Magelis XBTGC, XBTGT, XBTGK HMI Controller

Fonctions et variables système
Guide de la bibliothèque XBT PLCSystem

11/2015

EIO0000000627.07

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

L'objet de ce document est de vous familiariser avec les fonctions et variables système fournies par les contrôleurs XBTGC, XBTGK et XBTGT. La bibliothèque XBT PLCSystem contient des fonctions et des variables permettant de recevoir des informations du système XBT et de lui envoyer des commandes.

Ce document décrit les types de données associés aux fonctions et variables de la bibliothèque XBT PLCSystem.

Les connaissances fondamentales requises pour tirer profit de ce document sont les suivantes :

- connaissances de base sur les fonctionnalités, la structure et la configuration du contrôleur XBTGC ;, XBTGT et XBT GK HMI Controller
- 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 XBTGC, XBTGT et XBTGK

Présentation

Ce chapitre :

- propose une introduction aux variables système *(voir page 12)* ;
- décrit les variables système *(voir page 18)* disponibles avec la bibliothèque XBTPLC.

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 Magelis XBTGC HMI Controller.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Présentation des variables système

Introduction

Cette section décrit la mise en œuvre des variables système pour le contrôleur. Ces variables possèdent les attributs suivants :

- Les variables système permettent d'accéder à des informations générales sur le système, de réaliser des diagnostics système et de commander des actions simples.
- Les variables système sont des variables structurées selon les définitions et conventions de désignation de la norme CEI 61131. Elles sont accessibles en utilisant le 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`).
- Les variables système sont déclarées automatiquement comme des variables globales. Elles s'appliquent à l'ensemble du système et doivent être utilisées avec précaution, car tous les POU (unités organisationnelles de programme) d'une tâche peuvent y accéder.

Conventions de désignation des variables système

Les variables système sont identifiées par :

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

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

Voici un exemple de mise en œuvre des 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 dans son 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 affectées
- variables non affectées

Les variables affectées :

- 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 les programmes SoMachine selon la convention `nom_structure.nom_composant` expliquée précédemment (les adresses %MW comprises entre 0 et 59999 sont accessibles directement ; les adresses au-delà sont considérées comme étant hors plage par SoMachine et ne sont accessibles que par le biais de la convention `nom_structure.nom_composant`) ;

Les variables non affectées :

- ne se trouvent pas physiquement dans la zone %MW ;
- ne sont pas accessibles par un bus de terrain ou des requêtes réseau ;
- sont utilisées dans les programmes SoMachine conformément à la convention `nom_structure.nom_composant` expliquée précédemment.

Utilisation des variables système

Introduction

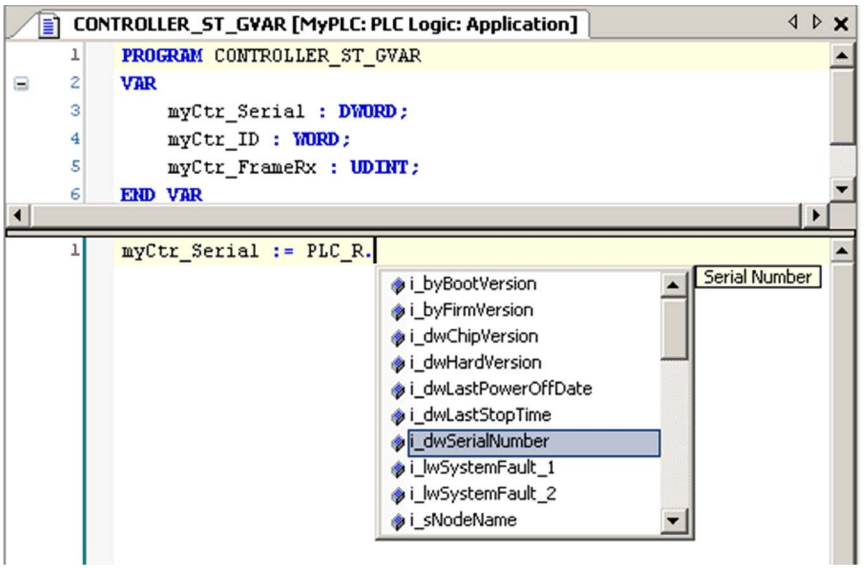
Cette rubrique décrit la procédure de programmation et d'utilisation des variables système dans SoMachine.

