Par conséquent, si cette fonction n'est pas nécessaire, réglez EndDly sur 0 ms.

Active

La valeur 1 indique qu'une opération XXMIT est en cours.

Done

La valeur 1 indique que l'opération XXMIT a réussi.

Erreur

La valeur 1 indique qu'une erreur est survenue ou que l'opération XXMIT courante est terminée.

MsgIn

MsgIn contient les données du message entrant pour une entrée ASCII terminée ou une entrée ASCII simple.

Dans le cas d'une opération Modbus, le type de données doit être identique au type du champ MsgOut.

RecCount

Cet élément affiche le nombre de caractères reçus.

NOTE : Lorsque trop de caractères sont reçus et qu'aucun caractère de fin n'a été détecté, une erreur est générée dans le paramètre Status (131) et le paramètre RecCount prend la valeur MsgLen-1.

Etat

Cet élément affiche un code d'erreur généré par le bloc XXMIT.

Le tableau ci-dessous présente une liste complète.

Etat d'erreur

Code d'erreur	Description de l'erreur
1	Exception Modbus - Fonction incorrecte
2	Exception Modbus - Adresse de données incorrecte
3	Exception Modbus - Valeur de données incorrecte
4	Exception Modbus - Erreur abonné esclave
5	Exception Modbus - Confirmation
6	Exception Modbus - Abonné esclave occupé
7	Exception Modbus - Confirmation négative
8	Exception Modbus - Erreur de parité mémoire
9... 99	Réservé
100	La zone de données Automate esclave ne peut être égale à zéro
101	La zone de données Automate maître ne peut être égale à zéro
102	bit (%M) non configuré
103	Mot mémoire (%MW) de l'automate maître non configuré
104	La longueur de données ne peut être égale à zéro.
105, 106	Réservé

Code d'erreur	Description de l'erreur
107	Timeout de transmission de message (cette erreur est générée lorsque l'émetteur-récepteur asynchrone universel ne parvient pas à terminer une transmission en 10 secondes maximum ; Cette erreur évite le compteur de répétitions et active la sortie d'erreur à la première erreur, voir Informations supplémentaires pour l'erreur 107 ci-dessous).
108	Erreur non définie
109	Le modem signale ERREUR
110	Le modem signale PAS DE PORTEUSE
111	Le modem signale PAS DE TONALITÉ
112	Le modem signale OCCUPÉ
113	Checksum LRC de l'automate esclave non valide, voir 1) ci-dessous
114	Checksum CRC de l'automate esclave non valide, voir 1) ci-dessous
115	Code fonction Modbus invalide
116	Timeout message de réponse Modbus, voir 2 ci-dessous
117	Timeout de réponse Modem
118	Le bloc XXMIT n'a pas obtenu d'accès au port de communication numéro 1 de l'automate
119	Le bloc XXMIT ne peut pas valider le récepteur de port de l'automate
120	Le bloc XXMIT n'a pas pu activer l'émetteur-récepteur asynchrone universel de l'automate
121	Réservé
122	Port incorrect
123	Réservé
124	Etat interne indéfini
125	Mode diffusion interdit pour cette fonction Modbus
126	DCE peut transmettre CTS
127	<ul style="list-style-type: none"> ● Configuration incorrecte (bits de données, vitesse, parité ou bits d'arrêt). ● Exécution dans une tâche autre que MAST. ● Code fonction Modbus incorrect. ● Adresse esclave supérieure à 247. ● Longueur de données supérieure aux limites indiquées dans ce document. ● Numéro de port différent de 1. ● Valeurs RespTout, StartDly, EndDly, RecCount, MsgLen, RetryLmt inférieures à 0.

Code d'erreur	Description de l'erreur
128	Réponse inattendue de la part de l'esclave Modbus, voir 1) ci-dessous
129	Paramétrage incorrect du mot de commande
130	Mot de commande modifié en cours d'activité
131	Compte de caractères incorrect
132	Réservé
133	Erreur de débordement de l'entrée FIFO ASCII
134	Nombre incorrect de caractères de départ ou de caractères de fin
135...149	Réservé
150	Le port configuré est déjà occupé par une autre instance du bloc XXMIT ou il n'est pas pris en charge sur cet automate.
151	MsgOut est inférieur à 12 octets alors que la fonction Messages Modbus maîtres est sélectionnée. Inclut également la valeur de paramètre MsgLen inférieure à 5
152	La variable connectée à MsgOut est inférieure à la valeur du paramètre MsgLen alors que la fonction Messages sous forme de chaînes ASCII est sélectionnée.
153	La variable connectée à MsgIn est inférieure à la valeur du paramètre MsgLen alors que la fonction Entrée ASCII terminée ou Entrée ASCII simple est sélectionnée.
154	Le bit Start du bloc XXMIT est à 1 dans un automate Quantum avec redondance d'UC qui n'est pas en mode Primaire.

NOTE : 1) Ce code d'erreur apparaît en cas de réponse trop rapide de l'esclave Modbus.

Si l'esclave Modbus utilisé est un automate Modicon, vérifiez sa configuration de port Modbus.

2) Pour le bloc XXMIT en mode Modbus maître, si l'erreur 116 persiste après une configuration correcte à l'aide du clavier ou de modifications en ligne, reconstruisez et transférez l'application sur l'automate.

Informations supplémentaires pour l'erreur 107

L'erreur 107 apparaît en cas de détection de différence de configuration XXMIT. Si le paramètre de port interne XXMIT est en mode RS232 et que le mot de commande est en mode RS485 (#2100). Le mode de communication doit être modifié.

Le tableau ci-dessous montre comment modifier le mode de communication en cas d'apparition de l'erreur 107 :

Etape	Opération
1	Connectez le PC à Unity
2	Placez l'automate en mode STOP
3	Modifiez les paramètres de port Modbus (RS232 ou RS 485)
4	Modifiez le mot de commande en fonction du mode demandé
5	Reconstruisez l'application
6	Téléchargez l'application
7	Placez l'automate en mode RUN

NOTE : Vérifiez que :

- Le passage du mode RS232 à RS485 prend 3 secondes quand les paramètres de ports internes et des mots de commande correspondent. Pendant ces 3 secondes le code d'erreur 126 (DCE non évalué, seulement pour le mode RS232) apparaît.
- Le passage du mode RS485 à RS232 est immédiat car il correspond à la configuration par défaut.

Tentative

La valeur affichée indique le nombre de tentatives effectuées par le bloc XXMIT. Cet élément est en lecture seule.

Fonctions de communication du bloc XXMIT

Mot de commande du bloc XXMIT

Le bloc XXMIT exécute les six fonctions illustrées ci-dessous. Pour chaque fonction, certains bits du mot de commande doivent être activés.

Bits du mot de commande

Fonctions du mot de commande en fonction des bits

Fonction	Bits du mot de commande qui doivent être définis sur 1	Bits qui DOIVENT être définis sur 0
Entrée ASCII terminée (bit 11=1) - Voir remarque ci-dessous.	14,7,6,5,4	10,9,8,3,2,1,0
Entrée ASCII simple (bit 10=1) - Voir remarque ci-dessous.	14, 7, 6, 5, 4	11, 9, 8, 3, 2, 1, 0
Sortie ASCII simple (bit 9=1)	14, 7, 6, 5, 4	11, 10, 8, 3, 2, 1, 0
Sortie modem (bit 9=1)	14, 3, 2, 1, 0	11, 10, 8, 7, 6, 5, 4 (et un SEUL des bits 3, 2, 1 ou 0 est défini sur 1, tandis que les trois autres bits doivent être définis sur 0)
Sortie des messages du maître Modbus (bit 8=1)	14	11, 10, 9, 7, 6, 5, 4, 3, 2, 1, 0

NOTE : Lorsque vous utilisez l'une de ces fonctions, vous DEVEZ régler le bit 7 (Validation du tampon FIFO de réception ASCII) sur 1. Le bit 15 (MSB) et les bits 13 et 12 sont réservés. (Voir le tableau relatif au *Command*, page 341.)

Fonctions ASCII du bloc XXMIT

Vue d'ensemble

Le bloc XXMIT prend en charge les fonctions de communication ASCII suivantes :

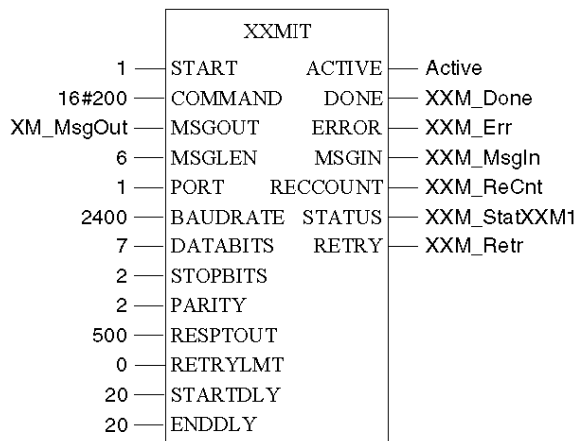
- Entrée ASCII terminée
- Entrée ASCII simple
- Messages sous forme de chaînes ASCII

Configuration du port Modbus

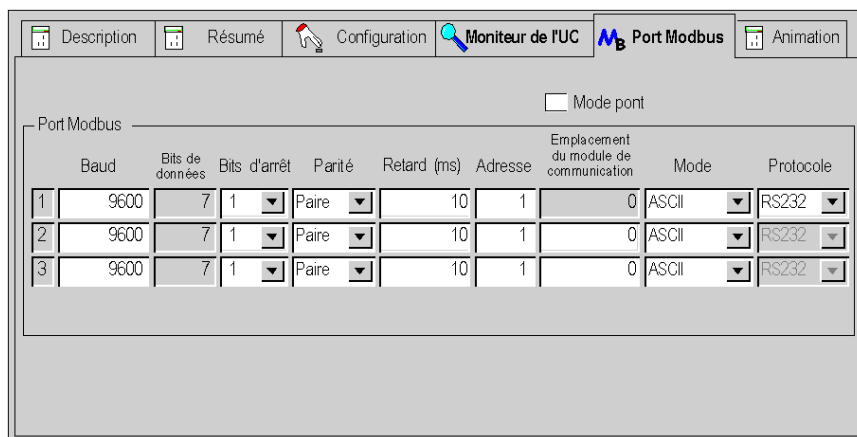
La fonction XXMIT est configurée à travers son bloc fonction. Le bloc fonction initialise le port Modbus 1 chaque fois qu'il est actif, avec les paramètres définis. Les paramètres XXMIT peuvent être affichés sur l'écran LCD de l'automate.

NOTE : les paramètres définis sur l'écran de configuration de l'UC ou au moyen de l'écran LCD de l'automate n'ont aucun effet sur les valeurs de transmission en mode maître.

Exemple de configuration du bloc fonction XXMIT :



Exemple d'écran de configuration d'UC avec un ensemble de valeurs différent :



Exemples de paramètres sur l'écran LCD de l'automate :

```
232 MB   ASCII: 1
9600,Even,7,2 >
```

Les trois graphiques ci-dessus présentent des valeurs différentes pour la fonction XXMIT. Indépendamment des éléments définis ou affichés sur l'écran de configuration de l'UC ou sur l'écran LCD, les valeurs réelles utilisées pour la transmission sont celles du bloc fonction XXMIT :

- Débit : 2400
- Parité : Paire
- Bits de données : 7
- Bits d'arrêt : 2

Fonction d'entrée ASCII terminée

Lorsque le bit 11 (Entrée ASCII terminée) du mot de commande est à 1, le tableau MsgOut doit contenir le tableau de définition d'entrée ASCII. En fonction du type de données sélectionné pour MsgOut, la longueur du tableau de définition d'entrée ASCII terminée est de trois mots ou six octets. Le tableau de définition d'entrée ASCII terminée est présenté ci-dessous.

Tableau de définition d'entrée ASCII terminée (type de données WordArray)

Mot	Octet de poids fort	Octet de poids faible
MsgOut[1]	Nombre de caractères de départ (contenu autorisé = 0, 1, 2)	Nombre de caractères de fin (contenu autorisé = 1, 2)
MsgOut[2]	Premier caractère de départ	Deuxième caractère de départ
MsgOut[3]	Premier caractère de fin	Deuxième caractère de fin

Tableau de définition d'entrée ASCII terminée (type de données ByteArray)

Octet	Fonction
MsgOut[1]	Longueur de la chaîne de départ (0, 1 ou 2)
MsgOut[2]	Longueur de la chaîne de fin (1 ou 2)
MsgOut[3]	Premier caractère de départ
MsgOut[4]	Deuxième caractère de départ
MsgOut[5]	Premier caractère de fin
MsgOut[6]	Deuxième caractère de fin

Au cours du processus, RecCount contient le cumul des caractères écrits dans le tableau MsgIn. Après réception de la chaîne de fin, la sortie Done du bloc XXMIT passe à 1 et RecCount contient la longueur totale de la chaîne reçue, y compris les chaînes de départ et de fin. A ce stade, le bloc XXMIT contrôle toujours le port et continue d'enregistrer les caractères récemment reçus dans le tampon FIFO de réception ASCII, car le bit 7 (Validation du tampon FIFO de réception ASCII) du mot de commande est à 1.

A l'aide du programme, vous pouvez mettre le bit Entrée ASCII simple à 0 avant le prochain cycle, tout en laissant le bit Validation du tampon FIFO de réception ASCII à 1. MsgIn n'est donc PAS écrasé par les données FIFO plus récentes, toujours collectées. A l'aide du programme, vous pouvez mettre à 0 le bit 7 (Validation du tampon FIFO de réception ASCII) et le bit 11 (Entrée ASCII terminée) pour redonner le contrôle du port à l'automate.

Lorsque le tableau MsgIn contient trop de caractères et qu'AUCUN caractère de fin n'a été détecté, ou que le tableau MsgIn est en dehors de la plage autorisée pour l'automate configuré, une erreur est générée dans Status et le paramètre RecCount n'est pas significatif. La limite de caractères est le nombre le plus petit entre 1024 et deux fois la taille du tableau MsgIn.

Exemple d'entrée ASCII terminée

Supposez que le bloc XXMIT est activé avec les bits 7 et 11 du mot de commande. Validez le tampon FIFO ASCII et l'entrée ASCII terminée. Le port reçoit la chaîne ASCII suivante : "AMScrlf\$weight = 1245 GRAMScrlf\$wei". Reportez-vous au tableau de définition d'entrée ASCII présentant le contenu indiqué par () utilisé dans cet exemple.

Tableau de définition d'entrée ASCII terminée (contenu pour le type de données ByteArray)

Octet	Contenu
MsgOut[1]	Nombre de caractères de départ (0x01)
MsgOut[2]	Nombre de caractères de fin (0x02)
MsgOut[3]	Premier caractère de départ ("\$")
MsgOut[4]	Deuxième caractère de départ (inutilisé)
MsgOut[5]	Premier caractère de fin ("cr")
MsgOut[6]	Deuxième caractère de fin ("lf")

Exemple de tableau de définition d'entrée ASCII terminée (contenu pour le type de données WordArray)

Mot	Octet de poids fort	Octet de poids faible
MsgOut[1]	Nombre de caractères de départ (0x01)	Nombre de caractères de fin (0x02)
MsgOut[2]	Premier caractère de départ ("\$")	Deuxième caractère de départ (inutilisé)
MsgOut[3]	Premier caractère de fin ("cr")	Deuxième caractère de fin ("lf")

Le bloc XXMIT s'ACTIVE, puis supprime les cinq premiers caractères du tampon FIFO ("AMScrlf"), car ils ne correspondent pas au premier caractère de départ ("\$"). Au cours du cycle logique, après réception de "\$", le bloc XXMIT reste ACTIF et copie le "\$" et les caractères suivants dans le tableau MsgIn, en mettant à jour RecCount avec le décompte effectué jusque-là, au fur et à mesure de l'entrée des caractères.

Après réception du dernier caractère de fin, la sortie Done est activée et MsgLen contient la longueur totale, égale à 22 caractères (0x0016). Le tableau MsgIn comporte : "\$weight = 1245 GRAMScrlf" pour le type de données ByteArray (ou : "\$w", "ei", "gh", "t", "=", "12", "45", "G", "RA", "MS", "crlf" pour le type de données WordArray).

Au cours du cycle où la sortie Done est activée, les caractères déjà reçus du message suivant ("\$wei"), arrivés après la chaîne de fin, restent dans le tampon FIFO d'entrée ASCII. Le programme peut ainsi désactiver l'entrée ASCII terminée avant l'exécution du cycle suivant du bloc XXMIT pour ce port, en gardant les caractères dans le tampon FIFO jusqu'à ce que l'automate ait terminé le traitement du message courant, ce qui peut prendre plusieurs cycles.

Fonction d'entrée ASCII simple

Tous les caractères entrants sont placés dans le tableau MsgIn. Si MsgIn est réglé sur ByteArray (recommandé), les caractères entrants sont simplement stockés ainsi : le premier caractère dans le premier élément de tableau, le deuxième caractère dans le deuxième, etc. Si MsgIn est réglé sur WordArray, deux caractères sont stockés dans chaque élément. Le premier caractère est stocké dans l'octet de poids fort du premier élément. Le deuxième caractère est stocké dans l'octet de poids faible du premier élément. Le troisième caractère est stocké dans l'octet de poids fort du deuxième élément, etc. La variable de longueur du message (MsgLen) contient la longueur du message (1 à 1 024 caractères).

NOTE : lorsque le bit 10 (Entrée ASCII simple) et le bit 7 (Tampon FIFO de réception ASCII) restent à 1, de nouveaux caractères sont constamment transférés du tampon FIFO vers le même tableau MsgIn ; ils écrasent donc constamment les caractères précédents qui y étaient stockés.

Messages sous forme de chaînes ASCII

Lorsque le bit 9 (Messages sous forme de chaînes) du mot de commande est à 1, le tableau MsgOut doit contenir les informations ASCII à transmettre. La longueur du message MsgLen doit être définie en fonction de la longueur du message à transmettre.

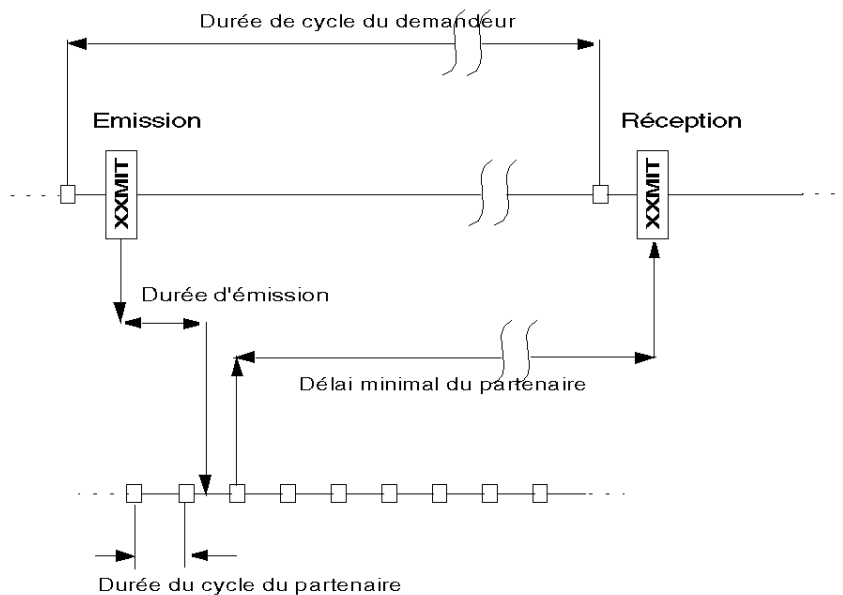
Comme indiqué dans la section Description détaillée des paramètres (*voir page 341*), MsgOut peut être de n'importe quel type de données. Pour les messages sous forme de chaînes ASCII, le type de données ByteArray reflète mieux la nature des chaînes : le premier octet contient le premier caractère, et ainsi de suite (pour plus d'informations, voir Envoi ASCII simple (*voir page 376*)).

