Altivar ATV IMC Drive Controller

Fonctions et variables système
Guide de la bibliothèque ATV-IMC PLCSystem

12/2015

EIO0000000597.05

www.schneider-electric.com

Schneider Electric

Le présent document comprend des descriptions générales et/ou des caractéristiques techniques des produits mentionnés. Il ne peut pas être utilisé pour définir ou déterminer l'adéquation ou la fiabilité de ces produits pour des applications utilisateur spécifiques. Il incombe à chaque utilisateur ou intégrateur de réaliser l'analyse de risques complète et appropriée, l'évaluation et le test des produits pour ce qui est de l'application à utiliser et de l'exécution de cette application. Ni la société Schneider Electric ni aucune de ses sociétés affiliées ou filiales ne peuvent être tenues pour responsables 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.

Toutes les réglementations de sécurité pertinentes locales doivent être observées 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 techniques de sécurité, suivez les instructions appropriées.

La non-utilisation du logiciel Schneider Electric ou d'un logiciel approuvé avec nos produits matériels 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.

Table des matières

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, de le réparer 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.

La présence de ce symbole sur une étiquette “Danger“ ou “Avertissement“ signale un risque d'électrocution qui provoquera des blessures physiques en cas de non-respect des consignes de sécurité.

Ce symbole est le symbole d'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 signale un risque qui, en cas de non-respect des consignes de sécurité, **provoque** la mort ou des blessures graves.

⚠ AVERTISSEMENT

AVERTISSEMENT signale un risque qui, en cas de non-respect des consignes de sécurité, **peut provoquer** la mort ou des blessures graves.

⚠ ATTENTION

ATTENTION signale un risque qui, en cas de non-respect des consignes de sécurité, **peut provoquer** des blessures légères ou moyennement graves.

AVIS

AVIS indique des pratiques n'entraînant pas de risques corporels.

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 ce matériel.

Une personne qualifiée est une personne disposant de compétences et de connaissances dans le domaine de la construction, du fonctionnement et de l'installation des équipements électriques, et ayant suivi une formation en sécurité leur permettant d'identifier et d'éviter les risques encourus.

A propos de ce manuel

Présentation

Objectif du document

Ce document est destiné à vous familiariser aux fonctions et variables que propose le contrôleur Altivar ATV IMC Drive Controller. La bibliothèque PLCSystem ATV IMC contient des fonctions et des variables permettant de communiquer avec le système Altivar ATV IMC Drive Controller (réception d'informations et envoi de commandes).

Ce document décrit les types de données, fonctions et variables de la bibliothèque PLCSystem du contrôleur ATV IMC.

Les connaissances de base requises sont les suivantes :

- connaissances de base sur les fonctionnalités, la structure et la configuration du contrôleur ATV IMC
- programmation en langages FBD, LD, ST, IL, SFC ou CFC,
- variables système (variables globales).

Champ d'application

Ce document a été actualisé pour la version de SoMachine V4.1 SP2.

Information spécifique au produit

⚠ AVERTISSEMENT

PERTE DE CONTROLE

- Le concepteur d'un circuit de commande doit tenir compte des modes de défaillance potentiels des canaux de commande et, pour certaines fonctions de commande critiques, prévoir un moyen d'assurer la sécurité en maintenant un état sûr pendant et après la défaillance. Par exemple, l'arrêt d'urgence, l'arrêt en cas de surcourse, la coupure de courant et le redémarrage sont des fonctions de commande cruciales.
- Des canaux de commande séparés ou redondants doivent être prévus pour les fonctions de commande critiques.
- Les liaisons de communication peuvent faire partie des canaux de commande du système. Une attention particulière doit être prêtée aux implications des délais de transmission non prévus ou des pannes de la liaison.
- Respectez toutes les réglementations de prévention des accidents ainsi que les consignes de sécurité locales.[1]
- Chaque implémentation de cet équipement doit être testée individuellement et entièrement pour s'assurer du fonctionnement correct avant la mise en service.

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

[1] Pour plus d'informations, consultez le document NEMA ICS 1.1 (dernière édition), « Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control » (Directives de sécurité pour l'application, l'installation et la maintenance de commande statique) et le document NEMA ICS 7.1 (dernière édition), « Safety Standards for Construction and Guide for Selection, Installation, and Operation of Adjustable-Speed Drive Systems » (Normes de sécurité relatives à la construction et manuel de sélection, installation et opération de variateurs de vitesse) ou son équivalent en vigueur dans votre pays.

⚠ AVERTISSEMENT

FONCTIONNEMENT INATTENDU DE L'EQUIPEMENT

- N'utilisez que le logiciel approuvé par Schneider Electric pour faire fonctionner cet équipement.
- Mettez à jour votre programme d'application chaque fois que vous modifiez la configuration matérielle physique.

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

Terminologie utilisée dans les normes

Les termes techniques, la terminologie, les symboles et les descriptions correspondantes employés dans ce manuel ou figurant dans ou sur les produits proviennent généralement des normes internationales.

Dans les domaines des systèmes de sécurité fonctionnelle, des variateurs et de l'automatisme en général, les termes employés sont *sécurité*, *fonction de sécurité*, *état sécurisé*, *défaut*, *réinitialisation du défaut*, *dysfonctionnement*, *panne*, *erreur*, *message d'erreur*, *dangereux*, etc.

Entre autres, les normes concernées sont les suivantes :

Norme	Description
EN 61131-2:2007	Automates programmables - Partie 2 : exigences et essais des équipements
ISO 13849-1:2008	Sécurité des machines - Parties des systèmes de commande relatives à la sécurité - Principes généraux de conception
EN 61496-1:2013	Sécurité des machines - Équipements de protection électro-sensibles - Partie 1 : prescriptions générales et essais
ISO 12100:2010	Sécurité des machines - Principes généraux de conception - Appréciation du risque et réduction du risque
EN 60204-1:2006	Sécurité des machines - Équipement électrique des machines - Partie 1 : règles générales
EN 1088:2008 ISO 14119:2013	Sécurité des machines - Dispositifs de verrouillage associés à des protecteurs - Principes de conception et de choix
ISO 13850:2006	Sécurité des machines - Fonction d'arrêt d'urgence - Principes de conception
EN/IEC 62061:2005	Sécurité des machines - Sécurité fonctionnelle des systèmes de commande électrique, électronique et électronique programmable relatifs à la sécurité
IEC 61508-1:2010	Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité - Exigences générales
IEC 61508-2:2010	Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité - Exigences pour les systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité
IEC 61508-3:2010	Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité - Exigences concernant les logiciels
IEC 61784-3:2008	Communications numériques pour les systèmes de mesure et de commande - Bus de terrain de sécurité fonctionnelle
2006/42/EC	Directive Machines
2004/108/EC	Directive sur la compatibilité électromagnétique
2006/95/EC	Directive sur les basses tensions

De plus, des termes peuvent être utilisés dans le présent document car ils proviennent d'autres normes telles que :

Norme	Description
Série IEC 60034	Machines électriques rotatives
Série IEC 61800	Entraînements électriques de puissance à vitesse variable
Série IEC 61158	Communications numériques pour les systèmes de mesure et de commande - Bus de terrain utilisés dans les systèmes de commande industriels

Enfin, le terme *zone de fonctionnement* utilisable pour décrire des dangers spécifiques correspond aux termes *zone dangereuse* ou *zone de danger* employés dans la *directive européenne Machines* (*EC/2006/42*) et la norme *ISO 12100:2010*.

NOTE : Les normes susmentionnées peuvent s'appliquer ou pas aux produits cités dans la présente documentation. Pour plus d'informations sur chacune des normes applicables aux produits décrits dans le présent document, consultez les tableaux de caractéristiques de ces références de produit.

Chapitre 1
Variables système du contrôleur ATV IMC

Présentation

Ce chapitre :

- fournit une introduction aux variables système *(voir page 12)* ;
- décrit les variables système *(voir page 18)* disponibles avec la bibliothèque PLCSystem de ATV IMC.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre 1.1
Variables système : définition et utilisation

Présentation

Cette section définit les variables système et explique leur mise en œuvre dans le Altivar ATV IMC Drive Controller.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Présentation des variables système

Introduction

Cette section décrit comment les variables système sont mises en œuvre. Les variables système :

- permettent d'accéder à des informations générales sur le système, de réaliser des diagnostics système et de commander des actions simples ;
- sont des variables structurées selon les définitions et conventions de désignation de la norme CEI 61131-3. Vous pouvez accéder aux variables système à l'aide du nom symbolique CEI `PLC_GVL`. Certaines variables `PLC_GVL` sont en lecture seule (par exemple, `PLC_R`) et d'autres sont en lecture-écriture (par exemple, `PLC_W`).
- sont déclarées automatiquement comme des variables globales. Elles s'appliquent à l'ensemble du système et toute POU (unité organisationnelle de programme) d'une tâche peut y accéder.

Convention de désignation

Les variables système sont identifiées par :

- un nom de structure qui représente la catégorie de variables système. Par exemple, `PLC_R` représente un nom de structure de variables en lecture seule utilisées pour le diagnostic du contrôleur.
- un ensemble de noms de composant qui identifie le rôle de la variable. Par exemple, `i_wVendorID` représente l'ID du fournisseur du contrôleur.