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_R : variables système en lecture seule de contrôleur

Structure de la variable

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

Nom de la variable	Type	Commentaire
i_wVendorID	WORD	ID du fournisseur du contrôleur. 101A hex = Schneider Electric
i_wProductID	WORD	ID de référence du contrôleur. **NOTE :** Les identifiants de fournisseur et de référence sont les composantes de l'identifiant cible du contrôleur indiqué dans l'écran des paramètres de communication (ID cible = XXXX 101 hex).
i_byFirmVersion[0..3]	ARRAY[0..3] OF BYTE	Version [aa.bb.cc.dd] du micrologiciel du contrôleur : ● i_byFirmVersion[0]= aa ● ... ● i_byFirmVersion[3]= dd
i_byBootVersion[0..3]	ARRAY[0..3] OF BYTE	Version [aa.bb.cc.dd] du démarrage du contrôleur : ● i_byBootVersion[0]= aa ● ... ● i_byBootVersion[3]= dd
i_dwHardVersion	DWORD	Version du matériel du contrôleur.
i_wStatus	PLC_R_STATUS *(voir page 40)*	Etat du contrôleur.
i_wBootProjectStatus	PLC_R_BOOT_PROJECT_STATUS *(voir page 39)*	Renvoie des informations sur l'application de démarrage stockée dans la mémoire Flash.
i_wLastStopCause	PLC_R_STOP_CAUSE *(voir page 41)*	Cause du dernier passage du mode RUN à un autre état.
i_wLastApplicationError	PLC_R_APPLICATION_ERROR *(voir page 37)*	Cause de la dernière exception détectée du contrôleur.
i_wIOStatus1	PLC_R_IO_STATUS *(voir page 36)*	Etat des E/S intégrées. **NOTE :** Valide uniquement pour le contrôleur XBT GC.
i_wIOStatus2	PLC_R_IO_STATUS *(voir page 36)*	Etat des E/S TM2. **NOTE :** Valide uniquement pour le contrôleurXBT GC.

Nom de la variable	Type	Commentaire
i_dwAppliSignature1	DWORD	Signature du premier des quatre DWORD (16 octets au total). La signature de l'application est créée par le logiciel pendant la génération.
i_dwAppliSignature2	DWORD	Signature du deuxième des quatre DWORD (16 octets au total). La signature de l'application est créée par le logiciel pendant la génération.
i_dwAppliSignature3	DWORD	Signature du troisième des quatre DWORD (16 octets au total). La signature de l'application est créée par le logiciel pendant la génération.
i_dwAppliSignature4	DWORD	Signature du quatrième des quatre DWORD (16 octets au total). La signature de l'application est créée par le logiciel pendant la génération.

PLC_W : variable système en lecture/écriture de contrôleur

Structure de la variable

Le tableau suivant décrit les paramètres inclus dans la variable système `PLC_W` (type `PLC_W_STRUCT`) :

Nom de la variable	Type	Commentaire
`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.
`q_wPLCControl`	`PLC_W_COMMAND` *(voir page 43)*	Commande RUN/STOP du contrôleur exécutée lorsque la valeur de la variable système `PLC_R.q_uiOpenPLCControl` passe de 0 à 6699.