Passage du mode d'émission au mode de réception

Si votre application exige la réception d'une réponse d'un autre équipement à la suite de l'émission d'un message (requête-réponse), vous aurez besoin du bloc XXMIT pour passer du mode d'émission au mode de réception et ainsi pouvoir lire la réponse fournie par votre partenaire de communication. Le passage au mode de réception du bloc XXMIT peut se faire au plus tôt au cours du cycle qui suit l'opération d'émission. Il incombe à l'utilisateur de s'assurer que la réponse intervient au moins un cycle après la demande de l'automate afin d'éviter l'échec de la communication.

Le délai de transmission de votre partenaire de communication est particulièrement important en cas de cycles longs pour le demandeur et de partenaires rapides.

Considérations relatives au délai du partenaire :



Le chiffre ci-dessus (qui n'est pas à l'échelle) vous permet de déterminer l'influence des durées de cycle et de transmission du demandeur ainsi que de la durée de cycle du partenaire étant asynchrones, celui du partenaire ne doit pas être pris en compte. La durée de transmission dépend de la longueur du télégramme envoyé et du débit en bauds. Par exemple, l'envoi d'un message comportant 18 caractères à 9 600 bauds nécessite 14 ms. Le cycle du demandeur représente sans nul doute la majeure partie de cette durée. Par conséquent, même si le délai minimal du partenaire peut être inférieur à la durée du cycle du demandeur, il est recommandé d'utiliser celle-ci comme délai minimal du partenaire en vue de garantir une communication efficace.

Fonctions de modem du bloc XXMIT

Présentation

Le bloc XXMIT vous permet de communiquer avec un modem compatible Hayes à l'aide des fonctions répertoriées dans le tableau suivant :

Fonctions de modem

Bit dans le mot de commande	Fonction
Bit 3	Modem à numérotation par impulsion
Bit 2	Raccrochage du modem
Bit 1	Modem à numérotation à tonalité
Bit 0	Initialisation du modem

Modem à numérotation par impulsion

Mettez le bit 3 du mot de commande à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez composer un numéro de téléphone en utilisant la numérotation par impulsion. Programmez le numéro de téléphone dans le tableau MsgOut. La longueur du message doit être définie dans MsgLen. Les numéros composés par impulsion sont envoyés au modem, automatiquement précédés d'ATDP et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le message de composition étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du numéro à composer.

Raccrochage du modem

Mettez le bit 2 du mot de commande à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes si vous souhaitez raccrocher le modem. Vous devez utiliser le programme pour mettre ce bit à 1. Le message de raccrochage étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du message. Les messages raccrochés sont envoyés au modem, automatiquement précédés de +++AT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le bloc XXMIT recherche une réponse de déconnexion correcte de la part du modem avant d'ACTIVER le signal de sortie Done qui indique que l'exécution a réussi.

Modem à numérotation à tonalité

Mettez le bit 1 du mot de commande à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez composer un numéro de téléphone en utilisant la numérotation à tonalité. Programmez le numéro de téléphone dans le tableau MsgOut. La longueur du message doit être définie dans MsgLen. Les numéros composés au clavier sont envoyés au modem, automatiquement précédés d'ATDT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le message de composition étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du numéro à composer.

Initialisation du modem

Mettez le bit 0 du mot de commande à 1 lorsque vous utilisez un modem à numérotation automatique compatible Hayes et que vous souhaitez initialiser le modem. Programmez le message d'initialisation dans le tableau MsgOut et la longueur du message dans MsgLen. Tous les messages sont envoyés au modem, automatiquement précédés d'AT et suivis d'un retour chariot <CR> et d'un retour à la ligne <LF>. Le message d'initialisation étant une chaîne ASCII, le bit 9 doit être à 1 avant l'envoi du message.

Fonctions Modbus du bloc XXMIT

Vue d'ensemble

Le bloc XXMIT prend en charge les codes de fonction Modbus suivants :

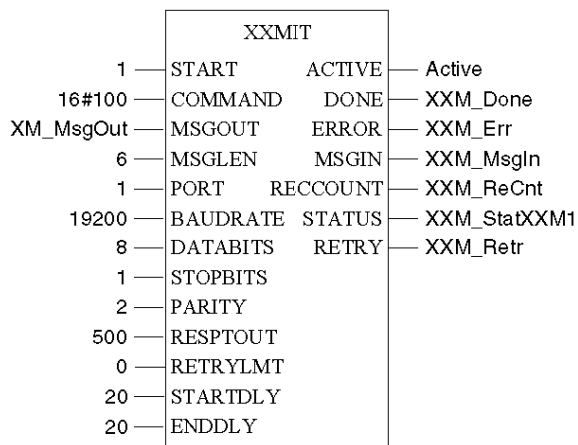
- 01 ... 06 et 15 à 16
- 08

Configuration du port Modbus

La fonction XXMIT est configurée à travers son bloc fonction. Le bloc fonction initialise le port Modbus 1 chaque fois qu'il est actif, avec les paramètres définis. Les paramètres XXMIT peuvent être affichés sur l'écran LCD de l'automate.

NOTE : les paramètres définis sur l'écran de configuration de l'UC ou au moyen de l'écran LCD de l'automate n'ont aucun effet sur les valeurs de transmission en mode maître.

Exemple de configuration du bloc fonction XXMIT :



Exemple d'écran de configuration d'UC avec un ensemble de valeurs différent :

	Baud	Bits de données	Bits d'arrêt	Parité	Retard (ms)	Adresse	Emplacement du module de communication	Mode	Protocole
1	9600	8	1	Paire	10	1	0	RTU	RS232
2	9600	8	1	Paire	10	1	0	RTU	RS232
3	9600	8	1	Paire	10	1	0	RTU	RS232

Exemples de paramètres sur l'écran LCD de l'automate :

```
232 MB      RTU: 1
9600,Even,8,1 >
```

Les trois graphiques ci-dessus présentent des valeurs différentes pour la fonction XXMIT. Indépendamment des éléments définis ou affichés sur l'écran de configuration de l'UC ou sur l'écran LCD, les valeurs réelles utilisées pour la transmission sont celles du bloc fonction XXMIT :

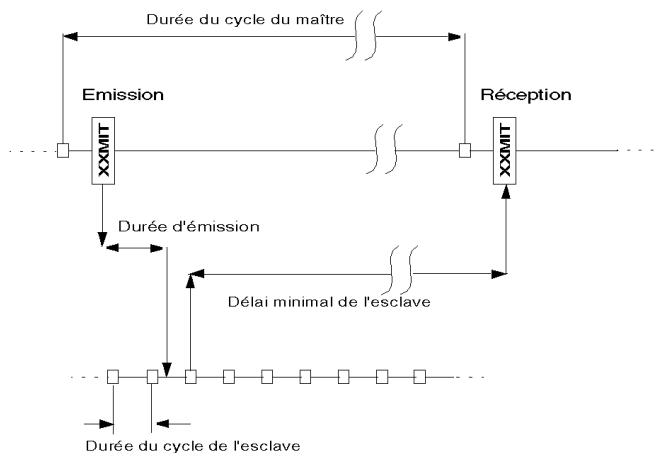
- Débit : 19200
- Parité : Paire
- Bits de données : 8
- Bits d'arrêt : 1

Passage du mode d'émission au mode de réception

A l'exception des messages de diffusion, toutes les fonctions Modbus nécessitent que le bloc XXMIT passe du mode d'émission au mode de réception afin de lire la réponse de l'esclave. Le bloc fonction XXMIT passe en mode de réception au cours du cycle qui suit l'opération d'émission. Il incombe à l'utilisateur de s'assurer que la réponse de l'esclave intervient au moins un cycle après celle du maître afin d'éviter l'échec de la communication.

Le délai de transmission de l'esclave est particulièrement important en cas de cycles longs pour le maître et d'esclaves rapides.

Considérations relatives au délai de l'esclave :



Le chiffre ci-dessus (qui n'est pas à l'échelle) vous permet de déterminer l'influence des durées de cycle et de transmission du maître ainsi que de la durée du cycle de l'esclave sur le délai requis pour ce dernier. Les cycles du maître et de l'esclave étant asynchrones, celui de l'esclave ne doit pas être pris en compte. La durée de transmission dépend du type de télégramme envoyé, du débit en bauds et du protocole utilisé. Par exemple, une demande de lecture standard à 9600 bauds utilisant le protocole ASCII dure 14 ms. Le cycle du maître représente sans nul doute la majeure partie de cette durée. Par conséquent, même si le délai minimal de l'esclave peut être inférieur à la durée du cycle du maître, il est recommandé d'utiliser ce dernier comme délai minimal de l'esclave en vue de garantir une communication efficace.

NOTE : pour les automates Quantum, vous pouvez préciser le délai dans la boîte de dialogue de configuration du port Modbus. Ce délai doit être compris entre 10 et 1000 ms, la valeur que vous indiquez étant automatiquement arrondie en un multiple de 10.

Vous devez saisir le délai requis **plus** 10 ms. Par exemple, si vous voulez instaurer un délai de 110 ms, saisissez 120 dans ce champ.

Codes de fonction Modbus (01 à 06, 15 et 16)

Pour les messages Modbus, le tableau MsgOut doit contenir le tableau de définition Modbus. Il doit être défini comme un champ de mots. Le tableau de définition Modbus pour le code de fonction Modbus 01, 02, 03, 04, 05, 06, 15 et 16 occupe cinq mots et vous devez paramétrer MsgLen sur 5 pour que l'opération XXMIT réussisse. Le tableau de définition Modbus est présenté ci-dessous.

Codes de fonction du tableau de définition Modbus (01 à 06, 15 et 16)

Contenu	Description
Code de fonction Modbus (MsgOut[1])	Le bloc XXMIT prend en charge les codes de fonction suivants : 01 = Lecture de plusieurs bits (%Q) 02 = Lecture de plusieurs bits TOR (%I) 03 = Lecture de plusieurs mots (%MW) 04 = Lecture de plusieurs mots d'entrée (%IW) 05 = Ecriture d'un seul bit (%Q) 06 = Ecriture d'un seul mot (%MW) 15 = Ecriture de plusieurs bits (%Q) 16 = Ecriture de plusieurs mots (%MW)
Quantité (MsgOut[2])	Entrez la quantité de données à écrire ou à lire dans l'automate esclave. Par exemple, entrez 100 pour lire 100 mots d'un automate esclave ou entrez 32 pour écrire 32 bits dans un automate esclave. Il existe une taille limite, qui dépend du modèle de l'automate. Reportez-vous à l'annexe A pour obtenir tous les détails sur les limites.
Adresse de l'automate esclave (MsgOut[3])	Entrez l'adresse de l'automate Modbus esclave. La plage des adresses Modbus va généralement de 1 à 247. Pour envoyer un message Modbus à plusieurs automates, entrez 0 comme adresse de l'automate esclave. Ce type de transmission est appelé mode diffusion. Le mode diffusion prend uniquement en charge les codes de fonction Modbus écrivant des données de l'automate maître vers des automates esclaves. Il NE prend PAS en charge les codes de fonction Modbus lisant des données des automates esclaves.

Contenu	Description
Zone de données de l'automate esclave (MsgOut[4])	Pour une commande de lecture, la zone de données de l'automate esclave est la source des données. Pour une commande d'écriture, la zone de données de l'automate esclave est la destination des données. Par exemple, lorsque vous lisez les bits (%I300 à %I500) d'un automate esclave, entrez 300 dans ce champ. Si vous voulez écrire des données d'un automate maître dans les mots (%MW100) d'un automate esclave, entrez 100 dans ce champ. Selon le type de commande Modbus (lecture ou écriture), les zones de données source et cible doivent être conformes à celles du tableau ci-dessous.
Zone de données de l'automate maître (MsgOut[5])	Pour une commande de lecture, la zone de données de l'automate maître est la destination des données renvoyées par l'esclave. Pour une commande d'écriture, la zone de données de l'automate maître est la source des données. Par exemple, lorsque vous voulez écrire des bits (%M16 à %M32) de l'automate maître dans un automate esclave, entrez 16 dans ce champ. Lorsque vous lisez les mots (%IW1 à %IW100) d'un automate esclave et placez les données dans le champ de données de l'automate maître (%MW100 à %MW199), saisissez 100 dans ce champ. Selon le type de commande Modbus (lecture ou écriture), les zones de données source et cible doivent être conformes à celles du tableau ci-dessous.

Zones de données source et cible pour les codes de fonction (01 à 06, 15 et 16)

Code de fonction	Zone de données de l'automate maître	Zone de données de l'automate esclave
03 (Lecture de plusieurs registres 4x)	%MW (cible)	%MW (source)
04 (Lecture de plusieurs registres 3x)	%MW (cible)	%IW (source)
01 (Lecture de plusieurs références 0x)	%M (cible)	%Q (source)
02 (Lecture de plusieurs références 1x)	%M (cible)	%I (source)
16 (Ecriture de plusieurs registres 4x)	%MW (source)	%MW (cible)
15 (Ecriture de plusieurs références 0x)	%M (source)	%Q (cible)
05 (Ecriture d'une seule référence 0x)	%M (source)	%Q (cible)
06 (Ecriture d'un seul registre 4x)	%MW (source)	%MW (cible)

Lorsque vous souhaitez envoyer 20 messages Modbus à partir de l'automate, vous devez transférer 20 tableaux de définition Modbus un par un dans MsgOut après chaque exécution réussie du bloc XXMIT ou programmer 20 blocs XXMIT différents, puis les activer un par un à partir du programme utilisateur.

Code de fonction Modbus (08)

Pour les messages Modbus, le tableau MsgOut doit contenir le tableau de définition Modbus. Il doit être défini comme un champ de mots. Le tableau de définition Modbus pour le code de fonction Modbus 08 occupe cinq mots et vous devez paramétrer MsgLen sur 5 pour que l'opération XXMIT réussisse. Le tableau de définition Modbus est présenté ci-dessous.

Codes de fonction du tableau de définition Modbus (08)

Contenu	Description	
Code de fonction Modbus (MsgOut[1])	Le bloc XXMIT prend en charge le code de fonction suivant : 08 = Diagnostic	
Diagnostic (MsgOut[2])	Entrez la valeur décimale du code de sous-fonction du diagnostic dans ce champ pour exécuter la fonction de diagnostic voulue. Les sous-fonctions de diagnostic suivantes sont prises en charge :	
	Code de sous-fonction	Description
	00	Interrogation en retour
	01	Relancer l'option comm.
	02	Renvoi du mot de diagnostic
	03	Changement de séparateur d'entrée ASCII
	04	Mode écoute seul
	05 ... 09	Réservé
	10	Effacer compteurs
	11	(et mots de diagnostic dans 384, 484)
	12	Renvoi du compte de messages bus
	13	Renvoi du compte d'erreurs de comm. bus
	14 ... 15	Renvoi du compte d'exceptions de bus
	16	Non pris en charge
	17	Donner le compte de NAK esclave
	18	Donner le compte d'esclaves occupés
19 ... 21	Renvoi du compte de dépassement de car. de bus	

Contenu	Description						
Adresse de l'automate esclave (MsgOut[3])	Entrez l'adresse de l'automate Modbus esclave. La plage des adresses Modbus va généralement de 1 à 247. Le code de fonction 8 NE prend PAS en charge le mode diffusion (adresse 0).						
Contenu du champ des données de fonction de diagnostic (MsgOut[4])	Vous devez saisir la valeur décimale nécessaire pour la zone de données de la sous-fonction de diagnostic particulière :						
	<table border="1"> <thead> <tr> <th>Code de sous-fonction</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>02, 04, 10, 11, 12, 13, 16, 17 et 18</td> <td>Cette valeur est automatiquement réglée sur 0 dans le message Modbus envoyé à l'esclave (elle n'est pas reflétée dans le tampon MsgOut).</td> </tr> <tr> <td>00, 01 et 03</td> <td>Vous devez saisir une valeur dans le champ de données. Pour plus de détails, reportez-vous au document <i>Modicon Modbus Protocol Reference Guide</i> (www.modbus.org).</td> </tr> </tbody> </table>	Code de sous-fonction	Description	02, 04, 10, 11, 12, 13, 16, 17 et 18	Cette valeur est automatiquement réglée sur 0 dans le message Modbus envoyé à l'esclave (elle n'est pas reflétée dans le tampon MsgOut).	00, 01 et 03	Vous devez saisir une valeur dans le champ de données. Pour plus de détails, reportez-vous au document <i>Modicon Modbus Protocol Reference Guide</i> (www.modbus.org).
	Code de sous-fonction	Description					
02, 04, 10, 11, 12, 13, 16, 17 et 18	Cette valeur est automatiquement réglée sur 0 dans le message Modbus envoyé à l'esclave (elle n'est pas reflétée dans le tampon MsgOut).						
00, 01 et 03	Vous devez saisir une valeur dans le champ de données. Pour plus de détails, reportez-vous au document <i>Modicon Modbus Protocol Reference Guide</i> (www.modbus.org).						
Zone de données de l'automate maître (MsgOut[5])	Pour toutes les sous-fonctions, la zone de données de l'automate maître est la destination des données renvoyées par l'esclave. Vous devez indiquer un mot mémoire %MW marquant le début de la zone de données dans laquelle les données renvoyées sont placées. Par exemple, pour placer les données dans la zone de données de l'automate maître commençant à (%MW100), saisissez 100 dans ce champ. La sous-fonction 04 NE RENVOIE PAS de réponse. Pour plus de détails, reportez-vous au document <i>Modicon Modbus Protocol Reference Guide</i> (www.modbus.org).						

Tampon FIFO et contrôle de flux

Présentation

Le bloc XXMIT permet à l'utilisateur de définir l'utilisation d'un tampon FIFO de réception, du contrôle de flux et de la fonction des caractères de retour arrière reçus.

Tampon FIFO de réception ASCII

La mise à 0 du bit 7 du mot de commande désactive la fonction. Lorsque le tampon FIFO reçoit 512 caractères, un débordement interne se produit. Dans ce cas, tous les caractères suivants sont ignorés, toutes les opérations d'entrée ASCII (au format simple et terminé) sont arrêtées et le bloc renvoie une erreur jusqu'à ce que vous basculiez l'état du bit 7. Lorsque vous basculez l'état du bit 7, toutes les données du tampon FIFO sont supprimées, les deux bits de contrôle des entrées ASCII sont ignorés (Entrée ASCII simple [bit 10] et Entrée ASCII terminée [bit 11]) et lorsque aucun contrôle de sortie ASCII n'est sélectionné, le contrôle du port série (1) revient à l'automate.

Vous devez mettre à 1 le bit 11 (Entrée ASCII terminée) ou le bit 10 (Entrée ASCII simple) pour supprimer les caractères ASCII du tampon FIFO pour le traitement. Un seul des trois bits suivants peut être activé à la fois : Entrée ASCII terminée (bit 11), Entrée ASCII simple (bit 10) ou Messages sous forme de chaînes ASCII (bit 9).

L'opération Full Duplex peut être réalisée en mettant à 1 à la fois le bit 7 (Tampon FIFO de réception ASCII) et le bit 9 (Messages sous forme de chaînes ASCII). Vous pouvez ainsi envoyer une entrée ASCII simple à partir de l'automate tout en continuant à recevoir des caractères ASCII dans le tampon FIFO. Cette fonction est utile lors de l'utilisation de terminaux muets. Lorsque le bit 7 (Tampon FIFO de réception ASCII) est à 1, aucun des contrôles de sortie ASCII suivants n'est autorisé : Messages Modbus maîtres (bit 8), Modem à numérotation par impulsion (bit 3), Raccrochage du modem (bit 2), Modem à numérotation à tonalité (bit 1) et Initialisation du modem (bit 0).

