Unity Pro
Système
Bibliothèque de blocs

07/2018

33002540.19

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

Lorsque des équipements sont utilisés pour des applications présentant des exigences 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.

AVANT DE COMMENCER

N'utilisez pas ce produit sur les machines non pourvues de protection efficace du point de fonctionnement. L'absence de ce type de protection sur une machine présente un risque de blessures graves pour l'opérateur.

⚠ AVERTISSEMENT

EQUIPEMENT NON PROTEGE

- N'utilisez pas ce logiciel ni les automatismes associés sur des appareils non équipés de protection du point de fonctionnement.
- N'accédez pas aux machines pendant leur fonctionnement.

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

Cet automatisme et le logiciel associé permettent de commander des processus industriels divers. Le type ou le modèle d'automatisme approprié pour chaque application dépendra de facteurs tels que la fonction de commande requise, le degré de protection exigé, les méthodes de production, des conditions inhabituelles, la législation, etc. Dans certaines applications, plusieurs processeurs seront nécessaires, notamment lorsque la redondance de sauvegarde est requise.

Vous seul, en tant que constructeur de machine ou intégrateur de système, pouvez connaître toutes les conditions et facteurs présents lors de la configuration, de l'exploitation et de la maintenance de la machine, et êtes donc en mesure de déterminer les équipements automatisés, ainsi que les sécurités et verrouillages associés qui peuvent être utilisés correctement. Lors du choix de l'automatisme et du système de commande, ainsi que du logiciel associé pour une application particulière, vous devez respecter les normes et réglementations locales et nationales en vigueur. Le document National Safety Council's Accident Prevention Manual (reconnu aux Etats-Unis) fournit également de nombreuses informations utiles.

Dans certaines applications, telles que les machines d'emballage, une protection supplémentaire, comme celle du point de fonctionnement, doit être fournie pour l'opérateur. Elle est nécessaire si les mains ou d'autres parties du corps de l'opérateur peuvent entrer dans la zone de point de pincement ou d'autres zones dangereuses, risquant ainsi de provoquer des blessures graves. Les produits logiciels seuls, ne peuvent en aucun cas protéger les opérateurs contre d'éventuelles blessures. C'est pourquoi le logiciel ne doit pas remplacer la protection de point de fonctionnement ou s'y substituer.

Avant de mettre l'équipement en service, assurez-vous que les dispositifs de sécurité et de verrouillage mécaniques et/ou électriques appropriés liés à la protection du point de fonctionnement ont été installés et sont opérationnels. Tous les dispositifs de sécurité et de verrouillage liés à la protection du point de fonctionnement doivent être coordonnés avec la programmation des équipements et logiciels d'automatisation associés.

NOTE : La coordination des dispositifs de sécurité et de verrouillage mécaniques/électriques du point de fonctionnement n'entre pas dans le cadre de cette bibliothèque de blocs fonction, du Guide utilisateur système ou de toute autre mise en œuvre référencée dans la documentation.

DEMARRAGE ET TEST

Avant toute utilisation de l'équipement de commande électrique et des automatismes en vue d'un fonctionnement normal après installation, un technicien qualifié doit procéder à un test de démarrage afin de vérifier que l'équipement fonctionne correctement. Il est essentiel de planifier une telle vérification et d'accorder suffisamment de temps pour la réalisation de ce test dans sa totalité.

⚠ AVERTISSEMENT

RISQUES INHERENTS AU FONCTIONNEMENT DE L'EQUIPEMENT

- Assurez-vous que toutes les procédures d'installation et de configuration ont été respectées.
- Avant de réaliser les tests de fonctionnement, retirez tous les blocs ou autres cales temporaires utilisés pour le transport de tous les dispositifs composant le système.
- Enlevez les outils, les instruments de mesure et les débris éventuels présents sur l'équipement.

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

Effectuez tous les tests de démarrage recommandés dans la documentation de l'équipement. Conservez toute la documentation de l'équipement pour référence ultérieure.

Les tests logiciels doivent être réalisés à la fois en environnement simulé et réel.

Vérifiez que le système entier est exempt de tout court-circuit et mise à la terre temporaire non installée conformément aux réglementations locales (conformément au National Electrical Code des Etats-Unis, par exemple). Si des tests diélectriques sont nécessaires, suivez les recommandations figurant dans la documentation de l'équipement afin d'éviter de l'endommager accidentellement.

Avant de mettre l'équipement sous tension :

- Enlevez les outils, les instruments de mesure et les débris éventuels présents sur l'équipement.
- Fermez le capot du boîtier de l'équipement.
- Retirez toutes les mises à la terre temporaires des câbles d'alimentation entrants.
- Effectuez tous les tests de démarrage recommandés par le fabricant.

FONCTIONNEMENT ET REGLAGES

Les précautions suivantes sont extraites du document NEMA Standards Publication ICS 7.1-1995 (la version anglaise prévaut) :

- Malgré le soin apporté à la conception et à la fabrication de l'équipement ou au choix et à l'évaluation des composants, des risques subsistent en cas d'utilisation inappropriée de l'équipement.
- Il arrive parfois que l'équipement soit déréglé accidentellement, entraînant ainsi un fonctionnement non satisfaisant ou non sécurisé. Respectez toujours les instructions du fabricant pour effectuer les réglages fonctionnels. Les personnes ayant accès à ces réglages doivent connaître les instructions du fabricant de l'équipement et les machines utilisées avec l'équipement électrique.
- Seuls ces réglages fonctionnels, requis par l'opérateur, doivent lui être accessibles. L'accès aux autres commandes doit être limité afin d'empêcher les changements non autorisés des caractéristiques de fonctionnement.

A propos de ce manuel

Présentation

Objectif du document

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

Champ d'application

Ce document est applicable à Unity Pro 13.1 ou version ultérieure.

Document(s) à consulter

Titre de documentation	Référence
Unity Pro - Langages de programmation et structure - Manuel de référence	35006144 (anglais), 35006145 (français), 35006146 (allemand), 35013361 (italien), 35006147 (espagnol), 35013362 (chinois)
Unity Pro - Bits et mots système - Manuel de référence	EIO0000002135 (anglais), EIO0000002136 (français), EIO0000002137 (allemand), EIO0000002138 (italien), EIO0000002139 (espagnol), EIO0000002140 (chinois)

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

Partie I
Informations générales

Présentation

Cette section contient des informations générales concernant la bibliothèque système.

NOTE : Pour obtenir une description détaillée des objets système (%S et %SW), reportez-vous au document *Unity Pro - Bits et mots système - Manuel de référence*.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 1
Types de module et leur utilisation

Vue d'ensemble

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

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Types de bloc

Types de bloc

Types de bloc

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

Une différence est faite entre les types de bloc suivants :

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

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

Fonction élémentaire

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

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

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

⚠ ATTENTION

COMPORTEMENT INATTENDU DE L'EQUIPEMENT

Pour Unity Pro V4.0 et les versions antérieures, n'utilisez pas de liens pour connecter les sorties des blocs fonction lorsque votre application repose sur des données de sortie persistantes d'un bloc EF.

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

NOTE : avec Unity Pro V4.0 et les versions antérieures, la désactivation d'un EF (EN=0) entraîne la réinitialisation des liens associés à ses entrées/sorties. Pour transférer l'état du signal, n'utilisez pas de lien. Une variable doit être connectée à la sortie de la fonction élémentaire et être utilisée pour connecter l'entrée de l'élément. Avec Unity Pro V4.1 et les versions ultérieures, vous pouvez maintenir les liens de sortie, même si un EF est désactivé en activant l'option **Maintenir les liens de sortie sur les EF désactivés (EN=0)** par l'intermédiaire du menu **Outils → Programme → Langues → Commun**.

Bloc fonction élémentaire

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

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

Bloc fonction dérivé

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

Procédure

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

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

Les procédures ne renvoient aucune valeur.

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

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

⚠ ATTENTION

COMPORTEMENT INATTENDU DE L'EQUIPEMENT

Pour Unity Pro V4.0 et les versions antérieures, n'utilisez pas de liens pour connecter les sorties des blocs fonction lorsque votre application repose sur des données de sortie persistantes d'un bloc EF.

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

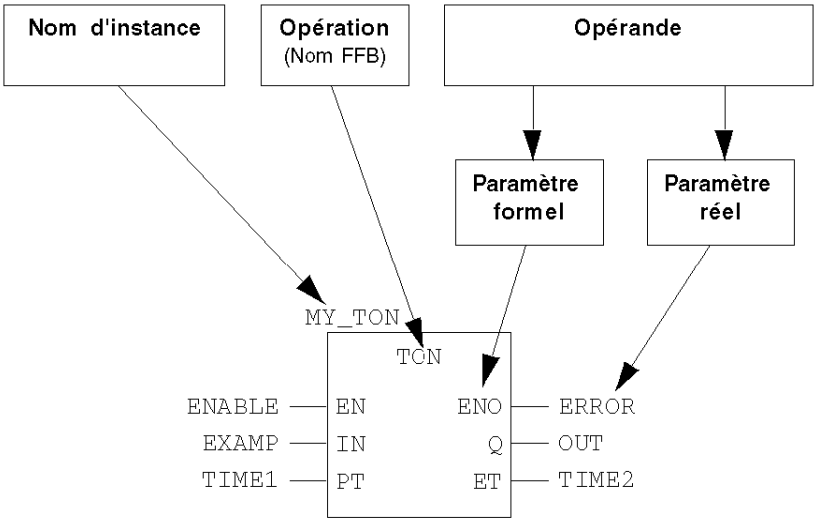
NOTE : avec Unity Pro V4.0 et les versions antérieures, la désactivation d'un EF (EN=0) entraîne la réinitialisation des liens associés à ses entrées/sorties. Pour transférer l'état du signal, n'utilisez pas de lien. Une variable doit être connectée à la sortie de l'EF et être utilisée pour connecter l'entrée de l'élément. Avec Unity Pro V4.1 et les versions ultérieures, vous pouvez maintenir les liens de sortie, même si un EF est désactivé en activant l'option **Maintenir les liens de sortie sur les EF désactivés (EN=0)** par l'intermédiaire du menu **Outils → Programme → Langues → Commun**.

Structure d'un FFB

Structure

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

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



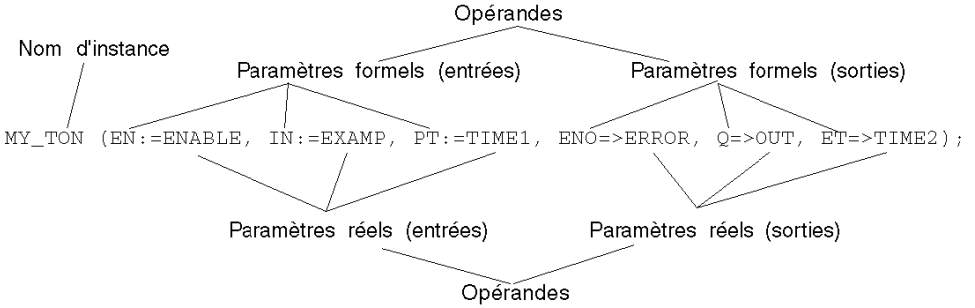
⚠ ATTENTION

COMPORTEMENT INATTENDU DE L'APPLICATION

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

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

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



Opération

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

Opérande

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

Paramètres formels et réels

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

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

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

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

Appel de FFB dans le langage IL/ST

Les FFB peuvent être appelés de deux manières dans les langages textuels IL et ST : formelle ou informelle. Pour obtenir des informations détaillées, reportez-vous au chapitre *Langage de programmation (voir Unity Pro, Langages de programmation et structure, Manuel de référence)*.

Exemple d'un appel de fonction formel :

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

Exemple d'un appel de fonction informel :

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

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

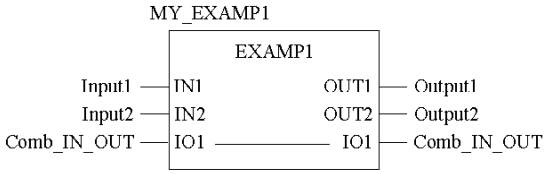
Variable VAR_IN_OUT

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

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

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

Bloc fonction avec la variable `VAR_IN_OUT` dans le langage FBD :



Bloc fonction avec la variable `VAR_IN_OUT` dans le langage ST :

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

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

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

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

- Les liaisons graphiques permettent uniquement de relier des sorties `VAR_IN_OUT` à des entrées `VAR_IN_OUT`.
- Seule une liaison graphique peut être associée à une entrée/sortie `VAR_IN_OUT`.
- Des variables ou des composantes de variables différentes peuvent être reliées à l'entrée `VAR_IN_OUT` et à la sortie `VAR_IN_OUT`. Dans ce cas, la valeur de la variable ou de la composante de variable en entrée est copiée dans la variable ou la composante de variable en sortie.
- Vous ne pouvez pas utiliser des négations sur les entrées/sorties `VAR_IN_OUT`.
- Une combinaison de variable/adresse et de liaisons graphiques n'est pas possible pour les sorties `VAR_IN_OUT`.

EN et ENO

Description

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

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

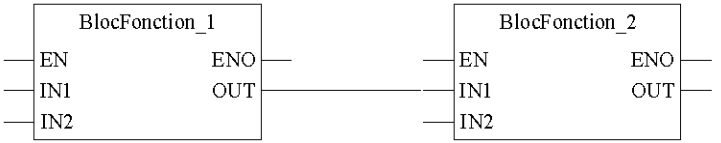
Si la valeur de EN est déjà réglée sur 1, lors de l'appel de FFB, les algorithmes définis par FFB sont exécutés. Une fois les algorithmes exécutés, la valeur de la sortie ENO est réglée sur « 1 ». Si certaines conditions d'erreur sont détectées durant l'exécution de ces algorithmes, ENO est réglé sur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

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

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

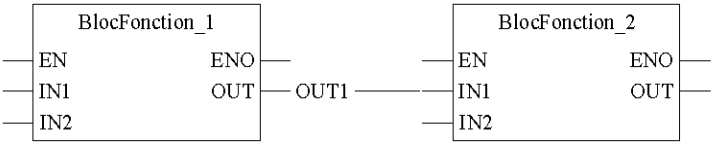
Si la valeur de ENO est réglée sur 0 (car EN = 0 ou en raison d'une condition d'erreur détectée lors de l'exécution ou de l'échec de l'exécution des algorithmes) :

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



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

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



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

- Fonctions/procédures

⚠ ATTENTION
COMPORTEMENT INATTENDU DE L'EQUIPEMENT Pour Unity Pro V4.0 et les versions antérieures, n'utilisez pas de liens pour connecter les sorties des blocs fonction lorsque votre application repose sur des données de sortie persistantes d'un bloc EF. **Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.**

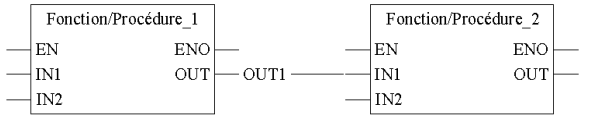
NOTE : avec Unity Pro V4.0 et les versions antérieures, la désactivation d'un EF (EN=0) entraîne la réinitialisation des liens associés à ses entrées/sorties. Pour transférer l'état du signal, n'utilisez pas de lien. Une variable doit être connectée à la sortie de l'EF et être utilisée pour connecter l'entrée de l'élément. Avec Unity Pro V4.1 et les versions ultérieures, vous pouvez maintenir les liens de sortie, même si un EF est désactivé en activant l'option **Maintenir les liens de sortie sur les EF désactivés (EN=0)** par l'intermédiaire du menu **Outils → Programme → Langues → Commun**.

Comme spécifié dans la norme CEI 61131-3, les sorties de fonctions désactivées (entrée `EN` réglée sur « 0 ») ne sont pas définies. (Cette caractéristique s'applique également aux procédures.)
Voici une explication des états des sorties dans un tel cas :

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

  Si l'entrée `EN` de `Function/Procedure_1` est réglée sur 0, la connexion de sortie `OUT` de `Function/Procedure_1` est également réglée sur 0.

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



  Si l'entrée `EN` de `Function/Procedure_1` est réglée sur 0, la connexion de sortie `OUT` de `Function/Procedure_1` est également réglée sur 0. La variable `OUT1` présente sur la même broche conserve son état précédent ou peut être modifiée de manière externe sans incidence sur la connexion. La variable et la liaison sont enregistrées indépendamment l'une de l'autre.

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

Appel de FFB conditionnel/inconditionnel

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

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

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

⚠ ATTENTION

EQUIPEMENT D'APPLICATION IMPREVU

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

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

Remarque concernant les langages IL et ST

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

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

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

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

Chapitre 2
Disponibilité du bloc sur les différentes plates-formes matérielles

Disponibilité des blocs sur les différentes plates-formes matérielles

Introduction

Tous les blocs ne sont pas disponibles sur toutes les plates-formes matérielles. Les blocs disponibles sur votre plate-forme matérielle sont indiqués dans les tableaux ci-dessous.