Sous-chapitre 1.3
Structures SERIAL_R et SERIAL_W

Présentation

Cette section répertorie et décrit les variables système des structures `SERIAL_R` et `SERIAL_W`.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

SERIAL_R[0..1] : variables système en lecture seule de ligne série

Introduction

SERIAL_R est un tableau de 2 éléments de type SERIAL_R_STRUCT. Chaque élément du tableau renvoie des **variables système** de diagnostic pour la ligne série correspondante :

- Serial_R[0] se rapporte au port COM1 (non pertinent pour le XBTGC1100).
- Serial_R[1] se rapporte au port COM2 (non pertinent pour le XBTGC).

Structure de la variable

Le tableau suivant décrit les paramètres de la variable système SERIAL_R[0..1] :

Nom de la variable	Type	Commentaire
i_udiFramesTransmittedOK	UDINT	Nombre de trames transmises avec succès.
i_udiFramesReceivedOK	UDINT	Nombre de trames reçues sans erreur détectée.
i_udiRX_MessagesError	UDINT	Nombre de trames reçues avec erreurs détectées (somme de contrôle, parité).

NOTE :
Les compteurs de SERIAL_R sont réinitialisés :

- en cas de téléchargement ;
- lors d'une réinitialisation de l'automate ;
- par la commande SERIAL_W[x].q_wResetCounter ;
- par la commande de réinitialisation associée au code fonction de requête Modbus n°8.

SERIAL_W[0..1] : variables système en lecture/écriture de ligne série

Introduction

SERIAL_W est un tableau de 2 éléments de type SERIAL_R_STRUCT. Chaque élément du tableau force la **variable système** SERIAL_R pour la ligne série correspondante à réinitialiser :

- Serial_W[0] se rapporte au port COM1 (non pertinent pour le XBTGC1100).
- Serial_W[1] se rapporte au port COM2 (non pertinent pour le XBTGC).

Structure de la variable

Le tableau suivant décrit les paramètres de la variable système SERIAL_W[0..1] :

Nom de la variable	Type	Commentaire
q_wResetCounter	WORD	Le passage de 0 à 1 réinitialise tous les compteurs de SERIAL_R[0..1]. 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 XBTGC , XBTGT et XBTGK

Sous-chapitre 2.1
Fonctions de lecture de XBTGC, XBTGT et XBTGK

Présentation

Cette section décrit les fonctions de lecture de la bibliothèque PLCSystem de XBTGC.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

HMI_GetRightBusStatus : renvoie l'état du bus d'extension

Description de la fonction

Cette fonction renvoie l'état du bus d'extension des E/S dans un champ de bits. A la suite du paramètre d'entrée (`Mask`), la fonction renvoie un diagnostic de configuration sur le bus d'extension des E/S (différence de configuration du bus d'extension des E/S avec les modules connectés) ou un diagnostic détaillé du module d'extension demandé pour les E/S analogiques.

NOTE : La fonction `HMI_GetRightBusStatus` ne s'applique qu'au HMI Controller XBT GC.

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 47)*.

Description des variables d'E/S

Le tableau suivant décrit le paramètre d'entrée :

Entrée	Type	Commentaire
`Mask`	`RIGHTBUS_GET_STATUS` *(voir page 44)*	Définit le type de diagnostic du bus d'extension d'E/S, renvoyé par la fonction : configuration du bus ou l'un des trois modèles d'extension possibles.

Le tableau suivant décrit les variables de sortie :

Sortie	Type	Commentaire
`HMI_GetRightBusStatus`	`WORD`	Diagnostic du bus d'extension d'E/S voir informations ci-après

Diagnostic générique du bus d'extension d'E/S

Le tableau suivant décrit le champ de bits renvoyé par la fonction `HMI_GetRightBusStatus` lorsque le paramètre d'entrée (`Mask`) est `RIGHT_BUS_GET_GEN_STATUS` *(voir page 44)* pour un diagnostic de configuration du bus d'extension (hex 0000 si aucune erreur n'est détectée) :

Bit	Description
0	Réservé (toujours 0)
1	TRUE si la configuration 1 du module TM2 ne correspond pas au module connecté.
2	TRUE si la configuration 2 du module TM2 ne correspond pas au module connecté.
3	TRUE si la configuration 3 du module TM2 ne correspond pas au module connecté.
4 à 15	Réservé (toujours 0)

Diagnostic des modules du bus d'extension d'E/S

Les tableaux ci-dessous décrivent le champ de bits renvoyé par la fonction `HMI_GetRight-BusStatus` lorsque le paramètre d'entrée (`Mask`) est `RIGHTBUS_GET_STATUSx` *(voir page 44)* (où x=1 à 3) pour un diagnostic détaillé du module d'extension "x".