Validation du caractère de retour arrière

Lorsqu'un caractère de retour arrière (BS) est détecté, il N'EST PAS stocké dans le tableau MsgIn ; en fait, il supprime le caractère précédent et diminue donc le compteur de caractères RecCount. Par comparaison, lorsqu'un caractère ASCII standard est détecté, il est stocké dans le tableau MsgIn et augmente le compteur de caractères RecCount.

NOTE : Les caractères de retour arrière NE PEUVENT PAS supprimer les caractères d'un tableau MsgIn vide. Par conséquent, le compteur de caractères RecCount ne peut jamais être inférieur à 0.

Cette fonctionnalité spéciale de caractère de retour arrière et un écho interne activé sur le terminal sont très utiles lors de l'utilisation de terminaux muets. Un seul bloc XXMIT d'entrée ASCII terminée recherchant un "cr" est activé lorsque le bit 7 (Tampon FIFO de réception ASCII) et le bit 6 (Caractère de retour arrière) sont à 1. Aucun autre programme n'est nécessaire lorsque vous entrez et éditez des caractères en utilisant la touche retour arrière à la volée. Lorsque vous tapez "cr", le bloc XXMIT active la sortie Done et les données corrigées sont toutes alignées correctement dans le tableau MsgIn.

Validation du contrôle de flux RTS/CTS

Les informations suivantes s'appliquent au mode de sortie. L'état du bloc XXMIT passe à BLOQUE lorsque l'équipement récepteur indique qu'il ne peut traiter les caractères supplémentaires en désactivant le signal CTS. De même, l'état du bloc XXMIT passe à DEBLOQUE lorsque le signal CTS est activé et que l'équipement récepteur indique qu'il PEUT traiter des caractères supplémentaires.

Lorsque l'émission est DEBLOQUEE et que le bit 9 (Sortie ASCII simple) et le bit 5 (Contrôle de flux RTS/CTS) sont à 1, les données de la sortie d'émission sont envoyées par paquets de 16 octets. Une fois tous les paquets envoyés, la sortie Done du bloc XXMIT est activée pour indiquer la réussite de l'opération.

Si, au cours d'une émission, elle passe soudainement à BLOQUEE, seuls les caractères restants dans le paquet de sortie courant sont envoyés (16 caractères au plus) et le bloc XXMIT reste ACTIF indéfiniment. La sortie ASCII reprend l'envoi de tous les paquets de sortie restants dès l'instant où le signal CTS est activé.

Les informations suivantes s'appliquent au mode d'entrée. Le signal RTS étant un signal de sortie, il peut être utilisé indépendamment du processus d'émission de sortie ASCII pour BLOQUER ou DEBLOQUER des équipements émetteurs. Lorsque le bit 7 (Tampon FIFO de réception ASCII) est à 1, le contrôle de flux RTS/CTS fonctionne en mode d'entrée. Lorsque le bit 7 (Tampon FIFO de réception ASCII) est à 1 et qu'aucune des deux entrées ASCII n'est activée (Entrée ASCII simple [bit 10] ou Entrée ASCII terminée [bit 11]), les caractères reçus remplissent le tampon FIFO dans lequel ils sont insérés. Au même moment, le contrôle de flux RTS (bit 5) est activé, permettant ainsi à l'équipement émetteur de poursuivre ses opérations.

Lorsque le tampon FIFO (512 caractères) est plus qu'aux trois-quarts plein, le bit 5 (Contrôle de flux RTS) est mis à 0 pour BLOQUER l'équipement émetteur. Ce bit reste à 0 jusqu'à ce que l'entrée ASCII simple (bit 10) ou l'entrée ASCII terminée (bit 11) ait supprimé assez de caractères du tampon FIFO pour réduire son contenu à moins d'un quart et permettre au bit 5 (Contrôle de flux RTS) de passer à 1.

ATTENTION

VERROUILLAGE DE LA TRANSMISSION

Si l'équipement récepteur bloque indéfiniment l'activation du signal CTS (ou en cas de problème de câblage qui a pour effet de désactiver le signal CTS), le bloc XXMIT ne termine jamais la transmission et l'équipement récepteur ne reçoit jamais la commande. Pour éviter ce type de verrouillage, il est conseillé de démarrer un temporisateur au démarrage du bloc XXMIT et de prévenir le programme d'application lorsque le délai interne défini par le temporisateur est écoulé. Il est également conseillé de réinitialiser le bit de démarrage XXMIT.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

NOTE : L'algorithme du contrôle de flux RTS/CTS est différent de celui du contrôle modem RTS/CTS. Le premier est lié au débordement du tampon de réception duplex. Le second concerne l'accès du processus de transmission à un support de transmission partagé. Par conséquent, il est interdit de demander simultanément ces deux algorithmes RTS/CTS.

NOTE : Vous NE POUVEZ PAS sélectionner n'importe quel type de contrôle de flux RTS/CTS (bit 5) lorsque le port est en mode RS 485 (bit 13), car ces signaux N'EXISTENT PAS en mode RS 485.

Validation du contrôle de flux Xon/Xoff

Les informations suivantes s'appliquent au mode de sortie. L'état du bloc XXMIT passe à BLOQUE lors de la réception d'un caractère Xoff. De même, l'état du bloc XXMIT passe à DEBLOQUE lors de la réception d'un caractère Xon. Xon ou Xoff ne sont en aucun cas insérés dans le tampon FIFO.

Lorsque l'émission est DEBLOQUEE et que le bit 9 (Sortie ASCII simple) et le bit 4 (Contrôle de flux Xon/Xoff) sont à 1, les données de la sortie d'émission sont envoyées par paquets de 16 octets. Une fois tous les paquets envoyés, la sortie Done du bloc XXMIT est activée.

Si, au cours d'une transmission, elle passe soudainement à BLOQUEE, seuls les caractères restants dans le paquet de sortie courant sont envoyés (16 caractères au plus) et le bloc XXMIT reste ACTIF indéfiniment. Ce n'est que lors de la réception du caractère Xon suivant que la sortie ASCII reprend l'envoi de tous les paquets de sortie restants.

Les informations suivantes s'appliquent au mode d'entrée. Le signal Xon/Xoff peut être utilisé pour BLOQUER ou DEBLOQUER des équipements émetteurs. Lorsque le bit 7 (Tampon FIFO de réception ASCII) est à 1, le contrôle de flux Xon/Xoff (bit 4) fonctionne en mode d'entrée. Lorsque le bit 7 (Tampon FIFO de réception ASCII) est à 1 et qu'aucune des deux entrées ASCII n'est activée (Entrée ASCII simple [bit 10] ou Entrée ASCII terminée [bit 11]), les caractères reçus remplissent le tampon FIFO dans lequel ils sont insérés.

Lorsque le tampon FIFO est plus qu'aux trois-quarts plein et qu'il reçoit des caractères supplémentaires, la variable d'état du tampon FIFO est mise à 1. Elle entraîne l'envoi de caractères XOFF à partir du port série après un retard de 16 caractères BLOQUANT l'émetteur et mettant à 0 la variable d'état du tampon FIFO.

Lorsque toutes les fonctions de sortie ASCII (bits 8, 3, 2, 1 et 0) sont désactivées et que le contrôle de flux Xon/Xoff (bit 4) est activé, le retard est par défaut d'un caractère. Par comparaison, lorsque toutes les fonctions de sortie ASCII (bits 8, 3, 2, 1 et 0) sont activées et que le contrôle de flux Xon/Xoff (bit 4) est activé, la sortie ASCII est répartie dans des paquets de 16 octets. Par conséquent, les caractères Xoff en attente N'ONT PAS BESOIN d'attendre que le temps nécessaire à la transmission de 16 caractères soit écoulé pour BLOQUER l'émetteur.

Une fois que l'émetteur a interrompu l'émission, l'automate supprime finalement les caractères du tampon FIFO en utilisant le bit 10 (Entrée ASCII simple) ou le bit 11 (Entrée ASCII terminée).

Lorsque le tampon FIFO est rempli à moins d'un quart, sa variable d'état est mise à 1 de façon à permettre l'envoi d'un caractère XON à partir du port série pour DEBLOQUER l'émetteur.

NOTE : Pour empêcher le verrouillage suite à la déconnexion d'un câble ou à d'autres erreurs de communication intermittentes lorsque l'émetteur est BLOQUE et qu'il n'a PAS reçu le caractère Xon correctement, nous utilisons l'algorithme suivant. Lorsque le tampon FIFO devient vide et qu'il ne reçoit aucun caractère par la suite, une chaîne régulière de caractères Xon est émise une fois toutes les 5 secondes.

NOTE : Le contrôle de flux Xon/Xoff (bit 4) est différent du contrôle de flux RTS/CTS (bit 5). Le premier utilise les caractères Xon et Xoff émis pour éviter le débordement du tampon de réception en mode Full Duplex. Le second utilise les signaux de synchronisation par matériel dans le même but. Par conséquent, il est interdit de demander simultanément ces deux algorithmes de contrôle de flux, car le contrôle de flux/contrôle modem RTS/CTS (bit 5) implique un réseau semi-duplex alors que le contrôle de flux Xon/Xoff (bit 4) implique un réseau Full Duplex.

Exemples d'application

Description

Le programme suivant est une courte application de démonstration avec quatre instances du bloc XXMIT illustrant les quatre fonctions principales :

- Maître Modbus
- Entrée ASCII simple
- Sortie Message ASCII
- Entrée ASCII terminée

Maître Modbus

L'opération suivante du maître Modbus est une requête de lecture adressée à un équipement esclave (adresse 4 par exemple) connecté au port 1 du maître :

- Lecture des mots %MW1 à %MW10 de l'esclave
- dans les mots locaux %MW11 à %MW20

L'esclave (adresse 4) doit être configuré avec les paramètres de port suivants :

- 9 600 bauds
- 8 bits de données
- 1 bit d'arrêt
- parité paire (2)

Le maître utilise les paramètres du bloc XXMIT.

Déclaration des variables pour le maître Modbus

Le tableau suivant présente les variables utilisées dans l'exemple de maître Modbus :

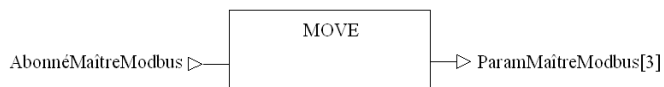
Nom de variable	Type de données	Valeur initiale	Commentaire
DémarrMaîtreModbus	BOOL		
MaîtreModbusActif	BOOL		
CommandeMaîtreModbus	WORD	16#0100	Bit 8 défini
MaîtreModbusEffect	BOOL		
ErreurMaîtreModbus	BOOL		
ParamMaîtreModbus	WordArr9		
ParamMaîtreModbus[1]		3	Code Modbus : lecture de plusieurs mots
ParamMaîtreModbus[2]		10	Nombre de mots à lire
ParamMaîtreModbus[3]		4	Adresse de l'automate Modbus esclave
ParamMaîtreModbus[4]		1	Mot source

Nom de variable	Type de données	Valeur initiale	Commentaire
ParamMaîtreModbus[5]		11	Mot cible
ParamMaîtreModbus[6]		0	inutilisé
ParamMaîtreModbus[7]			inutilisé
ParamMaîtreModbus[8]			inutilisé
ParamMaîtreModbus[9]			inutilisé
EtatMaîtreModbus	INT		
AbonnéMaîtreModbus	WORD	4	Saisie de l'adresse de l'esclave
CpteurErrMaîtreModbus	INT		
CpteurEffectMaîtreModbus	INT		

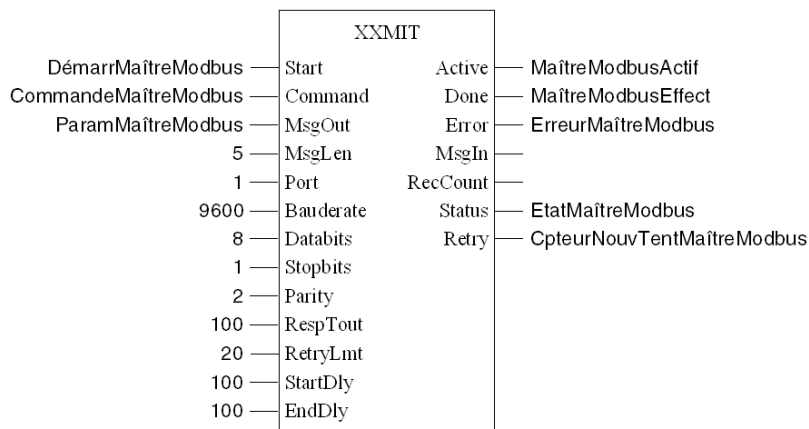
Section IEC pour maître Modbus

Utilisez le programme suivant dans une section FBD :

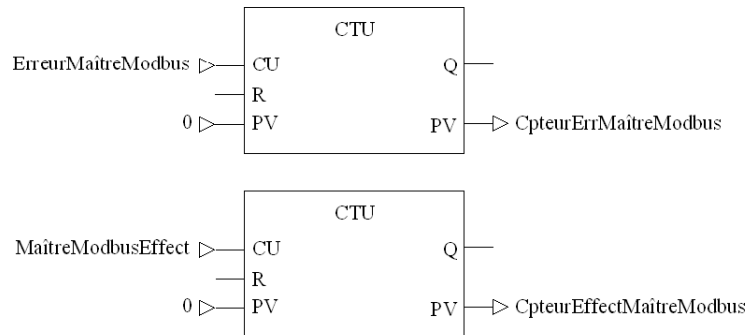
Affectation de l'adresse de l'abonné esclave



Affectations du bloc fonction XXMIT:



Comptage des échecs et des réussites



Réception ASCII simple

Reçoit toute valeur arrivant au port 1. La longueur du tampon de réception est associée à la variable LongRéceptSimple, dont la valeur initiale est égale à 10.

Les caractères reçus sont stockés dans le tableau MsgIn et le nombre de caractères reçus dans RecCount.

Déclaration des variables pour la réception ASCII simple

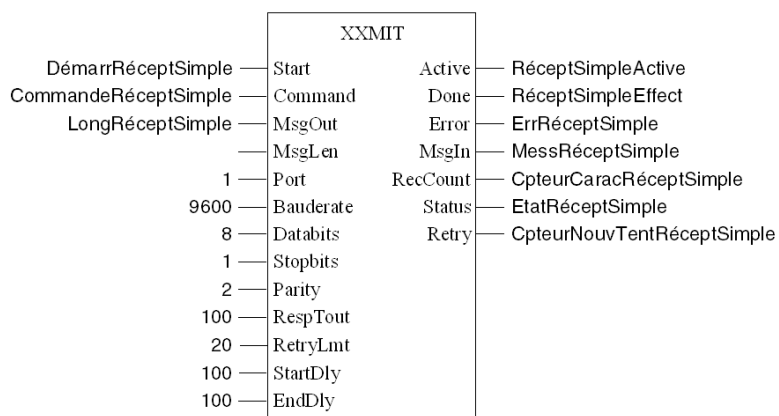
Le tableau suivant présente les variables utilisées dans l'exemple de réception ASCII simple :

Nom de variable	Type de données	Valeur initiale	Commentaire
DémarrRéceptSimple	BOOL		
RéceptSimpleActive	BOOL		
CpteurCaracRéceptSimple	INT		
CommandeRéceptSimple	WORD	16#0480	Bits 7 et 10 définis. FIFO activé
RéceptSimpleEffect	BOOL		
ErrRéceptSimple	BOOL		
LongRéceptSimple	INT	10	
CpteurNouvTentRéceptSimple	INT		

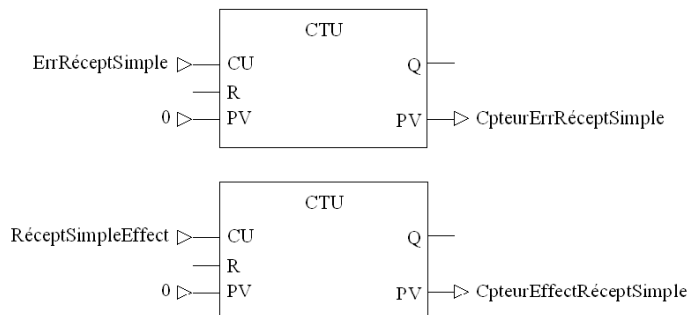
Nom de variable	Type de données	Valeur initiale	Commentaire
EtatRéceptSimple	INT		
MessRéceptSimple	ByteArr12		
CpteurEffectRéceptSimple	INT		
CpteurErrRéceptSimple	INT		

Section IEC pour réception ASCII simple

Utilisez le programme suivant dans une section FBD :



Comptage des échecs et des réussites



Envoi ASCII simple

Envoie un message ASCII simple depuis le port 1. Le message est « Hello World!! ».

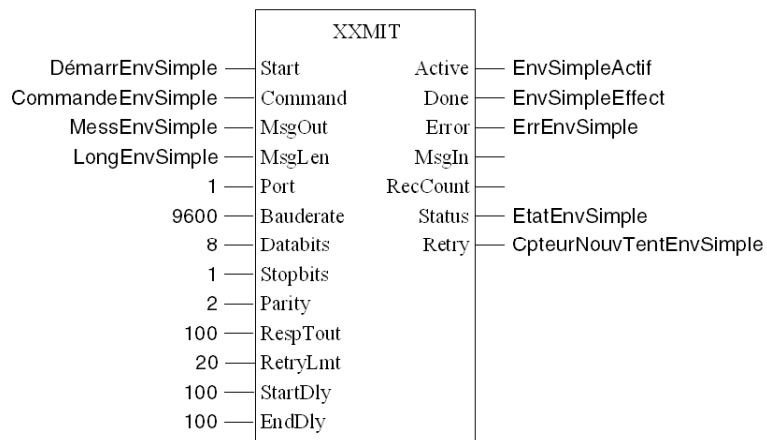
Déclaration des variables pour l'envoi ASCII simple

Le tableau suivant présente les variables utilisées dans l'exemple d'envoi ASCII simple :

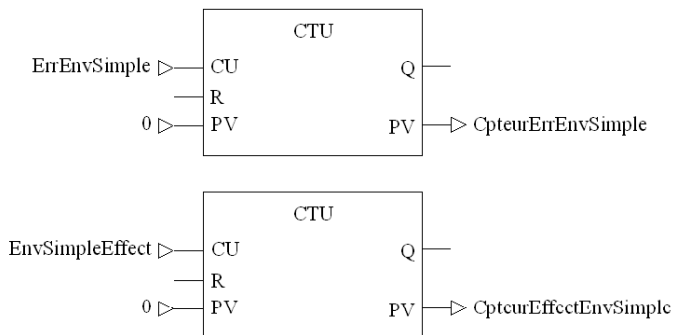
Nom de variable	Type de données	Valeur initiale	Commentaire
DémarrEnvSimple	BOOL		
EnvSimpleActif	BOOL		
CommandeEnvSimple	WORD	16#0200	Bit 9 défini
EnvSimpleEffect	BOOL		
ErrEnvSimple	BOOL		
LongEnvSimple	INT	14	Nombre de caractères à envoyer
MessEnvSimple	ByteArr36		« Hello World !! »
MessEnvSimple[1]		16#48	
MessEnvSimple[2]		16#65	
MessEnvSimple[3]		16#6C	
MessEnvSimple[4]		16#6C	
MessEnvSimple[5]		16#6F	
MessEnvSimple[6]		16#20	
MessEnvSimple[7]		16#57	
MessEnvSimple[8]		16#6F	
MessEnvSimple[9]		16#72	
MessEnvSimple[10]		16#6C	
MessEnvSimple[11]		16#64	
MessEnvSimple[12]		16#20	
MessEnvSimple[13]		16#21	
MessEnvSimple[14]		16#21	
CpteurNouvTentEnvSimple	INT		
EtatEnvSimple	INT		
CpteurEffectEnvSimple	INT		
CpteurErrEnvSimple	INT		

Section IEC pour envoi ASCII simple

Utilisez le programme suivant dans une section FBD :



Comptage des échecs et des réussites



Réception ASCII terminée

Après avoir reçu les caractères de départ « AB », le bloc fonction place tous les caractères reçus dans le tampon de réception MsgIn. Le récepteur s'arrêtera à la réception des caractères de fin « CD ». La sortie « Done » sera alors activée pour indiquer la réussite de l'opération. La longueur maximale du tampon de réception est associée à la variable « LongRéceptTerm », dont la valeur initiale est égale à 20 dans cet exemple.