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

Evénements

Disponibilité des blocs :

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
HALT	Procédure	+	+	+	+	+
ITCNTRL	Procédure	+	+	+	+	+
MASKEVT	Procédure	+	+	+	+	+
UNMASKEVT	Procédure	+	+	+	+	+
+ Oui **-** Non						

Redondance d'UC

Disponibilité des blocs :

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
HSBY_BUILD_OFFLINE	EFB	-	BME H58 ••••	-	-	-
HSBY_RD	EFB	-	-	140 CPU 67• 60/ 140 CPU 67• 61	-	-
HSBY_ST	EFB	-	-	140 CPU 67• 60/ 140 CPU 67• 61	-	-
HSBY_SWAP *(voir page 137)*	DFB	-	-	140 CPU 67• 60/ 140 CPU 67• 61	-	TSX H57 ••
+ Oui **-** Non						

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
HSBY_WR	EFB	-	-	140 CPU 67• 60/ 140 CPU 67• 61	-	-
REV_XFER	EFB	-	-	140 CPU 67• 60/ 140 CPU 67• 61	-	-
+ Oui **-** Non						

Gestion SFC

Disponibilité des blocs :

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
CLEARCHART	EF	+	+	+	+	+
FREEZECHART	EF	+	+	+	+	+
INITCHART	EF	+	+	+	+	+
RESETSTEP	Procédure	+	+	+	+	+
SETSTEP	Procédure	+	+	+	+	+
SFCCNTRL	EFB	+	+	+	+	+
SFC_RESTORE	EFB	-	+	+	+	+
+ Oui **-** Non						

Horloge système

Disponibilité des blocs :

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
FREERUN	EF	+	+	+	+	TSX P57 5••
GET_TS_EVT_M	EFB	+	+	-	-	-
GET_TS_EVT_Q	EFB	-	-	+(2)	-	-
PTC	Procédure	+	+	+	+	+
RRTC_DT	Procédure	+	+	+	-	+
R_NTPC	Procédure	+	+	+	+	+
RRTC_DT_MS	Procédure	+	+	+(1)	+	+
SCHEDULE	Procédure	+	+	+	+	+
WRTC_DT	Procédure	+	+	+	+	+
+ Oui **-** Non **(1)** Non disponible pour les anciens systèmes Quantum. **(2)** Station d'E/S distantes Ethernet M340 adressée à partir d'une plate-forme Quantum.						

Particularités système

Disponibilité des blocs :

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
FORCE_BIT	Procédure	+	+	+	+	+
IS_BIT_FORCED	EF	+	+	+	+	+
IS_PAR_CON	EF	+	+	+	+	+
PRJ_VERS	EFB	+	+	+	+	+
SIG_CHECK	EF	+	+	-	-	-
SIG_WRITE	EF	+	+	-	-	-
UNFORCE_BIT	Procédure	+	+	+	+	+
+ Oui - Non						

Gestion des fichiers de carte mémoire

Disponibilité des blocs :

Nom du bloc	Type de bloc	M340	M580	Quantum	Momentum	Premium
CREATE_FILE	EFB	+	+	-	-	-
OPEN_FILE	EFB	+	+	-	-	-
SET_FILE_ATTRIBUTES	EFB	+	+	-	-	-
GET_FILE_INFO	EFB	+	+	-	-	-
GET_FREESIZE	EFB	+	+	-	-	-
SEEK_FILE	EFB	+	+	-	-	-
WR_DATA_TO_FILE	EFB	+	+	-	-	-
RD_FILE_TO_DATA	EFB	+	+	-	-	-
CLOSE_FILE	EFB	+	+	-	-	-
DELETE_FILE	EFB	+	+	-	-	-
WRITE_V_PCMCIA	Procédure	-	-	+	-	+
WRITE_U_PCMCIA	Procédure	-	-	+	-	+
READ_V_PCMCIA	Procédure	-	-	+	-	+
READ_U_PCMCIA	Procédure	-	-	+	-	+
+ Oui - Non						

Partie II
Gestion des fichiers

Description

Cette section décrit les fonctions et blocs fonction élémentaires de la famille de `gestion des fichiers`.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 3
Mise en œuvre de la gestion des fichiers

Introduction

Ce chapitre décrit la mise en œuvre de la gestion des fichiers sur les cartes mémoire.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Gestion des fichiers sur les cartes mémoire

Présentation

Les tableaux ci-dessous présentent les méthodes d'utilisation de la carte mémoire, une fois celle-ci formatée.

NOTE : Les opérations présentées ci-dessous ne concernent que la gamme Modicon M580 et M340.

Création d'un fichier

Le tableau ci-dessous indique la méthode pour créer un fichier et y ajouter des informations.

Etape	Action	Fonction associée	Commentaire
1	Créer le fichier	CREATE_FILE *(voir page 43)*	Renvoie l'identifiant (descripteur de fichier) qui sera utilisé par les autres fonctions.
2	Définir les attributs du fichier	SET_FILE_ATTRIBUTES *(voir page 51)*	Facultatif
3	Afficher l'espace disponible	GET_FREESIZE *(voir page 57)*	Facultatif Cette fonction peut être appelée à tout moment.
4	Positionnement dans le fichier	SEEK_FILE *(voir page 59)*	
5	Écrire des données dans le fichier	WR_DATA_TO_FILE *(voir page 63)*	
6	Fermer le fichier	CLOSE_FILE *(voir page 69)*	

Ouverture d'un fichier en lecture/écriture

Le tableau ci-dessous présente les méthodes pour ouvrir un fichier afin d'y lire ou d'y écrire des données.

Etape	Action	Fonction associée	Commentaire
1	Ouvre le fichier	OPEN_FILE *(voir page 47)*	Renvoie l'identifiant (descripteur de fichier) qui sera utilisé par les autres fonctions.
2	Récupérer les attributs du fichier	GET_FILE_INFO *(voir page 53)*	Facultatif
3	Positionnement dans le fichier	SEEK_FILE *(voir page 59)*	Facultatif
4	Afficher l'espace disponible	GET_FREESIZE *(voir page 57)*	Facultatif Cette fonction peut être appelée à tout moment.
5	Écrire des données	WR_DATA_TO_FILE *(voir page 63)*	

Etape	Action	Fonction associée	Commentaire
6	Positionnement dans le fichier	SEEK_FILE *(voir page 59)*	
7	Lire des données	RD_FILE_TO_DATA *(voir page 65)*	
8	Fermer le fichier	CLOSE_FILE *(voir page 69)*	

Suppression d'un fichier

Le tableau ci-dessous présente la méthode pour supprimer un fichier

Etape	Action	Fonction associée
1	Supprime le fichier	DELETE_FILE *(voir page 71)*

Limitations des fonctions

Le nombre de fichiers ouvrables simultanément est limité à 16.

Le nombre de fichiers pouvant être créés dépend de la capacité de stockage de la carte mémoire.

Les fonctions ne sont utilisables que dans la tâche MAST.

La carte mémoire doit posséder des droits DATA-STORAGE pour permettre l'utilisation de ces fonctions.

Modes opératoires des blocs fonction

Présentation

Les différents modes opératoires des blocs fonction dépendent des paramètres suivants :

- REQ
- DONE
- ERROR
- STATUS

Description des paramètres

Tous les EFB de cette famille s'exécutent de manière asynchrone par rapport à l'application, et ce sans exception (le service peut être exécuté sur plusieurs cycles). Avant de commencer une nouvelle opération, les bits DONE et ERROR de l'opération précédente doivent être testés, et le mot STATUS doit être vérifié si le bit ERROR est à 1.

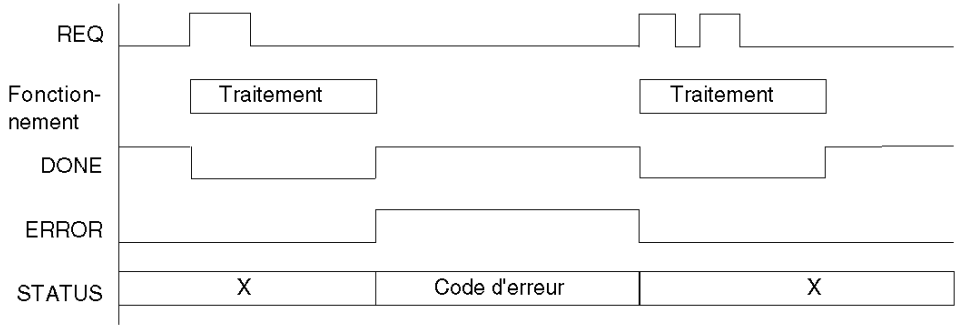
Tous les EF de gestion de fichiers sont configurés comme suit :

Paramètre	Type	Description
REQ	EBOOL	Démarre une opération complète sur un front montant.
DONE	BOOL	La valeur 1est définie lorsque l'opération est terminée. Cette sortie est remise à zéro lorsque le bloc est activé par un front montant sur l'entrée REQ.
ERROR	BOOL	La valeur 1 est définie en cas d'erreur lors de l'exécution de l'opération. Cette sortie est remise à zéro lorsque le bloc est activé par un front montant sur l'entrée REQ.
STATUS	WORD	Code d'erreur. *(voir page 73)* Ce mot n'est représentatif que lorsqu'une erreur a eu lieu (ERROR = 1).

Les EF du système de fichiers ne sont utilisables que dans la tâche MAST. **Ils sont tous exécutés de façon asynchrone par rapport à la tâche, sans exception**. Si les EF sont appelés par une autre tâche, le code d'erreur suivant est immédiatement renvoyé : `SDCARD_WRONG_USER_TASK`.

Illustration

La figure ci-dessous illustre le fonctionnement des paramètres des blocs :



Descripteur de fichier

Présentation

Un descripteur de fichier est une valeur abstraite qui contient toutes les informations nécessaires à l'utilisation d'un fichier :

- un pointeur vers le fichier,
- les droits d'accès à ce fichier,
- les conditions d'accès au fichier (lecture, écriture ou lecture/écriture),
- la position courante dans le fichier, etc.

Fonctionnement

Il est possible d'utiliser un ou plusieurs descripteurs simultanément ou séquentiellement sur un fichier afin de créer des accès multiples (lecture, écriture, lecture/écriture) sur une ou plusieurs positions (début, fin, milieu, etc.).

Un descripteur de fichier est :

- affecté (via `OPEN_FILE`, `CREATE_FILE`) ;
- utilisé (via `RD_FILE_TO_DATA`, `WR_DATA_TO_FILE`, `SEEK_FILE`, `GET_FILE_INFO`) ;
- libéré (via `CLOSE_FILE`).

Il permet d'identifier une transaction vers un fichier. Plusieurs transactions sont possibles sur un même fichier.

Instance EFB

L'instance EFB est considérée comme un outil qui applique une action à un fichier dans un contexte donné. Elle est représentée par le descripteur de fichier.

Avec la même instance EFB, vous pouvez travailler sur plusieurs transactions séquentielles par le biais d'un ou plusieurs descripteurs de fichier.

Ces descripteurs permettent de travailler sur le même fichier ou des fichiers différents.

NOTE : le descripteur de fichier est perdu et vous ne pouvez plus fermer le fichier. Dans ce cas, le nombre maximal de fichiers ouverts est atteint, car vous ne pouvez pas fermer les fichiers déjà ouverts. Un message d'erreur s'affiche à chaque nouvelle ouverture.

⚠ ATTENTION
COMPORTEMENT INATTENDU DE L'APPLICATION N'ouvrez pas un fichier avec la même variable comme descripteur lorsque le fichier n'est pas fermé. **Le non-respect de ces instructions peut provoquer des blessures ou des dommages matériels.**

NOTE : si le code d'erreur `FILE_DELETED` (code d'erreur 17) s'affiche, vous devez fermer le fichier correspondant pour libérer le descripteur de fichier associé.

Chapitre 4
Formatage d'une carte mémoire

Effacement de la carte mémoire

Effacement

Il est possible d'effacer la carte mémoire à l'aide du mot `%SW93` via la table d'animation Unity Pro (mais en principe cela n'est pas nécessaire). L'effacement ou le formatage de la carte sur un PC à l'aide de commandes Windows peut rendre la carte inutilisable.

Si vous effacez la carte mémoire, le répertoire par défaut */DataStorage/* est créé.

Avant l'effacement avec Unity Pro, l'utilisateur doit télécharger une application par défaut. Sinon, le mot système `%SW93` n'existe pas.

NOTE : L'effacement n'est possible que si l'automate est en mode STOP :

Mode de fonctionnement `%SW93`

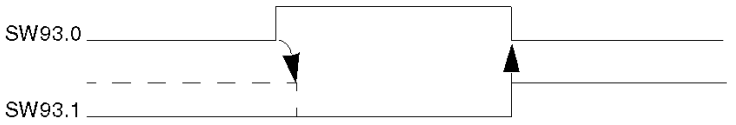
La commande d'effacement et l'état de la carte mémoire sont fournis par le mot système `%SW93`.

`SW93.0` : un front montant lance l'effacement.

`SW93.1` : fournit l'état de la partition du fichier système :

- 0 : partition du fichier système non valide (format incorrect, effacement en cours, etc.),
- 1 : partition du fichier système valide.

L'image ci-dessous indique les états des bits `%SW93.0` et `%SW93.1`.



Enregistrer

Il est conseillé à l'utilisateur de sauvegarder la partition fichier (données stockées, pages Web fabricant ou utilisateur) avec le client FTP, qui peut être lancé depuis Unity Pro.

L'utilisateur peut sauvegarder les pages Web avec FactoryCast ou WAD (Web Application Designer). Il peut ainsi restaurer ces pages en cas de problème.

Unity Loader permet également de sauvegarder des fichiers sur la carte mémoire.

Chapitre 5
CREATE_FILE : création d'un fichier de stockage de données

CREATE_FILE : création d'un fichier de stockage de données

Description de la fonction

La fonction CREATE_FILE permet de créer un fichier appelé *FILENAME* (s'il n'existe pas) et de l'ouvrir avec l'indicateur ModeFlag spécifié :

- O_RDONLY : ouverture en lecture seule (valeur 0)
- O_WRONLY : ouverture en écriture seule (valeur 1)
- O_RDWR : ouverture en lecture-écriture (valeur 2)

Cet indicateur est remis à 0 dès que le fichier est fermé.

Ce fichier est créé dans le sous-répertoire */DataStorage/*.

Si vous utilisez le mode simulateur, le fichier est créé dans le dossier utilisateur temporaire.

Pour accéder au fichier par FTP, consultez cette page *(voir Unity Pro, Modes de marche)*.

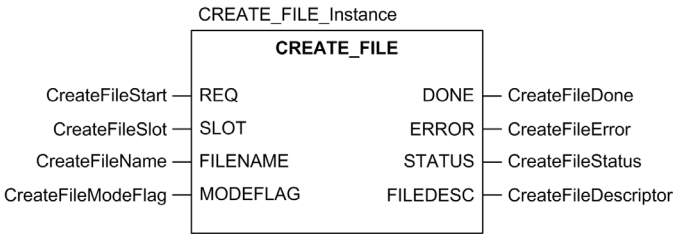
NOTE : à chaque ouverture d'un fichier, le paramètre FileDescriptor renvoie une valeur aléatoire. Celui-ci sera utilisé lors de la fermeture du fichier. Si vous ouvrez un nouveau fichier sans fermer le précédent, le paramètre FileDescriptor renvoie une nouvelle valeur et vous ne pourrez pas fermer le fichier précédent. Il est donc conseillé d'enregistrer le paramètre FileDescriptor après avoir ouvert un fichier.
Lorsque le nombre maximum de fichiers ouverts est atteint, vous recevez un message d'erreur si vous tentez d'ouvrir un autre fichier.