Vous pouvez accéder aux variables système en entrant leur nom de structure suivi du nom du composant.

Voici un exemple de mise en œuvre de variables système :

```
VAR
    myCtr_Serial : DWORD;
    myCtr_ID : DWORD;
    myCtr_FramesRx : UDINT;
END_VAR

myCtr_Serial := PLC_R.i_dwSerialNumber;
myCtr_ID := PLC_R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK;
```

NOTE : Dans l'exemple ci-dessus, le nom complet de la variable système est `PLC_GVL.PLC_R.i_wVendorID`. Le `PLC_GVL` est implicite lors de la déclaration d'une variable à l'aide de l'**Aide à la saisie**, mais vous pouvez aussi l'entrer en intégralité. Les bonnes pratiques de programmation préconisent souvent d'utiliser le nom complet de la variable dans les déclarations.

Emplacement des variables système

Deux sortes de variables système sont définies pour la programmation du contrôleur :

- variables localisées
- variables non localisées

Les variables localisées :

- ont un emplacement fixe dans une zone %MW statique :
  - %MW60000 à %MW60199 pour les variables système en lecture seule,
  - %MW62000 à %MW62199 pour les variables système en lecture/écriture ;
- sont utilisées dans des programmes SoMachine conformément à la convention `structure_name.component_name` expliquée précédemment. Les adresses %MW de 0 to 59999 sont accessibles directement. Les adresses supérieures sont considérées hors plage par SoMachine et sont uniquement accessibles via la convention `structure_name.component_name`.

Les variables non affectées :

- ne se trouvent pas physiquement dans la zone %MW ;.
- sont utilisées dans des programmes SoMachine conformément à la convention `structure_name.component_name` expliquée précédemment. Les adresses %MW de 0 à 59999 sont accessibles directement. Les adresses supérieures sont considérées hors plage par SoMachine et sont uniquement accessibles via la convention `structure_name.component_name`.

Utilisation des variables système

Introduction

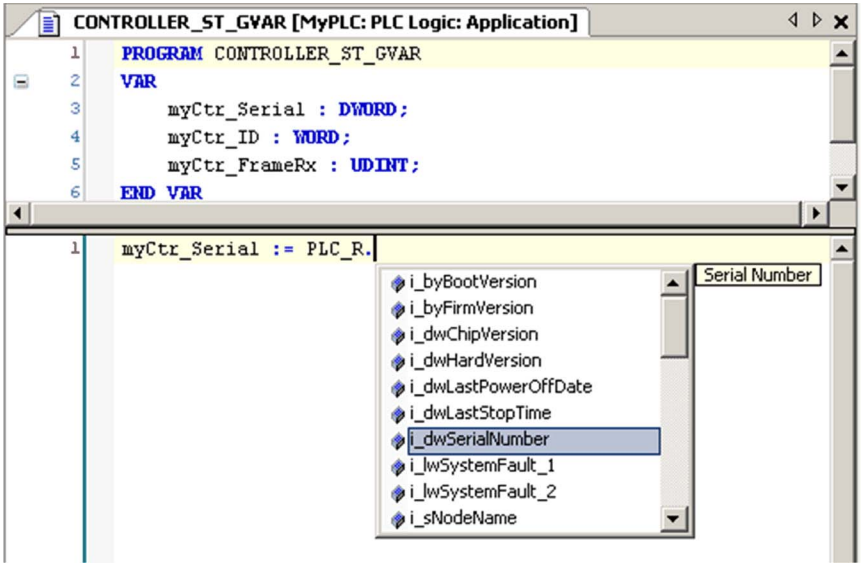
Cette rubrique décrit la procédure de programmation et d'utilisation des variables système dans SoMachine.

Les variables système ont un champ d'application global et vous pouvez les utiliser dans tous les POU (unité organisationnelle de programme) de l'application.

Il n'est pas nécessaire de déclarer les variables système dans la liste des variables globales (GVL). Elles sont déclarées automatiquement à partir de la bibliothèque système du contrôleur.

Utilisation des variables système dans un POU

SoMachine a une fonction de saisie automatique. Dans un **POU**, commencez par entrer le nom de structure de la variable système (`PLC_R`, `PLC_W`, ...) suivi d'un point. Les variables système s'affichent dans l'**Aide à la saisie**. Vous pouvez sélectionner la variable de votre choix ou entrer manuellement son nom en intégralité.



NOTE : Dans l'exemple ci-dessus, une fois que le nom de structure `PLC_R.` a été entré, SoMachine affiche un menu contextuel des noms de composants/variables possibles.

Exemple

L’exemple ci-dessous décrit l'utilisation de certaines variables système :

```
VAR myCtr_Serial : DWORD; myCtr_ID : WORD; myCtr_FramesRx : UDINT;
END_VAR
```

```
myCtr_Serial := PLC_R.i_dwSerialNumber; myCtr_ID := PLC_R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK;
```

Sous-chapitre 1.2
Structures PLC_R et PLC_W

Présentation

Cette section répertorie et décrit les variables système incluses dans les structures `PLC_R` et `PLC_W`.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

PLC_RPLC_R : Variables système en lecture seule du contrôleur

Structure de la variable

Ce tableau décrit les paramètres de la variable système PLC_R (type PLC_R_STRUCT) :

Adresse Modbus (1)	Nom de la variable	Type	Commentaire
60000	i_wVendorID	WORD	ID du fournisseur du contrôleur. 101A hex = Schneider Electric
60001	i_wProductID	WORD	ID de référence du contrôleur. **NOTE :** L'ID du fournisseur et l'ID de référence constituent l'ID cible du contrôleur, indiqué dans l'écran des paramètres de communication (ID cible = 101A XXXX hex).
60002	i_dwSerialNumber	DWORD	Numéro de série du contrôleur.
60004	i_byFirmVersion[0..3]	ARRAY[0..3] OF BYTE	Version du micrologiciel du contrôleur [aa.bb.cc.dd] : ● i_byFirmVersion[0] = aa ● ... ● i_byFirmVersion[3] = dd
60006	i_byBootVersion[0..3]	ARRAY[0..3] OF BYTE	Version du démarrage du contrôleur [aa.bb.cc.dd] : ● i_byBootVersion[0] = aa ● ... ● i_byBootVersion[3] = dd
60008	i_dwHardVersion	DWORD	Version du matériel du contrôleur.
60010	i_dwHardwareID	DWORD	Version du coprocesseur du contrôleur.
60012	i_wStatus	PLC_R_STATUS *(voir page 48)*	Etat du contrôleur.
60013	i_wBootProjectStatus	PLC_R_BOOT_ PROJECT_STATUS *(voir page 46)*	Renvoie des informations sur l'application de démarrage stockée dans la mémoire Flash.
60014	i_wLastStopCause	PLC_R_STOP_ CAUSE *(voir page 49)*	Cause du dernier passage du mode d'exécution (RUN) à un autre état.
60015	i_wLastApplicationError	PLC_R_ APPLICATION_ERROR *(voir page 43)*	Cause de la dernière exception du contrôleur.

Adresse Modbus (1)	Nom de la variable	Type	Commentaire
60016	i_lwSystemFault_1	LWORD	Le champ de bits FFFF FFFF FFFF FFFF hex indique qu'aucune erreur n'a été détectée. Un bit de niveau bas signifie qu'une erreur a été détectée : ● bit 0 = erreur détectée sur liaison interne ATV-IMC ● bit 1 = liaison Ethernet non connectée ● bit 2 = liaison USB non connectée ● bit 3 = liaison CANopen inopérante ● bit 4 = délai Modbus/TCP expiré ● bit 5 = adresse IP en double détectée ● bit 6 = surcharge détectée sur le réseau Ethernet ● bit 7 = erreur détectée sur le matériel Ethernet ● bit 8 = erreur détectée sur la mémoire non volatile ● bit 9 = erreur de messagerie détectée dans la communication CAN ● bit 10 = erreur détectée dans un dictionnaire d'objets ATV-IMC ● bit 11 = erreur d'horloge de surveillance système détectée ● bit 12 = erreur interne détectée ● bit 13 = erreur de sortie logique détectée (surchauffe) ● bit 14 = alimentation 24 V de sortie logique non opérationnelle ● bits 15 à 63 : inutilisés **NOTE :** Le bit 11 et le bit 12 peuvent être réinitialisés avec la fonction ResetInternalErrorDiag *(voir page 37)*.
60020	i_lwSystemFault_2	LWORD	Non utilisé.
60024	i_wIOStatus1	PLC_R_IO_STATUS *(voir page 47)*	Etat des E/S intégrées.
60025	i_wIOStatus2	PLC_R_IO_STATUS *(voir page 47)*	inutilisé (toujours FFFF hex).
60026	i_wBatteryStatus	PLC_R_BATTERY_ STATUS *(voir page 45)*	Etat de la batterie de l'horodateur.
60028	i_dwAppliSignature1	DWORD	Premier des 4 DWORD de la signature (16 octets au total). La signature de l'application est générée par le logiciel pendant la construction.