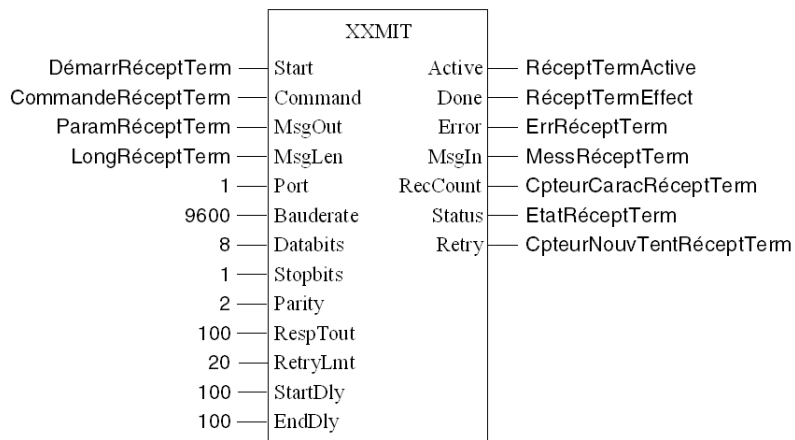
Déclaration des variables pour la réception ASCII terminée

Le tableau suivant présente les variables utilisées dans l'exemple de réception ASCII terminée :

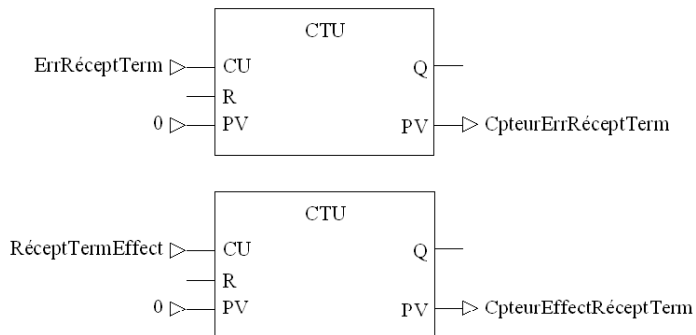
Nom de variable	Type de données	Valeur initiale	Commentaire
DémarrRéceptTerm	BOOL		
RéceptTermActive	BOOL		
CpteurCaracRéceptTerm	INT		
CommandeRéceptTerm	WORD	16#0880	Bits 11 et 7 définis. FIFO activé
RéceptTermEffect	BOOL		
ErrRéceptTerm	BOOL		
LongRéceptTerm	INT	20	
MessRéceptTerm	ByteArr36		Caractères reçus
CpteurNouvTentRéceptTerm	INT		
ParamRéceptTerm ParamRéceptTerm[1] ParamRéceptTerm[2] ParamRéceptTerm[3] ParamRéceptTerm[4] ParamRéceptTerm[5] ParamRéceptTerm[6]	ByteArr36	16#02 16#02 16#41 16#42 16#43 16#44	Longueur de la chaîne de fin (1 ou 2) Longueur de la chaîne de départ (0, 1 ou 2) Deuxième caractère de départ Premier caractère de départ Deuxième caractère de fin Premier caractère de fin
EtatRéceptTerm	INT		
CpteurEffectRéceptTerm	INT		
CpteurErrRéceptTerm	INT		

Section IEC pour réception ASCII terminée

Utilisez le programme suivant dans une section FBD :



Comptage des échecs et des réussites



Saisie des chaînes comme valeurs initiales

L'éditeur de données Unity Pro vous permet de saisir facilement des chaînes en tant que valeurs initiales dans des tableaux d'octets.

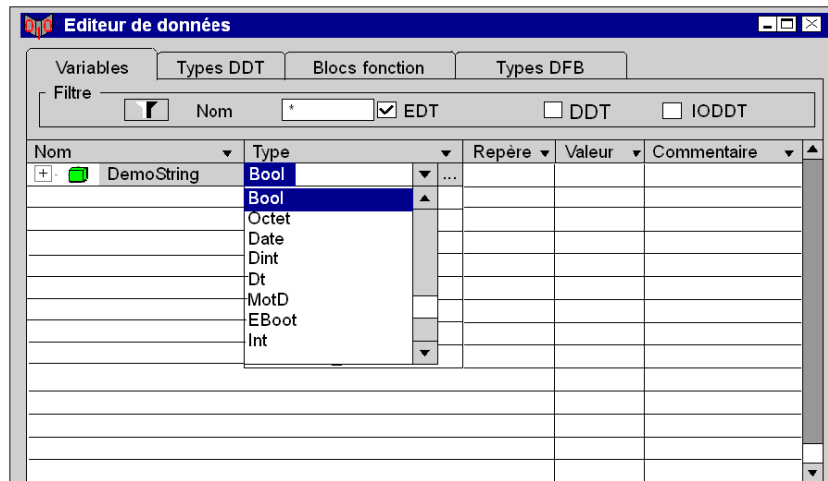
La section suivante explique brièvement comment définir une variable « ChaîneDémon » en tant que « ByteArr36 » et comment saisir une chaîne « Mon texte! » comme valeur initiale.

Ouverture de l'éditeur de données


Dans le menu principal, sélectionnez :

Outils -> Editeur de données.

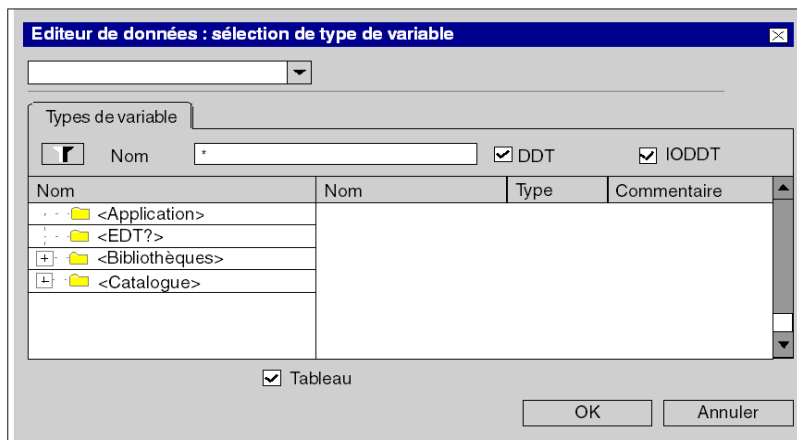
Editeur de données



Déclaration du nouveau nom de chaîne

Déclarez le nom de la chaîne du tableau en cliquant sur  en regard du type (sélectionnez bool dans le cas présent). Une fenêtre permettant de sélectionner le type de variable s'affiche alors.

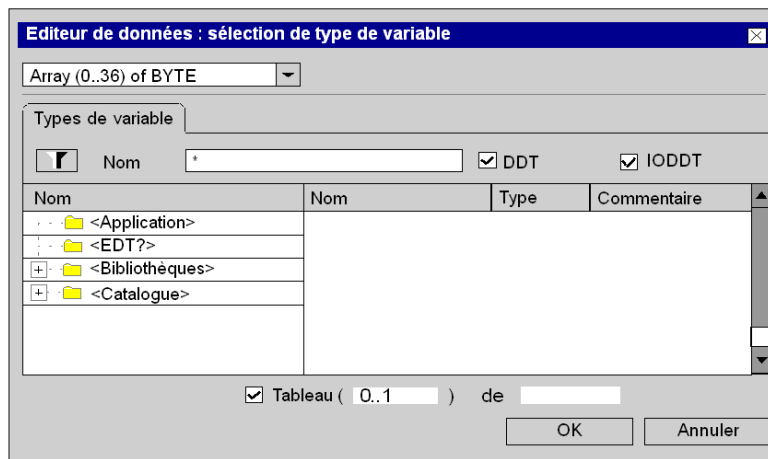
Définition du type d'élément 1



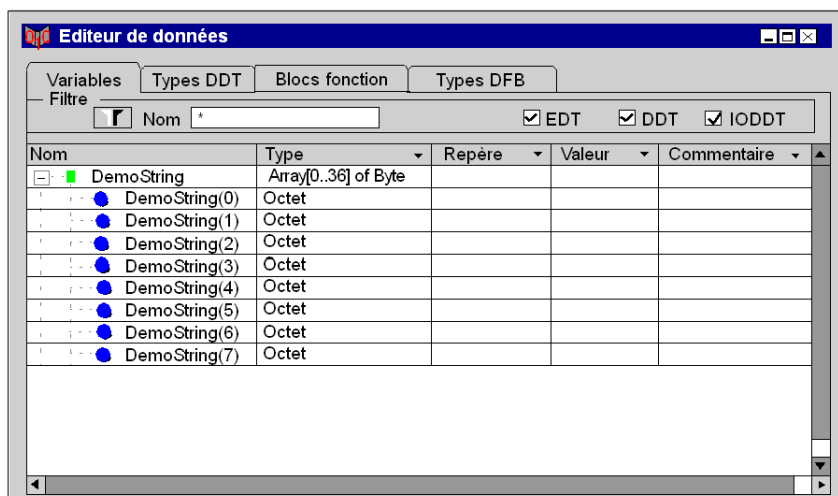
Définition du type d'élément

Cochez la case Tableau, puis définissez le nombre d'éléments et leur type. Vous pouvez maintenant afficher et modifier les valeurs de byteArr36 étendu.

Définition du type d'élément 2



Définition de ByteArr36

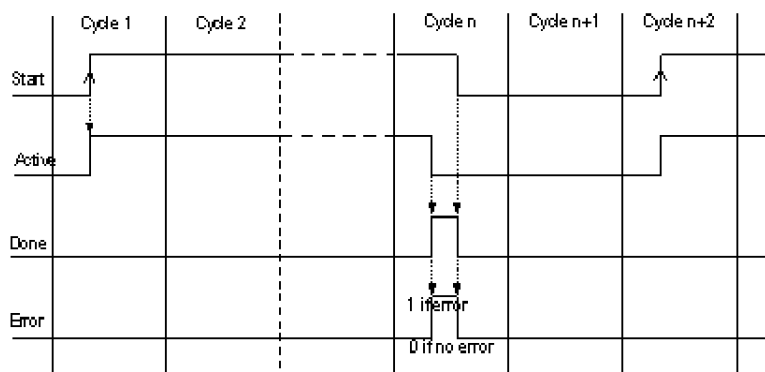


32.3 XXMIT : Règles de programmation

Règles de programmation du bloc XXMIT

Description des règles de programmation

Plusieurs cycles d'automate peuvent être nécessaires pour envoyer une requête Modbus ou une chaîne de caractères. Les bits de démarrage (Start), d'activation (Active), de fin d'opération (Done) et d'erreur (Error) fonctionnent de la manière suivante :



Le signal de démarrage est écrit par l'application. Les signaux Active, Done et Error sont lus par l'application.

⚠ ATTENTION

EMISSIONS MULTIPLES

Si la valeur du signal de démarrage reste à 1 après la fin de l'opération, le bloc XXMIT est redémarré. Cela va générer plusieurs émissions du même message Modbus ou ASCII vers l'équipement récepteur. Pour éviter cela, il est recommandé de réinitialiser le bit Start dès que le bit Done est à 1.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

Si la valeur du signal de démarrage reste à 1 après la fin de l'opération, le bloc XXMI est redémarré.

Plusieurs blocs fonction XXMIT ne peuvent pas être activés en même temps. Si plusieurs blocs fonction XXMIT sont activés, le premier bloc scruté va fonctionner et bloquer tous les autres blocs XXMIT jusqu'à ce qu'il ait terminé. L'appel du bloc suivant va immédiatement générer le code d'erreur 150.

Le bloc fonction XXMIT peut être utilisé uniquement pour une tâche MAST. Un code d'erreur (127) apparaît immédiatement si le bloc est activé dans les tâches FAST / AUX ou EVENT (Unity Pro n'effectue aucun contrôle lors de la compilation).

NOTE : Les paramètres d'entrée du bloc XXMIT doivent être initialisés pour pouvoir activer l'entrée START. Ils ne doivent pas être modifiés lorsque le bloc fonction est en cours d'exécution. Si le bit START est remis à 0 avant la fin de l'opération, le bloc fonction est arrêté (le bit Active passe à 0). Pour permettre l'exécution complète du bloc, le bit START doit rester à 1 tant que l'opération n'est pas terminée ou qu'aucune erreur n'est survenue.

32.4 Références techniques XXMIT

Vue d'ensemble

Cette section décrit les références techniques relatives à XXMIT.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Limites des paramètres de requête/réponse Modbus	386
Configuration de XXMIT à l'aide de modems à numérotation automatique compatibles Hayes (uniquement)	387
Exemple d'application Hayes	392

Limites des paramètres de requête/réponse Modbus

Limites des paramètres basées sur le code de fonction

Les paramètres de requête/réponse sont limités selon le type de code de fonction que vous utilisez. Reportez-vous au tableau ci-dessous.

Tableau des paramètres maximum de l'automate Quantum :

Code fonction	Description	Requête	Réponse
1	Lecture de plusieurs bits	2 000 bits	2 000 bits
2	Lecture de plusieurs bits	2 000 bits	2 000 bits
3	Lecture de plusieurs mots	125 mots	125 mots
4	Lecture de plusieurs mots	125 mots	125 mots
5	Ecriture d'un seul bit	1 bit	1 bit
6	Ecriture d'un seul mot	1 mot	1 mot
15	Ecriture de plusieurs bits	800 bits	800 bits
16	Ecriture de plusieurs mots	100 mots	100 mots

Configuration de XXMIT à l'aide de modems à numérotation automatique compatibles Hayes (uniquement)

Description

Vous devez vous familiariser avec trois commandes lors de la connexion de modems à numérotation automatique à XXMIT.

Ces commandes sont les suivantes :

- initialisation du modem
- numérotation du modem
- raccrochage du modem.

Avant qu'un message ASCII ou Modbus ne passe par le modem, vous devez commencer par envoyer une chaîne d'initialisation, puis une chaîne de numérotation au modem. Une fois que le modem a composé le numéro de téléphone et établi la connexion avec le modem déporté, vous pouvez envoyer un nombre illimité de messages ASCII ou Modbus par ce modem. Pour envoyer plusieurs messages, incrémentez le pointeur de message pour passer au message suivant après chaque exécution réussie de XXMIT. Lorsque tous les messages ont été envoyés, vous pouvez envoyer la chaîne de raccrochage au modem.

Message d'initialisation

Le message d'initialisation est un simple message ASCII ; il peut comporter un maximum de 512 caractères, sachant que 50 caractères sont souvent plus que suffisants pour initialiser un modem. Vous pouvez mettre en oeuvre n'importe quelle commande AT Hayes dans la chaîne d'initialisation. Nous recommandons les commandes suivantes lors de l'initialisation d'un modem utilisé avec XXMIT.

Message d'initialisation pour un modem à numérotation automatique

Message d'initialisation =	AT&F&K0&Q0&D0V1Q0X0E1
AT=	Réglage automatique du modem ¹
&F=	Reprise de la configuration usine comme configuration active ¹
&K0=	Désactivation du contrôle de flux local ²
&Q0=	Communication en mode asynchrone ²
&D0=	Ignorer l'état du signal DTR ¹
V1=	Affichage des codes de résultat sous forme de mots ¹ Si V1 n'est pas utilisé ou si le modem n'est pas capable de renvoyer des réponses prolixes, le bloc XXMIT renvoie l'erreur 117 (timeout de réponse du modem).

Message d'initialisation =	AT&F&K0&Q0&D0V1Q0X0E1
Q0=	Renvoi des codes de résultat ¹
X0=	Renvoi des codes de résultat de la progression des appels de base : Connexion, Pas de porteuse et Sonnerie ¹
E1=	Echo des caractères entrés au clavier vers l'écran dans l'état de commande ¹
<p>1 Ces paramètres doivent toujours faire partie de la chaîne d'initialisation pour que XXMIT fonctionne correctement.</p> <p>2 Ces paramètres doivent faire partie de la chaîne d'initialisation pour que XXMIT transmette correctement un message à un modem déporté. Ne modifiez ou n'utilisez ces paramètres que si vous êtes un utilisateur de modem expérimenté.</p>	

NOTE : Si certains fabricants de modems garantissent une compatibilité totale avec Hayes, il peut exister de légères différences. Par conséquent, nous recommandons de n'utiliser que les commandes ayant les mêmes définitions que celles indiquées ci-dessus.

Le message d'initialisation doit toujours commencer par une commande AT standard Hayes. Le bloc XXMIT ajoute automatiquement AT au début des messages de commande du modem et les fait suivre des caractères retour chariot (0x0D) et retour à la ligne (0x0A) puisque ces caractères sont nécessaires pour tous les messages de contrôle du modem. Il n'est pas nécessaire que les autres messages ASCII (sans contrôle) finissent par un retour chariot et un retour à la ligne.

Exemple de message d'initialisation typique envoyé par le bloc XXMIT au modem.

Message	Longueur
(AT)&F&K0&Q0&D0V1X0Q0 (<CR><LF>) ¹	17 caractères
1 Les caractères entre parenthèses sont automatiquement envoyés.	

Par exemple, le message d'initialisation peut également être utilisé pour régler les registres S du modem.

Message	Longueur
(AT)S0=1 (<CR><LF>) ¹	4 caractères
1 Les caractères entre parenthèses sont automatiquement envoyés.	

Pour que XXMIT envoie un message d'initialisation au modem, les bits 9 et 0 du mot de commande doivent être à 1. Lorsque le bit 0 est à 1, les bits 1 et 2 doivent être à 0, sinon le bloc XXMIT ne peut pas exécuter l'opération correctement. Pour que le message soit effectivement envoyé, l'entrée Start de XXMIT doit s'ACTIVER et rester ACTIVE jusqu'à ce que l'opération soit terminée ou qu'une erreur survienne. Lorsque XXMIT détermine que le message a bien été envoyé au modem, il ACTIVE la sortie Done. Lorsqu'une erreur survient, la sortie Error s'ACTIVE. La sortie Active est ACTIVE pendant l'envoi du message au modem.

NOTE : Pour limiter la programmation par schémas à contacts, vous pouvez initialiser le modem avec des paramètres via un programme de terminal et ne pas utiliser XXMIT. Une fois dans la mémoire du modem, les paramètres peuvent être sauvegardés dans la mémoire non volatile à l'aide d'une commande AT, habituellement &W.

Message de composition

Le message de composition est utilisé pour envoyer un numéro de téléphone au modem. Seules les commandes AT liées à la composition d'un numéro doivent être incluses dans le message. Vous trouverez ci-dessous des exemples de messages de composition typiques utilisés avec le bloc XXMIT.

Exemple de composition d'un numéro de téléphone à l'aide de la numérotation à tonalité.

Message	Longueur
(AT)DT)6800326 (<CR><LF>) ¹	7 caractères
1 Les caractères entre parenthèses sont automatiquement envoyés.	

Exemple de composition d'un numéro de téléphone à l'aide de la numérotation par impulsion.

Message	Longueur
(AT)DP)6800326 (<CR><LF>) ¹	7 caractères
1 Les caractères entre parenthèses sont automatiquement envoyés.	