NOTE : si le fichier existe déjà, le comportement de CREATE_FILE est similaire à celui de OPEN_FILE.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

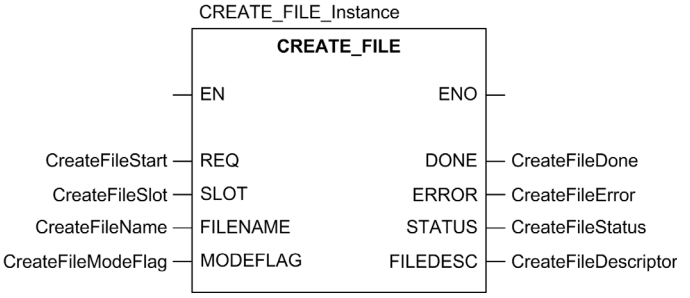
Représentation en FBD

La représentation en FBD de la fonction CREATE_FILE est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `CREATE_FILE` est la suivante :



Représentation en IL

Représentation :

```
CAL CREATE_FILE_Instance (REQ:=CreateFileStart, SLOT:=CreateFileSlot,
FILENAME:=CreateFileName, MODEFLAG:=CreateFileModeFlag,
DONE=>CreateFileDone, ERROR=>CreateFileError, STATUS=>CreateFileStatus,
FILEDESC=>CreateFileDescriptor)
```

Représentation en ST

Représentation :

```
CREATE_FILE_Instance (REQ:=CreateFileStart, SLOT:=CreateFileSlot,
FILENAME:=CreateFileName, MODEFLAG:=CreateFileModeFlag,
DONE=>CreateFileDone, ERROR=>CreateFileError, STATUS=>CreateFileStatus,
FILEDESC=>CreateFileDescriptor);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
SLOT	DINT	Ce paramètre indique l'adresse de l'emplacement de la carte mémoire (toujours 0).
FILENAME	STRING	Ce paramètre indique le nom du fichier à créer. Nombre maximum de caractères : 64 Les caractères « ? » ou « * » ne sont pas autorisés.
MODEFLAG	INT	Ce paramètre spécifie le mode d'ouverture. ● O_RDONLY (0) : ouvert en lecture seule ● O_WRONLY (1) : ouvert en écriture seule ● O_RDWR (2) : ouvert en lecture-écriture

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre permet à la fonction DATA_STORAGE d'accéder au fichier.

Description des paramètres communs *(voir page 36)*.

Chapitre 6
OPEN_FILE : ouverture d'un fichier

OPEN_FILE : ouverture d'un fichier

Description de la fonction

La fonction `OPEN_FILE` permet d'ouvrir un fichier appelé FileName (situé dans le répertoire *\DataStorage*). Ce fichier doit exister. Le mode argument a l'une des valeurs suivantes : `O_RDONLY`, `O_WRONLY` ou `O_RDRW`.

Le fichier s'ouvre en mode binaire. Cela signifie que les caractères spéciaux, comme CTRL-Z (marqueur de fin de fichier), les retours chariot et les sauts de ligne, ne sont pas convertis.

Vous pouvez ouvrir jusqu'à 16 fichiers simultanément.

Si vous utilisez le mode simulateur, le fichier est ouvert dans le dossier utilisateur temporaire.

Pour accéder au fichier par FTP, consultez cette page *(voir Unity Pro, Modes de marche)*.

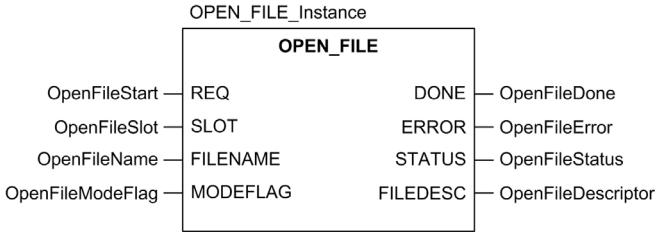
NOTE : à chaque ouverture d'un fichier, le paramètre FileDescriptor renvoie une valeur aléatoire. Celui-ci sera utilisé lors de la fermeture du fichier. Si vous ouvrez un nouveau fichier sans fermer le précédent, le paramètre FileDescriptor renvoie une nouvelle valeur et vous ne pourrez pas fermer le fichier précédent. Il est donc conseillé d'enregistrer le paramètre FileDescriptor après avoir ouvert un fichier.
Lorsque le nombre maximum de fichiers ouverts est atteint, vous recevez un message d'erreur si vous tentez d'ouvrir un autre fichier.

NOTE : il est toujours possible de modifier l'état d'un fichier dans le système de fichiers, même si ce fichier est déjà ouvert. L'état d'un fichier (lecture seule, lecture-écriture) n'est pas évalué à l'ouverture mais lors de la modification de son contenu. (Par exemple, il est possible d'ouvrir un fichier en lecture seule en mode Lecture-Ecriture sans générer d'erreur). Tant que le fichier est en lecture seule dans le système de fichiers, toute demande d'écriture génère une erreur (code 14), même si le fichier a été ouvert en mode Lecture-Ecriture. Dès que le fichier reprend son état Lecture-Ecriture, les demandes d'écriture sont traitées sans qu'il soit nécessaire de fermer et de rouvrir le fichier.

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

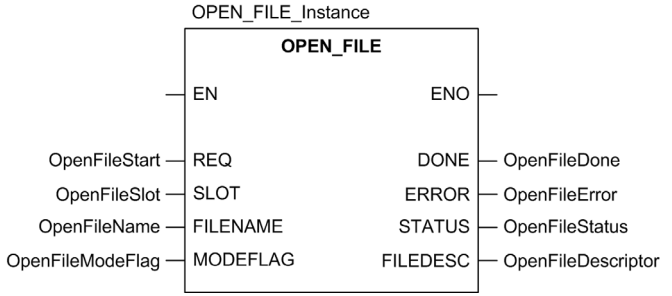
Représentation en FBD

La représentation en FBD de la fonction `OPEN_FILE` est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `OPEN_FILE` est la suivante :



Représentation en IL

Représentation :

```
CAL OPEN_FILE_Instance (REQ:=OpenFileStart, SLOT:=OpenFileSlot,
FILENAME:=OpenFileName, MODEFLAG:=OpenFileModeFlag, DONE=>OpenFileDone,
ERROR=>OpenFileError, STATUS=>OpenFileStatus,
FILEDESC=>OpenFileDescriptor)
```

Représentation en ST

Représentation :

```
OPEN_FILE_Instance (REQ:=OpenFileStart, SLOT:=OpenFileSlot,
FILENAME:=OpenFileName, MODEFLAG:=OpenFileModeFlag, DONE=>OpenFileDone,
ERROR=>OpenFileError, STATUS=>OpenFileStatus,
FILEDESC=>OpenFileDescriptor);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
SLOT	DINT	Ce paramètre indique l'adresse de l'emplacement de la carte mémoire (toujours 0).
FILENAME	STRING	Ce paramètre indique le nom du fichier à ouvrir.
MODEFLAG	INT	Ce paramètre spécifie le mode d'ouverture. • O_RDONLY (0) : ouvert en lecture seule • O_WRONLY (1) : ouvert en écriture seule • O_RDWR (2) : ouvert en lecture-écriture

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre permet à la fonction DATA_STORAGE d'accéder au fichier.

Description des paramètres communs *(voir page 36)*.

Chapitre 7
SET_FILE_ATTRIBUTES : définition des attributs du fichier

SET_FILE_ATTRIBUTES : configuration des attributs du fichier

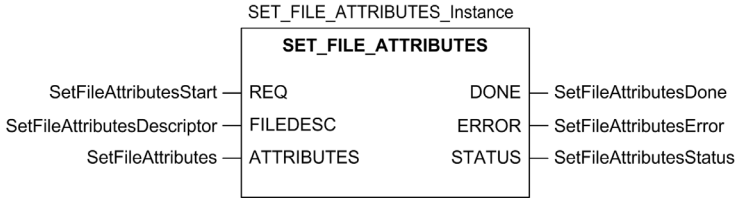
Description de la fonction

La fonction `SET_FILE_ATTRIBUTES` permet de configurer les attributs du fichier.

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

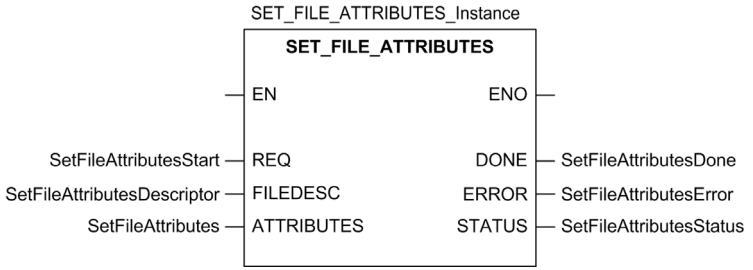
Représentation en FBD

La représentation en FBD de la fonction `SET_FILE_ATTRIBUTES` est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `SET_FILE_ATTRIBUTES` est la suivante :



Représentation en IL

Représentation :

```
CAL SET_FILE_ATTRIBUTES_Instance (REQ:=SetFileAttributesStart,
FILEDESC:=SetFileAttributesDescriptor, ATTRIBUTES:=SetFileAttributes,
DONE=>SetFileAttributesDone, ERROR=>SetFileAttributesError,
STATUS=>SetFileAttributesStatus)
```

Représentation en ST

Représentation :

```
SET_FILE_ATTRIBUTES_Instance (REQ:=SetFileAttributesStart,
FILEDESC:=SetFileAttributesDescriptor, ATTRIBUTES:=SetFileAttributes,
DONE=>SetFileAttributesDone, ERROR=>SetFileAttributesError,
STATUS=>SetFileAttributesStatus);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre spécifie le descripteur de fichier renvoyé par la fonction CREATE_FILE ou OPEN_FILE.
ATTRIBUTES	WORD	Ce paramètre spécifie les attributs à configurer. Le système n'accepte que deux attributs : • SET_RDONLY (1) : le fichier est protégé en écriture. • CLR_RDONLY (0) : l'attribut de lecture seule est supprimé.

Description des paramètres communs *(voir page 36)*.

Chapitre 8
GET_FILE_INFO : récupération d'attributs de fichier

GET_FILE_INFO : obtention d'informations sur le fichier

Description de la fonction

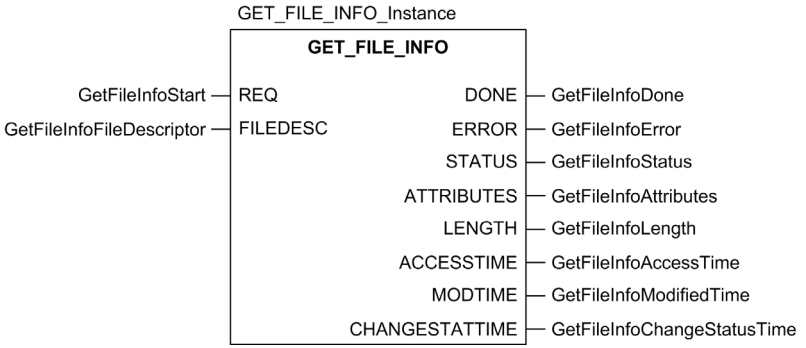
La fonction GET_FILE_INFO permet d'obtenir des informations sur le fichier.

La fonction OPEN_FILE *(voir page 47)* doit être activée avant le bloc GET_FILE_INFO, car la valeur de FileDescriptor *(voir page 49)* provient du paramètre de sortie du bloc OPEN_FILE ou CREATE_FILE.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

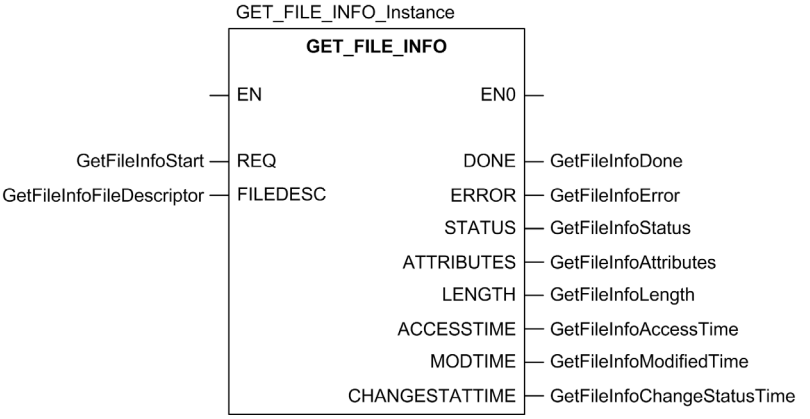
Représentation en FBD

La représentation en FBD de la fonction GET_FILE_INFO est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `GET_FILE_INFO` est la suivante :



Représentation en IL

Représentation :

```
CAL GET_FILE_INFO_Instance (REQ:=GetFileInfoStart,
FILEDESC:=GetFileInfoDescriptor, DONE=>GetFileInfoDone,
ERROR=>GetFileInfoError, STATUS=>GetFileInfoStatus,
ATTRIBUTES=>GetFileInfoAttributes, LENGTH=>GetFileInfoLength,
ACCESSTIME=>GetFileInfoAccessTime, MODTIME=>GetFileInfoModifiedTime,
CHANGESTATTIME=>GetFileInfoChangeStatusTime)
```

Représentation en ST

Représentation :

```
GET_FILE_INFO_Instance (REQ:=GetFileInfoStart,
FILEDESC:=GetFileInfoDescriptor, DONE=>GetFileInfoDone,
ERROR=>GetFileInfoError, STATUS=>GetFileInfoStatus,
ATTRIBUTES=>GetFileInfoAttributes, LENGTH=>GetFileInfoLength,
ACCESSTIME=>GetFileInfoAccessTime, MODTIME=>GetFileInfoModifiedTime,
CHANGESTATETIME=>GetFileInfoChangeStatusTime);
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre spécifie le descripteur de fichier renvoyé par les fonctions OPEN_FILE.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
ATTRIBUTES	WORD	Ce champ binaire spécifie les attributs de fichier : ● Bit 0 : fichier en lecture seule ● Bit 1 : fichier masqué ● Bit 2 : fichier système ● Bit 3 : libellé de volume (pas un fichier) ● Bit 4 : sous-répertoire ● Bit 5 : en cours d'archivage
LENGTH	UDINT	Ce paramètre spécifie la taille du fichier en octets.
ACCESSTIME	DATEANDTIME	Ce paramètre spécifie la date et l'heure du dernier accès au fichier. **NOTE :** il n'est pas mis à jour pour les automates M580.
MODTIME	DATEANDTIME	Ce paramètre spécifie la date et l'heure de la dernière modification du fichier.
CHANGESTATETIME	DATEANDTIME	Ce paramètre spécifie la date et l'heure de la dernière modification d'état du fichier.

Description des paramètres communs *(voir page 36)*.

Chapitre 9
GET_FREESIZE : affichage de l'espace disponible sur la carte mémoire

GET_FREESIZE : affichage de l'espace disponible dans la partition de fichiers sur la carte mémoire

Description de la fonction

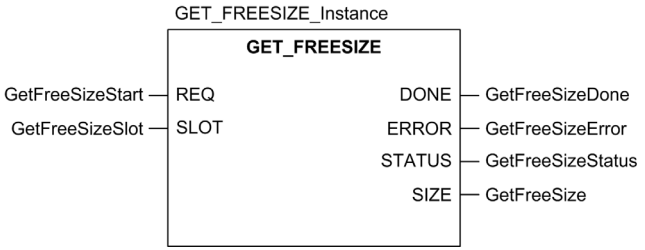
La fonction `GET_FREESIZE` affiche le volume d'espace disponible sur la carte mémoire.

Cet espace disponible est lié à la partition du système de fichiers, qui contient les données de stockage et les pages Web.

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

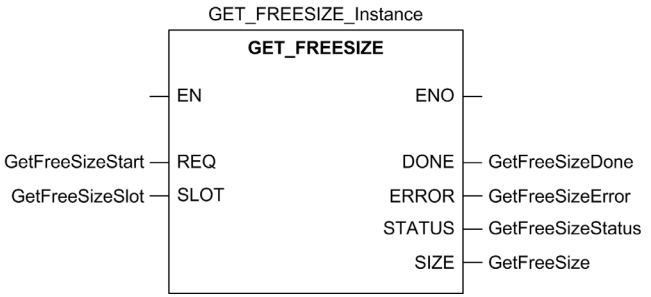
Représentation en FBD

La représentation en FBD de la fonction `GET_FREESIZE` est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `GET_FREESIZE` est la suivante :



Représentation en IL

Représentation :

```
CAL GET_FREESIZE_Instance (REQ:=GetFreeSizeStart, SLOT:=GetFreeSizeSlot,
DONE=>GetFreeSizeDone, ERROR=>GetFreeSizeError, STATUS=>GetFreeSizeStatus,
SIZE=>GetFreeSize)
```

