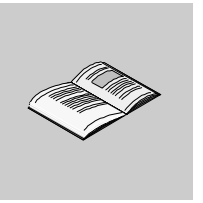
OUT_IN_MBUS
Fonction de communication Modbus

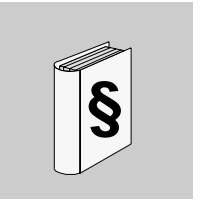
fre Janvier 2006

35011559_00

Table des matières



Consignes de sécurité



Informations importantes

AVIS

Veuillez lire soigneusement ces consignes et examiner l'appareil afin de vous familiariser avec lui avant son installation, son fonctionnement ou son entretien. Les messages particuliers qui suivent peuvent apparaître dans la documentation ou sur l'appareil. Ils vous avertissent de dangers potentiels ou attirent votre attention sur des informations susceptibles de clarifier ou de simplifier 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 dangereuse **entraînant** la mort, des blessures graves ou des dommages matériels.

⚠ AVERTISSEMENT

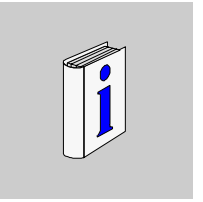
AVERTISSEMENT indique une situation présentant des risques **pouvant entraîner** la mort, des blessures graves ou des dommages matériels.

⚠ ATTENTION
ATTENTION indique une situation potentiellement dangereuse **pouvant entraîner** des lésions corporelles ou des dommages matériels.

REMARQUE IMPORTANTE

L'entretien du matériel électrique ne doit être effectué que par du personnel qualifié. Schneider Electric n'assume aucune responsabilité des conséquences éventuelles découlant de l'utilisation de cette documentation. Ce document n'a pas pour objet de servir de guide aux personnes sans formation.

A propos de ce manuel



Présentation

Objectif du document

L'objectif de ce manuel est de mettre en œuvre la fonction de communication `OUT_IN_MBUS`.

Document à consulter

Titre	Référence
Modbus - Guide utilisateur	TSX DG MDB

Commentaires utilisateur

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

OUT_IN_MBUS : fonction de communication Modbus

1

Présentation

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 :

1.1 Présentation générale du bloc de communication OUT_IN_MBUS

Présentation

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 :

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 caractère.

Combinée avec la possibilité de passer à la volée d'une configuration Modbus esclave à une configuration en mode caractère via la fonction WRITE_CMD, elle permet d'utiliser sur une même liaison l'automate tantôt en maître tantôt en mode esclave Modbus.

Note : cette fonction n'a d'intérêt que lorsque les 2 modes Modbus maître et esclave doivent cohabiter. Si ce n'est pas le cas, il est recommandé d'utiliser les EF standards READ_VAR et WRITE_VAR pour gérer la fonction Modbus maître (Modbus esclave est géré implicitement par le système). Se référer dans ce cas à la documentation Modbus.

AVERTISSEMENT

Risques liés à l'écriture de variables.

Les modifications incorrectes qui sont apportées à l'automate cible peuvent avoir des effets indésirables sur le comportement de l'application.
Il vous incombe de prendre toutes les précautions nécessaires avant de mettre en œuvre le bloc OUT_IN_MBUS.

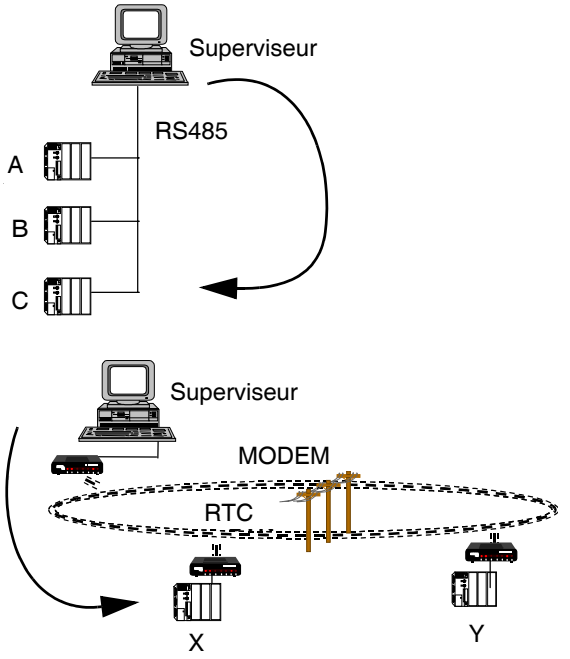
Le non-respect de cette instruction peut entraîner la mort, des blessures graves ou des dommages matériels.