Exemple de composition d'un numéro de téléphone à l'aide de la numérotation à tonalité, attente de la tonalité avant la composition du numéro, puis pause avant la composition du reste du numéro.

Message	Longueur
(AT)DTW,6800326 (<CR><LF>) ¹	9 caractères
1 Les caractères entre parenthèses sont automatiquement envoyés.	

Pour que le bloc XXMIT envoie un message de numérotation à tonalité au modem, les bits 9 et 1 du mot de commande doivent être à 1. Lorsque le bit 1 est à 1, les bits 0 et 2 doivent être à 0, sinon le bloc XXMIT ne peut pas exécuter l'opération correctement. Pour que le message soit effectivement envoyé, l'entrée Start de XXMIT doit s'ACTIVER et rester ACTIVE jusqu'à ce que l'opération soit terminée ou qu'une erreur survienne. Lorsque XXMIT détermine que le message a bien été envoyé au modem, il ACTIVE la sortie Done. Lorsqu'une erreur survient, la sortie Error s'ACTIVE. La sortie Active est ACTIVE pendant l'envoi du message au modem.

NOTE : L'établissement de la connexion entre un modem local et un modem déporté demandant beaucoup de temps, la valeur du timeout dans RespTout doit être assez longue lors de l'envoi d'un message de composition à un modem. Par exemple, réglez le timeout sur 30 000 ms lors de l'envoi d'un message de composition. Lorsque la valeur du timeout est trop faible, XXMIT émet un timeout de message. Vous devrez peut-être essayer plusieurs réglages avant de trouver la valeur optimale.

Message de raccrochage

Le message de raccrochage est utilisé pour raccrocher le modem. Seules les commandes AT liées au raccrochage du modem doivent être utilisées dans ce message. Vous trouverez ci-dessous un exemple de message de raccrochage typique.

Exemple de message de raccrochage du modem.

Message	Longueur
(+++AT)H0 (<CR><LF>) ¹	2 caractères
1 Les caractères entre parenthèses sont automatiquement envoyés.	

Lorsque le message de raccrochage est envoyé à un modem déjà connecté à un modem déporté, XXMIT doit d'abord régler le modem local en mode de commande en envoyant une séquence d'échappement +++ au modem. XXMIT suppose que +++ règle le modem en mode de commande. Certains fabricants de modems permettent au propriétaire de modifier cette séquence d'échappement par défaut. Pour que XXMIT fonctionne correctement, le modem doit être réglé de manière à accepter la séquence d'échappement +++.

Pour que le bloc XXMIT envoie un message de raccrochage au modem, les bits 9 et 2 du mot de commande doivent être à 1. Lorsque le bit 2 est à 1, les bits 0 et 1 doivent être à 0, sinon le bloc XXMIT ne peut pas exécuter l'opération correctement. Pour que le message soit effectivement envoyé, l'entrée Start de XXMIT doit s'ACTIVER et rester ACTIVE jusqu'à ce que l'opération soit terminée ou qu'une erreur survienne. Lorsque XXMIT détermine que le message a bien été envoyé au modem, il ACTIVE la sortie Done. Lorsqu'une erreur survient, la sortie Error s'ACTIVE. La sortie Done est ACTIVE pendant l'envoi du message au modem.

NOTE : Expert : Le raccrochage d'un modem local demandant beaucoup de temps après réception de la commande de raccrochage, la valeur du timeout dans RespTout doit être assez longue lors de l'envoi d'un message de composition à un modem. Par exemple, réglez le timeout sur 30 000 ms lors de l'envoi d'un message de composition. Lorsque la valeur du timeout est trop faible, XXMIT émet un timeout de message. Vous devrez peut-être essayer plusieurs réglages avant de trouver la valeur optimale.

Exemple d'application Hayes

Description

Le programme suivant est une petite application de démonstration pour l'envoi du message "Bonjour tout le monde" via un modem compatible Hayes (Com One, Deskline 56K).

Pour lancer l'application, la variable "stage" (étape) doit être réglée à 1 dans une table d'animation Unity Pro.

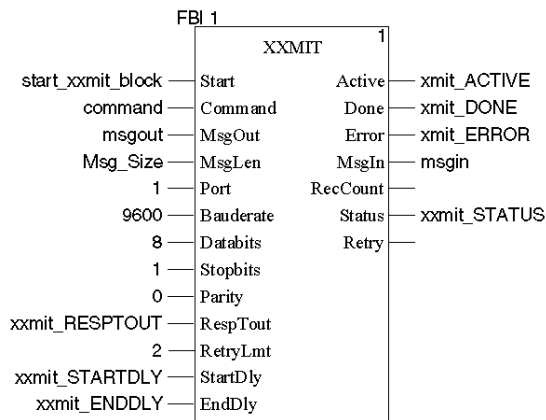
Types de données variables

start_xxmit_block	BOOL
command	WORD
msgout	ARRAY[0..40] OF BYTE
Msg_Size	INT
xmit_DONE	BOOL
xmit_ERROR	BOOL
xmit_ACTIVE	BOOL
xxmit_RESPTOUT	INT
xxmit_STARTDLY	INT
xxmit_ENDDLY	INT

xmit_STATUS_mem	INT
counter	INT
stage	WORD

Section IEC pour XXMIT

Effectuez la programmation suivante dans la section FBD :



Section IEC pour les commandes du modem

Effectuez la programmation suivante dans la section ST :

```
(* %S6 is used to generate a 2 second delay *) %m6 := %s6;
if xxmit_STATUS_Mem = 0 then
  if not(xxmit_STATUS = 0) then
    xxmit_STATUS_Mem := xxmit_STATUS;
  end_if; end_if;
(* If XXMIT error, the function block is stopped *) if
xmit_ERROR=1 then
  stage := 0;
  counter := 0;
  start_xxmit_block:=0; end_if;
if stage = 1 then
  (* Initialization message*)
  xxmit_STATUS_Mem := 0;
  counter := 0;
  xxmit_ENDDLY := 700;
  xxmit_STARTDLY := 600;
  xxmit_RESPTOUT := 700;
  (* Command word init *)
  command:=2#0000001000000001;
  msgout[0] := 16#26; (*&->26*)
  msgout[1] := 16#46; (*F->46*)
  msgout[2] := 16#26; (*&->26*)
  msgout[3] := 16#4B; (*K->46*)
  msgout[4] := 16#30; (*0->30*)
  msgout[5] := 16#26; (*&->26*)
  msgout[6] := 16#44; (*D->44*)
  msgout[7]
```

```

:= 16#30;(*0->30*)          msgout[8] := 16#56;(*V->56*)
                           msgout[9] := 16#31;(*1->31*)          msgout[10]
:= 16#51;(*Q->51*)          msgout[11] := 16#30;(*0->30*)
                           msgout[12] := 16#58;(*X->58*)
                           msgout[13] := 16#30;(*0->30*)
                           msgout[14] := 16#45;(*E->45*)
                           msgout[15] := 16#31;(*1->31*)          Msg_Size
:= 16;                      start_xxmit_block:=1;              stage := 2;
end_if;

if stage = 3 then          (* Dial message *)
    xxmit_STARTDLY := 100;          xxmit_ENDDLY :=
100;          xxmit_RESPTOUT := 32000;          (* Command
word init *)          command:=2#0000001000000010;
    (* Extension number *)          msgout[0] :=
16#32;(*2*)          msgout[1] := 16#35;(*5*)
    msgout[2] := 16#37;(*7*)          msgout[3] :=
16#34;(*4*)          start_xxmit_block:=1;
    Msg_Size := 4;          stage := 4; end_if;

if (stage = 5)then          if RE(%m6) then
    counter := counter + 1;          end_if;

    (* Two seconds delay *)          if stage = 5 and
counter = 2 then          counter := 0;
    stage := 7;          end_if; end_if;

if stage = 7 then          (* ASCII message to be send *)
    xxmit_STARTDLY := 300;          xxmit_ENDDLY :=
400;          xxmit_RESPTOUT := 32000;          (* Command
word init *)          command:=2#0100001000000000;
    msgout[0] := 16#48; (*H*)          msgout[1] :=
16#65; (*e*)          msgout[2] := 16#6C; (*1*)
    msgout[3] := 16#6C; (*1*)          msgout[4] :=
16#6F; (*o*)          msgout[5] := 16#20; (* *)
    msgout[6] := 16#57; (*W*)          msgout[7] :=
16#6F; (*o*)          msgout[8] := 16#72; (*r*)
    msgout[9] := 16#6C; (*1*)          msgout[10] :=
16#64; (*d*)          msgout[11] := 16#20; (* *)
    msgout[12] := 16#21; (*!*)          msgout[13] :=
16#21; (*!*)          Msg_Size := 14;
    start_xxmit_block:=1;          stage := 8;
end_if;

```

```
if stage = 100 then          (* Hangup message *)
    command:=2#0000001000000100;
    xxmit_STARTDLY := 300;          xxmit_ENDDLY :=
400;          xxmit_RESPTOUT := 500;          msgout[0] :=
16#48; (* *)          msgout[1] := 16#30; (*CR*)
    Msg_Size := 2;          start_xxmit_block:=1;
end_if;
(* Change of state after each XXMIT operation *) if xmit_DONE
= 1 then          start_xxmit_block:=0;          if stage
= 2 then          stage := 3;          end_if;
    if stage = 4 then          counter :=
0;          stage := 5;          end_if;
    if stage = 8 then          stage := 99;
end_if;
    if stage = 99 then          stage
:=100;          end_if; end_if;
```

32.5 Informations sur le câblage

Vue d'ensemble

Cette section décrit les câbles et le brochage des composants matériels utilisés avec XXMIT.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Sujet	Page
Brochage des câbles	397
Kits d'adaptateurs de câble	409

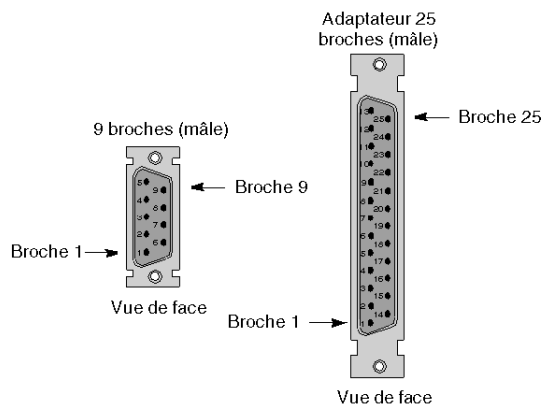
Brochage des câbles

Brochage des câbles d'interface

Vous devez installer un câble d'interface entre l'automate et le modem ou l'imprimante. Ce câble doit être branché sur le port pris en charge par l'automate et sur le port RS232 du modem ou de l'imprimante, ou directement sur le port Modbus d'un autre automate. Le bloc XXMIT prenant en charge de nombreux modems et imprimantes, le brochage risque de varier. Certains brochages sont indiqués ci-dessous.

9 broches (RS-232) vers 25 broches (modem) sans contrôle RTS/CTS

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.

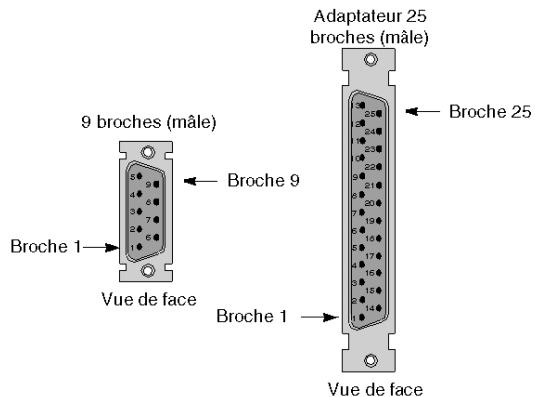


Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur 9 broches			Connecteur type SUB-D 25 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	2	Oui	3	RXD
TXD	3	Oui	2	TXD
RTS	7 (pontage)		4 (pontage)	RTS
CTS	8 (pontage)		5 (pontage)	CTS
DSR	4 (pontage)		6 (pontage)	DSR
DTR	6 (pontage)		20 (pontage)	DTR
GND	5	Oui	7	GND

9 broches (RS-232) vers 25 broches (modem) avec contrôle RTS/CTS

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.

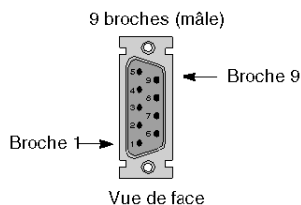


Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur 9 broches			Connecteur type SUB-D 25 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	2	Oui	3	RXD
TXD	3	Oui	2	TXD
RTS	7	Oui	4	RTS
CTS	8	Oui	5	CTS
DSR	4 (pontage)		6 (pontage)	DSR
DTR	6 (pontage)		20 (pontage)	DTR
GND	5	Oui	7	GND

9 broches vers 9 broches (Null Modem)

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.

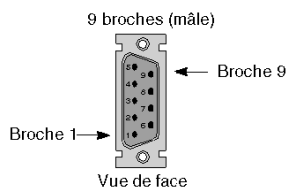


Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur 9 broches			Connecteur 9 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	2	Oui	3	TXD
TXD	3	Oui	2	RXD
RTS	7 (pontage)		7 (pontage)	RTS
CTS	8 (pontage)		8 (pontage)	CTS
DSR	4 (pontage)		4 (pontage)	DSR
DTR	6 (pontage)		6 (pontage)	DTR
GND	5	Oui	5	GND

9 broches vers 9 broches (Modem)

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



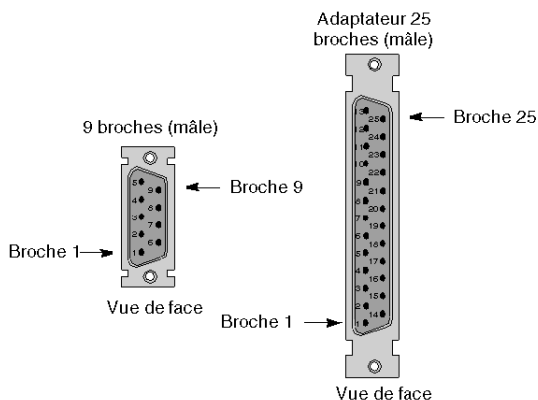
Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur 9 broches			Connecteur 9 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
TXD	2	Oui	2	TXD
RXD	3	Oui	3	RXD
RTS	7	Oui	7	RTS

Brochage des connecteurs				
Connecteur 9 broches			Connecteur 9 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
CTS	8	Oui	8	CTS
DSR	4 (pontage)		4 (pontage)	DSR
DTR	6 (pontage)		6 (pontage)	DTR
GND	5	Oui	5	GND

9 broches vers 25 broches (Null Modem)

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.

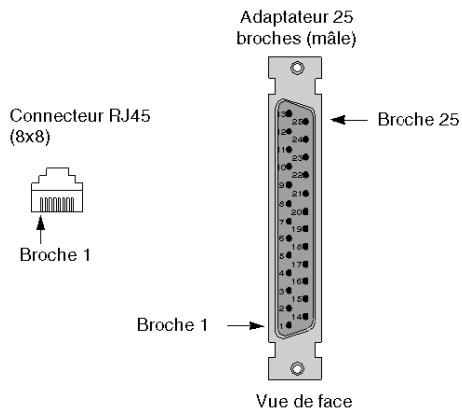


Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur 9 broches			Connecteur type SUB-D 25 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	2	Oui	2	TXD
TXD	3	Oui	3	RXD
RTS	7 (pontage)		4 (pontage)	RTS
CTS	8 (pontage)		5 (pontage)	CTS
DSR	4 (pontage)		6 (pontage)	DSR
DTR	6 (pontage)		20 (pontage)	DTR
GND	5	Oui	7	GND

RJ45-(8x8) vers 25 broches (Null Modem) 110XCA20401

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur type SUB-D 25 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	2	TXD
TXD	3	Oui	3	RXD
RTS	6 (pontage)		4 (pontage)	RTS
CTS	7 (pontage)		5 (pontage)	CTS
GND	5	Oui	7	GND
DSR	2	Oui	6	DSR
			20	DTR
Masse du châssis	8	Oui	1	Masse du châssis

⚠ ATTENTION

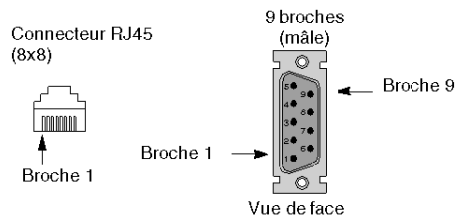
RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

RJ45-(8x8) vers 9 broches (Null Modem) 110XCA20301

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur type SUB-D 9 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	3	TXD
TXD	3	Oui	2	RXD
RTS	6 (pontage)		7 (pontage)	RTS
CTS	7 (pontage)		8 (pontage)	CTS
GND	5	Oui	5	GND
DSR	2	Oui	4	DTR
			6	DSR
Masse du châssis	8	Oui		Cas du connecteur

⚠ ATTENTION

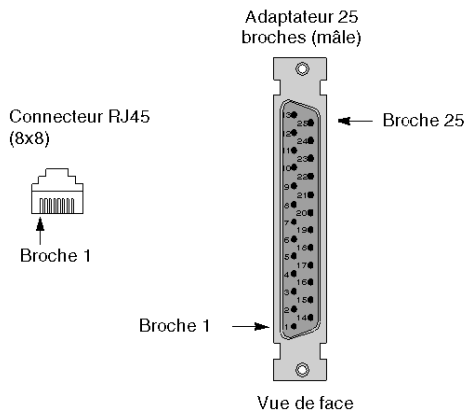
RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

RJ45-(8x8) vers 25 broches (Modem) 110XCA20401

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur type SUB-D 25 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	3	RXD
TXD	3	Oui	2	TXD
RTS	6 (pontage)		4 (pontage)	RTS
CTS	7 (pontage)		5 (pontage)	CTS
GND	5	Oui	7	GND
DSR	2	Oui	6 20	DSR
				DTR
Masse du châssis	8	Oui	1	Masse du châssis

⚠ ATTENTION

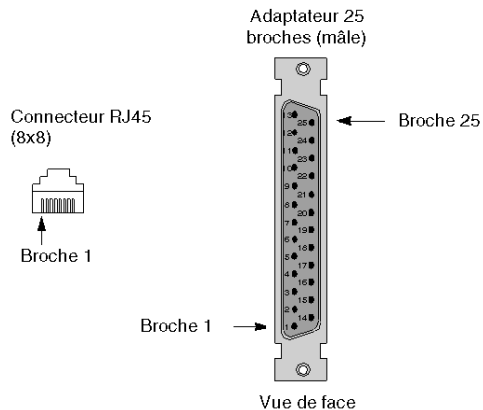
RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

RJ45-(8x8) vers 25 broches (Modem) 110XCA20401

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur type SUB-D 25 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	3	RXD
TXD	3	Oui	2	TXD
RTS	6	Oui	4	RTS
CTS	7	Oui	5	CTS
GND	5	Oui	7	GND
			6 (pontage)	DSR
			20 (pontage)	DTR
Masse du châssis	8	Oui	1	Masse du châssis

⚠ ATTENTION

RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

RJ45-(8x8) vers RJ45-(8x8) (Modem)

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.