NOTE : Le diagnostic détaillé est correct pour des modules d'E/S analogiques uniquement et la signification du champ de bits dépend du type du module analogique concerné.

Lorsque le module associé est un module d'E/S analogiques standard (jusqu'à 2 voies d'entrée) :

- TM2AMM3HT
- TM2ALM3LT
- TM2AMI2HT
- TM2AMI2LT
- TM2AVO2HT
- TM2AMO1HT

Bit	Description
0	Toutes les voies analogiques à l'état normal
1	Module à l'état d'initialisation
2	Dysfonctionnement de l'alimentation
3	Configuration incorrecte – analyse nécessaire
4	Processus de conversion de la voie d'entrée 0
5	Processus de conversion de la voie d'entrée 1
6	Paramètre non valide de la voie d'entrée 0
7	Paramètre non valide de la voie d'entrée 1
8	Non utilisée

Bit	Description
9	Non utilisée
10	Valeur de dépassement par le haut de la voie d'entrée 0
11	Valeur de dépassement par le haut de la voie d'entrée 1
12	Valeur de dépassement par le bas de la voie d'entrée 0
13	Valeur de dépassement par le bas de la voie d'entrée 1
14	Non utilisée
15	Paramètre non valide de la voie de sortie

Lorsque le module associé est l'un des modules à 4 ou 8 voies d'entrées analogiques :
- TM2ARI8HT
- TM2AMI8HT
- TM2ARI8LT
- TM2ARI8LRJ
- TM2AMI4LT
- TM2AMM6HT

Bit	Description	Signification
0, 1	Etat de la voie 0	00 : voie analogique à l'état normal 01 : paramètre non valide de la voie d'entrée 10 : valeur d'entrée indisponible (module en phase d'initialisation, processus de conversion) 11 : valeur non valide de la voie d'entrée (valeur supérieure au seuil maximal ou inférieure au seuil minimal)
2, 3	Etat de la voie 1	voir bit 0, 1
4, 5	Etat de la voie 2	voir bit 0, 1
6, 7	Etat de la voie 3	voir bit 0, 1
8, 9	Etat de la voie 4	voir bit 0, 1 (pour les modules à 8 voies d'entrée uniquement)
10, 11	Etat de la voie 5	voir bit 0, 1 (pour les modules à 8 voies d'entrée uniquement)
12, 13	Etat de la voie 6	voir bit 0, 1 (pour les modules à 8 voies d'entrée uniquement)
14, 15	Etat de la voie 7	voir bit 0, 1 (pour les modules à 8 voies d'entrée uniquement)

NOTE : Lorsque le module d'extension ciblé est un module d'E/S numériques, le diagnostic renvoyé est non valide (hex 0000).

Exemple

L'exemple suivant décrit une méthode utilisant `HMI_GetRightBusStatus` pour le diagnostic du bus d'extension d'E/S et des modules :

```
VAR
(*diagnostic de conf des modules 1 à 3 = MyRightBusStatus bits 1 à 3*)
MyRightBusStatus: WORD;
(*Codes de diagnostic des modules 1 à 3*)
    ModuleError:Array [1..3] of WORD;
END_VAR
```

```
(*Diagnostic de conf sur le bus d'extension*)
MyRightBusStatus:=HMI_GetRightBusStatus(RIGHTBUS_GET_GEN_STATUS);
IF MyRightBusStatus<>0 THEN
(*Diff de conf détectée => définir une alarme, vérifier les valeurs des
bits...*)END_IF;
(*Obtention du diagnostic des modules : erreur détectée si diag <> 0*)
(*Limiter la liste aux modules analogiques configurés*)
ModuleError[1]:=HMI_GetRightBusStatus(RIGHTBUS_GET_STATUS1);
ModuleError[2]:=HMI_GetRightBusStatus(RIGHTBUS_GET_STATUS2);
ModuleError[3]:=HMI_GetRightBusStatus(RIGHTBUS_GET_STATUS3);
```