Adresse Modbus (1)	Nom de la variable	Type	Commentaire
60030	i_dwAppliSignature2	DWORD	Deuxième des 4 DWORD de la signature (16 octets au total). La signature de l'application est générée par le logiciel pendant la construction.
60032	i_dwAppliSignature3	DWORD	Troisième des 4 DWORD de la signature (16 octets au total). La signature de l'application est générée par le logiciel pendant la construction.
60034	i_dwAppliSignature4	DWORD	Quatrième des 4 DWORD de la signature (16 octets au total). La signature de l'application est générée par le logiciel pendant la construction.
(1) Inaccessible via l'application.			

s/o	i_sVendorName	STRING(31)	Nom du fournisseur : « Schneider Electric ».
s/o	i_sProductRef	STRING(31)	Référence du contrôleur.

NOTE : s/o signifie qu'aucun mappage d'adresse Modbus n'est prédéfini pour cette variable système.

PLC_W : variable système en lecture/écriture de contrôleur

Structure de la variable

Ce tableau décrit les paramètres de la variable système PLC_W (type PLC_W_STRUCT) :

%MW	Nom de la variable	Type	Commentaire
62000	q_uiOpenPLCControl	UINT	Lorsque la valeur passe de 0 à 6699, la commande inscrite précédemment dans la variable PLC_W.q_wPLCControl suivante est exécutée.
62001	q_wPLCControl	PLC_W_COMMAND *(voir page 50)*	La commande RUN/STOP du contrôleur est exécutée lorsque la valeur de la variable système PLC_R.q_uiOpenPLCControl passe de 0 à 6699.

Sous-chapitre 1.3
Structures ETH_R et ETH_W

Présentation

Cette section répertorie et décrit les variables système incluses dans les structures `ETH_R` et `ETH_W`.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

ETH_R : variables système en lecture seule du port Ethernet

Structure de la variable

Le tableau suivant décrit les paramètres de la variable système ETH_R (type ETH_R_STRUCT) :

%MW	Nom de la variable	Type	Commentaire
60050	i_byIPAddress[0..3]	ARRAY[0..3] OF BYTE	Adresse IP [aaa.bbb.ccc.ddd] : ● i_byIPAddress[0] = aaa ● ... ● i_byIPAddress[3] = ddd
60052	i_bySubNetMask[0..3]	ARRAY[0..3] OF BYTE	Masque de sous-réseau [aaa.bbb.ccc.ddd] : ● i_bySub-netMask[0] = aaa ● ... ● i_bySub-netMask[3] = ddd
60054	i_byGateway[0..3]	ARRAY[0..3] OF BYTE	Adresse de la passerelle [aaa.bbb.ccc.ddd] : ● i_byGateway[0] = aaa ● ... ● i_byGateway[3] = ddd
60056	i_byMACAddress[0..5]	ARRAY[0..5] OF BYTE	Adresse MAC [aa.bb.cc.dd.ee.ff] : ● i_byMACAddress[0] = aa ● ... ● i_byMACAddress[5] = ff
60059	i_sDeviceName	STRING(16)	Nom utilisé pour obtenir l'adresse IP auprès du serveur.
60067	i_wIpMode	ETH_R_IP_ MODE *(voir page 52)*	Méthode utilisée pour obtenir une adresse IP.
60068	i_byFDRServerIPAddress[0..3]	ARRAY[0..3] OF BYTE	Adresse IP [aaa.bbb.ccc.ddd] du serveur DHCP ou BootP : ● i_byFDRServerIPAddress[0] = aaa ● ... ● i_byFDRServerIPAddress[3] = ddd Egale à 0.0.0.0 en cas d'utilisation d'une adresse IP enregistrée ou par défaut.
60070	i_udiOpenTcpConnections	UDINT	Nombre de connexions TCP ouvertes.
60073	i_udiFramesTransmittedOK	UDINT	Nombre de trames transmises correctement. Réinitialisation lors de la mise sous tension ou avec la commande de réinitialisation ETH_W.q_wResetCounter.
s/o signifie qu'aucun mappage %MW n'est prédéfini pour cette variable système.			

%MW	Nom de la variable	Type	Commentaire
60075	i_udiFramedReceivedOK	UDINT	Nombre de trames reçues correctement. Réinitialisation lors de la mise sous tension ou avec la commande de réinitialisation ETH_W.q_wResetCounter.
60077	i_udiTransmitBufferErrors	UDINT	Nombre de trames transmises avec détection d'erreurs. Réinitialisation lors de la mise sous tension ou avec la commande de réinitialisation ETH_W.q_wResetCounter.
60079	i_udiReceiveBufferErrors	UDINT	Nombre de trames reçues avec détection d'erreurs. Réinitialisation lors de la mise sous tension ou avec la commande de réinitialisation ETH_W.q_wResetCounter.
60081	i_wPortALinkStatus	ETH_R_PORT_LINK_STATUS *(voir page 55)*	Liaison du port Ethernet (0 = aucune liaison, 1 = liaison connectée à un autre équipement Ethernet).
60082	i_wPortASpeed	ETH_R_PORT_SPEED *(voir page 56)*	Débit réseau du port Ethernet (10 ou 100 Mbits/s).
60083	i_wPortADuplexStatus	ETH_R_PORT_DUPLEX_STATUS *(voir page 54)*	Etat duplex du port Ethernet (0 = semi duplex ou 1 = duplex intégral).
s/o signifie qu'aucun mappage %MW n'est prédéfini pour cette variable système.			

ETH_W : variables système en lecture/écriture du port Ethernet

Structure de la variable

Le tableau suivant décrit les paramètres de la variable système ETH_W (type ETH_W_STRUCT) :

%MW	Nom de la variable	Type	Commentaire
62066	q_wResetCounter	WORD	Le passage de 0 à 1 réinitialise tous les compteurs ETH_R. Pour réinitialiser à nouveau les compteurs, il est nécessaire de mettre ce registre à 0 de sorte qu'un nouveau passage de 0 à 1 puisse intervenir.

Chapitre 2
Fonctions système de ATV IMC

Présentation

Ce chapitre décrit les fonctions système disponibles dans la bibliothèque PLCSystem de ATV IMC.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre 2.1
Fonctions de lecture de ATV IMC

Présentation

Cette section décrit les fonctions de lecture de la bibliothèque PLCSystem de ATV IMC.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

GetEventsNumber : renvoie le nombre d'événements externes détectés

Description de la fonction

Cette fonction renvoie le nombre d'événements qui se sont produits depuis le dernier démarrage à froid, notamment ceux détectés sur les entrées et les événements de comparaison de seuil HSC.

Représentation graphique

Représentation en IL et en ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Le tableau suivant décrit la variable de sortie :

Sortie	Type	Commentaire
`GetEventsNumber`	`UINT`	Valeur représentant le nombre d'événements qui se sont produits depuis le dernier démarrage à froid. La remise à 0 est effectuée avec un appel de la fonction ResetEventsNumber *(voir page 36)*.

GetLastStopTime : renvoie la date et l'heure du dernier arrêt détecté.

Description de la fonction

Cette fonction renvoie la date et l'heure du dernier passage du mode d'exécution (RUN) à un autre état.

Représentation graphique

Représentation en IL et en ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Le tableau suivant décrit la variable de sortie :

Sortie	Type	Commentaire
`GetLastStopTime`	`DWORD`	Heure du dernier arrêt (STOP) détecté en secondes, en commençant au 1er janvier 1970 à 00:00:00.

IsFirstMastColdCycle : indique si le cycle est le premier cycle MAST après un démarrage à froid

Description de la fonction

Cette fonction renvoie TRUE au cours du premier cycle MAST après un démarrage à froid (premier cycle après téléchargement ou réinitialisation à froid).

Représentation graphique

Représentation en IL et en ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Ce tableau décrit la variable de sortie :

Sortie	Type	Commentaire
IsFirstMastColdCycle	BOOL	TRUE au cours du premier cycle de la tâche MAST après un démarrage à froid.

Exemple

Reportez-vous à la description de la fonction IsFirstMastCycle *(voir page 32)*.

IsFirstMastCycle : indique si le cycle est le premier cycle MAST

Description de la fonction

Cette fonction renvoie TRUE lors du premier cycle MAST après un démarrage.

Représentation graphique

Représentation en IL et en ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Sortie	Type	Commentaire
IsFirstMastCycle	BOOL	TRUE lors du premier cycle de la tâche MAST après un démarrage.

Exemple

Cet exemple décrit les trois fonctions IsFirstMastCycle, IsFirstMastColdCycle et IsFirstMastWarmCycle utilisées ensemble.