RJ45 connector
(8x8)



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur RJ45	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	4	RXD
TXD	3	Oui	3	TXD
RTS	6	Oui	6	RTS
CTS	7	Oui	7	CTS
GND	5	Oui	5	GND
DSR	2	Oui	2	DSR
Masse du châssis	8	Oui	8	Masse du châssis

ATTENTION

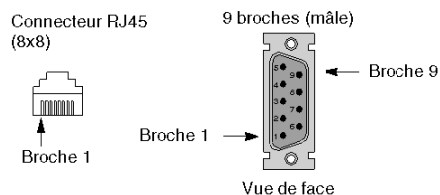
RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

9 broches vers RJ45-(8x8) (Modem) 110XCA20301

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur 9 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	2	RXD
TXD	3	Oui	3	TXD
RTS	6 (pontage)		7 (pontage)	RTS
CTS	7 (pontage)		8 (pontage)	CTS
GND	5	Oui	5	GND
DSR	2	Oui	6	DSR
			4	DTR
Masse du châssis	8	Oui	Cas du connecteur	

⚠ ATTENTION

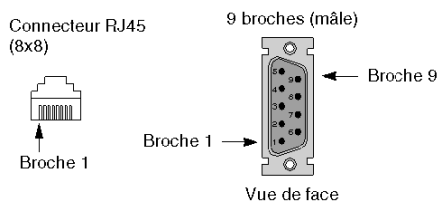
RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

9 broches vers RJ45-(8x8) (Modem) 110XCA20301

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.



Reportez-vous au tableau de brochage des connecteurs.

Brochage des connecteurs				
Connecteur RJ45			Connecteur 9 broches	
Nom du signal	Broche	Broche connectée à ...	Broche	Nom du signal
RXD	4	Oui	2	RXD
TXD	3	Oui	3	TXD
RTS	6	Oui	7	RTS
CTS	7	Oui	8	CTS
GND	5	Oui	5	GND
			6 (pontage)	DSR
			4 (pontage)	DTR
Masse du châssis	8	Oui	Cas du connecteur	

⚠ ATTENTION

RISQUE DE COURT-CIRCUIT DE 5 V

La broche 1 du connecteur RJ45 reçoit 5 V de l'automate.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

Connecteur RS 422/RS 485 de processeur avancé RJ 45

Reportez-vous à la figure pour obtenir une vue de face des connecteurs.

Connecteur RJ45
(8x8)



Broche 1

Tableau de brochage du connecteur RS 422/RS 485 Quantum avancé RJ 45

Broche	Signal RS-422	Signal RS-485
1	RX -	D -
2	RX +	D +
3	TX +	
4	Nc	Nc
5	GND	GND
6	TX -	
7	Nc	Nc
8	PE facultatif	PE facultatif

NOTE : pour le protocole RS-485, les broches 1, 2, 3 et 6 doivent être court-circuitées.

Kits d'adaptateurs de câble

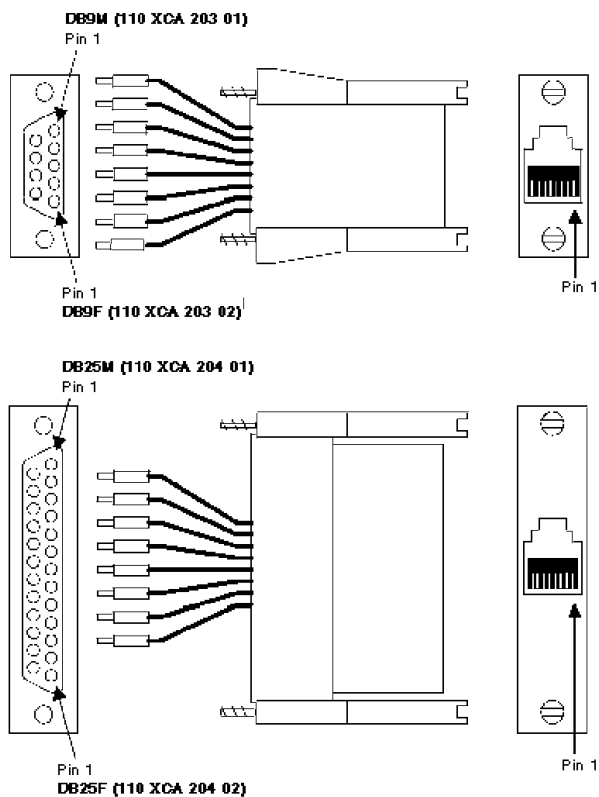
Kits d'adaptateurs de câble pour RJ45

Il est recommandé d'acheter des kits d'adaptateurs de câble afin de répondre aux exigences RJ45 (8x8). Le tableau suivant contient une liste des kits disponibles.

Kits d'adaptateurs de câble disponibles

Description	Référence de produit
RJ45-(8x8) vers 25 broches (mâle)	110XCA20401
RJ45-(8x8) vers 9 broches (mâle)	110XCA20301
RJ45-(8x8) vers 9 broches (femelle)	110XCA20302
RJ45-(8x8) vers 25 broches (femelle)	110XCA20402

Kits d'adaptateurs DB/RJ45

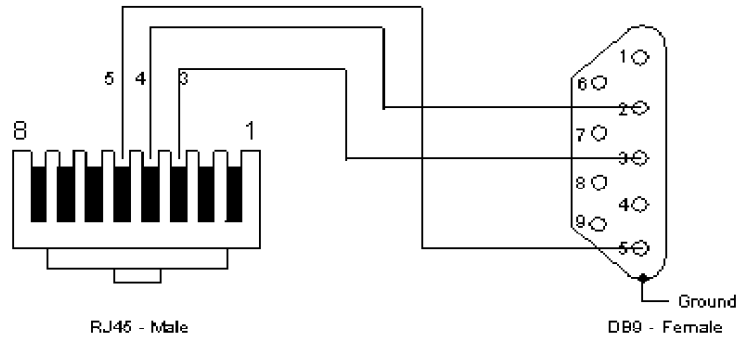


Connexion des câbles - Exemple 1

Pour des terminaux standard à vitesse faible ou utilisant un contrôle de flux logiciel, vous pouvez utiliser une connexion simple à 3 broches.

RJ45 à DB9

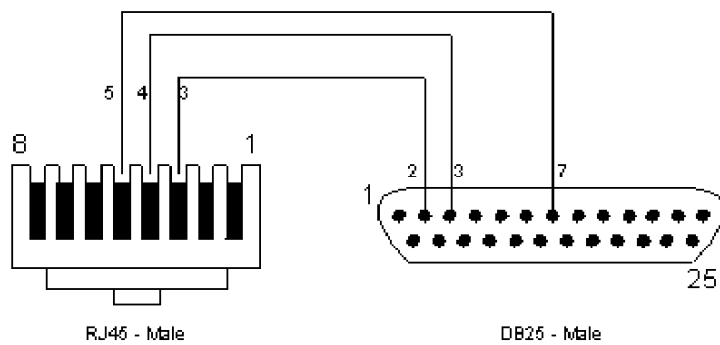
Automate Quantum		Connecteur du modem	
RJ45		DB9	
RJ45-3	TXD	RXD	DB25-2
RJ45-4	RXD	TXD	DB-25-3
RJ45-5	GND	GND	DB25-5



Connexion des câbles - Exemple 2

RJ45 à DB25

Automate Quantum		Connecteur du modem	
RJ45		DB25	
RJ45-3	TXD	RXD	DB25-3
RJ45-4	RXD	TXD	DB-25-2
RJ45-5	GND	GND	DB25-7



Annexes



Objet de ce chapitre

Cette section contient les annexes.

Contenu de cette annexe

Cette annexe contient les chapitres suivants :

Chapitre	Titre du chapitre	Page
A	codes et valeurs d'erreur des EFB	415
B	Objets système	421

codes et valeurs d'erreur des EFB



Introduction

Les tableaux présentés dans cette section répertorient les codes et les valeurs d'erreur générés pour les EFB de la bibliothèque de communications.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Tableau des codes d'erreur pour la bibliothèque de communications	416
Erreurs courantes relatives aux valeurs en virgule flottante	419

Tableau des codes d'erreur pour la bibliothèque de communications

Introduction

Le tableau suivant répertorie les codes et les valeurs d'erreur générés pour les EFB de la bibliothèque de communications.

Booléen étendu

Tableau des codes et valeurs d'erreur générés pour les EFB du type Booléen étendu.

Nom EFB	Code d'erreur	Etat ENO en cas d'erreur	Valeur d'erreur (format décimal)	Valeur d'erreur (format hexa décimal)	Description de l'erreur
CREAD_REG	E_EFB_MSTR_ERROR	F	-30 191	16#8A11	Erreur de communication MSTR
CREAD_REG	E_EFB_NOT_STATE_RAM_4X	F	-30 531	16#88BD	Variable non affectée à la zone %MW (4x)
CREAD_REG	-	F	8 195	16#2003	Valeur affichée dans le mot d'état (apparaît avec E_EFB_MSTR_ERROR)
CREAD_REG	-	F	8 206	16#200E	Valeur affichée dans le mot d'état (apparaît avec E_EFB_NOT_STATE_RAM_4X)
CREAD_REG	-	F	-	-	Voir les tableaux suivants : <ul style="list-style-type: none"> • Codes d'erreur Modbus Plus et EtherNet SY/MAX (voir page 137) • Codes d'erreur spécifiques SY/MAX (voir page 142) • Codes d'erreur EtherNet TCP/IP (voir page 144)
CWRITE_REG	E_EFB_MSTR_ERROR	F	-30 191	16#8A11	Erreur de communication MSTR
CWRITE_REG	-	F	8 195	16#2003	Valeur affichée dans le mot d'état (apparaît avec E_EFB_MSTR_ERROR)
CWRITE_REG	-	F	8 206	16#200E	Valeur affichée dans le mot d'état (apparaît avec E_EFB_NOT_STATE_RAM_4X)

Nom EFB	Code d'erreur	Etat ENO en cas d'erreur	Valeur d'erreur (format décimal)	Valeur d'erreur (format hexa décimal)	Description de l'erreur
CWRITE_REG	-	F	-	-	Voir les tableaux suivants : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus et EtherNet SY/MAX (voir page 137) ● Codes d'erreur spécifiques SY/MAX (voir page 142) ● Codes d'erreur EtherNet TCP/IP (voir page 144)
MBP_MSTR	E_EFB_OUT_OF_RANGE	F	-30 192	16#8A10	Erreur interne : L'EFB a détecté une violation, par exemple l'écriture a dépassé les limites %MW (4x)
MBP_MSTR	E_EFB_NOT_STATE_RAM_4X	F	-30 531	16#88BD	Variable non affectée à la zone %MW (4x)
MBP_MSTR	-	F	8 195	16#2003	Valeur affichée dans le mot d'état (apparaît avec E_EFB_MSTR_ERROR dans l'état du bloc de commande)
MBP_MSTR	-	F	8 206	16#200E	Valeur affichée dans le mot d'état (apparaît avec E_EFB_NOT_STATE_RAM_4X dans l'état du bloc de commande)
MBP_MSTR	-	F	-	-	Voir les tableaux suivants : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus et EtherNet SY/MAX (voir page 137) ● Codes d'erreur spécifiques SY/MAX (voir page 142) ● Codes d'erreur EtherNet TCP/IP (voir page 144)
READ_REG	W_WARN_OUT_OF_RANGE	F	30 110	16#759E	Paramètre hors limites
READ_REG	E_EFB_NOT_STATE_RAM_4X	F	-30 531	16#88BD	Variable non affectée à la zone %MW (4x)
READ_REG	E_EFB_MSTR_ERROR	F	-30 191	16#8A11	Erreur de communication MSTR
READ_REG	-	F	8 195	16#2003	Valeur affichée dans le mot d'état (apparaît avec W_WARN_OUT_OF_RANGE)

Nom EFB	Code d'erreur	Etat ENO en cas d'erreur	Valeur d'erreur (format décimal)	Valeur d'erreur (format hexa décimal)	Description de l'erreur
READ_REG	MBPUNLOC	F	8 206	16#200E	Valeur affichée dans le mot d'état (apparaît avec E_EFB_NOT_STATE_RAM_4X)
READ_REG	-	F	-	-	Voir les tableaux suivants : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus et EtherNet SY/MAX (voir page 137) ● Codes d'erreur spécifiques SY/MAX (voir page 142) ● Codes d'erreur EtherNet TCP/IP (voir page 144)
WRITE_REG	W_WARN_OUT_OF_RANGE	F	30 110	16#759E	Paramètre hors limites
WRITE_REG	E_EFB_NOT_STATE_RAM_4X	F	-30 531	16#88BD	Variable non affectée à la zone %MW (4x)
WRITE_REG	E_EFB_MSTR_ERROR	F	-30 191	16#8A11	Erreur de communication MSTR
WRITE_REG	-	F	8 195	16#2003	Valeur affichée dans le mot d'état (apparaît avec W_WARN_OUT_OF_RANGE)
WRITE_REG	-	F	8 206	16#200E	Valeur affichée dans le mot d'état (apparaît avec E_EFB_NOT_STATE_RAM_4X)
WRITE_REG	-	F	-	-	Voir les tableaux suivants : <ul style="list-style-type: none"> ● Codes d'erreur Modbus Plus et EtherNet SY/MAX (voir page 137) ● Codes d'erreur spécifiques SY/MAX (voir page 142) ● Codes d'erreur EtherNet TCP/IP (voir page 144)

Erreurs courantes relatives aux valeurs en virgule flottante

Introduction

Le tableau ci-dessous répertorie les codes d'erreur et les valeurs générés par des erreurs relatives aux valeurs en virgule flottante.

Erreurs courantes relatives aux valeurs en virgule flottante

Tableau des erreurs courantes relatives aux valeurs en virgule flottante

Codes d'erreur	Valeur d'erreur (format décimal)	Valeur d'erreur (format hexadécimal)	Description de l'erreur
FP_ERROR	-30 150	16#8A3A	Valeur de base (n'apparaît pas comme une valeur d'erreur)
E_FP_STATUS_FAILED_IE	-30 151	16#8A39	Opération incorrecte de virgule flottante
E_FP_STATUS_FAILED_DE	-30 152	16#8A38	L'opérande n'est pas un nombre de type REAL valide
E_FP_STATUS_FAILED_ZE	-30 154	16#8A36	Division par zéro incorrecte
E_FP_STATUS_FAILED_ZE_IE	-30 155	16#8A35	Opération incorrecte de virgule flottante / Division par zéro
E_FP_STATUS_FAILED_OE	-30 158	16#8A32	Dépassement de virgule flottante
E_FP_STATUS_FAILED_OE_IE	-30 159	16#8A31	Opération incorrecte de virgule flottante / Dépassement
E_FP_STATUS_FAILED_OE_ZE	-30 162	16#8A2E	Dépassement de virgule flottante / Division par zéro
E_FP_STATUS_FAILED_OE_ZE_IE	-30 163	16#8A2D	Opération incorrecte de virgule flottante / Dépassement / Division par zéro
E_FP_NOT_COMPARABLE	-30 166	16#8A2A	Erreur interne

Objets système



B

Objet de ce chapitre

Ce chapitre décrit les bits et mots système du langage Unity Pro.

Note : les symboles, associés à chaque objet bit ou mot système, mentionnées dans les tableaux descriptifs de ces objets ne sont pas implémentés de base dans le logiciel, ils peuvent être saisis à l'aide de l'éditeur de données.

Ils sont proposés afin de rendre homogène leur appellation dans les différentes applications.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Sujet	Page
Présentation des bits système	422
Description des bits système %S15 à %S21	423
Description des mots système %SW12 à %SW29	426

Présentation des bits système

Généralités

Les automates Modicon M340, Premium, Atrium et Quantum disposent de bits système %Si qui indiquent les états de l'automate ou permettent d'agir sur le fonctionnement de celui-ci

Ces bits peuvent être testés dans le programme utilisateur afin de détecter toute évolution de fonctionnement devant entraîner une procédure particulière de traitement.

Certains de ces bits doivent être remis dans leur état initial ou normal par le programme. Cependant, les bits système qui sont remis dans leur état initial ou normal par le système ne doivent pas l'être par le programme ou par le terminal

Description des bits système %S15 à %S21

Description détaillée

Description des bits système %S15 à %S21 :

Bit Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%S15 STRINGERROR	Défaut chaîne de caractères	Normalement à l'état 0, ce bit est mis à l'état 1 quand la zone de destination d'un transfert de chaîne de caractères n'a pas la taille suffisante (comprenant le nombre de caractères et le caractère de fin de chaîne de caractères) pour accueillir cette chaîne de caractères. L'application s'arrête en erreur si le bit %S78 a été mis à 1. Ce bit doit être remis à 0 par l'application. Ce bit n'est pas disponible sur les automates de sécurité Quantum.	0	OUI	OUI	OUI (sauf pour les automates de sécurité)
%S16 IOERTSK	Défaut d'entrées/s orties tâche	Normalement à l'état 1, ce bit est réglé sur 0 par le système quand un défaut sur un module en rack ou un équipement sur Fipio est détecté (configuration non conforme, défaut d'échange, défaut matériel, etc.). Ce bit doit être remis à 1 par l'utilisateur.	1	OUI	OUI	OUI

ATTENTION

COMPORTEMENT INATTENDU DE L'APPLICATION - COMPORTEMENT SPECIFIQUE DE VARIABLES

Sur Quantum, les erreurs de communication réseau avec des équipements distants qui sont détectées par les modules de communication (NOM, NOE, NWM, CRA, CRP) et les modules de commande de mouvement (MMS) ne sont pas signalées sur les bits %S10, %S16 et %S119.

Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.