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) :

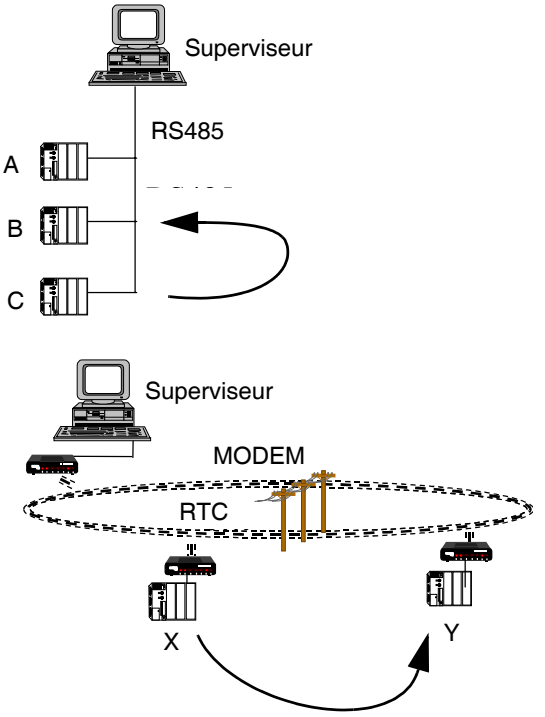


Le mode d'exception

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 supporte :

- les codes Modbus 1, 2, 3, 4, 5, 6, 15, 16 (Voir *Le paramètre* MbusCmd, p. 21),
- les modes ASCII et RTU (Voir *Le paramètre* DataBits, p. 25),
- l'adressage complet (Voir *Le paramètre* MbusCmd, p. 21).

La fonction OUT_IN_MBUS se présentant sous forme d'un DFB, son exécution doit être relancée à chaque cycle automate tant que le bit d'activité est à 1.

La fonction est disponible sur les automates Premium au travers d'une carte de communication TSXSCP111 ou 114 installée dans le processeur ou un module d'accueil (TSXSCY21601 ou TSXSCY11601). Sa mise en œuvre s'effectue à l'aide d'Unity Pro V2.2.

Restrictions

Sur chacun des ports, il ne faut pas activer simultanément plus d'un DFB OUT_IN_MBUS. De la même façon, il ne faut pas utiliser simultanément les blocs PRINT_CHAR, INPUT_CHAR ou OUT_IN_CHAR.

La fonction OUT_IN_MBUS ne supporte pas :

- le changement des paramètres des couches physique ou liaison : débit, format des caractères, passage de RTU à ASCII ou inversement, gestion des signaux RS232,
- les conflits et erreurs de communication pouvant résulter de la présence de 2 maîtres Modbus actifs simultanément sur la même liaison,
- tous les modes de marche pouvant résulter d'un défaut ou d'une connexion temporaire d'une carte de communication,
- la configuration et la gestion des modems.

1.2 Description du bloc de communication OUT_IN_MBUS

Présentation

Objet de ce sous-chapitre

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

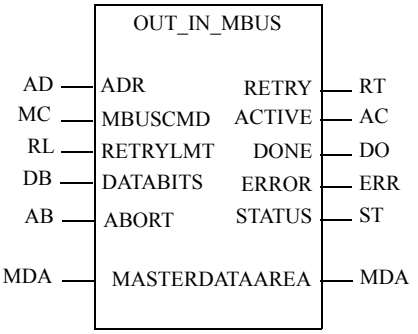
Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Représentations et paramètres