Représentation en ST

Représentation :

```
WRITE GET_FREESIZE_Instance (REQ:=GetFreeSizeStart, SLOT:=GetFreeSizeSlot,
DONE=>GetFreeSizeDone, ERROR=>GetFreeSizeError, STATUS=>GetFreeSizeStatus,
SIZE=>GetFreeSize);
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
SLOT	BYTE	Ce paramètre indique l'adresse de l'emplacement de la carte mémoire (toujours 0).

Le tableau suivant décrit le paramètre de sortie :

Paramètre	Type	Commentaire
SIZE	UDINT	Ce paramètre indique en octets l'espace disponible sur la carte mémoire.

Description des paramètres communs *(voir page 36)*.

Chapitre 10
SEEK_FILE : positionnement dans un fichier

SEEK_FILE : position dans le fichier

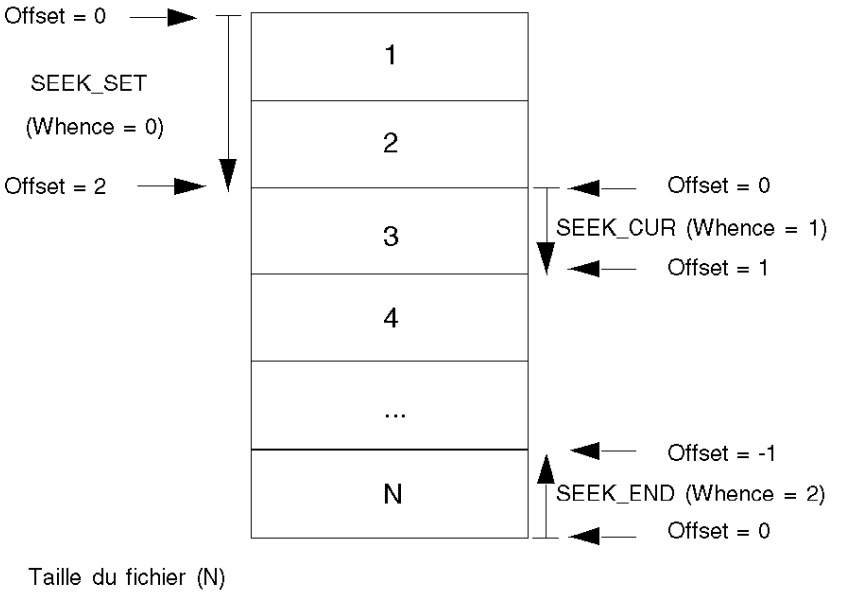
Description de la fonction

La fonction `SEEK_FILE` définit la position courante dans le fichier, en fonction de l'argument `Offset`.

L'argument `Whence` définit la position du pointeur dans le fichier. Les trois valeurs possibles sont les suivantes :

- `SEEK_SET` (valeur 0) : cette valeur place le pointeur à la position correspondant à l'offset (valeur par défaut : modifier le fichier).
- `SEEK_CUR` (valeur 1) : place le pointeur à la position courante + l'offset.
- `SEEK_END` (valeur 2) : place le pointeur à la position correspondant à la taille du fichier + l'offset. Cette valeur est très utile pour remplacer les dernières variables du fichier (offset négatif) ou pour écrire dans le fichier à partir de la date, afin de ne pas écraser de données.

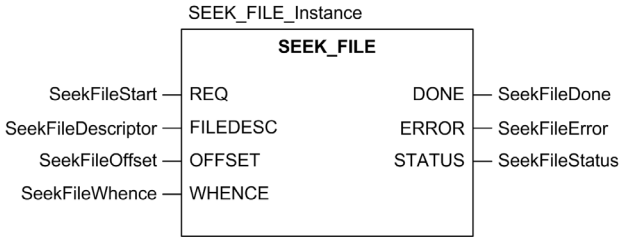
Le schéma suivant montre le positionnement du pointeur en fonction de l'offset (différentes valeurs d'offset indiquées) :



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

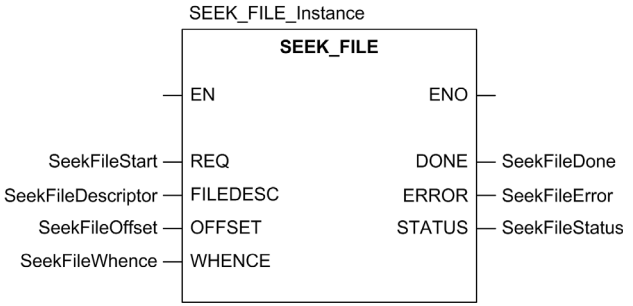
Représentation en FBD

La représentation en FBD de la fonction `SEEK_FILE` est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction SEEK_FILE est la suivante :



Représentation en IL

Représentation :

```
CAL SEEK_FILE_Instance (REQ:=SeekFileStart, FILEDESC:=SeekFileDescriptor,
OFFSET:=SeekFileOffset, WHENCE:=SeekFileWhence, DONE=>SeekFileDone,
ERROR=>SeekFileError, STATUS=>SeekFileStatus)
```

Représentation en ST

Représentation :

```
SEEK_FILE_Instance (REQ:=SeekFileStart, FILEDESC:=SeekFileDescriptor,
OFFSET:=SeekFileOffset, WHENCE:=SeekFileWhence, DONE=>SeekFileDone,
ERROR=>SeekFileError, STATUS=>SeekFileStatus);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre spécifie le descripteur de fichier renvoyé par la fonction CREATE_FILE ou OPEN_FILE.
OFFSET	DINT	Ce paramètre spécifie l'offset du mouvement.
WHENCE	INT	Mode de recherche : • SEEK_SET (0) - réglé sur l'offset • SEEK_CUR (1) - réglé sur la position courante + l'offset • SEEK_END (2) - réglé sur la taille du fichier + l'offset

Description des paramètres communs *(voir page 36)*.

Chapitre 11
WR_DATA_TO_FILE : écriture de données dans un fichier

WR_DATA_TO_FILE : écriture de données dans un fichier

Description de la fonction

La fonction WR_DATA_TO_FILE permet d'écrire dans un fichier une variable à adresse directe (%MW1000), une variable localisée (VAR0 @ %MW1000) ou une variable non localisée.

La valeur écrite est ajoutée après la position courante dans le fichier. Après l'écriture, la position courante du fichier est mise à jour.

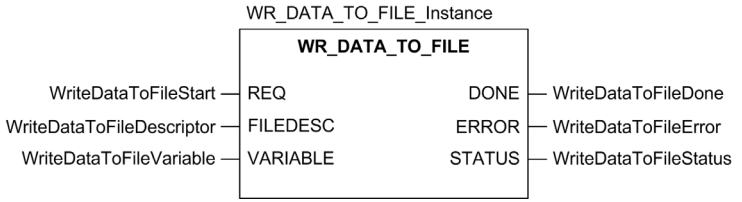
NOTE : lorsque vous utilisez la fonction WR_DATA_TO_FILE avec le type de données STRING, elle crée dans le fichier un tableau d'octets de la taille maximale déclarée. Par exemple, si vous déclarez STRING[100], la fonction écrit 100 octets dans le fichier. Si vous lisez ce fichier avec un EFB de Unity Pro, la fonction lit les 100 octets, mais ne voit que les premiers caractères avant le premier caractère NULL. Si les 100 octets contiennent un grand nombre de caractères NULL, le logiciel ou l'outil tiers utilisé pourrait afficher plusieurs chaînes et non une seule.

NOTE : il est toujours possible de modifier l'état d'un fichier dans le système de fichiers, même si ce fichier est déjà ouvert. L'état d'un fichier (Lecture seule, Lecture/Ecriture) n'est pas évalué à l'ouverture mais lors de la modification de son contenu.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

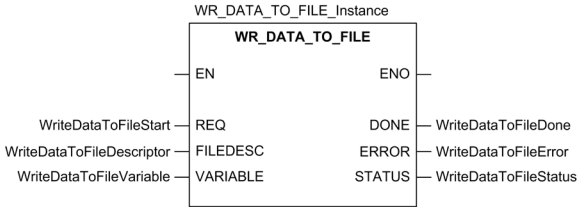
Représentation en FBD

La représentation en FBD de la fonction WR_DATA_TO_FILE est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction WR_DATA_TO_FILE est la suivante :



Représentation en IL

Représentation :

```
CAL WR_DATA_TO_FILE_Instance (REQ:=WriteDataToFileStart,
FILEDESC:=WriteDataToFileDescriptor, VARIABLE:=WriteDataToFileVariable,
DONE=>WriteDataToFileDone, ERROR=>WriteDataToFileError,
STATUS=>WriteDataToFileStatus)
```

Représentation en ST

Représentation :

```
WR_DATA_TO_FILE_Instance (REQ:=WriteDataToFileStart,
FILEDESC:=WriteDataToFileDescriptor, VARIABLE:=WriteDataToFileVariable,
DONE=>WriteDataToFileDone, ERROR=>WriteDataToFileError,
STATUS=>WriteDataToFileStatus);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre spécifie le descripteur de fichier renvoyé par la fonction CREATE_FILE ou OPEN_FILE.
VARIABLE	ANY	Ce paramètre spécifie la variable à écrire dans le fichier. Toute variable à adresse directe (%MW1000), localisée (VAR0 @ %MW1000) ou non localisée (simple, table, structure) peut être écrite. Pour écrire un tableau de %MW ou de variables directes, la syntaxe utilisée est %MW1000:10 (10 mots à partir de %MW1000).

Description des paramètres communs *(voir page 36)*.

Chapitre 12
RD_FILE_TO_DATA : lecture de données dans un fichier

RD_FILE_TO_DATA : lecture des données d'un fichier

Description de la fonction

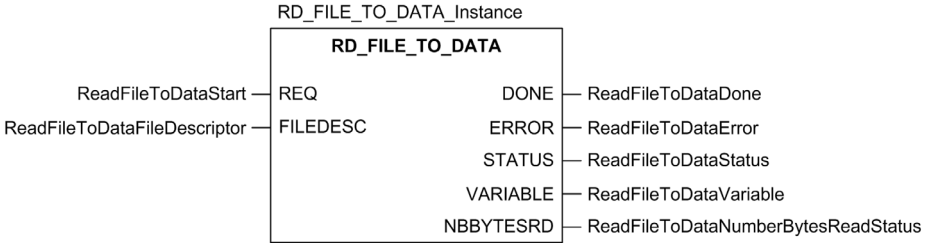
La fonction `RD_FILE_TO_DATA` permet de lire les données d'un fichier, à l'emplacement courant du fichier, et de copier ces données dans une variable à adresse directe (`%MW1000`), une variable localisée (`VAR0 @ %MW1000`) ou une variable non localisée. Après la lecture, la position courante du fichier est mise à jour.

Cette fonction élémentaire lit un nombre d'octets correspondant à la taille de la variable dans le fichier spécifié.

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

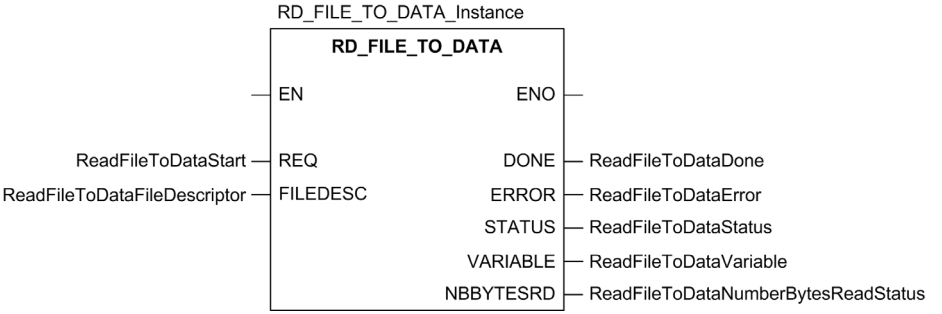
Représentation en FBD

La représentation en FBD de la fonction `RD_FILE_TO_DATA` est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `RD_FILE_TO_DATA` est la suivante :



Représentation en IL

Représentation :

```
CAL RD_FILE_TO_Instance (REQ:=ReadFileToDataStart,
FILEDESC:=ReadFileToDataDescriptor, DONE=>ReadFileToDataDone,
ERROR=>ReadFileToDataError, STATUS=>ReadFileToDataStatus,
VARIABLE=>ReadFileToDataVariable,
NBBYTESRD=>ReadFileToDataNumberBytesReadStatus)
```

Représentation en ST

Représentation :

```
RD_FILE_TO_Instance (REQ:=ReadFileToDataStart,
FILEDESC:=ReadFileToDataDescriptor, DONE=>ReadFileToDataDone,
ERROR=>ReadFileToDataError, STATUS=>ReadFileToDataStatus,
VARIABLE=>ReadFileToDataVariable,
NBBYTESRD=>ReadFileToDataNumberBytesReadStatus);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
FILEDESC	DINT	Ce paramètre spécifie le descripteur de fichier renvoyé par la fonction CREATE_FILE ou OPEN_FILE.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
VARIABLE	ANY	Ce paramètre spécifie la variable ou le tableau qui va recevoir les données. Toute variable à adresse directe (%MW1000), localisée (VAR0 @ %MW1000) ou non localisée (simple, table, structure) peut être utilisée comme destination. Pour copier dans un tableau ou des variables directes, la syntaxe utilisée est %MW1000:10 (10 mots à partir de %MW1000).
NBBYTESRD	UDINT	Ce paramètre spécifie la taille effectivement lue. Elle peut être inférieure à la taille de la variable. La principale raison pour laquelle NBBYTESRD peut être inférieur à la taille de la variable est la suivante : Taille du fichier - position de début de la lecture < Taille de la variable Ensuite, seul le nombre d'octets lus est mis à jour dans la variable, les autres octets de données provenant de la dernière lecture.

Description des paramètres communs *(voir page 36)*.

Chapitre 13
CLOSE_FILE : fermeture d'un fichier

CLOSE_FILE : fermeture du fichier

Description de la fonction

La fonction `CLOSE_FILE` permet de fermer le fichier spécifié et de libérer son descripteur.

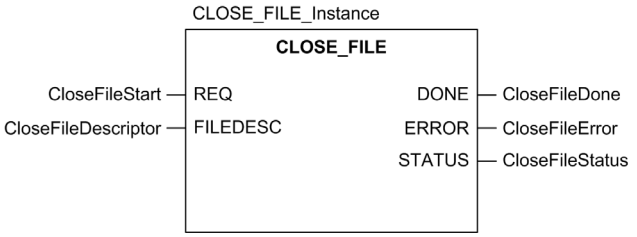
NOTE : si l'utilisateur souhaite désactiver l'accès à la carte pour extraire le fichier par exemple, il doit mettre le bit `%S65` à 1. Dans ce cas, tous les accès en cours sont révolus, ce qui entraîne la fermeture automatique de tous les fichiers ouverts. (Pour réactiver l'accès à la carte, réglez le bit `%S65` à 0.)

NOTE : à chaque ouverture d'un fichier, le paramètre FileDescriptor renvoie une valeur aléatoire. Celui-ci sera utilisé lors de la fermeture du fichier. Si vous ouvrez un nouveau fichier sans fermer le précédent, le paramètre FileDescriptor renvoie une nouvelle valeur et vous ne pourrez pas fermer le fichier précédent. Il est donc conseillé d'enregistrer le paramètre FileDescriptor après avoir ouvert un fichier.
Lorsque le nombre maximum de fichiers ouverts est atteint, vous recevez un message d'erreur si vous tentez d'ouvrir un autre fichier.

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

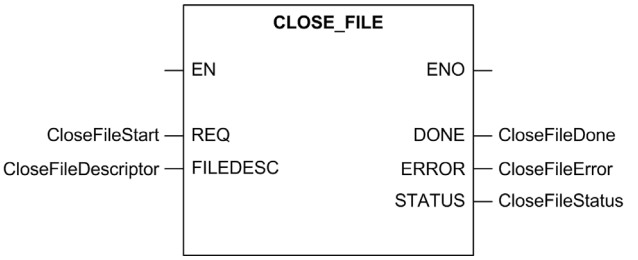
Représentation en FBD

La représentation en FBD de la fonction `CLOSE_FILE` est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `CLOSE_FILE` est la suivante :



Représentation en IL

Représentation :

```
CAL CLOSE_FILE_Instance (REQ:=CloseFileStart,
FILEDESC:=CloseFileDescriptor, DONE=>CloseFileDone, ERROR=>CloseFileError,
STATUS=>CloseFileStatus)
```

Représentation en ST

Représentation :

```
CLOSE_FILE_Instance (REQ:=CloseFileStart, FILEDESC:=CloseFileDescriptor,
DONE=>CloseFileDone, ERROR=>CloseFileError, STATUS=>CloseFileStatus);
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
`FILEDESC`	`DINT`	Ce paramètre spécifie le descripteur de fichier renvoyé par `CREATE_FILE` ou `OPEN_FILE`.

Description des paramètres communs *(voir page 36)*.

Chapitre 14
DELETE_FILE : suppression d'un fichier

DELETE_FILE : suppression d'un fichier

Description de la fonction

La fonction DELETE_FILE permet de supprimer le fichier appelé FileName.

NOTE :
Lorsque vous utilisez la fonction DELETE_FILE, fermez toujours un fichier à l'aide de la fonction CLOSE_FILE avant de le supprimer. Sinon :

- Le compteur de fichiers ouverts n'est pas mis à jour. Consultez la section, Limitations concernant les fichiers ouverts *(voir page 35)*.
- Le paramètre FileDescriptor (FD) n'est pas libéré.

⚠ AVERTISSEMENT

RISQUE DE DEPASSEMENT DU NOMBRE DE DESCRIPTEURS DE FICHIER

Fermez systématiquement un fichier ouvert avant de le supprimer à l'aide de la fonction DELETE_FILE.

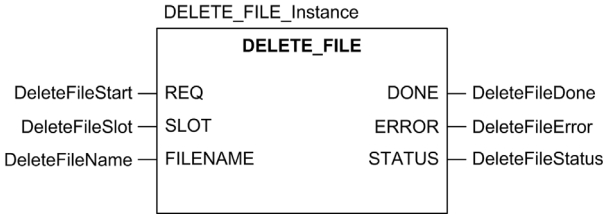
Un fichier peut être supprimé sans que l'utilisateur en soit informé et peut incrémenter le nombre de descripteurs de fichier au-delà de la limite autorisée.

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

Les paramètres supplémentaires EN et ENO peuvent être configurés.

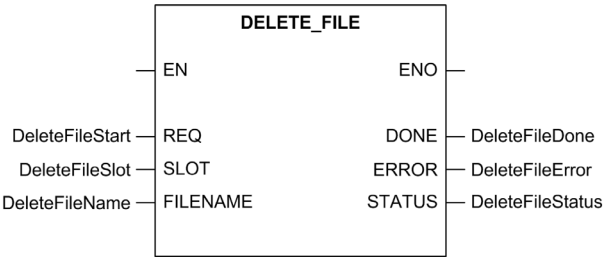
Représentation en FBD

La représentation en FBD de la fonction DELETE_FILE est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `DELETE_FILE` est la suivante :



Représentation en IL

Représentation :

```
CAL DELETE_FILE_Instance (REQ:=DeleteFileStart, SLOT:=DeleteFileSlot,
FILENAME:=DeleteFileName, DONE=>DeleteFileDone, ERROR=>DeleteFileError,
STATUS=>DeleteFileStatus)
```

Représentation en ST

Représentation :

```
DELETE_FILE_Instance (REQ:=DeleteFileStart, SLOT:=DeleteFileSlot,
FILENAME:=DeleteFileName, DONE=>DeleteFileDone, ERROR=>DeleteFileError,
STATUS=>DeleteFileStatus);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
`SLOT`	`BYTE`	Ce paramètre spécifie l'emplacement de la carte mémoire (toujours 0).
`FILENAME`	`STRING`	Ce paramètre indique le nom du fichier à supprimer. Nombre maximum de caractères : 64

Description des paramètres communs *(voir page 36)*.