HMI_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 1er 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 47)*.

Description des variables d'E/S

Le tableau suivant décrit la variable de sortie :

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

Exemple

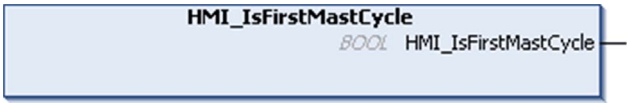
Reportez-vous à la fonction `HMI_IsFirstMastCycle` *(voir page 32)*.

HMI_IsFirstMastCycle: indique si Cycle est le premier cycle Mast

Description de la fonction

Cette fonction renvoie TRUE au cours du 1er 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 47)*.

Description des variables d'E/S

Sortie	Type	Commentaire
`HMI_IsFirstMastCycle`	`BOOL`	TRUE au cours du premier cycle de la tâche MAST après un démarrage.

Exemple

Cet exemple décrit les trois fonctions `HMI_IsFirstMastCycle`, `HMI_IsFirstMast-ColdCycle` et `HMI_IsFirstMastWarmCycle` utilisées ensemble :

NOTE : Les fonctions doivent être utilisées dans la tâche MAST, faute de quoi des actions d'initialisation risquent d'être réalisées plusieurs fois ou au contraire de ne jamais être exécutées (une tâche supplémentaire pourrait être appelée plusieurs fois ou ne pas être appelée du tout pendant un cycle de tâche MAST).

```
IF HMI_IsFirstMastWarmCycle() THEN
(*Premier cycle MAST après un démarrage à chaud: toutes les variables
sont définies à leurs valeurs d'initialisation sauf les variables
conservées*)
(*=> initialiser les variables nécessaires pour que votre application
s'exécute comme prévu dans ce cas*)
END_IF;
IF HMI_IsFirstMastColdCycle() THEN
(*Premier cycle Mast après un démarrage à froid : toutes les variables
sont définies à leurs valeurs d'initialisation y compris les variables
conservées*)
(*=> initialiser les variables nécessaires pour que votre application
s'exécute comme prévu dans ce cas*)
END_IF;
IF HMI_IsFirstMastCycle() THEN
(*Premier cycle Mast après un démarrage : à froid, à chaud ou après des
commandes STOP/RUN*)
(*=> initialiser les variables nécessaires pour que votre application
s'exécute comme prévu dans ce cas*)
END_IF;]
```

HMI_IsFirstMastWarmCycle : indique si Cycle est le premier cycle de démarrage à chaud Mast

Description de la fonction

Cette fonction renvoie TRUE au cours du 1er 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 47)*.

Description des variables d'E/S

Le tableau suivant décrit la variable de sortie :

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

Exemple

Reportez-vous à la fonction `HMI_IsFirstMastCycle` *(voir page 32)*.

Chapitre 3
Types de données de la bibliothèque XBT PLCSystem

Présentation

Ce chapitre décrit les **types de données** de la bibliothèque XBT PLCSystem.

2 sortes de **types de données** sont disponibles :

- Les **types de données de variable système** sont utilisés par les **variables système** *(voir page 11)* de la bibliothèque XBT PLCSystem (PLC_R, PLC_W,...).
- Les **types de données de fonction système** sont utilisés par les **fonctions système** *(voir page 25)* de lecture/écriture de la bibliothèque XBT PLCSystem.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

PLC_R_IO_STATUS : codes d'état E/S

Type énumération - Description

Le type de données énumération `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 E/S incorrects détectés.