Utilisez cet exemple dans la tâche MAST. Sinon, il peut s'exécuter plusieurs fois ou jamais (une tâche supplémentaire peut être appelée plusieurs fois ou éventuellement aucune fois pendant un cycle de tâche MAST) :

```
VAR MyIsFirstMastCycle : BOOL; MyIsFirstMastWarmCycle : BOOL; MyIsFirst-
MastColdCycle : BOOL; END_VAR

MyIsFirstMastWarmCycle := IsFirstMastWarmCycle(); MyIsFirstMastColdCycle
:= IsFirstMastColdCycle(); MyIsFirstMastCycle := IsFirstMastCycle();

IF (MyIsFirstMastWarmCycle) THEN

(*Il s'agit du premier cycle MAST après un démarrage à chaud : toutes les
variables prennent sur leurs valeurs d'initialisation, à l'exception des
variables conservées.*)

(*=> initialisez les variables nécessaires pour que votre application
s'exécute comme prévu dans ce cas*)

END_IF;
```

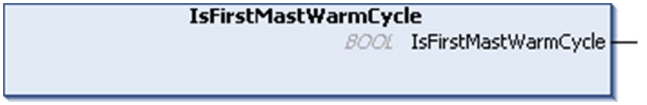
```
IF (MyIsFirstMastColdCycle) THEN
(*Il s'agit du premier cycle MAST après un démarrage à froid : toutes les
variables prennent sur leurs valeurs d'initialisation, y compris les
variables conservées.*)
(*=> initialisez les variables nécessaires pour que votre application
s'exécute comme prévu dans ce cas*)
END_IF;
IF (MyIsFirstMastCycle) THEN
(*Il s'agit du premier cycle MAST après un démarrage, c'est-à-dire après
un démarrage à chaud ou à froid ou l'exécution de commandes STOP/RUN*)
(*=> initialisez les variables nécessaires pour que votre application
s'exécute comme prévu dans ce cas*)
END_IF;
```

IsFirstMastWarmCycle : indique si le cycle est le premier cycle MAST après un démarrage à chaud

Description de la fonction

Cette fonction renvoie TRUE lors du premier cycle MAST après un démarrage à chaud.

Représentation graphique



Représentation en IL et en ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Ce tableau décrit la variable de sortie :

Sortie	Type	Commentaire
IsFirstMastWarmCycle	BOOL	TRUE au cours du premier cycle de la tâche MAST après un démarrage à chaud.

Exemple

Reportez-vous à la description de la fonction IsFirstMastCycle *(voir page 32)*.

Sous-chapitre 2.2
Fonctions d'écriture de l'automate ATV IMC

Vue d'ensemble

Cette section décrit les fonctions d'écriture de la bibliothèque PLCSystem de l'automate ATV IMC.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

ResetEventsNumber : remise à zéro du nombre d'événements

Description de la fonction

Cette fonction remet à 0 le nombre d'événements qui se sont produits depuis le dernier démarrage à froid, dont les événements détectés au niveau des entrées et les événements de comparaison de seuil de comptage rapide.

NOTE : Le nombre d'événements détectés au niveau des entrées ou des seuils de comptage rapide est envoyé par la fonction `GetEventsNumber` *(voir page 29)*.

Représentation graphique

Représentation en IL et en ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Le tableau suivant décrit les variables de sortie :

Sortie	Type	Commentaire
`ResetEventsNumber`	`BOOL`	Prend la valeur TRUE si le compteur a été remis à 0 correctement.

ResetInternalErrorDiag : remise à zéro des indicateurs générés en cas de détection d'une erreur d'horloge de surveillance du système ou d'une erreur interne

Description de la fonction

Cette fonction remet à zéro des indicateurs générés en cas de détection d'une erreur d'horloge de surveillance du système ou d'une erreur interne.

NOTE : Une erreur d'horloge de surveillance du système et une erreur interne sont consignées respectivement par bit 11 et bit 12 du registre `i_lwSystemFault_1` dans la variable système `PLC_R` *(voir page 18)*.

Représentation graphique

Représentation en langage IL et ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction* *(voir page 67)*.

Description des variables d'E/S

Le tableau suivant décrit la variable d'entrée :

Entrée	Type	Commentaire
`erreur`	`PLC_ERROR_TYPE` *(voir page 63)*	Erreur détectée lors de la remise à 0.

Exemple

```
VAR
     hw_watchdog_error : PLC_ERROR_TYPE := _ERR_IMC_WATCHDOG
     internal_error : PLC_ERROR_TYPE := _ERR_IMC_INT_ERROR
END_VAR

ResetInternalErrorDiag(hw_watchdog_error);
ResetInternalErrorDiag(internal_error);
```

SetLEDBehaviour : détermine le fonctionnement d'un voyant

Description de la fonction

Cette fonction contrôle le voyant USER de l'application.

Représentation graphique

Représentation en langage IL et ST

Pour voir la représentation générale en langage IL ou ST, reportez-vous au chapitre *Représentation des fonctions et blocs fonction (voir page 67)*.

Description des variables d'E/S

Le tableau suivant décrit les paramètres d'entrée :

Entrées	Type	Commentaire
LedId	LED_ID *(voir page 62)*	ID du voyant de l'application.
LedColor	LED_COLOR *(voir page 61)*	Couleur du voyant de l'application.
LedBhv	LED_BHV *(voir page 58)*	Mode du voyant de l'application.

Le tableau suivant décrit la variable de sortie :

Sortie	Type	Commentaire
SetLEDBehaviour	LED_BHV_ERROR *(voir page 60)*	Renvoie NO_ERROR (00 hex) si la commande est correcte, ou bien le code d'identification de l'erreur détectée.

Exemple

Cet exemple indique comment procéder pour que le voyant USER soit allumé en vert :

```
VAR myLEDStatus : LED_BHV_ERROR; myLED : LED_ID := LED_0; myLEDColor :
LED_COLOR := LED_GREEN; myLEDMode : LED_BHV := LED_ON; END_VAR
myLEDStatus := SetLedBehaviour(myLED, myLEDColor, myLEDMode);
```

NOTE : Vous pouvez contrôler séparément et mélanger les couleurs des voyants, par conséquent désactivez le code de couleur en cours avant d'allumer le nouveau. Le tableau ci-dessous présente un exemple de séquence de commandes `SetLedBehaviour` et le fonctionnement correspondant des voyants :

étape	LedId	LedColor	LedBhv	Mode VERT clignotant	Mode ROUGE clignotant
1	LED_0	-	-	Eteint	Eteint
2	LED_0	LED_GREEN	LED_ON	Allumé	Eteint
3	LED_0	LED_GREEN	LED_OFF	Eteint	Eteint
4	LED_0	LED_RED	LED_ON	Eteint	Allumé

Chapitre 3
Types de données de la bibliothèque PLCSystem ATV IMC

Présentation

Ce chapitre décrit les types de données de la bibliothèque PLCSystem de ATV IMC.

Deux types de données sont disponibles :

- Les types de données de variable système sont utilisés par les variables système *(voir page 11)* (PLC_R, PLC_W,...) de la bibliothèque PLCSystem de ATV IMC.
- Les types de données de fonction système sont utilisés par les fonctions système *(voir page 27)* de lecture/écriture de la bibliothèque PLCSystem de ATV IMC.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

Sous-chapitre 3.1
Types de données des variables système PLC_R/W

Présentation

Cette section répertorie et décrit les types de données de variable système, inclus dans les structures `PLC_R` et `PLC_W`.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

PLC_R_APPLICATION_ERRORPLC_R_APPLICATION_ERROR : Codes d'état des erreurs de l'application détectées

Description du type énumération

Le type de données énumération `PLC_R_APPLICATION_ERROR` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire	Que faire
`PLC_R_APP_ERR_UNKNOWN`	FFFF hex	Erreur indéfinie détectée.	Contactez le service client le plus proche.
`PLC_R_APP_ERR_NOEXCEPTION`	0000 hex	Aucune erreur détectée.	–
`PLC_R_APP_ERR_WATCHDOG`	0010 hex	Tâche chien de garde expirée.	Vérifiez votre application Voir le chapitre . Une réinitialisation est nécessaire pour entrer en mode Run.
`PLC_R_APP_ERR_HARDWAREWATCHDOG`	0011 hex	Chien de garde du système expiré.	Si le problème se reproduit, vérifiez la présence de ports de communication déconnectés. Si le problème persiste, mettez à jour le micrologiciel. Si le problème persiste encore, contactez votre service d'assistance local.
`PLC_R_APP_ERR_IO_CONFIG_ERROR`	0012 hex	Paramètres de configuration d'E/S incorrects détectés.	Il est possible que votre application soit endommagée. Pour résoudre ce problème, utilisez l'une de ces méthodes : **1. Run → Build → Clean All** **2.** Exportez/Importez votre application. **3.** Mettez à niveau SoMachine à la dernière version.
`PLC_R_APP_ERR_UNRESOLVED_EXTREFS`	0018 hex	Fonctions indéfinies détectées.	Supprimez les fonctions non résolues de l'application.