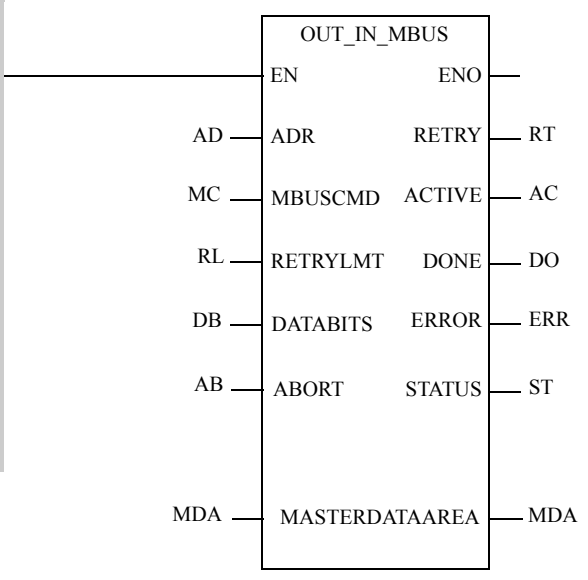
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Address
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
ADR	ADDR_TYPE	Information de l'adresse du port de communication de l'équipement esclave. Cette adresse est obligatoirement de la forme : ADDR('r.m.c.SYS'). Abréviations utilisées : r = rack, m = emplacement du module, c = voie.
MBUSCMD	ARRAY [1.. 4] OF INT	Tableau de définition (Voir *Définition du paramètre MbusCmd, p. 21*) Modbus.
RETRYLMT	INT	Nombre de tentatives (Voir *Le paramètre* RetryLmt, p. 24) d'envoi d'un message effectuées par le bloc.
DATABITS	BOOL	Messages (Voir *Le paramètre* DataBits, p. 25) Modbus à envoyer en mode ASCII (DATABITS =0) ou RTU (DATABITS=1).
RESPTOUT	INT	Temps d'attente (Voir *Le paramètre* RespTout, p. 26) du bloc.
ABORT	BOOL	Bit d'annulation du DFB.

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

Paramètre	Type	Commentaire
MASTERDATAAREA	ARRAY [x.. y] OF INT	Zone de données (Voir *Le paramètre* MasterDataArea, p. 27) de l'automate maître.

Note : le paramètre d'entrée/sortie utilise un tableau dynamique. Dans ce cas il faut cocher dans Unity Pro 'Autoriser tableaux dynamiques [ANY_ARRAY_XXX]' sous **outil** → **option du projet** → **extension de langage**.

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

Paramètre	Type	Commentaire
RETRY	INT	La valeur affichée indique le nombre courant de tentatives 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 a réussi.
ERROR	BOOL	La valeur 1 indique qu'une erreur est survenue.

Paramètre	Type	Commentaire
STATUS	INT	La valeur affiche un code d'état (Voir *Le paramètre* Status, p. 28) 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	Définition
MbusCmd[1]	Adresse esclave	Ce mot contient l'adresse de l'automate Modbus esclave. 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. 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. 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.
MbusCmd[2]	Code fonction Modbus	Le bloc OUT_IN_MBUS prend en charge les codes de fonction suivants : • 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). **Note** : quand l'automate esclave est de type Premium, tous les bits deviennent des **%M** et les registres deviennent des **%MW**.
MbusCmd[3]	Zone de données de l'automate esclave	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 : • 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. 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 *MbusCmd[3], p. 22*).

Contenu	Description	Définition
MbusCmd[4]	Quantité	Ce registre contient la quantité de données à écrire ou à lire dans l'automate esclave. 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. 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 *MbusCmd[4], p. 23*). Cette taille est non significative pour les codes fonctions 5 et 6. **Note** : la zone mémoire est limitée en fonction de l'équipement et du paramétrage de l'esclave

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)
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 `%KWr.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

1.3 Mise en oeuvre du bloc de communication OUT_IN_MBUS

Présentation

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 :

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 :