PLC_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 votre service d'assistance local.
`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. Compiler → Tout nettoyer** **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.
`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 de ces méthodes : **1. Compiler → Tout nettoyer** **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.

Enumérateur	Valeur	Commentaire	Que faire
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_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_BOOT_PROJECT_STATUS : codes d'état de projet de démarrage

Description du type énumération

Le type de données énumération `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.

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 d'énumération `PLC_R_STOP_CAUSE` contient les valeurs suivantes :

Enumérateur	Valeur	Commentaire	Que faire
`PLC_R_STOP_REASON_ UNKNOWN`	0000 hex	La valeur initiale ou la cause de l'arrêt ne sont pas définies.	Contactez votre service d'assistance local en cas de raison d'arrêt non défini.
`PLC_R_STOP_REASON_HW_ WATCHDOG`	0001 hex	Arrêté suite à une horloge de surveillance matérielle.	Contactez votre service d'assistance local.
`PLC_R_STOP_REASON_RESET`	0002 hex	Arrêté suite à une réinitialisation.	Voir les possibilités de réinitialisation dans le chapitre . Une réinitialisation est nécessaire pour entrer en mode Run.
`PLC_R_STOP_REASON_ EXCEPTION`	0003 hex	Arrêté en cas d'exception détectée.	Vérifiez votre application Voir le chapitre .
`PLC_R_STOP_REASON_USER`	0004 hex	Arrêté suite à une requête de l'utilisateur.	Voir le chapitre .
`PLC_R_STOP_REASON_ IECPROGRAM`	0005 hex	Arrêté suite à une requête de commande de programme (par exemple, commande de contrôle avec le paramètre `PLC_W.q_wPLCControl:= PLC_W_COMMAND.PLC_W_ STOP;`).	–
`PLC_R_STOP_REASON_DELETE`	0006 hex	Arrêté suite à une commande de suppression d'application.	Voir le chapitre .
`PLC_R_STOP_REASON_ DEBUGGING`	0007 hex	Arrêté suite au passage en mode de débogage.	–
`PLC_R_STOP_FROM_NETWORK_ REQUEST`	000A hex	Arrêté suite à une requête du réseau (clé USB ou commande PLC_W).	–
`PLC_R_STOP_FROM_INPUT`	000B hex	Arrêt requis par une entrée du contrôleur.	–
`PLC_R_STOP_REASON_RETAIN_ MISMATCH`	000C hex	Passage à l'état ARRETE après un redémarrage et la détection d'une différence dans la définition des variables rémanentes.	Les variables conservées ont été supprimées car elle n'étaient pas référencées dans l'application. Si l'application définit les variables conservées sur leur valeur initiale, la est disponible.

Enumérateur	Valeur	Commentaire	Que faire
PLC_R_STOP_REASON_BOOT_APPLI_MISMATCH	000D hex	Passage à l'état ARRETE après un redémarrage et la détection d'une différence dans l'application de démarrage.	Créez une application de démarrage valide.
PLC_R_STOP_REASON_POWERFAIL	000E hex	Le contrôleur a été arrêté en raison d'une coupure de courant.	Vérifiez la source d'alimentation électrique.

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.

RIGHTBUS_GET_STATUS : codes de paramétrage de fonction HMI_GetRightBusStatus

Descriptions du type énuméré

Le type de données énumération contient les valeurs suivantes :

Enumérateur	Valeur	Description
RIGHTBUS_GET_GEN_STATUS	Hex 00	Paramètre d'un diagnostic de configuration de bus d'extension
RIGHTBUS_GET_STATUS1	Hex 01	Paramètre de diagnostic du module 1 du bus d'extension
RIGHTBUS_GET_STATUS2	Hex 02	Paramètre de diagnostic du module 2 du bus d'extension
RIGHTBUS_GET_STATUS3	Hex 03	Paramètre de diagnostic du module 3 du bus d'extension

NOTE : Pour plus d'informations sur l'utilisation du type de paramètre RIGHTBUS_GET_STATUS, consultez la fonction HMI_GetRightBusStatus *(voir page 27)*.