Enumérateur	Valeur	Commentaire	Que faire
PLC_R_APP_ERR_IEC_TASK_CONFIG_ERROR	0025 hex	Paramètres de configuration de tâche incorrects détectés.	Il est possible que votre application soit endommagée. Pour résoudre ce problème, utilisez l'une des méthodes suivantes : **1. Exécuter → Compiler → Tout effacer** **2.** Exportez/Importez votre application. **3.** Mettez à niveau SoMachine à la dernière version.
PLC_R_APP_ERR_ILLEGAL_INSTRUCTION	0050 hex	Instruction indéfinie détectée.	Procédez au débogage de votre application pour résoudre le problème.
PLC_R_APP_ERR_ACCESS_VIOLATION	0051 hex	Tentative d'accès à la zone mémoire réservée.	Procédez au débogage de votre application pour résoudre le problème.
PLC_R_APP_ERR_RTSEXCPT_DIVIDEBYZERO	0102 hex	Division d'un entier par 0 détectée.	Procédez au débogage de votre application pour résoudre le problème.
PLC_R_APP_ERR_PROCESSORLOAD_WATCHDOG	0105 hex	Processeur surchargé par les tâches de l'application.	Réduisez la charge de travail de l'application en améliorant son architecture. Augmentez la durée du cycle de tâche. Réduisez la fréquence des événements.
PLC_R_APP_ERR_RTSEXCPT_FPU_DIVIDEBYZERO	0152 hex	Division d'une décimale par 0 détectée.	Procédez au débogage de votre application pour résoudre le problème.

PLC_R_BATTERY_STATUS : codes d'état de la batterie

Description du type énuméré

Le type de données énuméré `PLC_R_BATTERY_STATUS` contient les valeurs suivantes :

Enumérateur	Valeur	Description
`BATTERY_OK`	0000 hex	La batterie est en état de marche et l'horodateur (RTC) est configuré.
`BATTERY_NOT_CONFIGURED`	0001 hex	La batterie est en état de marche, mais l'horodateur n'est pas configuré.
`BATTERY_UNPLUGGED`	0002 hex	La batterie est débranchée ou doit être remplacée.

PLC_R_BOOT_PROJECT_STATUS : codes d'état du projet de démarrage

Description du type énuméré

Le type de données énuméré `PLC_R_BOOT_PROJECT_STATUS` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`PLC_R_NO_BOOT_PROJECT`	0000 hex	Le projet de démarrage n'existe pas dans la mémoire Flash.
`PLC_R_BOOT_PROJECT_CREATION_IN_PROGRESS`	0001 hex	Le projet de démarrage est en cours de création.
`PLC_R_DIFFERENT_BOOT_PROJECT`	0002 hex	Le projet de démarrage dans la mémoire Flash est différent du projet chargé dans la RAM.
`PLC_R_VALID_BOOT_PROJECT`	FFFF hex	Le projet de démarrage dans la mémoire Flash est identique au projet chargé dans la RAM.

types de donnéesPLC_R_IO_STATUS : codes d'état des E/S

Description du type énuméré

Le type de données énuméré PLC_R_IO_STATUS contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
PLC_R_IO_OK	FFFF hex	Les entrées/sorties sont opérationnelles.
PLC_R_IO_NO_INIT	0001 hex	Les entrées/sorties ne sont pas initialisées.
PLC_R_IO_CONF_FAULT	0002 hex	Paramètres de configuration d'E/S incorrects détectés.
PLC_R_IO_SHORTCUT_FAULT	0003 hex	Court-circuit des entrées/sorties détecté.
PLC_R_IO_POWER_SUPPLY_FAULT	0004 hex	Erreur d'alimentation des E/S détectée.

PLC_R_STATUS : codes d'état du contrôleur

Description du type énuméré

Le type de données énuméré `PLC_R_STATUS` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`PLC_R_EMPTY`	0000 hex	Le contrôleur ne contient aucune application.
`PLC_R_STOPPED`	0001 hex	Le contrôleur est arrêté.
`PLC_R_RUNNING`	0002 hex	Le contrôleur fonctionne.
`PLC_R_HALT`	0004 hex	Le contrôleur est à l'état HALT. (Reportez-vous au diagramme d'état du contrôleur dans le *guide de programmation* de votre contrôleur.)
`PLC_R_BREAKPOINT`	0008 hex	Le contrôleur s'est interrompu au point d'arrêt.

PLC_R_STOP_CAUSE : codes expliquant le passage de l'état RUN à un autre état

Description du type énumération

Le type de données énumération PLC_R_STOP_CAUSE contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire	Que faire
PLC_R_STOP_FROM_STOP	00 hex	Contrôleur arrêté en raison d'une requête d'arrêt d'application.	–
PLC_R_STOP_FROM_POWER_ FAIL	01 hex	Contrôleur arrêté en raison d'une panne de courant.	–
PLC_R_STOP_FROM_NETWORK_ REQUEST	02 hex	Arrêté suite à une requête du réseau (clé USB ou commande PLC_W).	–
PLC_R_HALT_FROM_APP_ WATCHOG	03 hex	Contrôleur arrêté en raison d'un événement d'horloge de surveillance de la tâche.	Vérifiez votre application Voir le chapitre . Une réinitialisation est nécessaire pour entrer en mode Run.
PLC_R_HALT_FROM_CPU_ OVERLOAD	04 hex	Contrôleur arrêté en raison d'une surcharge de l'UC.	Réduisez la charge de travail de l'application en améliorant son architecture. Augmentez la durée du cycle de tâche. Réduisez la fréquence des événements.
PLC_R_HALT_FROM_ INTERNAL_ ERROR	05 hex	Contrôleur arrêté en raison d'une erreur interne détectée.	Contactez le service client le plus proche.

PLC_W_COMMAND : codes de commande de contrôle

Description du type énuméré

Le type de données énuméré `PLC_W_COMMAND` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`PLC_W_STOP`	0001 hex	Commande d'arrêt du contrôleur.
`PLC_W_RUN`	0002 hex	Commande d'exécution du contrôleur.
`PLC_W_RESET_COLD`	0004 hex	Commande de lancement d'une réinitialisation à froid du contrôleur.
`PLC_W_RESET_WARM`	0008 hex	Commande de lancement d'une réinitialisation à chaud du contrôleur.

Sous-chapitre 3.2
Types de données des variables système ETH_R/W

Présentation

Cette section répertorie et décrit les types de données de variable système, inclus dans les structures `ETH_R` et `ETH_W`.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

ETH_R_IP_MODE : codes sources d'adresse IP

Description du type énuméré

Le type de données énuméré `ETH_R_IP_MODE` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`ETH_R_STORED`	00 hex	L'adresse IP stockée est utilisée.
`ETH_R_BOOTP`	01 hex	Le protocole Bootstrap est utilisé pour obtenir une adresse IP.
`ETH_R_DHCP`	02 hex	Le protocole DHCP est utilisé pour obtenir une adresse IP.
`ETH_DEFAULT_IP`	FF hex	L'adresse IP par défaut est utilisée.

ETH_R_FRAME_PROTOCOL : codes de protocole de transmission de trames

Description du type énuméré

Le type de données énuméré `ETH_R_FRAME_PROTOCOL` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`ETH_R_802_3`	00 hex	Le protocole utilisé pour la transmission des trames est IEEE 802.3.
`ETH_R_ETHERNET_II`	01 hex	Le protocole utilisé pour la transmission des trames est Ethernet II.

ETH_R_PORT_DUPLEX_STATUS : codes du mode de transmission

Description du type énumération

Le type de données énumération `ETH_R_PORT_DUPLEX_STATUS` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`ETH_R_PORT_HALF_DUPLEX`	00 hex	Le mode de transmission en semi-duplex est utilisé.
`ETH_R_FULL_DUPLEX`	01 hex	Le mode de transmission en duplex intégral est utilisé.

ETH_R_PORT_LINK_STATUS : codes d'état de la liaison de communication

Description du type énumération

Le type de données énumération `ETH_R_PORT_LINK_STATUS` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`ETH_R_LINK_DOWN`	00 hex	Liaison de communication du serveur vers l'équipement.
`ETH_R_LINK_UP`	01 hex	Liaison de communication de l'équipement vers le serveur.

ETH_R_PORT_SPEED : codes de la vitesse de communication des ports Ethernet

Description du type énumération

Le type de données énumération `ETH_R_PORT_SPEED` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
`ETH_R_SPEED_10_MB`	10 déc	Le débit réseau est de 10 mégabits par seconde.
`ETH_R_100_MB`	100 déc	Le débit réseau est de 100 mégabits par seconde.