Chapitre 15
Codes d'erreur des EFB de gestion des fichiers de carte mémoire

Codes d'erreur

Présentation

Un code d’erreur est valide jusqu’au prochain lancement de l’EF qui l’a levé.

Tableau des codes d’erreur

Le tableau suivant décrit les codes d’erreurs pouvant être levés pour les fonctions EFB :

Valeur	Description
1	La carte mémoire est manquante dans le logement.
2	La carte mémoire n'autorise pas le stockage de données (référence erronée) ou la structure du système de fichiers a été supprimée sur la carte mémoire (la carte mémoire a sans doute été reformatée sur un PC).
3	Le numéro de logement est incorrect (doit être 0).
4	Tentative d'écriture sur une carte mémoire protégée en écriture par le contacteur.
5	Tentative d'ouverture de plus de 16 fichiers en même temps.
6	Tentative d'écriture alors qu'il n'y a plus de place disponible dans la carte mémoire pour le stockage de données.
7	Tentative d'ouverture d'un fichier qui n'existe pas sur la carte mémoire.
11	Le nom de fichier dépasse 64 caractères.
13	Le nom de fichier est incorrect ou vide.
14	Tentative d'écriture d'un fichier ouvert en mode lecture seule.
17	Le fichier a été supprimé (survient quand il existe un descripteur de fichier et que le fichier a été supprimé par une commande **DELETE_FILE** EFB ou FTP).
21	Descripteur de fichier incorrect.
23	Tentative d'exécution d'un bloc de fonction de stockage de données dans une tâche différente de MAST.
-1	Autres erreurs que celles ci-dessus.

Chapitre 16
Exemples d'EF de gestion des fichiers de carte mémoire

Généralités

Ce chapitre fournit des exemples d'utilisation des EFB de gestion des fichiers de carte mémoire.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Définition de l'exemple et déclaration des variables

Définition

Les EFB WR_DATA_TO_FILE et RD_FILE_TO_DATA permettent respectivement d'écrire des variables dans un fichier et de lire les données d'un fichier. Ces fonctions élémentaires forment une partie de la base de la gestion de fichiers, et peuvent ainsi être utilisées pour construire n'importe quelle application ayant besoin de ces fonctions.

L'exemple utilise les quatre fonctions suivantes :

- OPEN_FILE, pour ouvrir un fichier déjà crée *(voir page 43)*,
- WR_DATA_TO_FILE, pour écrire des données,
- RD_FILE_TO_DATA, pour lire les données,
- CLOSE_FILE pour fermer le fichier.

Déclaration des variables

Le tableau ci-dessous recense le détail des valeurs à déclarer pour les paramètres utilisés :

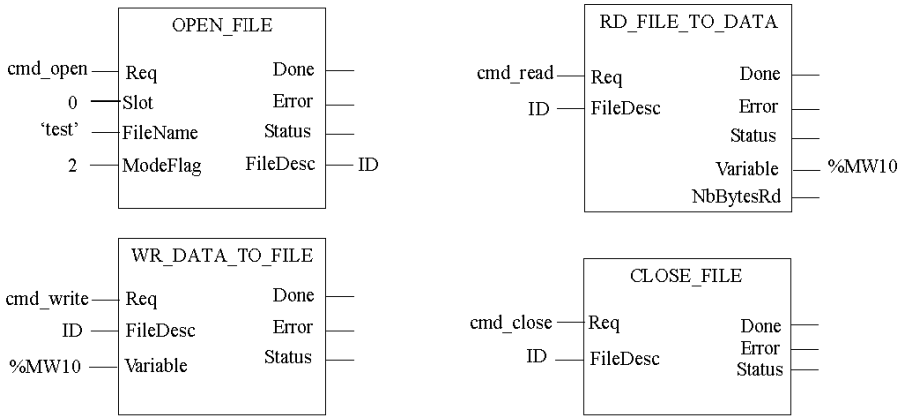
Nom de la variable	Type	Commentaire
cmd_open	EBOOL	Bit d'ouverture du fichier.
cmd_write	EBOOL	Bit d'écriture de données dans le fichier.
cmd_read	EBOOL	Bit de lecture des données du fichier.
cmd_close	EBOOL	Bit de fermeture du fichier.
FileName	STRING	Nom du fichier. La valeur par défaut est "test".
FileDesc	INT	FileDesc est le numéro attribué automatiquement lors de l'ouverture du fichier. Ce numéro d'identification unique et temporaire sera attribué par la fonction OPEN_FILE.

NOTE : le numéro d'identification temporaire FileDesc permet, entre autres, à plusieurs personnes d'ouvrir un même fichier en même temps pour travailler dessus. Le numéro utilisé pour FileDesc est libéré une fois le fichier fermé avec la fonction CLOSE_FILE. FileDesc est géré automatiquement par le système.

Exemple hors ligne

Présentation

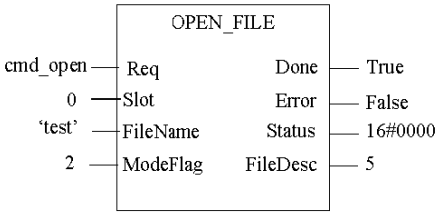
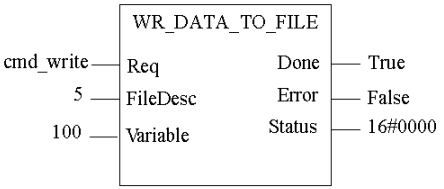
La figure ci-dessous décrit les blocs fonction élémentaires avec les variables déclarées auparavant, hors ligne :

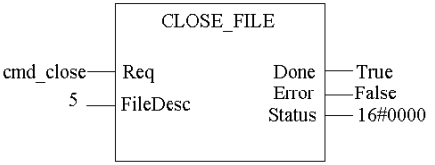
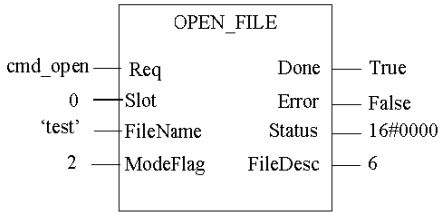


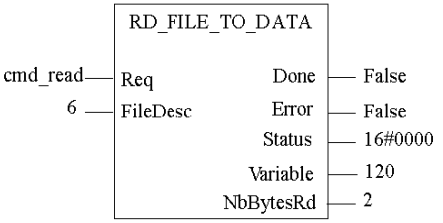
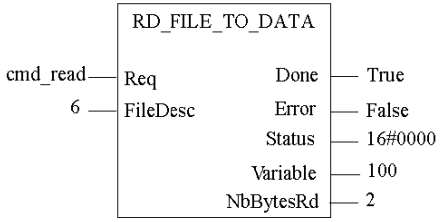
Exemple en ligne : Procédure

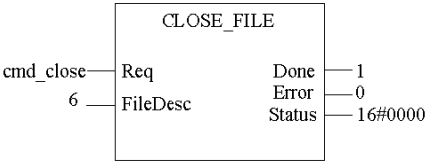
Procédure

Dans cet exemple, les données doivent être écrites dans un fichier, lequel est ensuite fermé puis rouvert pour que ses données soient lues. Le tableau ci-dessous décrit la procédure à suivre pour cet exemple :

Etape	Action
1	Exécutez la fonction `OPEN_FILE` en activant le bit `cmd_open` à l'aide d'un clic droit sur le paramètre d'entrée Req. **Résultat :** Un numéro d'identification unique et temporaire est attribué à l'instance d'ouverture en tant que sortie FileDesc. (Ici, nous avons attribué la valeur 5 à cette sortie. Utilisez cette valeur pour déclarer la variable d'entrée FileDesc sur les fonctions `WR_DATA_TO_FILE` et `CLOSE_FILE`.) Le graphique ci-dessous représente la fonction après son exécution :  Désactivez `cmd_open` par clic droit sur le paramètre d'entrée Req.
2	Dans le bloc `WR_DATA_TO_FILE`, attribuez à la variable `%MW10` la valeur 100 par clic droit sur le paramètre d'entrée Variable.
3	Exécutez la fonction `WR_DATA_TO_FILE` en activant le bit `cmd_write` par clic droit sur le paramètre d'entrée Req. **Résultat :** La valeur de la variable est écrite dans le fichier *test*. Le graphique ci-dessous représente la fonction après son exécution :  Désactivez `cmd_write` par clic droit sur le paramètre d'entrée Req.

Etape	Action
4	Exécutez la fonction `CLOSE_FILE` en activant le bit `cmd_close` par clic droit sur le paramètre d'entrée Req. **Résultat :** Le fichier est fermé et le numéro d'identification FileDesc est libéré en vue d'un usage ultérieur. Le graphique ci-dessous représente la fonction après son exécution :  Désactivez `cmd_close` par clic droit sur le paramètre d'entrée Req.
5	Exécutez à nouveau la fonction `OPEN_FILE` en activant le bit `cmd_open` par clic droit sur le paramètre d'entrée Req. **Résultat :** Comme il s'agit d'une nouvelle instance d'ouverture, un nouveau numéro FileDesc est attribué, ici 6. Utilisez ces valeurs pour déclarer la variable d'entrée FileDesc dans les fonctions `RD_FILE_TO_DATA` et `CLOSE_FILE`. Le graphique ci-dessous représente la fonction après son exécution :  Désactivez `cmd_open` par clic droit sur le paramètre d'entrée Req.

Etape	Action
6	Dans le bloc RD_FILE_TO_DATA, attribuez à la variable %MW10 la valeur 120 par clic droit sur le paramètre d'entrée Variable. Ce faisant, il est possible de vérifier que la valeur récupérée par la fonction RD_FILE_TO_DATA est correcte. Le graphique ci-dessous représente la fonction avant son exécution : 
7	Exécutez la fonction RD_FILE_TO_DATA en activant le bit cmd_read par clic droit sur le paramètre d'entrée Req. **Résultat :** La fonction lit les données du fichier, c'est-à-dire la valeur de la variable écrite auparavant à l'aide de la fonction WR_DATA_TO_FILE. Cette valeur se trouve dans le paramètre de sortie Variable. Le graphique ci-dessous représente la fonction après son exécution :  Désactivez cmd_read par clic droit sur le paramètre d'entrée Req.

Etape	Action
8	Exécutez la fonction `CLOSE_FILE` en activant le bit `cmd_close` par clic droit sur le paramètre d'entrée Req. **Résultat :** Le fichier est fermé et le numéro d'identification FileDesc est libéré en vue d'un usage ultérieur. Le graphique ci-dessous représente la fonction après son exécution :  Désactivez `cmd_close` par clic droit sur le paramètre d'entrée Req.

Exemple écrit en langage ST

Présentation

Dans cet exemple, les blocs fonction élémentaires (EFB) suivants sont utilisés pour effectuer certaines opérations sur un fichier :

- CREATE_FILE
- OPEN_FILE
- SEEK_FILE
- WR_DATA_TO_FILE
- RD_FILE_TO_DATA
- CLOSE_FILE

Déclaration des variables

Le tableau ci-dessous présente en détail les valeurs à déclarer pour les paramètres utilisés :

Nom de variable	Type	Commentaire
close_req	EBOOL	Bit d'activation de la fonction CLOSE_FILE
create_req	EBOOL	Bit d'activation de la fonction CREATE_FILE
fileDesc	DINT	FileDesc est le numéro attribué automatiquement lors de l'ouverture du fichier. Ce numéro d'identification unique et temporaire est attribué par la fonction OPEN_FILE.
NbBytesRd	UDINT	Indication de la taille effectivement lue de la variable.
read_req	EBOOL	Bit d'activation de la fonction RD_FILE_TO_DATA
variable	INT	Variable lue dans le fichier
seek_req	EBOOL	Bit d'activation de la fonction SEEK_FILE
write_req	EBOOL	Bit d'activation de la fonction WR_DATA_TO_FILE
variable_tab	ARRAY[0..9] OF INT	Tableau des différentes valeurs des variables écrites dans le fichier
cmd_write	EBOOL	Bit d'écriture du fichier
GO_STORE	EBOOL	Bit utilisé pour lancer la création du fichier
cmd_seek	EBOOL	Bit de recherche du fichier
Error_WR	BOOL	Bit indiquant une erreur lors de l'exécution de la fonction WR_DATA_TO_FILE
cmd_close	EBOOL	Bit de fermeture du fichier
GO_RESTORE	EBOOL	Bit utilisé pour amorcer la réouverture du fichier
open_req	EBOOL	Bit d'activation de la fonction OPEN_FILE

Nom de variable	Type	Commentaire
cmd_read	EBOOL	Bit de lecture de données dans le fichier
Done_WR	BOOL	Bit indiquant que l'opération d'écriture dans un fichier est terminée
Done_OPEN	BOOL	Bit indiquant que l'opération d'ouverture d'un fichier est terminée
Status_WR	WORD	Code d'erreur (utile lorsque Error_WR:=1)
Done_SEEK	BOOL	Bit indiquant que la recherche d'un fichier est terminée
Error_SEEK	BOOL	Bit indiquant une erreur lors de l'exécution de la fonction SEEK_FILE
Status_SEEK	WORD	Code d'erreur (utile lorsque Error_SEEK:=1)
Done_CREATE	BOOL	Bit indiquant que la création d'un fichier est terminée
Error_CREATE	BOOL	Bit indiquant une erreur lors de l'exécution de la fonction CREATE_FILE
Status_CREATE	WORD	Code d'erreur (utile lorsque Error_CREATE:=1)
Done_CLOSE	BOOL	Bit indiquant que l'opération de fermeture d'un fichier est terminée
Error_CLOSE	BOOL	Bit d'erreur de la fonction CLOSE_FILE
Status_CLOSE	WORD	Code d'erreur (utile lorsque Error_CLOSE:=1)
Error_OPEN	BOOL	Bit indiquant une erreur lors de l'exécution de la fonction OPEN_FILE
Status_OPEN	WORD	Code d'erreur (utile lorsque Error_OPEN:=1)
Done_RD	BOOL	Bit indiquant que l'opération de lecture d'un fichier est terminée
Error_RD	BOOL	Bit indiquant une erreur lors de l'exécution de la fonction RD_FILE_TO_DATA
Status_RD	WORD	Code d'erreur (utile lorsque Error_RD:=1)

Code

Ce code exécute les actions suivantes :