Bit Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%S17 CARRY	Sortie décalage circulaire	Normalement à l'état 0. Lors d'une opération de décalage circulaire, ce bit prend l'état du bit sortant.	0	OUI	OUI	OUI
%S18 OVERFLOW	Dépassement ou erreur arithmétique	<p>Normalement à l'état 0, ce bit est mis à l'état 1 en cas de dépassement de capacité dans les cas suivants :</p> <ul style="list-style-type: none"> ● un résultat supérieur à + 32 767 ou inférieur à - 32 768, en mono-longueur, ● un résultat supérieur à + 65 535, en entier non signé, ● un résultat supérieur à + 2 147 483 647 ou inférieur à - 2 147 483 648, en double-longueur, ● un résultat supérieur à +4 294 967 296, en double-longueur ou en entier non signé, ● valeurs réelles hors bornes, ● division par 0, ● racine d'un nombre négatif, ● forçage à un pas inexistant sur un programmeur cyclique, ● empilage d'un registre plein, dépilage d'un registre vide. <p>Il n'existe qu'un seul cas où le bit %S18 n'est pas augmenté par les automates Modicon M340 lorsque des valeurs réelles sont hors bornes. C'est lorsque des opérands non normalisés ou certaines opérations générant des résultats non normalisés sont utilisés (dépassement progressif par valeur inférieure).</p> <p>Doit être testé par le programme utilisateur après chaque opération présentant un risque de dépassement, puis remis à 0 par l'utilisateur en cas de dépassement.</p> <p>Lorsque le bit %S18 passe à 1, l'application s'arrête en erreur si le bit %S78 a été réglé sur 1.</p>	0	OUI	OUI	OUI

Bit Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%S19 OVERRUN	Dépassement période de tâche (scrutation périodique)	Normalement à l'état 0, ce bit est réglé sur 1 par le système en cas de dépassement de la période d'exécution (temps d'exécution de tâche supérieur à la période définie par l'utilisateur dans la configuration ou programmée dans le mot %SW associé à la tâche). Ce bit doit être remis à 0 par l'utilisateur. Chaque tâche gère son propre bit %S19.	0	OUI	OUI	OUI
%S20 INDEXOVF	Dépassement d'index	Normalement à l'état 0, ce bit est réglé sur 1 lorsque l'adresse de l'objet indexé devient inférieure à 0 ou dépasse le nombre d'objets déclaré dans la configuration. Dans ce cas, l'index est considéré comme étant égal à 0. Doit être testé par le programme utilisateur, après chaque opération où il y a risque de dépassement, puis remis à 0 en cas de dépassement. Lorsque le bit %S20 passe à 1, l'application s'arrête en erreur si le bit %S78 a été mis à 1. Ce bit n'est pas disponible sur les automates de sécurité Quantum.	0	OUI	OUI	OUI (sauf pour les automates de sécurité)
%S21 1RSTASKRUN	Premier cycle de tâche	Testé dans une tâche (Mast, Fast, Aux0, Aux1, Aux2, Aux3), le bit %S21 indique le premier cycle de cette tâche, y compris après un démarrage à froid avec démarrage automatique en mode Run et un démarrage à chaud. %S21 est mis à 1 en début de cycle et remis à zéro en fin de cycle. Remarque : le bit %S21 ne possède pas la même signification sous PL7 et sous Unity Pro.	0	OUI	OUI	OUI

Description des mots système %SW12 à %SW29

Description détaillée

Description des mots système %SW12 à %SW29 :

Mot Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%SW12 UTWPORTADDR	Adresse du port série du processeur	Pour Premium : adresse Uni-Telway du port terminal (en mode esclave) définie dans la configuration et chargée dans ce mot lors d'un démarrage à froid. La modification de la valeur de ce mot n'est pas prise en compte par le système. Pour Modicon M340 : adresse esclave Modbus du port série de l'UC. La modification de la valeur de ce mot n'est pas prise en compte. Ce bit est mis à 0 lorsque l'UC ne dispose pas de port série.	-	OUI	OUI	NON (voir%SW12 ci-dessous)
%SW12 APMODE	Mode du processeur de l'application	Pour les automates de sécurité Quantum uniquement, ce mot indique le mode de fonctionnement du processeur d'application du module d'UC. <ul style="list-style-type: none"> ● 16#A501 = mode de maintenance ● 16#5AFE = mode sûr Toute autre valeur génère une erreur. Remarque : Avec un système de sécurité de redondance d'UC, ce mot est échangé entre l'automate primaire et l'automate redondant afin d'informer ce dernier de l'activation du mode sûr ou de maintenance.	16#A501	NON	NON	OUI Uniquement sur les automates de sécurité

Mot Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%SW13 XWAYNETWADDR	Adresse principale de la station	Ce mot fournit les données suivantes relatives au réseau principal (Fipway ou Ethway) : <ul style="list-style-type: none"> ● le numéro de station (octet de poids faible), compris entre 0 et 127, ● le numéro de réseau (octet de poids fort), compris entre 0 et 63, (la valeur des micro-interrupteurs sur la carte PCMCIA).	254 (16#00FE)	NON	OUI	NON (voir %SW13 ci-dessous)
%SW13 INTELMODE	Mode du processeur Intel	Pour les automates de sécurité Quantum uniquement, ce mot indique le mode de fonctionnement du processeur Intel Pentium du module d'UC. <ul style="list-style-type: none"> ● 16#501A = mode de maintenance ● 16#5AFE = mode sûr Toute autre valeur génère une erreur. Remarque : Avec un système de sécurité de redondance d'UC, ce mot est échangé entre l'automate primaire et l'automate redondant afin d'informer ce dernier de l'activation du mode sûr ou de maintenance.	-	NON	NON	OUI Uniquement sur les automates de sécurité
%SW14 OSCOMMVERS	Version commerciale du processeur de l'automate	Ce mot contient la version actuelle du système d'exploitation du processeur de l'automate. Exemple : 16#0135 version : 01 numéro de révision : 35	-	OUI	OUI	OUI
%SW15 OSCOMPATCH	Version du patch processeur automate	Ce mot contient la version commerciale du patch pour le processeur automate. Le codage s'effectue sur l'octet de poids faible du mot. Codage : 0 = pas de patch, 1 = A, 2 = B... Exemple : 16#0003 correspond au patch C.	-	OUI	OUI	OUI

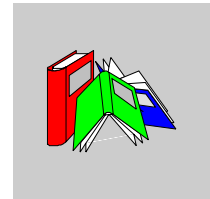
Mot Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%SW16 OSINTVERS	Numéro de version du micrologiciel	Ce mot contient le numéro de version en hexadécimal du micrologiciel du processeur de l'automate. Exemple : 16#0011 version : 2.1 numéro de révision : 17	-	OUI	OUI	OUI
%SW17 FLOATSTAT	Etat d'erreur sur opération flottante	Lorsqu'une erreur est détectée dans une opération arithmétique flottante, le bit %S18 est réglé sur 1 et l'état d'erreur du mot %SW17 est mis à jour selon le codage suivant : <ul style="list-style-type: none"> ● %SW17.0 = opération non valide/le résultat n'est pas un nombre, ● %SW17.1 = opérande non normalisé/résultat acceptable (indicateur non géré par Modicon M340 ou les automates de sécurité Quantum), ● %SW17.2 = division par 0 / le résultat est l'infini, ● %SW17.3 = dépassement / le résultat est l'infini, ● %SW17.4 = dépassement par valeur inférieure/le résultat est 0, ● %SW17.5 à 15 = non utilisés. <p>Ce mot est remis à 0 par le système lors d'un démarrage à froid, mais aussi par le programme pour pouvoir être réutilisé.</p> <p>Ce mot n'est pas disponible sur les automates de sécurité Quantum.</p>	0	OUI	OUI	OUI Uniquement sur les automates de sécurité

Mot Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%SD18 %SW18 et %SW19 100MSCOUNTER	Compteur de temps absolu	<p>%SW18 représente les octets de poids faible et %SW19 les octets de poids fort du double mot %SD18, qui est incrémenté par le système chaque 1/10^e de seconde.</p> <p>L'application peut lire ou écrire ces mots afin d'effectuer des calculs de durée.</p> <p>%SD18 est incrémenté systématiquement, même en mode STOP et dans des états équivalents. Cependant, les durées au cours desquelles l'automate est éteint ne sont pas prises en compte (fonction non liée au programmeur en temps réel, mais uniquement à l'horodateur).</p> <p>Pour les automates de sécurité Quantum, sachant que les 2 processeurs doivent traiter exactement les mêmes données, la valeur de %SD18 est mise à jour au début de la tâche maître, puis reste fixe tout au long de l'exécution de l'application.</p>	0	OUI	OUI	OUI

Mot Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%SD20 : %SW20 et %SW21 MSCOUNTER	Compteur de temps absolu	Pour les automates M340 et Quantum, %SD20 est incrémenté tous les 1/1 000e de seconde par le système (même lorsque l'automate est en mode STOP ; %SD20 n'est plus incrémenté si l'automate est éteint). %SD20 peut être lu par le programme utilisateur ou par le terminal. %SD20 est réinitialisé lors d'un démarrage à froid. %SD20 n'est pas réinitialisé lors d'une reprise à chaud. Dans le cas des automates Premium TSX P57 1•4M/2•4M/3•4M/C024 M/024M et TSX PCI57 204M/354M, %SD20 est incrémenté de 5 tous les 5/1 000e de seconde par le système. Pour tous les autres automates Premium, le compteur de temps de %SD20 est paramétré sur 1 ms comme pour les automates Quantum et M340. Pour les automates de sécurité Quantum, sachant que les 2 processeurs doivent traiter exactement les mêmes données, la valeur de %SD18 est mise à jour au début de la tâche maître, puis reste fixe tout au long de l'exécution de l'application.	0	OUI	OUI	OUI
%SW23	Valeur du commutateur rotatif	L'octet de poids faible contient le commutateur rotatif du processeur Ethernet. Il peut être lu par le programme utilisateur ou par le terminal.	-	OUI	NON	NON
%SW26	Nombre de requêtes traitées	Ce mot système permet de vérifier, côté serveur, le nombre de requêtes traitées par l'automate à chaque cycle.	-	OUI	OUI	OUI

Mot Symbole	Fonction	Description	Etat initial	Modicon M340	Premium Atrium	Quantum
%SW27 %SW28 %SW29	Durée surdébit	<ul style="list-style-type: none"> ● %SW27 contient la dernière durée du surdébit. ● %SW28 contient la durée maximum du surdébit. ● %SW29 contient la durée minimum du surdébit. <p>La durée du surdébit dépend de la configuration (nombre d'E/S, etc.) et des requêtes de cycle en cours (communication, diagnostic). Durée du surdébit = temps de cycle Mast - temps d'exécution du code utilisateur. Ces mots peuvent être lus et écrits par le programme utilisateur ou par le terminal.</p>	-	OUI	NON	NON

Glossaire



0-9

%I

Selon la norme CEI, %I indique un objet langage de type entrée TOR.

%IW

Selon la norme CEI, %IW indique un objet langage de type entrée analogique.

%KW

Selon la norme CEI, %KW indique un objet langage de type mot constante.

%M

Selon la norme CEI, %M indique un objet langage de type bit mémoire.

%MW

Selon la norme CEI, %MW indique un objet langage de type mot mémoire.

%Q

Selon la norme CEI, %Q indique un objet langage de type sortie TOR.

%QW

Selon la norme CEI, %QW indique un objet langage de type sortie analogique.

A

ADDM_TYPE

Ce type prédéfini est utilisé comme sortie pour la fonction ADDM. C'est un tableau ARRAY[0..8] OF Int. Vous pouvez le trouver dans la bibliothèque, dans la même famille que les EF qui l'utilisent.

ADDR_TYPE

Ce type prédéfini est utilisé comme sortie pour la fonction ADDR. C'est un tableau ARRAY[0.0,5] OF Int. Vous pouvez le trouver dans la bibliothèque, dans la même famille que les EF qui l'utilisent.

ANL_IN

ANL_IN est l'abréviation du type de données entrée analogique. Ce type est employé lors du traitement de valeurs analogiques. Les adresses %IW du module d'entrée analogique configuré, qui sont indiquées dans la liste des composants d'E/S, sont automatiquement affectées à des types de données et doivent par conséquent être occupées uniquement par des variables non affectées.

ANL_OUT

ANL_OUT est l'abréviation du type de données sortie analogique. Ce type est employé lors du traitement de valeurs analogiques. Les adresses %MW du module d'entrée analogique configuré, qui sont indiquées dans la liste des composants d'E/S, sont automatiquement affectées à des types de données et doivent par conséquent être occupées uniquement par des variables non affectées.

ANY

Une hiérarchie existe entre les différents types de données. Dans les DFB, il est parfois possible de déclarer les variables pouvant contenir plusieurs types de valeurs. On utilise alors les types `ANY_XXX`.

La figure suivante décrit cette structure hiérarchisée :

```

ANY
  ANY_ELEMENTARY
    ANY_MAGNITUDE_OR_BIT
      ANY_MAGNITUDE
        ANY_NUM
          ANY_REAL
            REAL
          ANY_INT
            DINT, INT, UDINT, UINT
        TIME
      ANY_BIT
        DWORD, WORD, BYTE, BOOL
    ANY_STRING
      STRING
    ANY_DATE
      DATE_AND_TIME, DATE, TIME_OF_DAY
      EBOOL
  ANY_DERIVED
    ANY_ARRAY
      ANY_ARRAY_ANY_EDT
      ANY_ARRAY_ANY_MAGNITUDE
        ANY_ARRAY_ANY_NUM
          ANY_ARRAY_ANY_REAL
            ANY_ARRAY_REAL
          ANY_ARRAY_ANY_INT
            ANY_ARRAY_DINT
            ANY_ARRAY_INT
            ANY_ARRAY_UDINT
            ANY_ARRAY_UINT
          ANY_ARRAY_TIME
        ANY_ARRAY_ANY_BIT
          ANY_ARRAY_DWORD
          ANY_ARRAY_WORD
          ANY_ARRAY_BYTE
          ANY_ARRAY_BOOL
        ANY_ARRAY_ANY_STRING
          ANY_ARRAY_STRING
        ANY_ARRAY_ANY_DATE
          ANY_ARRAY_DATE_AND_TIME
          ANY_ARRAY_DATE
          ANY_ARRAY_TIME_OF_DAY
        ANY_ARRAY_EBOOL
      ANY_ARRAY_ANY_DDT
    ANY_STRUCTURE
      ANY_DDT
      ANY_IODDT
      ANY_FFB
        ANY_EFB
        ANY_DFB

```

ARRAY

Un `ARRAY` est un tableau d'éléments de même type.

La syntaxe est la suivante : `ARRAY [<limites>] OF <Type>`

Exemple :

`ARRAY [1..2] OF BOOL` est un tableau à une dimension composé de deux éléments de type `BOOL`.

`ARRAY [1..10, 1..20] OF INT` est un tableau à deux dimensions composé de 10x20 éléments de type `INT`.

B

BCD

BCD est l'abréviation du format Binary Coded Decimal (décimal codé en binaire).

Le format BCD permet de représenter des nombres décimaux compris entre 0 et 9 à l'aide d'un groupe de quatre bits (demi-octet).

Dans ce format, les quatre bits employés pour coder les nombres décimaux ont une plage de combinaisons inutilisée.

Exemple de codage BCD :

- Le nombre 2 450
- est ainsi codé : 0010 0100 0101 0000

BOOL

`BOOL` est l'abréviation du type booléen. Il s'agit du type de données de base en informatique. Une variable de type `BOOL` possède l'une ou l'autre des valeurs suivantes : 0 (`FALSE`) ou 1 (`TRUE`).

Un bit extrait d'un mot est de type `BOOL`, par exemple : `%MW10.4`.

BYTE

Lorsque 8 bits sont regroupés, on parle alors de `BYTE` (octet). La saisie d'un `BYTE` s'effectue soit en mode binaire, soit en base 8.

Le type `BYTE` est codé dans un format 8 bits qui, au format hexadécimal, s'étend de `16#00` à `16#FF`.

C

CEI 61131-3

Norme internationale : automates programmables

Partie 3 : langages de programmation

Conventions de désignation (identificateur)

Un identificateur est une suite de lettres, de chiffres et de signes de soulignement commençant par une lettre ou un signe de soulignement (par exemple, le nom d'un type de bloc fonction, d'une instance, d'une variable ou d'une section). Les lettres accentuées (comme ö, ü, é et õ) peuvent être utilisées, sauf dans les noms de projet et DFB. Les signes de soulignement sont significatifs dans les identificateurs. Par exemple, A_BCD et AB_CD sont interprétés comme des identificateurs différents. L'utilisation de plusieurs soulignés consécutifs ou au début d'un identificateur est incorrecte.

Les identificateurs ne peuvent pas contenir d'espace. Ils ne distinguent pas les majuscules des minuscules. Par exemple, ABCD et abcd sont interprétés comme un seul et même identificateur.

Selon la norme CEI 61131-3, les chiffres initiaux ne sont pas autorisés dans les identificateurs. Cependant, vous pouvez les utiliser si vous cochez, dans la boîte de dialogue **Outils** → **Options du projet**, onglet **Extensions de langage**, la case **Chiffres en début autorisés**.

Les identificateurs ne peuvent pas être des mots clés.

D

DATE

Le type DATE, codé en BCD dans un format de 32 bits, contient les informations suivantes :

- l'année codée dans un champ de 16 bits ;
- le mois codé dans un champ de 8 bits ;
- le jour codé dans un champ de 8 bits.

Le type DATE doit être saisi comme suit : **D#** <Année> - <Mois> - <Jour>

Le tableau ci-après donne les limites inférieure/supérieure de chaque élément :

Élément	Limites	Commentaire
Année	[1990,2099]	Année
Mois	[01,12]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.
Jour	[01,31]	Pour les mois 01/03/05/07/08/10/12
	[01,30]	Pour les mois 04/06/09/11
	[01,29]	Pour le mois 02 (années bissextiles)
	[01,28]	Pour le mois 02 (années non bissextiles)

DATE_AND_TIME

Voir DT.

DBCD

Représentation d'un entier double au format double BCD.

Le format BCD (Binary Coded Decimal) est utilisé pour représenter des nombres décimaux compris entre 0 et 9 à l'aide d'un groupe de quatre bits.

Dans ce format, les quatre bits employés pour coder les nombres décimaux ont une plage de combinaisons inutilisée.

Exemple de codage DBCD :

- Le nombre 78 993 016
- est ainsi codé : 0111 1000 1001 1001 0011 0000 0001 0110

DDT

DDT est l'abréviation de Derived Data Type (type de données dérivées).

Un type de données dérivées est un ensemble d'éléments de même type (ARRAY) ou de types différents (structure).

DFB

DFB est l'abréviation de Derived Function Block (bloc fonction dérivé).

Les types DFB sont des blocs fonction programmables par l'utilisateur en langage ST, IL, LD ou FBD.

L'utilisation de ces types DFB dans une application permet :

- de simplifier la conception et la saisie du programme ;
- d'accroître la lisibilité du programme ;
- de faciliter sa mise au point ;
- de diminuer le volume de code généré.

DINT

DINT est l'abréviation du format Double INTeget (entier double) (codé sur 32 bits).

Les limites inférieure et supérieure sont les suivantes : -(2 puissance 31) à (2 puissance 31) -1.

Exemple :

-2147483648, 2147483647, 16#FFFFFFFF.

DT

DT est l'abréviation de Date and Time (date et heure).

Le type DT, codé en BCD dans un format de 64 bits, contient les informations suivantes :

- l'année codée dans un champ de 16 bits ;
- le mois codé dans un champ de 8 bits ;
- le jour codé dans un champ de 8 bits ;
- l'heure codée dans un champ de 8 bits ;
- les minutes codées dans un champ de 8 bits ;
- les secondes codées dans un champ de 8 bits.

NOTE : les 8 bits de poids faible ne sont pas utilisés.

Le type DT doit être saisi comme suit :

DT# <Année> - <Mois> - <Jour> - <Heure> : <Minutes> : <Secondes>

Le tableau ci-après donne les limites inférieure/supérieure de chaque élément :

Champ	Limites	Commentaire
Année	[1990,2099]	Année
Mois	[01,12]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.
Jour	[01,31]	Pour les mois 01/03/05/07/08/10/12
	[01,30]	Pour les mois 04/06/09/11
	[01,29]	Pour le mois 02 (années bissextiles)
	[01,28]	Pour le mois 02 (années non bissextiles)
Heure	[00,23]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.
Minute	[00,59]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.
Seconde	[00,59]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.

DWORD

DWORD est l'abréviation de Double Word (mot double).

Le type **DWORD** est codé dans un format de 32 bits.

Le tableau ci-dessous donne les limites inférieure/supérieure des bases qui peuvent être utilisées :

Base	Limite inférieure	Limite supérieure
Hexadécimale	16#0	16#FFFFFFFF
Octale	8#0	8#3777777777
Binaire	2#0	2#11111111111111111111111111111111

Exemples de représentation :

Données	Représentation dans l'une des bases
00000000000010101101110011011110	16#ADCDE
000000000000001000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

E**EBOOL**

EBOOL est l'abréviation du type Extended BOOLEan (booléen étendu). Une variable de type **EBOOL** possède une valeur (0 pour **FALSE** ou 1 pour **TRUE**), mais également des fronts montants ou descendants et des fonctions de forçage.

Elle occupe un octet de mémoire.

L'octet contient les informations suivantes :

- un bit pour la valeur ;
- un bit pour l'historique (chaque fois que l'objet change d'état, la valeur est copiée dans ce bit) ;
- un bit pour le forçage (égal à 0 si l'objet n'est pas forcé, égal à 1 s'il est forcé).

La valeur par défaut de chaque bit est 0 (**FALSE**).

EF

EF est l'acronyme de Elementary Function (fonction élémentaire).

Il s'agit d'un bloc, utilisé dans un programme, qui réalise une fonction logique prédéterminée.

Une fonction ne dispose pas d'informations sur l'état interne. Plusieurs appels de la même fonction à l'aide des mêmes paramètres d'entrée fournissent toujours les mêmes valeurs de sortie. Vous trouverez des informations sur la forme graphique de l'appel de fonction dans le « [bloc fonctionnel (instance)] ». Contrairement aux appels de bloc fonction, les appels de fonction comportent uniquement une sortie qui n'est pas nommée et dont le nom est identique à celui de la fonction. Dans FBD, chaque appel est indiqué par un [numéro] unique via le bloc graphique. Ce numéro est généré automatiquement et ne peut pas être modifié.