Sous-chapitre 3.3
Types de données des fonctions système

Présentation

Cette section décrit les différents types de données des fonctions système de la bibliothèque PLCSystem de ATV IMC.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

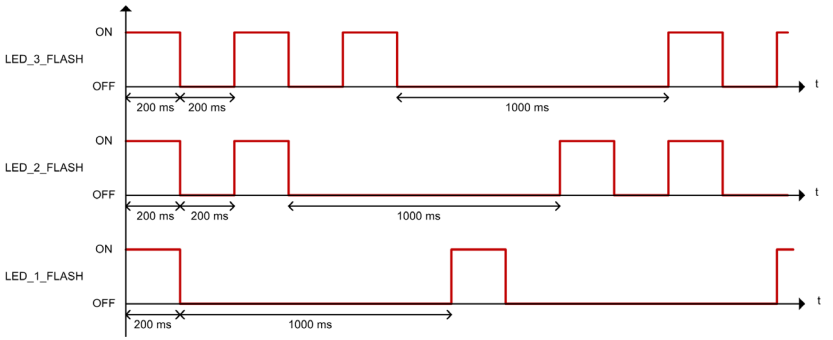
LED_BHV : Codes des paramètres LedBhv de la fonction SetLedBehaviour

Type énumération - Description

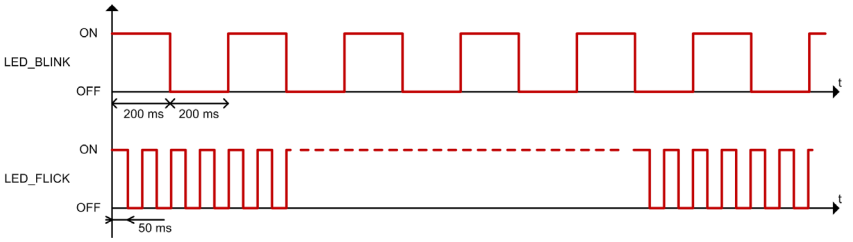
Le type de données énumération LED_BHV contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
LED_3_FLASH	-3 dec	Le voyant d'état clignote en mode 3 (voir diagramme ci-dessous).
LED_2_FLASH	-2 dec	Le voyant d'état clignote en mode 2 (voir diagramme ci-dessous).
LED_1_FLASH	-1 dec	Le voyant d'état clignote en mode 1 (voir diagramme ci-dessous).
LED_OFF	0 dec	Le voyant d'état est éteint en permanence.
LED_ON	1 dec	Le voyant d'état est allumé en permanence.
LED_BLINK	2 dec	Le voyant d'état clignote à 2,5 Hz (voir diagramme ci-dessous).
LED_FLICK	3 dec	Le voyant d'état clignote à 10 Hz (voir diagramme ci-dessous).

Le diagramme suivant décrit les modes de clignotement des voyants Application LED_x_FLASH :



Le diagramme suivant décrit les modes de clignotement des voyants Application `LED_BLINK` et `LED_FLICK`:



LED_BHV_ERROR : codes des erreurs détectées de la fonction SetLEDBehaviour

Description du type énumération

Le type de données énumération LED_BHV_ERROR contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
NO_ERROR	00 hex	Fonction de paramétrage du comportement du voyant exécutée sans détection d'erreur.
UNKNOWN_LED	01 hex	Le paramètre LED_ID est inconnu.
UNKNOWN_COLOR	02 hex	Le paramètre LED_COLOR est inconnu.
UNKNOWN_STATE	03 hex	L'état du voyant, contenu dans le paramètre LED_BHV, est inconnu.

MINUTE : codes du paramètre LedColor de la fonction SetLEDBehaviour

Description du type énumération

Le type de données énumération LED_COLOR contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
LED_RED	00 hex	Le voyant s'allume en rouge.
LED_GREEN	01 hex	Le voyant s'allume en vert.

LED_ID : codes du paramètre LedId de la fonction SetLEDBehaviour

Description du type énumération

Le type de données énumération LED_ID contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire
LED_0	00 hex	Identificateur du voyant USER de l'application.

PLC_ERROR_TYPE : ResetInternalErrorDiag Erreur de fonction Codes de paramètre

Description de la structure

Le type de données énumération PLC_ERROR_TYPE contient les valeurs suivantes :

Enumérateur	Valeur	Description
_ERR_IMC_WATCHDOG	0B hex	Erreur d'horloge de surveillance système détectée
_ERR_IMC_INT_ERROR	0C hex	Autre erreur interne détectée

Annexes

Annexe A
Représentation des fonctions et blocs fonction

Présentation

Chaque fonction peut être représentée dans les langages suivants :
- IL : (Instruction List) liste d'instructions
- ST : (Structured Text) littéral structuré
- LD : (Ladder Diagram) schéma à contacts
- FBD : Function Block Diagram (Langage à blocs fonction)
- CFC : Continuous Function Chart (Diagramme fonctionnel continu)

Ce chapitre fournit des exemples de représentations de fonctions et blocs fonction et explique comment les utiliser dans les langages IL et ST.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Différences entre une fonction et un bloc fonction

Fonction

Une fonction :

- est une POU (Program Organization Unit ou unité organisationnelle de programme) qui renvoie un résultat immédiat ;
- est directement appelée par son nom (et non par une instance) ;
- ne conserve pas son état entre deux appels ;
- peut être utilisée en tant qu'opérande dans des expressions.

Exemples : opérateurs booléens (`AND`), calculs, conversions (`BYTE_TO_INT`)

Bloc fonction

Un bloc fonction :

- est une POU qui renvoie une ou plusieurs sorties ;
- doit être appelé par une instance (copie de bloc fonction avec nom et variables dédiées).
- Chaque instance conserve son état (sorties et variables internes) entre deux appels à partir d'un bloc fonction ou d'un programme.

Exemples : temporisateurs, compteurs

Dans l'exemple, `Timer_ON` est une instance du bloc fonction `TON` :

```
1  PROGRAM MyProgram_ST
2  VAR
3      Timer_ON: TON;  // Function Block Instance
4      Timer_RunCd: BOOL;
5      Timer_PresetValue: TIME := T#5S;
6      Timer_Output: BOOL;
7      Timer_ElapsedTime: TIME;
8  END_VAR
```

```
1  Timer_ON(
2      IN:=Timer_RunCd,
3      PT:=Timer_PresetValue,
4      Q=>Timer_Output,
5      ET=>Timer_ElapsedTime);
```

Utilisation d'une fonction ou d'un bloc fonction en langage IL

Informations générales

Cette partie explique comment mettre en œuvre une fonction et un bloc fonction en langage IL.

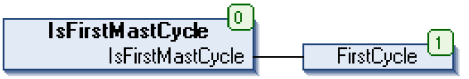
Les fonctions `IsFirstMastCycle` et `SetRTCDrift`, ainsi que le bloc fonction `TON`, sont utilisés à titre d'exemple pour illustrer les mises en œuvre.

Utilisation d'une fonction en langage IL

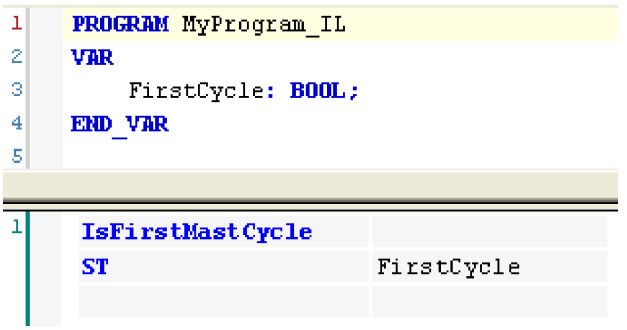
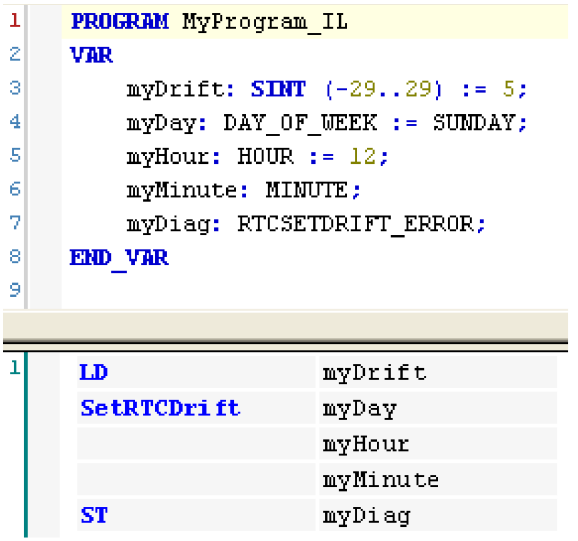
La procédure suivante explique comment insérer une fonction en langage IL :

Etape	Action
1	Ouvrez ou créez un POU en langage IL (Instruction List, ou liste d'instructions). **NOTE :** La procédure de création d'un POU n'est pas détaillée ici. Pour plus d'informations, reportez-vous à la section Ajout et appel de POU *(voir SoMachine, Guide de programmation)*.
2	Créez les variables nécessaires à la fonction.
3	Si la fonction possède une ou plusieurs entrées, chargez la première entrée en utilisant l'instruction LD.
4	Insérez une nouvelle ligne en dessous et : ● saisissez le nom de la fonction dans la colonne de l'opérateur (champ de gauche), ou ● utilisez l'**Aide à la saisie** pour choisir la fonction (sélectionnez **Insérer l'appel de module** dans le menu contextuel).
5	Si la fonction a plusieurs entrées et que l'Aide à la saisie est utilisée, le nombre requis de lignes est automatiquement créé avec `???` dans les champs situés à droite. Remplacez les `???` par la valeur ou la variable appropriée en fonction de l'ordre des entrées.
6	Insérez une nouvelle ligne pour stocker le résultat de la fonction dans la variable appropriée : saisissez l'instruction ST dans la colonne de l'opérateur (champ de gauche) et un nom de variable dans le champ situé à droite.