- création ou recherche du fichier si celui-ci existe déjà ;
- écriture des données dans le fichier ;
- fermeture du fichier ;
- ouverture du fichier fermé ;
- lecture des données écrites dans le fichier ;
- fermeture du fichier.

```
(*********************************************************
Créez un fichier 'TEST.bin' s'il n'existe pas et ouvrez-le en
mode d'ajout (SEEK).
Ecrivez les données et fermez le fichier
*********************************************************)
(******* Début de la section CREATE *******)
if RE(GO_STORE) then (* Définir GO_STORE pour démarrer *)
  create_req := 1;
end_if;
CREATE (REQ := create_req,(* Front montant sur l'entrée REQ pour démarrer
CREATE *)
SLOT := 0,
FILENAME := 'TEST.bin',
MODEFLAG := 2,(* mode lecture/écriture *)
DONE => Done_CREATE,
ERROR => Error_CREATE,
STATUS => Status_CREATE,
FILEDESC => fileDesc);
create_req := 0;
if GO_STORE then
   if (Done_CREATE and not Error_CREATE) then
     GO_STORE := 0 ;
     seek_req := 1;
     cmd_seek := 1;
   end_if;
end_if;
(******* Fin de la section CREATE *******)
(******* Début de la section SEEK *******)
SEEK (REQ := seek_req;(* Front montant sur l'entrée REQ pour
démarrer SEEK *)
FILEDESC := fileDesc
OFFSET := 0,
WHENCE := 2;
(* SEEK_END *)
```

```
DONE => Done_SEEK,
ERROR => Error_SEEK,
STATUS => Status_SEEK);
seek_req := 0;
if cmd_seek  then
   if (Done_SEEK and not Error_SEEK) then
     cmd_seek := 0;
     write_req := 1;
     cmd_write := 1;
   end_if;
 end_if;
(******* Fin de la section SEEK *******)
(******* Début de la section WRITE *******)
variable_tab[0] := variable_tab[0] + 1;
WRITE (REQ := write_req,(* Front montant sur l'entrée REQ pour
démarrer WRITE *)
FILEDESC := fileDesc,
VARIABLE := variable_tab,
DONE => Done_WR,
ERROR => Error_WR,
STATUS => Status_WR);
write_req := 0;
if cmd_write then
  if (Done_WR and not Error_WR) then
    cmd_write := 0;
    close_req := 1;
    cmd_close := 1;
  end_if;
end_if;
(******* Fin de la section CREATE *******)
(******* Début de la section CLOSE *******)
CLOSE (REQ := close_req,(* Front montant sur l'entrée REQ pour
démarrer CLOSE *)
FILEDESC := fileDesc,
```

```
DONE => Done_CLOSE,
ERROR => Error_CLOSE,
STATUS => Status_CLOSE);
close_req := 0;
if cmd_close then
  if (Done_CLOSE and not Error_CLOSE) then
    cmd_close := 0;
  end_if;
end_if;
(******* Fin de la section CLOSE *******)
(*********************************************************
Ouvrez le fichier 'TEST.bin',
lisez les premières données et fermez le fichier
*********************************************************
(******* Début de la section OPEN *******)
if RE(GO_RESTORE) then(* Définir GO_RESTORE pour démarrer *)
  open_req := 1;
end_if;
OPEN (REQ := open_req,(* Front montant sur l'entrée REQ pour démarrer
OPEN *)
SLOT := 0,
FILENAME := 'TEST.bin',
MODEFLAG := 0,
DONE => Done_OPEN,
ERROR => Error_OPEN,
STATUS => Status_OPEN,
FILEDESC => fileDesc);
open_req := 0;
if GO_RESTORE then
  if (Done_OPEN and not Error_OPEN) then
    GO_RESTORE := 0 ;
    read_req := 1;
    cmd_read := 1;
  end_if;
```

```
end_if;
(******* Fin de la section OPEN *******)
(******* Début de la section READ *******)
READ (REQ := read_req,(* Front montant sur l'entrée REQ pour
démarrer READ *)
FILEDESC := fileDesc,
DONE => Done_RD,
ERROR => Error_RD,
STATUS => Status_RD,
VARIABLE => variable,
NBBYTESRD => NbBytesRd);
read_req := 0;
if cmd_read then
  if (Done_RD and not Error_RD) then
    cmd_read := 0;
   close_req := 1;(* Section CLOSE identique à CREATE/SEEK/WRITE/CLOSE *)
    cmd_close := 1;
  end_if;
end_if;
(******* Fin de la section READ *******)
```

Chapitre 17
WRITE_U_PCMCIA : écriture de données sur la carte mémoire

Description

Description de la fonction

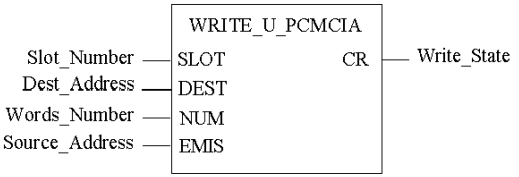
La fonction `WRITE_U_PCMCIA` permet de transférer les données issues de la mémoire RAM de l'automate vers la zone d'archivage de la carte mémoire de l'utilisateur. Elle permet d'adresser le début de la zone de l'automate à copier sur la carte PCMCIA jusqu'à 65 535 (au lieu de 32 767 pour la fonction `WRITE_PCMCIA`). L'utilisation de cette fonction est donc recommandée en lieu et place de la fonction `WRITE_PCMCIA`.

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

Un exemple de la fonction `WRITE_U_PCMCIA` est disponible, Exemple des fonctions WRITE_U_PCMCIA et READ_U_PCMCIA *(voir page 97)*.

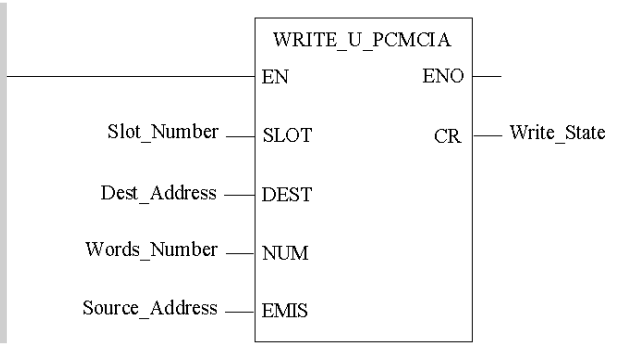
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Slot_Number
WRITE_U_PCMCIA Dest_Address, Words_Number, Source_Address, Write_State
```

Représentation en ST

Représentation :

```
WRITE_U_PCMCIA(Slot_Number, Dest_Address, Words_Number, Source_Address,
Write_State);
```

Désignation des paramètres

Ce tableau décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
`Slot_Number`	INT	Emplacement de la carte PCMCIA : ● 0 = emplacement supérieur ● 1 = emplacement inférieur
`Dest_Address`	BYTE	Première adresse d'écriture des données sur la carte mémoire.
`Words_Number`	INT	Nombre de mots à écrire.
`Source_Address`	UINT	Première adresse de lecture des données sur l'automate (%MW). La limite supérieure de l'adresse est 65 535.

Ce tableau décrit les paramètres de sortie :

Paramètre	Type	Commentaire
`Write_State`	INT	Code fournissant le résultat de l'exécution de la commande d'écriture : ● 16#0000 : écriture effectuée correctement ● 16#0102 : `Source_Address + Words_Number - 1` supérieur au nombre maximum de mots déclarés dans l'automate ● 16#0104 : aucune application, ni mot valide dans l'automate ● 16#0201 : aucune zone de fichiers dans la carte mémoire ● 16#0202 : erreur de carte mémoire ● 16#0204 : carte mémoire protégée en écriture ● 16#0241 : `Dest_Address` < 0 ● 16#0242 : `Dest_Address + Words_Number - 1` supérieur à l'adresse la plus élevée de la carte mémoire ● 16#0401 : `Words_Number` ≤ 0 ● 16#0402 : `Slot_Number` différent de 0 et 1 ● 16#0501 : service non pris en charge

Chapitre 18
READ_U_PCMCIA : lecture de données de la carte mémoire

Description

Description de la fonction

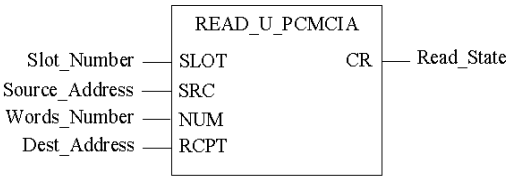
La fonction `READ_U_PCMCIA` permet de transférer les données issues de la zone d'archivage de la carte mémoire de l'utilisateur vers la mémoire RAM de l'automate. Elle permet d'adresser le début de la zone de l'automate à copier sur la carte PCMCIA jusqu'à 65 535 (au lieu de 32 767 pour la fonction `READ_PCMCIA`). L'utilisation de cette fonction est donc recommandée en lieu et place de la fonction `READ_PCMCIA`.

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

Un exemple de la fonction `READ_U_PCMCIA` est disponible : exemple d'utilisation des fonctions WRITE_U_PCMCIA et READ_U_PCMCIA *(voir page 97)*.

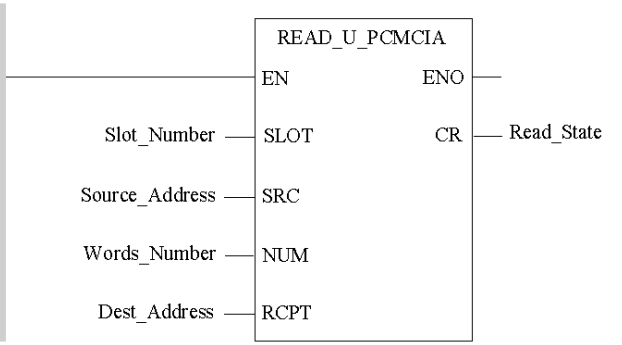
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Slot_Number
READ_U_PCMCIA Source_Address, Words_Number, Dest_Address, Read_State
```

Représentation en ST

Représentation :

```
READ_U_PCMCIA (Slot_Number, Source_Address, Words_Number, Dest_Address,
Read_State);
```

Description des paramètres

Ce tableau décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
`Slot_Number`	INT	Emplacement de la carte PCMCIA : ● 0 = emplacement supérieur ● 1 = emplacement inférieur
`Source_Address`	BYTE	Première adresse de lecture des données sur l'automate (0...).
`Words_Number`	INT	Nombre de mots à lire.
`Dest_Address`	UINT	Première adresse d'écriture des données sur la carte mémoire (%MW). La limite supérieure de l'adresse est 65 535.

Ce tableau décrit les paramètres de sortie :

Paramètre	Type	Commentaire
`Read_State`	INT	Code fournissant le résultat de l'exécution de la commande de lecture : ● 16#0000 : lecture effectuée correctement ● 16#0102 : `Dest_Address + Words_Number - 1` supérieur au nombre maximum de mots déclarés dans l'automate ● 16#0104 : aucune application, ni mot valide dans l'automate ● 16#0201 : aucune zone de fichiers dans la carte mémoire ● 16#0202 : erreur de carte mémoire ● 16#0204 : carte mémoire protégée en écriture ● 16#0241 : `Source_Address` < 0 ● 16#0242 : `Dest_Address + Words_Number - 1` supérieur à l'adresse la plus élevée de la carte mémoire ● 16#0401 : `Words_Number` ≤ 0 ● 16#0402 : `Slot_Number` différent de 0 et 1 ● 16#0501 : `Words_Number` = service non pris en charge

Chapitre 19
Exemple de fonctions WRITE_U_PCMCIA et READ_U_PCMCIA

Exemple WRITE_U_PCMCIA et READ_U_PCMCIA

Objectifs

Cet exemple montre comment utiliser les blocs de fonction READ_U_PCMCIA et WRITE_U_PCMCIA en :

- Ecrivant des valeurs de mots (%MW100 à %MW109) sur une carte mémoire.
- Lisant les valeurs de la carte mémoire dans des mots (%MW110 et %MW119).

NOTE : Dans cet exemple, pour utiliser ces blocs de fonction, une carte mémoire doit être reliées à l'automate.

Configuration de l'UC

Dans cet exemple, la carte mémoire TSX MRP C007M SRAM est insérée dans le logement supérieur A de l'UC (paramètre SLOT = 0 pour les blocs de fonction). De plus, pour le stockage de données, l'UC est configuré avec 2000 Ko de mémoire.

NOTE : Le stockage de données est utilisé pour les blocs de fonctions READ_U_PCMCIA et WRITE_U_PCMCIA. Les 2000 Ko de stockage de données représentent :

$$2000 kBytes \times 1024 = 2048000 Bytes$$

Les fonctions READ_U_PCMCIA et WRITE_U_PCMCIA travaillent avec des adresses de mot :

$$\frac{2048000 Bytes}{2} = 1024000 Words$$

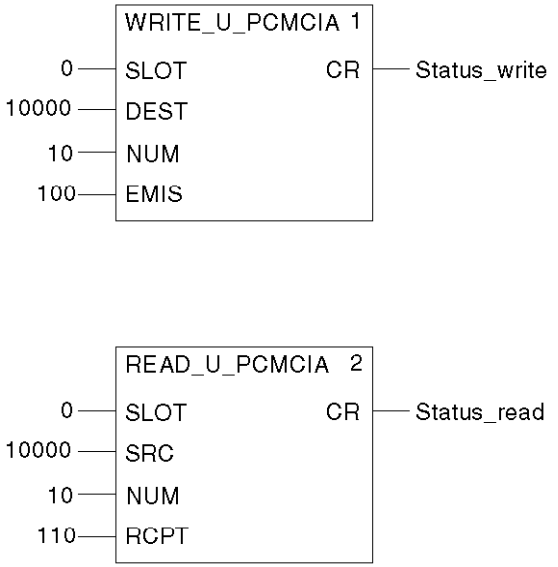
Les adresses de 0 à 1024000 pour la carte mémoire peuvent être utilisées.

Programmation de la section MAST

Dans la section MAST du programme, programmez comme suit :

- La carte mémoire est reliée au logement 0.
- La fonction WRITE_U_PCMCIA écrit 10 mots sur l'adresse 10000 de la carte mémoire à partir de %MW100.
- La fonction READ_U_PCMCIA lit 10 mots à partir de l'adresse 10000 sur la carte mémoire vers %MW110.

Représentation de WRITE_U_PCMCIA et READ_U_PCMCIA en langage FBD :



Représentation de WRITE_U_PCMCIA et READ_U_PCMCIA en langage ST :

```
WRITE_U_PCMCIA (0,10000,10,100,Status_write);
READ_U_PCMCIA (0,10000,10,110,Status_read);
```

Essai de l'exemple

En utilisant des tables d'animation avec les mots : %MW100 à %MW119, les valeurs de %MW100 à %MW109 sont copiées vers %MW110 à %MW119 en passant par la carte mémoire.

Chapitre 20
WRITE_V_PCMCIA : écriture de la variable dans la carte PCMCIA

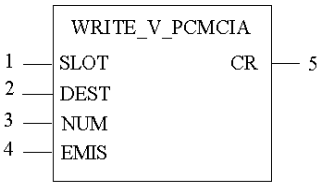
Description

Description de la fonction

La fonction `WRITE_V_PCMCIA` permet de transférer des variables depuis la mémoire RAM de l'automate vers une zone donnée de la carte mémoire de l'utilisateur.

Représentation en FBD

La représentation en `FBD` de la fonction `WRITE_V_PCMCIA` est la suivante :

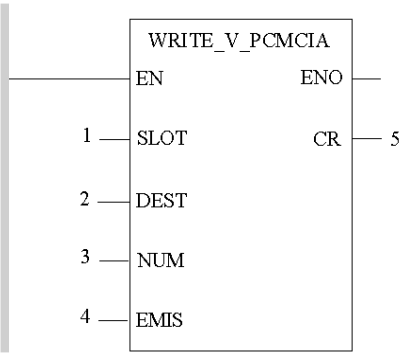


1 Numéro de l'emplacement
2 Adresse de la carte mémoire
3 Nombre d'octets à écrire
4 Adresse de la variable source
5 Etat d'écriture

Représentation en LD

Les paramètres supplémentaires EN et ENO peuvent être configurés.

La représentation en LD de la fonction WRITE_V_PCMCIA est la suivante :



1 Numéro de l'emplacement
2 Adresse de la carte mémoire
3 Nombre d'octets à écrire
4 Adresse de la variable source
5 Etat d'écriture

Représentation en IL

La représentation en IL de la fonction WRITE_V_PCMCIA est la suivante :

```
LD SLOT
WRITE_V_PCMCIA SLOT, DEST, NUM, EMIS, CR
```

Représentation en ST

La représentation en ST de la fonction WRITE_V_PCMCIA est la suivante :

```
WRITE_V_PCMCIA(SLOT, DEST, NUM, EMIS, CR);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'**entrée** :

Paramètre	Type	Commentaire
SLOT	INT	Emplacement de la carte PCMCIA : ● 0 = emplacement supérieur ● 1 = emplacement inférieur
DEST	DINT	Première adresse d'écriture de la variable sur la carte mémoire.
NUM	UDINT	Nombre d'octets à écrire.
EMIS	ANY	Première adresse de la variable dans l'automate.

Le tableau suivant décrit le paramètre de **sortie** :

Paramètre	Type	Commentaire
CR	INT	Code fournissant le résultat de l'exécution de la commande d'écriture : ● 0000 hex : écriture effectuée correctement ● 0201 hex : aucune zone de fichiers dans la carte mémoire ● 0202 hex : erreur de carte mémoire ● 0204 hex : carte mémoire protégée en écriture ● 0241 hex : DEST < 0 ● 0401 hex : NUM = 0 ● 0402 hex : SLOT différent de 0 et de 1 ● 0501 hex : service non pris en charge

Chapitre 21
READ_V_PCMCIA : lecture de la variable à partir de la carte PCMCIA

Description

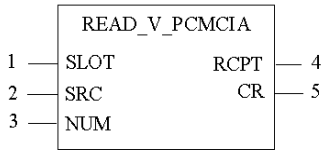
Description de la fonction

La fonction READ_V_PCMCIA permet de transférer des variables de la carte mémoire de l'utilisateur vers la mémoire RAM de l'automate.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

Représentation en FBD

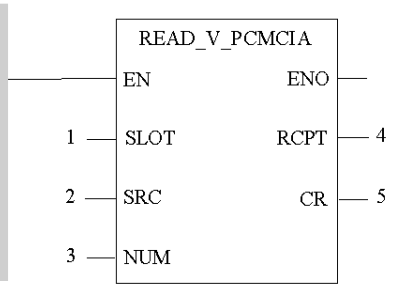
Représentation :



1 Numéro de l'emplacement
2 Adresse de la carte mémoire
3 Nombre d'octets à lire
4 Adresse cible de la variable
5 Etat de la lecture

Représentation en LD

Représentation :



1 Numéro de l'emplacement
2 Adresse de la carte mémoire
3 Nombre d'octets à lire
4 Adresse cible de la variable
5 Etat de la lecture

Représentation en IL

Représentation :

```
LD SLOT
READ_V_PCMCIA SRC, NUM
```

Représentation en ST

Représentation :

```
READ_V_PCMCIA(SLOT, SRC, NUM);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'**entrée** de la fonction READ_V_PCMCIA :

Paramètre	Type	Commentaire
SLOT	INT	Emplacement de la carte PCMCIA : ● 0 = emplacement supérieur ● 1 = emplacement inférieur
SRC	DINT	Adresse source depuis laquelle la variable est lue sur la carte mémoire (0…).
NUM	UDINT	Nombre d'octets à lire. Utilisez la fonction SIZEOF *(voir Unity Pro, Standard, Bibliothèque de blocs)* pour déterminer ce nombre.