Vous positionnez et paramétrez ces fonctions dans votre programme afin d'exécuter votre application.

Vous pouvez également développer d'autres fonctions à l'aide du kit de développement SDKC.

EFB

EFB est l'abréviation de Elementary Function Block (bloc fonction élémentaire).

Il s'agit d'un bloc, utilisé dans un programme, qui réalise une fonction logique prédéterminée.

Les EFB possèdent des états et des paramètres internes. Même si les entrées sont identiques, les valeurs des sorties peuvent différer. Par exemple, un compteur possède une sortie qui indique que la valeur de présélection est atteinte. Cette sortie est réglée sur 1 lorsque la valeur en cours est égale à la valeur de présélection.

EN

EN correspond à **EN**able (activer) ; il s'agit d'une entrée de bloc facultative. Lorsque l'entrée EN est activée, une sortie ENO est automatiquement établie.

Si EN = 0, le bloc n'est pas activé, son programme interne n'est pas exécuté et ENO est réglé sur 0.

Si EN = 1, le programme interne du bloc est exécuté et ENO est réglé sur 1. Si une erreur survient, ENO reprend la valeur 0.

Si l'entrée EN n'est pas connectée, elle est automatiquement réglée sur 1.

ENO

ENO correspond à **Error NOT**ification (notification d'erreur) ; il s'agit de la sortie associée à l'entrée facultative EN.

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

- l'état des sorties de blocs fonction reste identique à celui dans lequel elles étaient lors du dernier cycle de scrutation exécuté correctement ;
- la ou les sorties de la fonction, ainsi que les procédures, sont réglées sur 0.

F

FBD

FBD est l'abréviation de Function Block Diagram (langage en blocs fonction).

FBD est un langage de programmation graphique qui fonctionne comme un logigramme. Par l'ajout de blocs logiques simples (ET, OU, etc.), chaque fonction ou bloc fonction du programme est représenté(e) sous cette forme graphique. Pour chaque bloc, les entrées se situent à gauche et les sorties à droite. Les sorties des blocs peuvent être liées aux entrées d'autres blocs afin de former des expressions complexes.

FFB

Terme générique pour EF (fonction élémentaire), EFB (bloc fonction élémentaire) et DFB (bloc fonction dérivé).

Fonction

Voir EF.

Fonction élémentaire

Voir EF.

FTP

File Transfer Protocol (protocole de transfert de fichiers).

G

Global Data

Global Data fournit un échange automatique de variables de données pour la coordination d'applications d'automates.

GRAY

Le code Gray, ou « binaire réfléchi », permet de coder une valeur numérique développée en chaîne de configurations binaires pouvant être différenciées par le changement d'état d'un seul bit.

Ce code peut être utilisé, par exemple, pour éviter l'événement aléatoire suivant : en binaire pur, le changement de la valeur 0111 en 1000 peut produire des nombres aléatoires compris entre 0 et 1 000, étant donné que les bits ne changent pas tous de valeur simultanément.

Equivalence entre décimal, BCD et Gray :

Décimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Gray	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101

H

HTTP

Protocole de transfert hypertexte

I

IL

IL est l'abréviation de Instruction List (liste d'instructions).

Ce langage est une suite d'instructions simples.

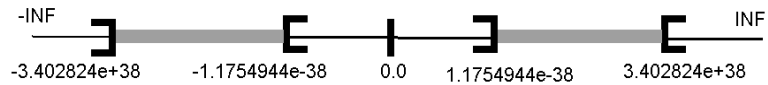
Il est très proche du langage d'assemblage utilisé pour programmer les processeurs.

Chaque instruction est composée d'un code instruction et d'un opérande.

INF

Utilisé pour signifier qu'un nombre dépasse les limites autorisées.

Pour un nombre de type entier, les plages de valeurs (indiquées en gris) sont les suivantes :



Lorsqu'un résultat est :

- inférieur à $-3,402824e+38$, le symbole $-INF$ (pour -infini) est affiché ;
- supérieur à $+3,402824e+38$, le symbole INF (pour +infini) est affiché.

INT

INT est l'abréviation du format single **INT**eger (entier simple) (codé sur 16 bits).

Les limites inférieure et supérieure sont les suivantes : $-(2 \text{ puissance } 15)$ à $(2 \text{ puissance } 15) - 1$.

Exemple :

-32768 , 32767 , $2\#11111110001001001$, $16\#9FA4$.

IODDT

IODDT est l'abréviation de **I**nterface **O**utput **D**erived **D**ata **T**ype (type de données dérivées d'E/S).

Cet acronyme désigne un type de données structuré représentant un module ou une voie d'un module automate. Chaque module expert possède ses propres **IODDT**.

J**Jeton**

Etape active d'un SFC.

Jeton unique

Mode de fonctionnement d'un diagramme SFC pour lequel une seule étape peut être active à la fois.

L

Langage en blocs fonction

Voir FBD.

LD

LD est l'abréviation de Ladder Diagram (langage à contacts).

LD est un langage de programmation représentant les instructions à exécuter sous forme de schémas graphiques très proches d'un schéma électrique (contacts, bobines, etc.).

M

Mot clé

Un mot clé est une combinaison de caractères unique employée en tant qu'élément syntaxique d'un langage de programmation. (Voir la définition fournie à l'annexe B de la norme CEI 61131-3. Tous les mots clés utilisés dans Unity Pro et mentionnés dans la norme CEI 61131-3 sont répertoriés à l'annexe C de cette norme. Ils ne peuvent pas servir d'identificateurs [noms de variables, sections, types DFB, etc.] dans votre programme.)

Multijeton

Mode de fonctionnement d'un SFC. En mode multijeton, le SFC peut posséder plusieurs étapes actives simultanément.

N

NAN

Utilisé pour signifier que le résultat d'une opération n'est pas un nombre (NAN = Not A Number).

Exemple : calcul de la racine carrée d'un nombre négatif.

NOTE : la norme CEI 559 définit deux classes de NAN : le NAN silencieux (QNaN) et le NAN de signalisation (SNaN). Un QNaN est un NAN (Not a Number) avec un bit de fraction de poids fort tandis qu'un SNaN est un NAN sans bit de fraction de poids fort (numéro de bit 22). Les QNaN peuvent être propagés par la plupart des opérations arithmétiques sans générer d'exception. Quant aux SNaN, ils signalent en général une opération non valide lorsqu'ils sont utilisés en tant qu'opérandes dans des opérations arithmétiques (voir %SW17 et %S18).

P

Peer Cop

Le service Peer Cop est un mécanisme d'échange automatique entre des stations connectées sur le même segment Modbus Plus.

Procédure

Les procédures sont des vues techniquement fonctionnelles. L'unique différence par rapport aux fonctions élémentaires est que les procédures peuvent inclure plusieurs sorties et qu'elles prennent en charge le type de données VAR_IN_OUT. En apparence, les procédures ne sont pas différentes des fonctions élémentaires.

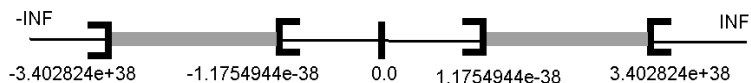
Les procédures sont un supplément à la norme CEI 61131-3.

R

REAL

Le type REAL (réel) est un type codé sur 32 bits.

Les plages de valeurs possibles sont illustrées dans la figure suivante :



Lorsqu'un résultat est :

- compris entre $-1,175494e-38$ et $1,175494e-38$, il est considéré comme étant un `DEN` ;
- inférieur à $-3,402824e+38$, le symbole `-INF` (pour -infini) est affiché ;
- supérieur à $+3,402824e+38$, le symbole `INF` (pour +infini) est affiché ;
- indéfini (racine carrée d'un nombre négatif), le symbole `NAN` est affiché.

NOTE : la norme standard CEI 559 définit deux classes de `NAN` : le `NAN` silencieux (`QNAN`) et le `NAN` de signalisation (`SNAN`). Un `QNAN` est un `NAN` (Not a Number) avec un bit de fraction de poids fort tandis qu'un `SNAN` est un `NAN` sans bit de fraction de poids fort (numéro de bit 22). Les `QNAN` peuvent être propagés par la plupart des opérations arithmétiques sans générer d'exception. Quant aux `SNAN`, ils signalent en général une opération non valide lorsqu'ils sont utilisés en tant qu'opérandes dans des opérations arithmétiques (voir `%SW17` et `%S18`).

NOTE : lorsqu'un `DEN` (nombre non normalisé) est utilisé en tant qu'opérande, le résultat n'est pas significatif.

Réseau

Il existe deux significations du terme réseau.

- En LD :
Un réseau est un ensemble d'éléments graphiques interconnectés. La portée d'un réseau est locale, par rapport à l'unité (la section) organisationnelle du programme dans laquelle le réseau est situé.
- Avec des modules de communication experts :
Un réseau est un groupe de stations qui communiquent entre elles. Le terme réseau est également utilisé pour définir un groupe d'éléments graphiques interconnectés. Ce groupe constitue ensuite une partie d'un programme qui peut être composée d'un groupe de réseaux.

S

Scrutation d'E/S

Interrogation continuelle des modules d'E/S afin de rassembler les bits de données et les informations d'état, d'erreur et de diagnostic. Ce processus permet de surveiller les entrées et les sorties de contrôle.

SFC

SFC est l'abréviation de Sequential Function Chart (diagramme fonctionnel en séquence).

Le SFC permet de représenter graphiquement et de façon structurée le fonctionnement d'un automatisme séquentiel. Cette description graphique du comportement séquentiel de l'automatisme et des différentes situations qui en découlent s'effectue à l'aide de symboles graphiques simples.

SIL

Niveau d'intégrité de la sécurité

Les fonctions de sécurité sont exécutées dans le but de sécuriser un système, puis de le maintenir dans cet état. La norme IEC 61508 stipule 4 niveaux de performances de sécurité pour une fonction de sécurité. Ces niveaux sont appelés niveaux d'intégrité de la sécurité (SIL) et portent les numéros 1 (niveau le plus faible) à 4 (niveau le plus élevé). L'utilisation de l'automate de sécurité Quantum dans les applications SIL2, dans lesquelles l'état de sécurité est l'état de non-alimentation, est certifiée, comme dans un système d'arrêt d'urgence.

Vous pouvez recourir aux produits de sécurité de Schneider pour mettre au point une solution de redondance d'UC si vous souhaitez disposer d'un système de sécurité hautement disponible.

SNMP

Protocole SNMP (Simple Network Management Protocol)

ST

ST est l'abréviation de Structured Text (langage littéral structuré).

Le langage littéral structuré est un langage élaboré proche des langages de programmation informatiques. Il permet de structurer des suites d'instructions.

STRING

Une variable de type `STRING` est une chaîne de caractères ASCII. La longueur maximale d'une chaîne de caractères est de 65 534 caractères.

T

TIME

Le type `TIME` exprime une durée en millisecondes. Codé sur 32 bits, ce type permet d'obtenir des durées de 0 à $2^{32}-1$ millisecondes.

Les unités du type `TIME` sont les suivantes : jours (d), heures (h), minutes (m), secondes (s) et millisecondes (ms). Une valeur littérale de type `TIME` est représentée par une combinaison des types précédents associés au préfixe `T#`, `t#`, `TIME#` ou `time#`.

Exemples : `T#25h15m`, `t#14`, `7S`, `TIME#5d10h23m45s3ms`

TIME_OF_DAY

Voir `TOD`.

TOD

`TOD` est l'abréviation de Time Of Day (heure de la journée).

Le type `TOD`, codé en BCD dans un format de 32 bits, contient les informations suivantes :

- l'heure codée dans un champ de 8 bits ;
- les minutes codées dans un champ de 8 bits ;
- les secondes codées dans un champ de 8 bits.

NOTE : les 8 bits de poids faible ne sont pas utilisés.

Le type Time Of Day doit être saisi comme suit : `TOD# <Heure> : <Minutes> : <Secondes>`

Le tableau ci-après donne les limites inférieure/supérieure de chaque élément :

Champ	Limites	Commentaire
Heure	[00,23]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.
Minute	[00,59]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.
Seconde	[00,59]	Le 0 initial est toujours affiché; il peut être omis lors de la saisie.

Exemple : `TOD#23:59:45`.

TOPO_ADDR_TYPE

Ce type prédéfini est utilisé comme sortie pour la fonction `READ_TOPO_ADDR`. C'est un tableau `ARRAY[0..4] OF Int`. Vous pouvez le trouver dans la bibliothèque, dans la même famille que les EF qui l'utilisent.

U

UDINT

UDINT est l'abréviation du format Unsigned Double INTeger (entier double non signé) (codé sur 32 bits). Les limites inférieure et supérieure sont les suivantes : 0 à (2 puissance 32) - 1.

Exemple :

```
0, 4294967295, 2#111111111111111111111111111111111111111111111111111, 8#377777777777, 16#FFFFFFFF.
```

UDP

User Datagram Protocol (protocole datagramme utilisateur). UDP est un protocole Internet de communication sans connexion défini par l'IETF RFC 768. Il permet la transmission directe de datagrammes sur des réseaux IP. Les messages UDP/IP n'attendent pas de réponse et, de ce fait, ils sont particulièrement adaptés aux applications dans lesquelles aucune retransmission des paquets envoyés n'est nécessaire (comme dans la vidéo en continu ou les réseaux exigeant des performances en temps réel).

UINT

UINT est l'abréviation du format Unsigned INTeger (entier non signé) (codé sur 16 bits). Les limites inférieure et supérieure sont les suivantes : 0 à (2 puissance 16) - 1.

Exemple :

```
0, 65535, 2#11111111111111111111111111111111, 8#177777, 16#FFFF.
```

V

Valeur littérale d'entier

Une valeur littérale d'entier est utilisée pour saisir des valeurs de type entier dans le système décimal. Les valeurs peuvent être précédées d'un signe (+/-). Les signes de soulignement (_) séparant les nombres ne sont pas significatifs.

Exemple :

```
-12, 0, 123_456, +986
```

Valeur littérale de temps

Les unités du type `TIME` sont les suivantes : jours (`d`), heures (`h`), minutes (`m`), secondes (`s`) et millisecondes (`ms`). Une valeur littérale de type `TIME` est représentée par une combinaison des types précédents associés au préfixe `T#`, `t#`, `TIME#` ou `time#`.

Exemples : `T#25h15m`, `t#14,7S`, `TIME#5d10h23m45s3ms`

Valeur littérale en base 10

Une valeur littérale en base 10 est utilisée pour représenter une valeur entière décimale. Cette valeur peut être précédée des signes « + » et « - ». Si le caractère « _ » est utilisé dans la valeur littérale, il n'est pas significatif.

Exemple :

`-12,0`, `123_456`, `+986`

Valeur littérale en base 16

Une valeur littérale en base 16 est utilisée pour représenter un entier hexadécimal. La base est déterminée par le nombre « 16 » et le signe « # ». Les signes « + » et « - » sont interdits. Pour faciliter la lecture, vous pouvez utiliser le signe « _ » entre les bits.

Exemple :

`16#F_F` ou `16#FF` (qui correspond au nombre décimal 255)

`16#E_0` ou `16#E0` (qui correspond au nombre décimal 224)

Valeur littérale en base 2

Une valeur littérale en base 2 est utilisée pour représenter un entier binaire. La base est déterminée par le nombre « 2 » et le signe « # ». Les signes « + » et « - » sont interdits. Pour faciliter la lecture, vous pouvez utiliser le signe « _ » entre les bits.

Exemple :

`2#1111_1111` ou `2#11111111` (qui correspond au nombre décimal 255)

`2#1110_0000` ou `2#11100000` (qui correspond au nombre décimal 224)

Valeur littérale en base 8

Une valeur littérale en base 8 est utilisée pour représenter un entier octal. La base est déterminée par le nombre « 8 » et le signe « # ». Les signes « + » et « - » sont interdits. Pour faciliter la lecture, vous pouvez utiliser le signe « _ » entre les bits.

Exemple :

`8#3_77` ou `8#377` (qui correspond au nombre décimal 255)

`8#34_0` ou `8#340` (qui correspond au nombre décimal 224)

Valeur littérale réelle

Une valeur littérale réelle est un nombre exprimé avec une ou plusieurs décimales.

Exemple :

-12,0,0,0,+0,456,3,14159_26

Valeur littérale réelle avec exposant

Nombre pouvant être exprimé à l'aide d'une notation scientifique standard. La représentation est alors la suivante : mantisse + exposant.

Exemple :

-1,34E-12 ou -1,34e-12

1,0E+6 ou 1,0e+6

1,234E6 ou 1,234e6

Variable

Entité mémoire du type `BOOL`, `WORD`, `DWORD`, etc., dont le contenu peut être modifié par le programme en cours d'exécution.

Variable non affectée

Variable dont la position dans la mémoire de l'automate ne peut pas être connue. Une variable à laquelle aucune adresse n'a été associée est dite non affectée.

Variables affectées

Variable dont la position dans la mémoire de l'automate peut être connue. Par exemple, la variable `Pression_eau` est associée au repère `%MW102`. `Pression_eau` est dite affectée.

W

WORD

Le type `WORD` est codé dans un format de 16 bits et est utilisé pour effectuer des traitements sur des chaînes de bits.

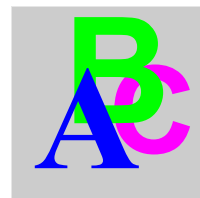
Le tableau ci-dessous donne les limites inférieure/supérieure des bases qui peuvent être utilisées :

Base	Limite inférieure	Limite supérieure
Hexadécimale	16#0	16#FFFF
Octale	8#0	8#177777
Binaire	2#0	2#1111111111111111

Exemples de représentation

Données	Représentation dans l'une des bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index



A

ADDM, 43
ADDM_TYPE, 43
ADDR, 47
ADDR_TYPE, 47

C

CANCEL, 49
codes d'erreur, 415
communication - instructions
 ADDM, 43
 ADDR, 47
communication, instruction
 CANCEL, 49
 CREAD_REG, 53
 CWRITE_REG, 61
 DATA_EXCH, 69
 informations générales sur Premium et

Atrium, 27
INPUT_BYTE, 79
INPUT_CHAR, 83
MBP_MSTR, 95
ModbusP_ADDR, 151
OUT_IN_CHAR, 157
OUT_IN_MBUS, 167
PRINT_CHAR, 205
RCV_TLG, 217
READ_ASYN, 223
READ_GDATA, 227
READ_REG, 229
READ_VAR, 239
SEND_REQ, 261
SEND_TLG, 279
SYMAX_IP_ADDR, 285
TCP_IP_ADDR, 291
UNITE_SERVER, 297
WRITE_ASYN, 303
WRITE_GDATA, 307
WRITE_REG, 309
WRITE_VAR, 319
XXMIT, 333

CREAD_REG, 53
CWRITE_REG, 61

D

DATA_EXCH, 69
disponibilité des instructions, 25

G

gestion des chaînes, instruction
 INPUT_CHAR, 83
 OUT_IN_CHAR, 157
 PRINT_CHAR, 205

I

INPUT_BYTE, 79
INPUT_CHAR, 83
instructions
 disponibilité, 25

M

MBP_MSTR, 95
ModbusP_ADDR, 151

O

OUT_IN_CHAR, 157
OUT_IN_MBUS, 167

P

PRINT_CHAR, 205

R

RCV_TLG, 217
READ_ASYN, 223
READ_GDATA, 227
READ_REG, 229
READ_VAR, 239

S

SEND_REQ, 261
SEND_TLG, 279
SYMAX_IP_ADDR, 285

T

TCP_IP_ADDR, 291

U

UNITE_SERVER, 297

W

WRITE_ASYN, 303
WRITE_GDATA, 307
WRITE_REG, 309
WRITE_VAR, 319

X

XXMIT, 333