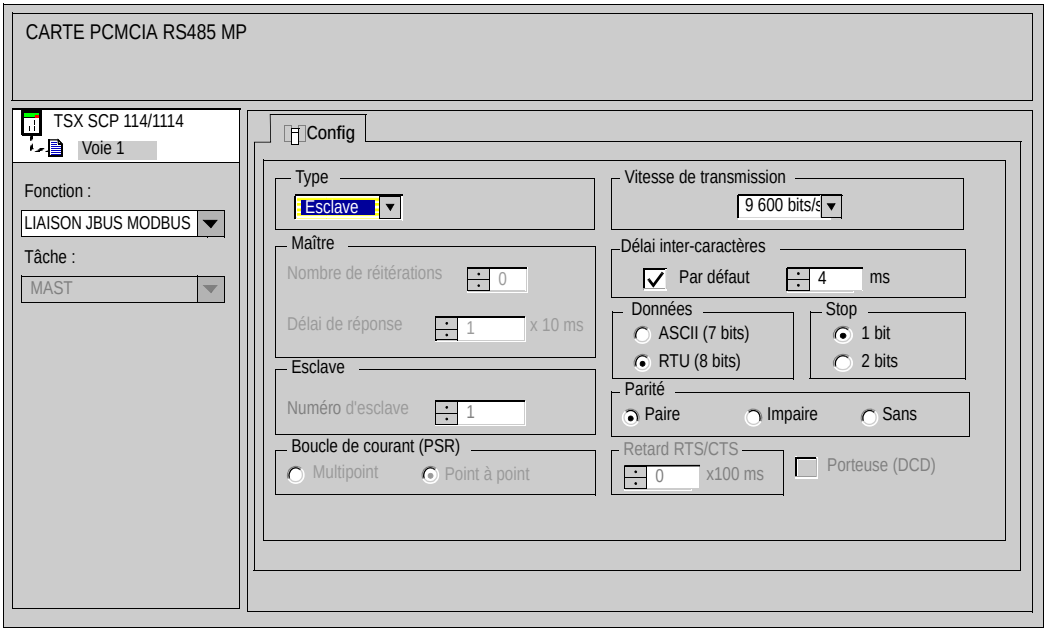
- la vitesse de transmission,
- le délai inter-caractères,
- le bit de données,
- le bit de stop,
- la parité.

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 :

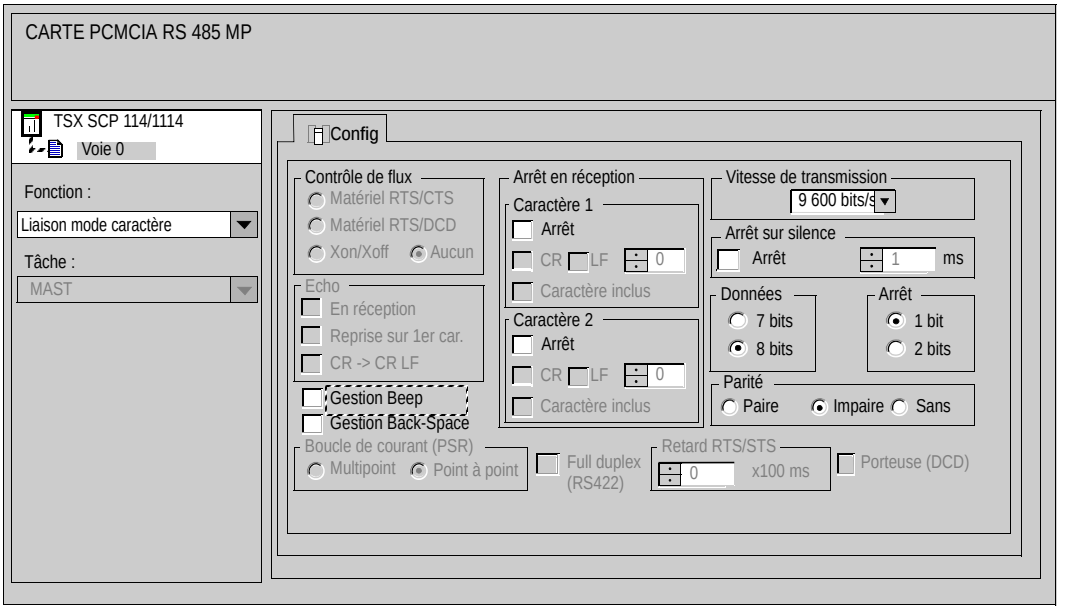


Note : la valeur du paramètre d'entrée `DataBits` (Voir *Le paramètre* `DataBits`, p. 25) du bloc `OUT_IN_MBUS` doit être conforme à la valeur cochée dans la fenêtre 'Données' de l'écran de configuration.

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), et 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 `DataBits` (Voir *Le paramètre* `DataBits`, p. 25) 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 réitérations (Voir *Le paramètre* `RetryLmt`, p. 24),
- le délai de réponse (Voir *Le paramètre* `RespTout`, p. 26).

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.	• 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 *Ecrire les mots de commande sur un port de communication, p. 35*). • 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 *Utilisation d'un modem, p. 36*). • 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 *Utilisation d'un modem, p. 36*).
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.	• 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.	• Dans le cas d'une transmission modem, envoyez la commande HAYES pour envoyer un message de déconnexion au modem (Voir *Utilisation d'un modem, p. 36*) 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 *Ecrire les mots de commande sur un port de communication, p. 35*).