Pour illustrer la procédure, utilisons les fonctions `IsFirstMastCycle` (sans paramètre d'entrée) et `SetRTCDrift` (avec paramètres d'entrée) représentées graphiquement ci-après :

Fonction	Représentation graphique
sans paramètre d'entrée : `IsFirstMastCycle`	
avec paramètres d'entrée : `SetRTCDrift`	

En langage IL, le nom de la fonction est utilisé directement dans la colonne de l'opérateur :

Fonction	Représentation dans l'éditeur IL de POU de SoMachine
Exemple IL d'une fonction sans paramètre d'entrée : `IsFirstMastCycle`	
Exemple IL d'une fonction avec des paramètres d'entrée : `SetRTCDrift`	

Utilisation d'un bloc fonction en langage IL

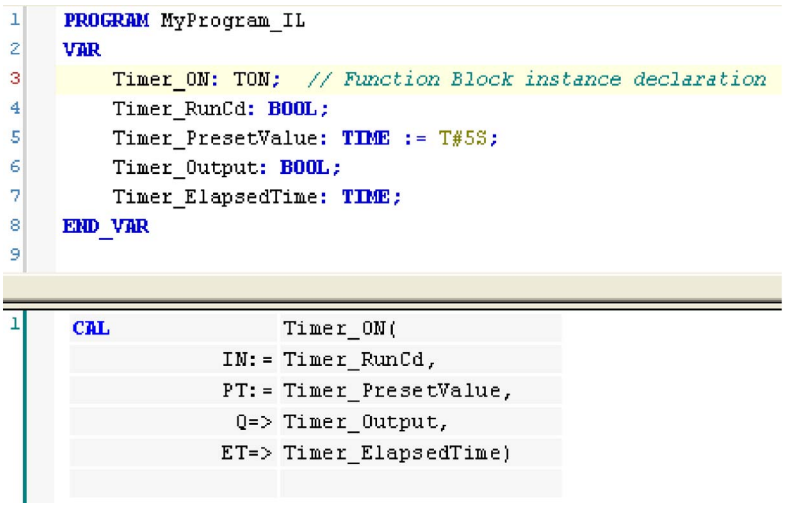
La procédure suivante explique comment insérer un bloc fonction en langage IL :

Etape	Action
1	Ouvrez ou créez un POU en langage IL (Instruction List, ou liste d'instructions). **NOTE :** La procédure de création d'un POU n'est pas détaillée ici. Pour plus d'informations, reportez-vous à la section Ajout et appel de POU *(voir SoMachine, Guide de programmation)*.
2	Créez les variables nécessaires au bloc fonction (y compris le nom de l'instance).
3	L'appel de blocs fonction nécessite l'utilisation d'une instruction `CAL` : ● Utilisez l'**Aide à la saisie** pour sélectionner le bloc fonction (cliquez avec le bouton droit et sélectionnez **Insérer l'appel de module** dans le menu contextuel). ● L'instruction `CAL` et les E/S nécessaires sont automatiquement créées. Chaque paramètre (E/S) est une instruction : ● Les valeurs des entrées sont définies à l'aide de « `:=` ». ● Les valeurs des sorties sont définies à l'aide de « `=>` ».
4	Dans le champ `CAL` de droite, remplacez les `???` par le nom de l'instance.
5	Remplacez les autres `???` par une variable ou une valeur immédiate appropriée.

Pour illustrer la procédure, utilisons le bloc fonction `TON` représenté graphiquement ci-après :

Bloc fonction	Représentation graphique
`TON`	

En langage IL, le nom du bloc fonction est utilisé directement dans la colonne de l'opérateur :

Bloc fonction	Représentation dans l'éditeur IL de POU de SoMachine
TON	

```
PROGRAM MyProgram_IL
VAR
    Timer_ON: TON;  // Function Block instance declaration
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
CAL          Timer_ON(
        IN:= Timer_RunCd,
        PT:= Timer_PresetValue,
         Q=> Timer_Output,
        ET=> Timer_ElapsedTime)
```

Utilisation d'une fonction ou d'un bloc fonction en langage ST

Informations générales

Cette partie décrit comment mettre en œuvre une fonction ou un bloc fonction en langage ST.

La fonction `SetRTCDrift` et le bloc fonction `TON` sont utilisés à titre d'exemple pour illustrer les mises en œuvre.

Utilisation d'une fonction en langage ST

La procédure suivante explique comment insérer une fonction en langage ST :

Etape	Action
1	Ouvrez ou créez un POU en langage ST (Structured Text ou Littéral structuré). **NOTE :** La procédure de création d'un POU n'est pas détaillée ici. Pour plus d'informations, reportez-vous à la section Ajout et appel de POU *(voir SoMachine, Guide de programmation)*.
2	Créez les variables nécessaires à la fonction.
3	Utilisez la syntaxe générale dans l'**éditeur ST de POU** pour la représentation en langage ST d'une fonction. La syntaxe générale est la suivante : `RésultatFonction:= NomFonction(VarEntrée1, VarEntrée2, … VarEntréex);`

Pour illustrer la procédure, utilisons la fonction `SetRTCDrift` représentée graphiquement ci-après :

Fonction	Représentation graphique
`SetRTCDrift`	

La représentation en langage ST de cette fonction est la suivante :

Fonction	Représentation dans l'éditeur ST de POU de SoMachine

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Utilisation d'un bloc fonction en langage ST

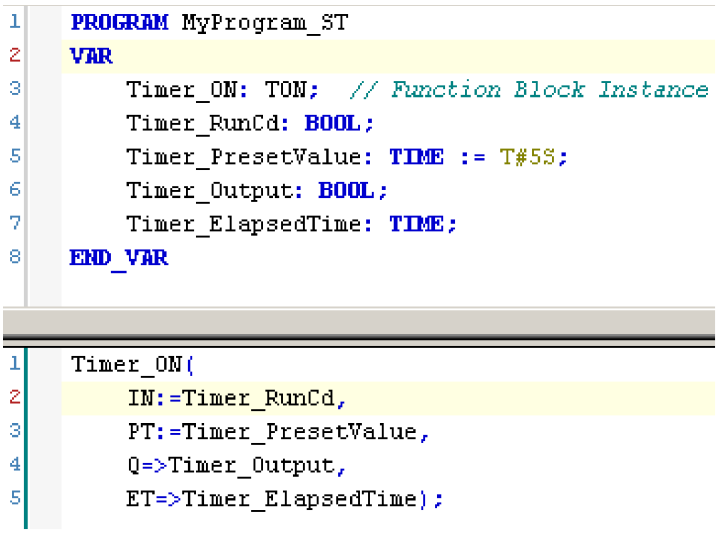
La procédure suivante explique comment insérer un bloc fonction en langage ST :

Etape	Action
1	Ouvrez ou créez un POU en langage IL (Instruction List, ou liste d'instructions). **NOTE :** La procédure de création d'un POU n'est pas détaillée ici. Pour plus d'informations sur l'ajout, la déclaration et l'appel de POU, reportez-vous à la documentation *(voir SoMachine, Guide de programmation)* associée.
2	Créez les variables d'entrée, les variables de sortie et l'instance requises pour le bloc fonction : ● Les variables d'entrée sont les paramètres d'entrée requis par le bloc fonction. ● Les variables de sortie reçoivent la valeur renvoyée par le bloc fonction.
3	Utilisez la syntaxe générale dans l'**éditeur ST de POU** pour la représentation en langage ST d'un bloc fonction. La syntaxe générale est la suivante : `BlocFonction_NomInstance(Entrée1:=VarEntrée1, Entrée2:=VarEntrée2,… Sortie1=>VarSortie1, Sortie2=>VarSortie2,…);`

Pour illustrer la procédure, utilisons le bloc fonction `TON` représenté graphiquement ci-après :

Bloc fonction	Représentation graphique
`TON`	

Le tableau suivant montre plusieurs exemples d'appel de bloc fonction en langage ST :

Bloc fonction	Représentation dans l'éditeur ST de POU de SoMachine
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossaire

!

%

Selon la norme IEC, % est un préfixe qui identifie les adresses mémoire internes des contrôleurs logiques pour stocker la valeur de variables de programme, de constantes, d'E/S, etc.

%MW

Selon la norme IEC, %MW représente un registre de mots mémoire (par exemple, un objet langage de type mot mémoire).

A

adresse MAC

(*media access control*) Nombre unique sur 48 bits associé à un élément matériel spécifique. L'adresse MAC est programmée dans chaque carte réseau ou équipement lors de la fabrication.

application

Programme comprenant des données de configuration, des symboles et de la documentation.

application de démarrage

(*boot application*). Fichier binaire qui contient l'application. En général, il est stocké dans le contrôleur et permet à ce dernier de démarrer sur l'application que l'utilisateur a générée.

ARRAY

Agencement systématique d'objets de données d'un même type sous la forme d'un tableau défini dans la mémoire d'un contrôleur logique. La syntaxe est la suivante : `ARRAY [<dimension>] OF <Type>`

Exemple 1 : `ARRAY [1..2] OF BOOL` est un tableau à 1 dimension composé de 2 éléments de type `BOOL`.

Exemple 2 : `ARRAY [1..10, 1..20] OF INT` est un tableau à 2 dimensions composés de 10 x 20 éléments de type `INT`.