Le tableau suivant décrit les paramètres de sortie de la fonction **READ_V_PCMCIA** :

Paramètre	Type	Commentaire
RCPT	ANY	Adresse cible de chaque variable localisée ou non localisée (DDT).
CR	INT	Code fournissant le résultat de la commande de lecture : ● 0000 (hex.) : lecture réussie ● 0201 (hex.) : aucune zone de fichiers dans la carte mémoire ● 0202 (hex.) : erreur de carte mémoire détectée ● 0204 (hex.) : carte mémoire protégée en écriture ● 0241 (hex.) : SRC < 0 ● 0280 (hex.) : valeur d'entrée NUM non cohérente avec la sortie RCT ou la structure de la mémoire PCMCIA ● 0401 (hex.) : NUM = 0 ● 0402 (hex.) : SLOT différent de 0 et 1 ● 0501 (hex.) : NUM = service non pris en charge

Chapitre 22
PRJ_VERS : version du projet

PRJ_VERS : version du projet

Description de la fonction

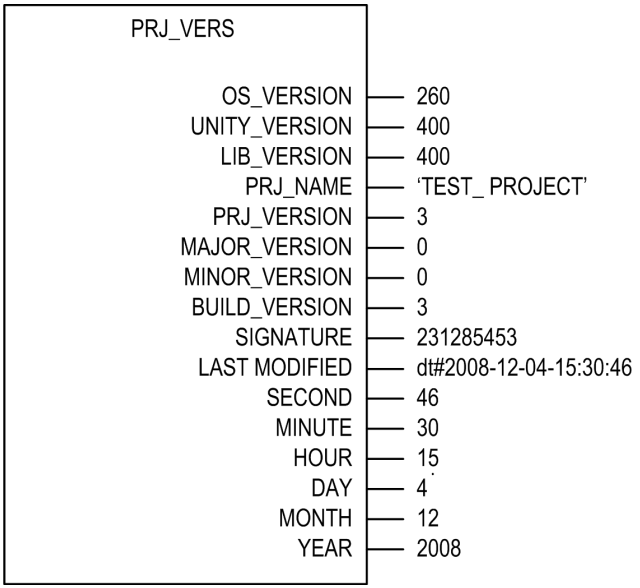
La fonction `PRJ_VERS` vous permet d'obtenir des informations sur le projet ouvert dans Unity Pro.

Pour accéder à ces informations dans Unity Pro :

Etape	Action
1	Dans le menu **Outils**, cliquez sur **Ecran de l'automate**. **Résultat** : la fenêtre **Ecran de l'automate** apparaît.
2	Cliquez sur l'onglet **Informations**.
3	Dans la partie gauche de la fenêtre **Ecran de l'automate**, cliquez sur **INFORMATIONS SYSTEME → APPLICATION → INFORMATION**. **Résultat** : les informations concernant l'application s'affichent dans la partie droite de la fenêtre **Ecran de l'automate**.

Représentation en FBD

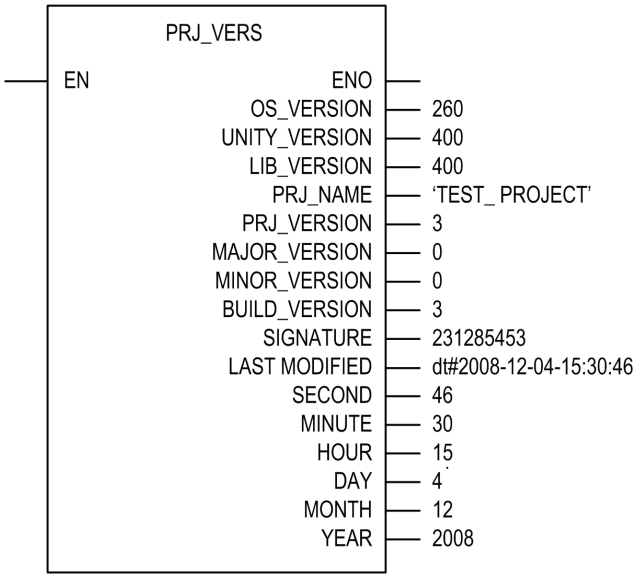
La représentation en FBD de la fonction PRJ_VERS (avec des données exemples) est la suivante :



NOTE : pour plus d'informations, reportez-vous à la section Description des paramètres *(voir page 110)*.

Représentation en LD

La représentation en LD de la fonction PRJ_VERS (avec des données exemples) est la suivante :



Représentation en IL

La représentation en IL de la fonction PRJ_VERS est la suivante :

```
CAL FBI_PRJ_VERS (
 OS_VERSION :=> OS_VERSION,
 UNITY_VERSION := > UNITY_VERSION,
 LIB_VERSION := > LIB_VERSION,
 PRJ_NAME := > PRJ_NAME,
 PRJ_VERSION := > PRJ_VERSION,
 MAJOR_VERSION :=> MAJOR_VERSION,
 MINOR_VERSION :=> MINOR_VERSION,
 BUILD_VERSION :=> BUILD_VERSION,
 SIGNATURE :=> SIGNATURE,
 LAST_MODIFIED :=> LAST_MODIFIED,
 SECOND :=> SECOND,
 MINUTE :=> MINUTE,
 HOUR :=> HOUR,
```

DAY :=> DAY,
MONTH :=> MONTH,
YEAR :=> YEAR
)

Représentation en ST

La représentation en ST de la fonction PRJ_VERS est la suivante :

```
FBI_PRJ_VERS(OS_VERSION, UNITY_VERSION, LIB_VERSION, PRJ_NAME,
PRJ_VERSION, MAJOR_VERSION, MINOR_VERSION, BUILD_VERSION, SIGNATURE,
LAST_MODIFIED, SECOND, MINUTE, HOUR, DAY, MONTH, YEAR);
```

Description des paramètres

Le tableau suivant décrit les paramètres de **out** de PRJ_VERS :

Paramètre	Type	Commentaire
OS_VERSION	INT*100	Version du processeur
UNITY_VERSION	INT*100	Version d'Unity Pro
LIB_VERSION	INT*100	Version de LibSet
PRJ_NAME	Chaîne[68]	Nom du projet
PRJ_VERSION	UDINT	Version du projet
MAJOR_VERSION	INT	Version de projet (majeure)
MINOR_VERSION	INT	Version de projet (mineure)
BUILD_VERSION	INT	N° de compilation
SIGNATURE	UDINT	Signature de version
LAST_MODIFIED	DT	Dernière compilation
SECOND	INT	Date de modification de la compilation (secondes)
MINUTE	INT	Date de modification de la compilation (minutes)
HOUR	INT	Date de modification de la compilation (heures)
DAY	INT	Date de modification de la compilation (jour)
MONTH	INT	Date de modification de la compilation (mois)
YEAR	INT	Date de modification de la compilation (année)

Partie III
Traitement d'événements

Aperçu

Cette partie décrit les fonctions de base et les blocs fonction de base de la famille `Traitement d'événements`.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 23
HALT : arrêt du programme

Description

Description de la fonction

Utilisée dans un programme d'application, la fonction HALT bloque l'exécution de ce dernier. Toutes les tâches sont arrêtées et les objets variables sont figés.

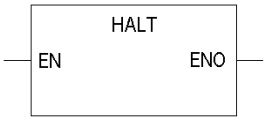
Pour reprendre son exécution, le programme doit être initialisé (à l'aide de la commande **INIT**). Les instructions qui suivent l'instruction **HALT** ne sont donc pas exécutées.

NOTE : lorsque l'automate est en HALT, les tâches sont arrêtées *(voir Unity Pro, Modes de marche)*. Vérifiez le comportement des E/S associées.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

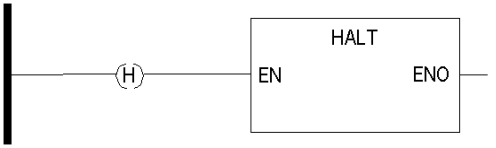
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
HALT
```

Représentation en ST

Représentation :

```
HALT();
```

Chapitre 24
ITCNTRL : déclenchement du traitement événementiel de type TIMER

Description

Description de la fonction

La fonction ITCNTRL est un temporisateur qui déclenche le traitement d'événements de type TIMER *(voir Unity Pro, Langages de programmation et structure, Manuel de référence)* sélectionnés par l'entrée EVENT, lorsque la valeur courante atteint la valeur de présélection.

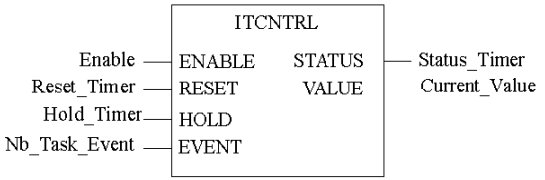
Les valeurs de présélection et de base de temps sont sélectionnées dans la boîte de dialogue des propriétés de traitement des événements *(voir Unity Pro, Modes de marche)*.

NOTE : le traitement d'événements TIMER n'est pas disponible sur tous les automates. Consultez la liste des fonctions disponibles pour chaque type d'automate.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

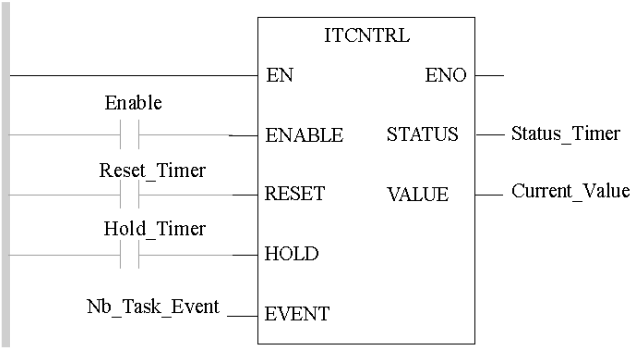
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Enable
ITCNTRL Reset_Timer, Hold_Timer, Nb_task_Event, Status_Timer,
Current_Value
```

Représentation en ST

Représentation :

```
ITCNTRL(ENABLE := Enable_Timer, RESET := Reset_Timer, HOLD := Hold_Timer,
EVENT := Nb_Task_Event, STATUS => Status_Timer, VALUE => Current_Value);
```

Description des paramètres

Le tableau ci-après décrit les paramètres d'entrée.

Paramètre	Type	Commentaire
`Enable`	BOOL	Activation de l'entrée sélectionnée A l'état 1 : le traitement des événements est déclenché lorsque le délai du temporisateur est écoulé. A l'état 0 : aucun événement n'est émis.
`Reset_Timer`	BOOL	A l'état 1 : réinitialise le temporisateur.
`Hold_Timer`	BOOL	A l'état 1 : bloque l'incrémentation du temporisateur.
`Nb_Task_Event`	BYTE	Mot d'entrée déterminant le numéro de traitement de l'événement TIMER à déclencher.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
`Status_Timer`	WORD	Mot d'état : ● Bit 0 = 1 : exécution différée par un masquage de l'interruption. ● Bit 1 = 1 : numéro de traitement de l'événement non valide. ● Bit 2 = 1 : temporisateur validé (`activation` de l'image d'entrée). ● Bit 3 = 1 : temporisateur figé (`interruption du temporisateur` de l'image d'entrée). ● Bit 4 = 1 : dès que ITCNTRL est appelé la première fois avec l'entrée `Reset_Timer` ou `Hold_Timer` à 1 (mode déphasé). Il est remis à 0 en cas de démarrage à froid. ● Bit 5 = 1 : pile de mémoire FIFO des interruptions saturées.
`Current_Value`	TIME	Valeur courante du temporisateur. Cette valeur passe de 0 à la valeur de présélection. Lorsque la valeur de présélection est atteinte, elle revient à 0. Si le traitement d'événement de type TIMER est confirmé, il est exécuté.

Chapitre 25
MASKEVT : masquage global des événements

Description

Description de la fonction

La fonction MASKEVT effectue un masquage global des événements (1).

Les événements sont stockés dans la mémoire de l'automate, mais toute tâche de traitement de l'événement associé reste inactive, tant que l'opération de masquage est valide, jusqu'à la prochaine instruction UNMASKEVT.

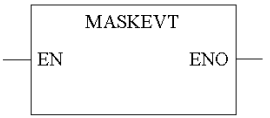
NOTE : Il est recommandé d'utiliser cette instruction uniquement pour une séquence de masquage courte, dans le même cycle de tâche. Dans les autres cas, vous devez utiliser %S38. Les événements sont automatiquement démasqués par le système lors du changement de mode (Stop -> Run, ...). Il est déconseillé d'exécuter les séquences de masquage lorsque des temporisateurs d'événements sont actifs, car ils sont remis à 0.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

(1) excepté les interruptions générées par des télégrammes FIPWAY.

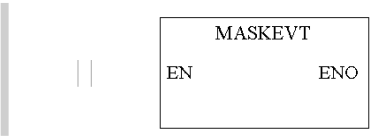
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
MASKEVT
```

Représentation en ST

Représentation :

```
MASKEVT();
```

Chapitre 26
UNMASKEVT : démasquage global des événements

Description

Description de la fonction

La fonction UNMASKEVT effectue un démasquage global des événements.

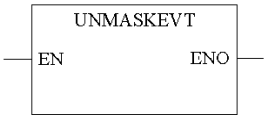
Les événements stockés au cours de la période de masquage sont traités. Le traitement événementiel fonctionne jusqu'à la prochaine instruction MASKEVT.

NOTE : Il est recommandé d'utiliser cette instruction uniquement pour une séquence de masquage courte, dans le même cycle de tâche. Dans les autres cas, vous devez utiliser %S38. Les événements sont automatiquement démasqués par le système lors du changement de mode (Stop -> Run, ...). Il est déconseillé d'exécuter les séquences de masquage lorsque des temporisateurs d'événements sont actifs, car ils sont remis à 0.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

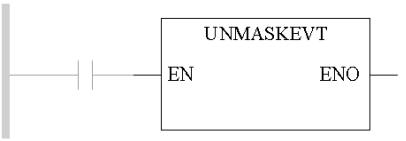
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
UNMASKEVT
```

Représentation en ST

Représentation :

```
UNMASKEVT();
```

Partie IV
Hot Stand By

Aperçu

Cette partie décrit les fonctions de base et les blocs fonction de base de la famille `Hot Stand By`.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 27
HSBY_BUILD_OFFLINE : Différence de génération en local autorisée

HSBY_BUILD_OFFLINE

Description de la fonction

Le bloc fonction `HSBY_BUILD_OFFLINE` permet d'effectuer des modifications en mode local dans un programme d'application d'un système Modicon M580 à redondance d'UC. Ce bloc fonction permet ou non les différences de génération en local.

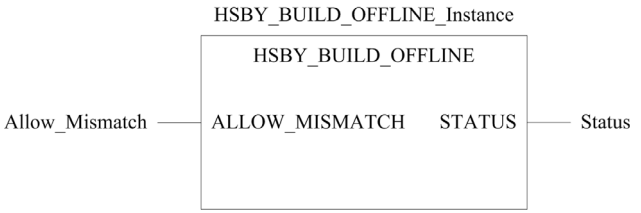
Les paramètres supplémentaires `EN` et `ENO` peuvent aussi être configurés.

NOTE : Consultez la documentation sur la Redondance d'UC Modicon M580 pour plus d'informations sur la modification d'application en local avec différence de génération en local autorisée *(voir Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures)*.

NOTE : La fonction `HSBY_BUILD_OFFLINE` est disponible sur les UC avec micrologiciel SE version 2.15 ou ultérieure.