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. • Avant d'exécuter un WRITE_CMD, testez si un échange est en cours en utilisant l'objet langage %MWr.m.c.0. Pour rafraîchir ce mot, il convient d'utiliser le bloc READ_STS.
2	Affectez le mot de commande. • 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 %MWr.m.c.15. Par exemple, pour commuter du mode Modbus au mode caractères, %MWr.m.c.15 est mis à 16#4000 (%MWr.m.c.15.14=1). **Note** : un seul bit de commande doit être commuté de 0 à 1 avant de transmettre le WRITE_CMD.
3	Envoyez la commande. • Finalement, un WRITE_CMD doit être exécuté pour acquitter la commande.

Dans l'exemple (Voir *Exemple d'utilisation du bloc de communication* OUT_IN_MBUS, p. 38) 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 interfacer des modems téléphoniques à des automates. Ces commandes sont :

- initialiser le modem,
- numéroter,
- déconnecter le modem.

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

Lorsque la connexion entre les deux modems a réussi, 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 Hyperterminal) 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`.

Numéroter 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 : numéroter par fréquence :

- `AT DT 6800326 <CR><LF>`

Exemple 2 : numéroter par impulsion :

- `AT DP 6800326 <CR><LF>`

Exemple 3 : numéroter 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 *Le paramètre* `Status`, p. 28) 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 *Le paramètre* `Status`, p. 28) 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.

1.4 Exemple d'utilisation du bloc de communication OUT_IN_MBUS

Présentation

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

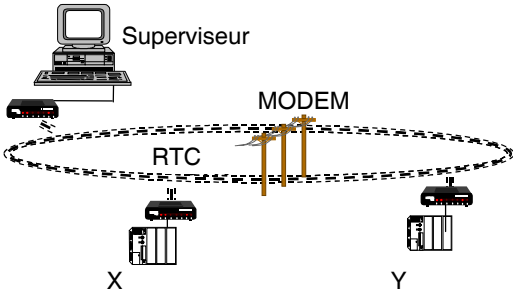
Ce sous-chapitre contient les sujets suivants :

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 `Start_4` 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 statut du port série.
	● S'il y a une erreur sur le port série, alors :
	  ● `Error_4` est à -2,
	  ● passage à l'étape 65.
	● S'il n'y a aucune erreur sur le port série :
	  ● et le mode caractère est actif, alors passage à l'étape 15,
	  ● et le mode caractère n'est pas 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 `Error_4` est à -1, et passage à l'étape 65.

15	Envoi d'une commande de numérotation au modem par le bloc `PRINT_CHAR` et attente de la réponse via `INPUT_CHAR`. Passage à l'étape 20.
20	Si le résultat de `PRINT_CHAR` est bien concluant, alors passage à l'étape 25, sinon passage à l'étape 65 avec `Error_4` à -3.
25	Si le résultat de `INPUT_CHAR` est bien concluant, alors passage à l'étape 30, sinon passage à l'étape 65 avec `Error_4` à -4.
30	Si le modem répond, alors passage à l'étape 35, sinon passage à l'étape 65 avec `Error_4` à -5.
35	Initialisation du paramétrage du bloc `OUT_IN_MBUS`. Passage en étape 40.

40	● Appel du bloc `OUT_IN_MBUS`.
	● Si le bit `Active_4` est à 0,
	  ● et si le bit `Flag_Error_4` est à 0, alors passage à l'étape 45,
	  ● et si le bit `Flag_Error_4` est à 1, alors passage à l'étape 45 avec `Error_4` à -6.

45	Envoi d'une commande de déconnexion au modem par le bloc `PRINT_CHAR`. Passage à l'étape 50.
50	Si le résultat de `PRINT_CHAR` est concluant, alors passage à l'étape 55, sinon passage à l'étape 65 avec `Error_4` à la valeur 1.

Numéro d'étape	Description de l'étape
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 statut du port série. ● S'il y a une erreur sur le port série, alors : ● `Error_4` est à 3, ● passage à l'étape 65. ● S'il n'y a aucune erreur sur le port série : ● et il y a passage en mode Modbus, alors passage à l'étape 65, ● et il n'y a pas passage 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 `Error_4` est à 2, et passage à l'étape 65.
65	Retour à l'étape 0 (l'état initial de la fonction) et bit `Start_4` 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;
```

Index