B

BOOL

(*booléen*) Type de données informatique standard. Une variable de type `BOOL` peut avoir l'une des deux valeurs suivantes : 0 (`FALSE`), 1 (`TRUE`). Un bit extrait d'un mot est de type `BOOL` ; par exemple, `%MW10.4` est le cinquième bit d'un mot mémoire numéro 10.

BOOTP

(*bootstrap protocol*). Protocole réseau UDP qu'un client réseau peut utiliser pour obtenir automatiquement une adresse IP (et éventuellement d'autres données) à partir d'un serveur. Le client s'identifie auprès du serveur à l'aide de son adresse MAC. Le serveur, qui gère un tableau préconfiguré des adresses MAC des équipements client et des adresses IP associées, envoie au client son adresse IP préconfigurée. A l'origine, le protocole BOOTP était utilisé pour amorcer à distance les hôtes sans lecteur de disque à partir d'un réseau. Le processus BOOTP affecte une adresse IP de durée illimitée. Le service BOOTP utilise les ports UDP 67 et 68.

C

CFC

Acronyme de *continuous function chart*, diagramme fonctionnel continu. Langage de programmation graphique (extension de la norme IEC 61131-3) basé sur le langage de diagramme à blocs fonction et qui fonctionne comme un diagramme de flux. Toutefois, il n'utilise pas de réseaux et le positionnement libre des éléments graphiques est possible, ce qui permet les boucles de retour. Pour chaque bloc, les entrées se situent à gauche et les sorties à droite. Vous pouvez lier les sorties de blocs aux entrées d'autres blocs pour créer des expressions complexes.

chien de garde

Temporisateur spécial utilisé pour garantir que les programmes ne dépassent pas le temps de scrutation qui leur est alloué. Le chien de garde est généralement réglé sur une valeur supérieure au temps de scrutation et il est remis à 0 à la fin de chaque cycle de scrutation. Si le temporisation chien de garde atteint la valeur prédéfinie (par exemple, lorsque le programme est bloqué dans une boucle sans fin) une erreur est déclarée et le programme s'arrête.

configuration

Agencement et interconnexions des composants matériels au sein d'un système, ainsi que les paramètres matériels et logiciels qui déterminent les caractéristiques de fonctionnement du système.

contrôleur

Automatise des processus industriels. On parle également de contrôleur logique programmable (PLC) ou de contrôleur programmable.

D

DHCP

Acronyme de *dynamic host configuration protocol*. Extension avancée du protocole BOOTP. Bien que DHCP soit plus avancé, DHCP et BOOTP sont tous les deux courants. (DHCP peut gérer les requêtes de clients BOOTP.)

DWORD

Abréviation de *double word*, mot double. Codé au format 32 bits.

E

E/S

Entrée/sortie

F

FB

Acronyme de *function block*, bloc fonction. Mécanisme de programmation commode qui consolide un groupe d'instructions de programmation visant à effectuer une action spécifique et normalisée telle que le contrôle de vitesse, le contrôle d'intervalle ou le comptage. Un bloc fonction peut comprendre des données de configuration, un ensemble de paramètres de fonctionnement interne ou externe et généralement une ou plusieurs entrées et sorties de données.

G

GVL

Acronyme de *global variable list*, liste de variables globales. Gère les variables globales qui peuvent être transmises entre contrôleurs sur un réseau Ethernet TCP/IP Modbus.

H

hex

(*hexadécimal*)

HSC

Abréviation de *high-speed counter*, compteur rapide

I

ID

(*identificateur/identification*)

IEC

Acronyme de *International Electrotechnical Commission*, Commission Electrotechnique Internationale (CEI). Organisation internationale non gouvernementale à but non lucratif, qui rédige et publie les normes internationales en matière d'électricité, d'électronique et de domaines connexes.

IEC 61131-3

Partie 3 d'une norme en 3 parties de l'IEC pour les équipements d'automatisation industriels. La norme IEC 61131-3 traite des langages de programmation des contrôleurs. Elle définit 2 normes pour la programmation graphique et 2 normes pour la programmation textuelle. Les langages de programmation graphiques sont le schéma à contacts (LD) et le langage à blocs fonction (FBD). Les langages textuels comprennent le texte structuré (ST) et la liste d'instructions (IL).

IEEE 802.3

Ensemble de normes IEEE définissant la couche physique et la sous-couche MAC de la couche de liaison de données de l'Ethernet câblé.

IL

Acronyme de *instruction list*, liste d'instructions. Un programme écrit en langage IL est composé d'instructions textuelles qui sont exécutées séquentiellement par le contrôleur. Chaque instruction comprend un numéro de ligne, un code d'instruction et un opérande (voir la norme IEC 61131-3).

INT

Abréviation de *integer*), nombre entier codé sur 16 bits.

IP

Acronyme de *Internet Protocol*, protocole Internet. Le protocole IP fait partie de la famille de protocoles TCP/IP, qui assure le suivi des adresses Internet des équipements, achemine les messages sortants et reconnaît les messages entrants.

L

langage en blocs fonctionnels

Un des 5 langages de programmation de logique ou de commande pris en charge par la norme IEC 61131-3 pour les systèmes de commande. FBD est un langage de programmation orienté graphique. Il fonctionne avec une liste de réseaux où chaque réseau contient une structure graphique de zones et de lignes de connexion représentant une expression logique ou arithmétique, un appel de bloc fonction ou une instruction de retour.

LD

Acronyme de *ladder diagram*, schéma à contacts. Représentation graphique des instructions d'un programme de contrôleur, avec des symboles pour les contacts, les bobines et les blocs dans une série de réseaux exécutés séquentiellement par un contrôleur (voir IEC 61131-3).

M

MAST

Tâche de processeur exécutée par le biais de son logiciel de programmation. La tâche MAST comprend deux parties :

- **IN :** les entrées sont copiées dans la section IN avant exécution de la tâche MAST.
- **OUT :** les sorties sont copiées dans la section OUT après exécution de la tâche MAST.

mémoire flash

Mémoire non volatile qui peut être écrasée. Elle est stockée dans une puce EEPROM spéciale, effaçable et reprogrammable.

micrologiciel

Représente le BIOS, les paramètres de données et les instructions de programmation qui constituent le système d'exploitation d'un contrôleur. Le micrologiciel est stocké dans la mémoire non volatile du contrôleur.

O

octet

Type codé sur 8 bits, de 00 à FF au format hexadécimal.

P

PLC

Acronyme de *programmable logic controller*, contrôleur logique programmable. Ordinateur industriel utilisé pour automatiser des processus de fabrication et autres processus électromécaniques. Les PLCs diffèrent des ordinateurs courants par le fait qu'ils sont conçus pour utiliser plusieurs tableaux d'entrées et de sorties et pour accepter des conditions de choc, de vibration, de température et d'interférences électriques plus rudes.

POU

Acronyme de *program organization unit*, unité organisationnelle de programme. Déclaration de variables dans le code source et jeu d'instructions correspondant. Les POUs facilitent la réutilisation modulaire de programmes logiciels, de fonctions et de blocs fonction. Une fois déclarées, les POUs sont réutilisables.

programme

Composant d'une application constitué de code source compilé qu'il est possible d'installer dans la mémoire d'un contrôleur logique.

R

RTC

Acronyme de *real-time clock*, horloge en temps réel. Horloge horaire et calendaire supportée par une batterie qui fonctionne en continu, même lorsque le contrôleur n'est pas alimenté, jusqu'à la fin de l'autonomie de la batterie.

run

Commande qui ordonne au contrôleur de scruter le programme d'application, lire les entrées physiques et écrire dans les sorties physiques en fonction de la solution de la logique du programme.

S

ST

Acronyme de *structured text*, texte structuré. Langage composé d'instructions complexes et d'instructions imbriquées (boucles d'itération, exécutions conditionnelles, fonctions). Le langage ST est conforme à la norme IEC 61131-3.

STOP

Commande ordonnant au contrôleur de cesser d'exécuter un programme d'application.

T

tâche

Ensemble de sections et de sous-programmes, exécutés de façon cyclique ou périodique pour la tâche MAST, ou périodique pour la tâche FAST.

Une tâche présente un niveau de priorité et des entrées et sorties du contrôleur lui sont associées. Ces E/S sont actualisées par rapport à la tâche.

Un contrôleur peut comporter plusieurs tâches.

TCP

Acronyme de *transmission control protocol*, protocole de contrôle de transmission. Protocole de couche de transport basé sur la connexion qui assure la transmission de données simultanée dans les deux sens. Le protocole TCP fait partie de la suite de protocoles TCP/IP.

U

UINT

Abréviation de *unsigned integer*, entier non signé. Valeur codée sur 16 bits.

V

variable
Unité de mémoire qui est adressée et modifiée par un programme.

variable non localisée
Variable qui n'a pas d'adresse (voir *variable localisée*).

variable système
Variable qui fournit des données de contrôleur et des informations de diagnostic et permet d'envoyer des commandes au contrôleur.

W

WORD
Type de données codé sur 16 bits.

Index

R

S

T

V