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.

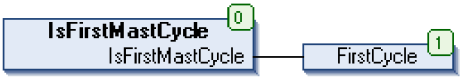
Les fonctions `IsFirstMastCycle` et `SetRTCDrift`, ainsi que le bloc fonction `TON`, sont utilisés à titre d'exemple pour illustrer les mises en œuvre.

Utilisation d'une fonction en langage IL

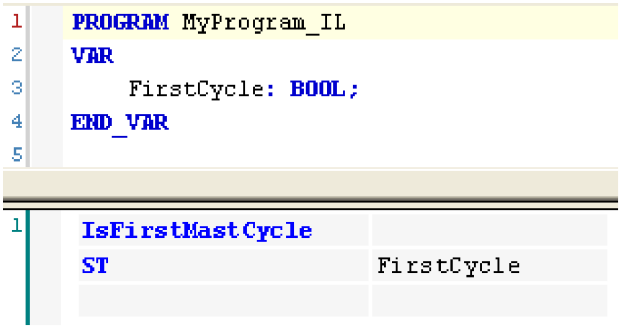
La procédure suivante explique comment insérer une fonction en langage IL :

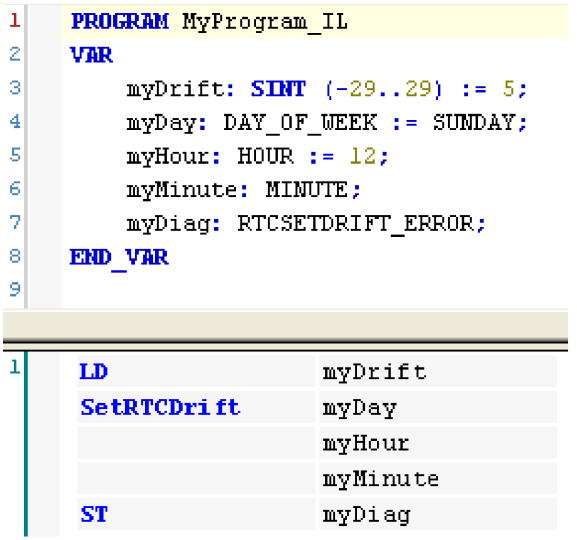
Etape	Action
1	Ouvrez ou créez un POU en langage IL (Instruction List, ou liste d'instructions). **NOTE :** La procédure de création d'un POU n'est pas détaillée ici. Pour plus d'informations, reportez-vous à la section Ajout et appel de POU *(voir SoMachine, Guide de programmation)*.
2	Créez les variables 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`	

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

```
1  PROGRAM MyProgram_IL
2  VAR
3      myDrift: SINT (-29..29) := 5;
4      myDay: DAY_OF_WEEK := SUNDAY;
5      myHour: HOUR := 12;
6      myMinute: MINUTE;
7      myDiag: RTCSETDRIFT_ERROR;
8  END_VAR
9
```

```
1  LD          myDrift
   SetRTCDrift myDay
               myHour
               myMinute
   ST          myDiag
```

Utilisation d'un bloc fonction en langage IL

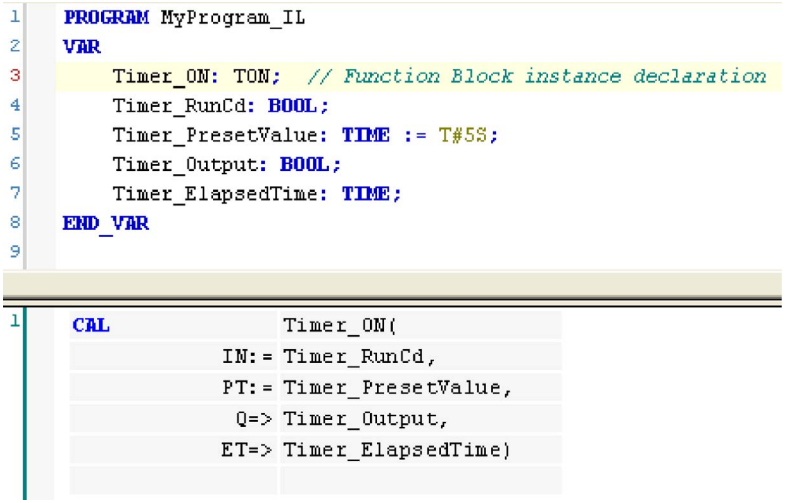
La procédure suivante explique comment insérer un bloc fonction en langage IL :

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	

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

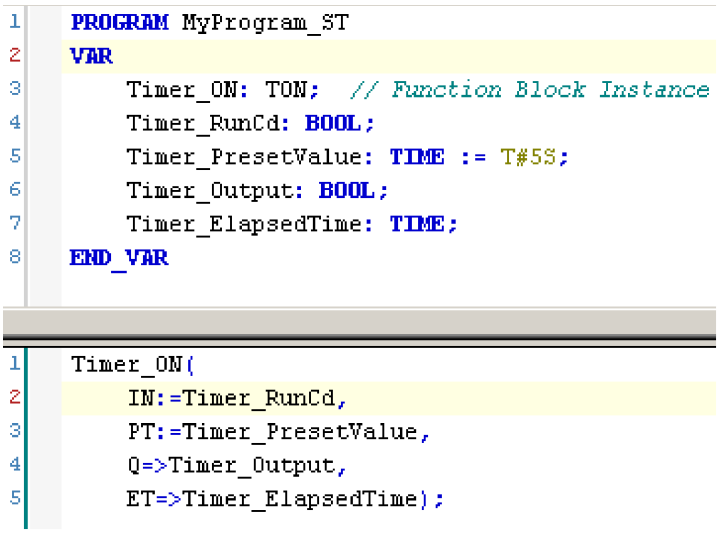
La procédure suivante explique comment insérer un bloc fonction en langage ST :

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

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

bus d'extension

Bus de communication électronique entre des modules d'E/S d'extension et un contrôleur.

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

DWORD

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

E

E/S

Entrée/sortie

E/S numérique

(*Entrée/sortie numérique*) Connexion de circuit individuelle au niveau du module électronique qui correspond directement à un bit de table de données. Ce bit de table de données contient la valeur du signal au niveau du circuit d'E/S. Il permet à la logique de contrôle un accès numérique aux valeurs d'E/S.

élément

Raccourci pour l'élément d'un ARRAY.

entrée analogique

Convertit les niveaux de tension ou de courant reçus en valeurs numériques. Vous pouvez stocker et traiter ces valeurs au sein du contrôleur logique.

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.

fonction

Unité de programmation possédant 1 entrée et renvoyant 1 résultat immédiat. Contrairement aux blocs fonction (FBs), une fonction est appelée directement par son nom (et non via une instance), elle n'a pas d'état persistant d'un appel au suivant et elle peut être utilisée comme opérande dans d'autres expressions de programmation.

Exemples : opérateurs booléens (AND), calculs, conversion (BYTE_TO_INT).

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*)

HMI

Acronyme de *human machine interface*, interface homme-machine (IHM). Interface opérateur (généralement graphique) permettant le contrôle d'équipements industriels par l'homme.

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.

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.

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.

Modbus

Protocole qui permet la communication entre de nombreux équipements connectés au même réseau.

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

réseau

Système d'équipements interconnectés qui partageant un chemin de données et un protocole de communications communs.

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.

V

variable

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

variable localisée

Voir *variable non localisée*

variable non localisée

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

X

XBT

Tout panneau graphique Magelis XBT.

Index