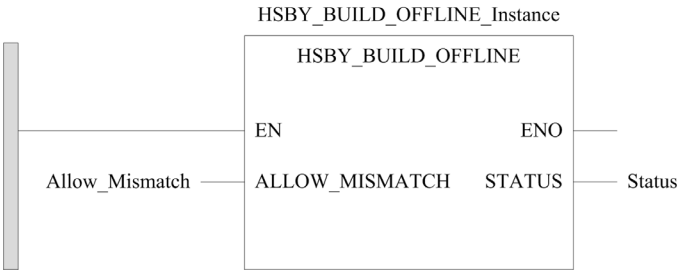
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL HSBY_BUILD_OFFLINE_Instance
(ALLOW_MISMATCH:=Allow_Mismatch, STATUS=>Status)
```

Représentation en ST

Représentation :

```
HSBY_BUILD_OFFLINE_Instance
(ALLOW_MISMATCH:=Allow_Mismatch, STATUS=>Status)
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
`Allow_Mismatch`	`BOOL`	Description des paramètres : **1 :** Autoriser une différence de génération en local. **0 :** Ne pas autoriser de différence de génération en local.

Le tableau suivant décrit le paramètre de sortie :

Paramètre	Type	Commentaire
`Status`	`WORD`	Rapport d'état : **0000 hex** Opération effectuée avec succès. **0001 hex :** La version EFB n'est pas compatible avec la version du micrologiciel de l'UC. **0002 hex :** L'application n'est pas une application à Redondance d'UC.

Description des paramètres `ENO` optionnels :

Valeur `ENO`	Description
1	Opération effectuée avec succès. Le paramètre de sortie `Status` a pour valeur 0.
0	Opération non traitée. Le paramètre de sortie `Status` n'est pas égal à 0 comme décrit dans le tableau de sortie précédent.

Chapitre 28
HSBY_RD : lecture du registre de la commande de redondance d'UC

HSBY_RD

Description de la fonction

Cet EFB vous permet d'utiliser la fonction de redondance d'UC. Il parcourt (avec les autres EFB de redondance d'UC) la configuration des automates Quantum à la recherche des composants qui lui sont nécessaires.

Ces composants correspondent à du matériel réellement connecté. Par conséquent, il est impossible de garantir que cet EFB se comportera comme convenu sur les simulateurs.

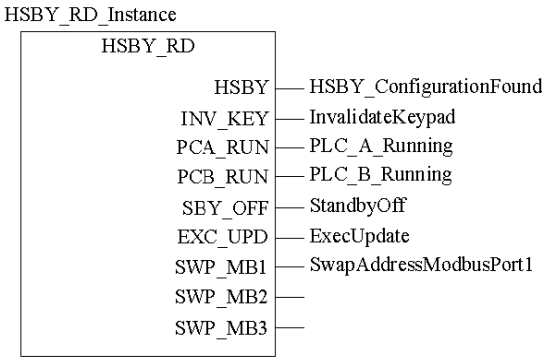
L'EFB **HSBY_RD** examine le mot système %SW60 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* pour vérifier la présence d'une configuration de redondance d'UC :

- En présence d'une configuration de redondance d'UC, le contenu du registre de commande est renvoyé et le paramètre de sortie `HSBY_ConfigurationFound` est mis à 1.
- Dans le cas contraire, le contenu du registre de commande est renvoyé et le paramètre de sortie `HSBY_ConfigurationFound` est mis à 0.

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

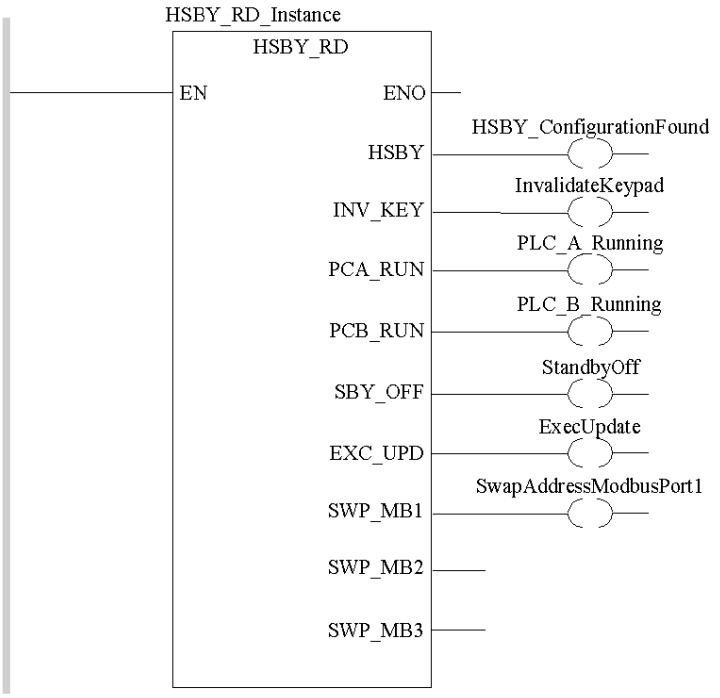
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL HSBY_RD_Instance (HSBY=>HSBY_ConfigurationFound,
        INV_KEY=>InvalidateKeypad, PCA_RUN=>PLC_A_Running,
        PCB_RUN=>PLC_B_Running, SBY_OFF=>StandbyOff,
        EXC_UPD=>ExecUpdate, SWP_MB1=>SwapAddressModbusPort1)
```

Représentation en ST

Représentation :

```
HSBY_RD_Instance (HSBY=>HSBY_ConfigurationFound,
        INV_KEY=>InvalidateKeypad, PCA_RUN=>PLC_A_Running,
        PCB_RUN=>PLC_B_Running, SBY_OFF=>StandbyOff,
        EXC_UPD=>ExecUpdate,
        SWP_MB1=>SwapAddressModbusPort1);
```

Description des paramètres

Description des paramètres de sortie :

Paramètre	Type de données	Signification
HSBY	BOOL	1 = Configuration de redondance d'UC trouvée 0 = Configuration de redondance d'UC introuvable
INV_KEY	BOOL	1 = Le sous-menu du bouton de l'automate du système de redondance d'UC est désactivé. 0 = Le sous-menu du bouton de l'automate du système de redondance d'UC n'est pas désactivé.
PCA_RUN	BOOL	Pour l'automate du rack local avec l'UC A du système de redondance d'UC : 1 = Le registre de commande est sélectionné pour le mode RUN. 0 = Le registre de commande est sélectionné pour le mode LOCAL.
PCB_RUN	BOOL	Pour l'automate du rack local avec l'UC B du système de redondance d'UC : 1 = Le registre de commande est sélectionné pour le mode RUN. 0 = Le registre de commande est sélectionné pour le mode LOCAL.
SBY_OFF	BOOL	1 = ??? 0 = L'automate redondant passe en mode LOCAL dès que les deux automates reçoivent un programme différent.
EXC_UPD	BOOL	1 = Le système d'exploitation de l'automate redondant peut être mis à jour alors que l'automate primaire est en cours d'exécution. 0 = ??? (Une fois le système d'exploitation mis à jour, l'automate redondant repasse en mode CONNECTE.)
SWP_MB1	BOOL	Conséquence d'un basculement pour les ports Modbus 1 : 1 = Aucune permutation des adresses 0 = Permutation des adresses
SWP_MB2	BOOL	Non utilisé. Réservé.
SWP_MB3	BOOL	Non utilisé. Réservé.

Chapitre 29
HSBY_ST : lecture du registre d'état de redondance d'UC

HSBY_ST

Description de la fonction

Cet EFB vous permet d'utiliser la fonction de redondance d'UC. Il parcourt (avec les autres EFB de redondance d'UC) la configuration des automates Quantum à la recherche des composants qui lui sont nécessaires.

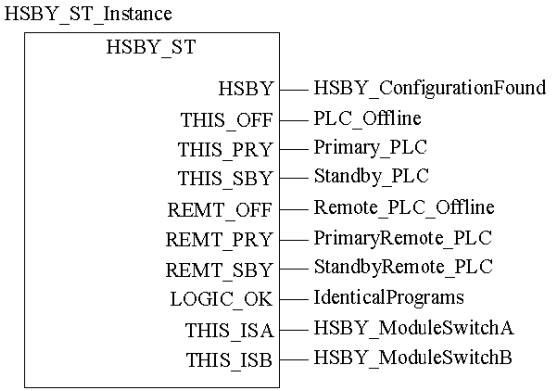
Ces composants correspondent à du matériel réellement connecté. Par conséquent, il est impossible de garantir que cet EFB se comportera comme convenu sur les simulateurs.

Cet EFB permet de lire le registre d'état du système de redondance d'UC CEI, à savoir %SW61 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*. En l'absence de configuration de redondance d'UC, la sortie `HSBY` est mise à 0.

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

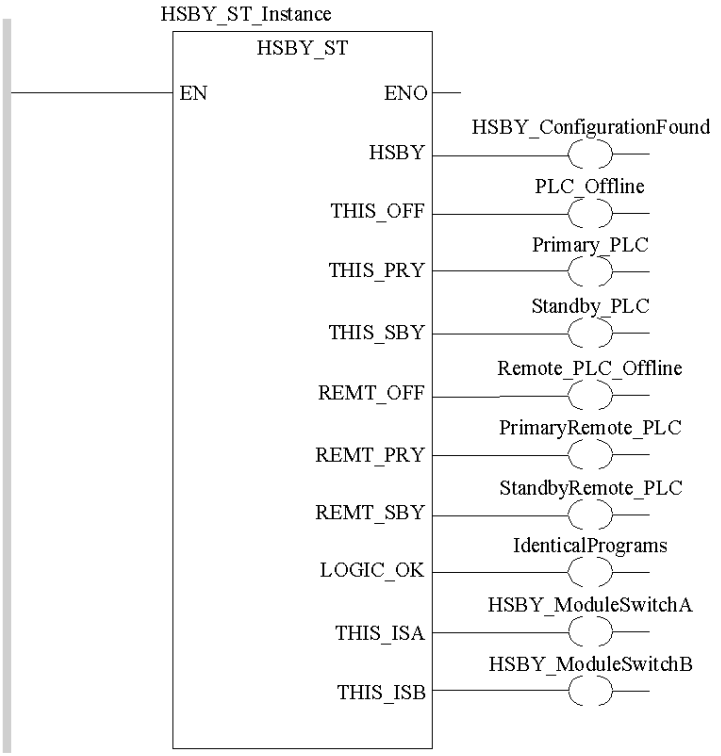
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL HSBY_ST_Instance (HSBY=>HSBY_ConfigurationFound,
         THIS_OFF=>PLC_Offline, THIS_PRY=>Primary_PLC,
         THIS_SBY=>Standby_PLC,
         REMT_OFF=>Remote_PLC_Offline,
         REMT_PRY=>PrimaryRemote_PLC,
         REMT_SBY=>StandbyRemote_PLC,
         LOGIC_OK=>IdenticalPrograms,
         THIS_ISA=>HSBY_ModuleSwitchA,
         THIS_ISB=>HSBY_ModuleSwitchB)
```

Représentation en ST

Représentation :

```
HSBY_ST_Instance (HSBY=>HSBY_ConfigurationFound,
         THIS_OFF=>PLC_Offline, THIS_PRY=>Primary_PLC,
         THIS_SBY=>Standby_PLC,
         REMT_OFF=>Remote_PLC_Offline,
         REMT_PRY=>PrimaryRemote_PLC,
         REMT_SBY=>StandbyRemote_PLC,
         LOGIC_OK=>IdenticalPrograms,
         THIS_ISA=>HSBY_ModuleSwitchA,
         THIS_ISB=>HSBY_ModuleSwitchB);
```

Description des paramètres

Description des paramètres de sortie :

Paramètre	Type de données	Signification
HSBY	BOOL	1 = Configuration de redondance d'UC trouvée 0 = Configuration de redondance d'UC introuvable
THIS_OFF	BOOL	1 = Cet automate est en mode Local. 0 = Cet automate n'est pas en mode Local.
THIS_PRY	BOOL	1 = Cet automate est l'automate primaire. 0 = Cet automate n'est pas l'automate primaire.
THIS_SBY	BOOL	1 = Cet automate est l'automate redondant. 0 = Cet automate n'est pas l'automate redondant.
REMT_OFF	BOOL	1 = L'autre automate (redondant) est en mode LOCAL. 0 = L'autre automate (redondant) n'est pas en mode LOCAL.
REMT_PRY	BOOL	1 = L'autre automate est l'automate primaire. 0 = L'autre automate n'est pas l'automate primaire.
REMT_SBY	BOOL	1 = L'autre automate est l'automate redondant. 0 = L'autre automate n'est pas l'automate redondant.
LOGIC_OK	BOOL	1 = Les programmes sont identiques sur les deux automates et la non-correspondance des applications est active. 0 = Les programmes diffèrent.

Paramètre	Type de données	Signification
THIS_ISA	BOOL	1 = Parmi les deux UC du système de redondance d'UC, cet automate sélectionne celle associée à l'adresse IP inférieure. Il s'agit de l'UC A. 0 = Il ne s'agit pas de l'UC A.
THIS_ISB	BOOL	1 = Parmi les deux UC du système de redondance d'UC, cet automate sélectionne celle associée à l'adresse IP supérieure. Il s'agit de l'UC B. 0 = Il ne s'agit pas de l'UC B.

Chapitre 30
HSBY_SWAP : déclenchement de l'échange entre l'UC primaire et l'UC redondante

Utilisation du système à redondance d'UC avec le bloc fonction HSBY_SWAP DFB

Description de la fonction

Le bloc fonction `HSBY_SWAP` permet de basculer entre l'UC primaire et l'UC redondante.

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

Ce bloc fonction permet de déclencher une permutation par programme. Cette permutation entre l'UC primaire et l'UC redondante ne peut s'effectuer qu'en mode Redondance d'UC.

Consultez les documents suivants :

- Modicon Premium, Système de redondance d'UC avec Unity, Manuel utilisateur *(voir Premium, Système de redondance d'UC avec Unity, Manuel utilisateur)* pour plus d'informations sur les UC redondantes Premium.
  Modicon Quantum, Système de redondance d'UC avec Unity, Manuel utilisateur *(voir Modicon Quantum, Système de redondance d'UC, Manuel utilisateur)* pour plus d'informations sur les UC redondantes Quantum.

De fait, lorsque le bloc fonction HSBY est exécuté, l'automate redondant devient l'automate primaire et l'ancien automate primaire devient l'automate redondant activé par le logiciel du programme.

NOTE : il est impératif d'exécuter cette fonction avant d'effectuer une permutation à chaud. En fait, vous pouvez utiliser le registre `%SW60`, comme indiqué dans la section Conditions déclenchant un basculement *(voir Premium, Système de redondance d'UC avec Unity, Manuel utilisateur)* pour les UC Premium ou la section Conditions déclenchant un basculement *(voir Modicon Quantum, Système de redondance d'UC, Manuel utilisateur)* pour les UC Quantum.

NOTE : n'utilisez pas le DFB dans la première section de l'application.

⚠ AVERTISSEMENT

COMPORTEMENT INATTENDU DE L'EQUIPEMENT

Le DFB HSBY_SWAP ne doit être appelé que pour tester l'application.

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

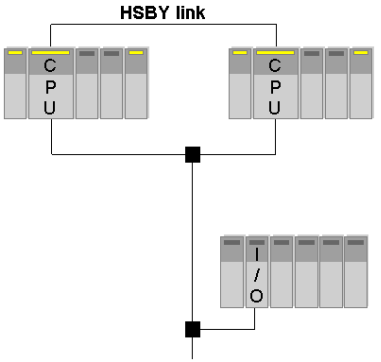
Avantage de la fonction de permutation

Les avantages de la permutation sont les suivants :

- L'intégrité de l'automate redondant est surveillée. La capacité de l'automate redondant à prendre le relais fait l'objet d'une vérification.
- La permutation peut être testée à intervalles réguliers.

Exemple d'une application à redondance d'UC

L'illustration suivante montre un exemple d'application à redondance d'UC :



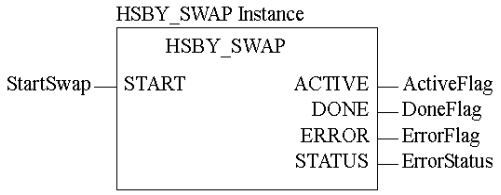
Etapes de la modification de l'état

Une fonction de permutation s'exécute comme suit :

Etape	Action
1	**Etat :** l'automate A est le contrôleur primaire, et l'automate B est le contrôleur redondant. L'automate A se met en mode local. **Résultat :** L'automate B devient le contrôleur primaire.
2	**Etat :** l'automate A est en mode local, et l'automate B est le contrôleur primaire. L'automate B fait passer l'automate A en mode RUN. **Résultat :** L'automate A est le contrôleur redondant.
3	**Etat :** l'automate A est le contrôleur redondant, et l'automate B est le contrôleur primaire. Les sorties de l'EFB sont définies. **Résultat :** L'exécution de la fonction de permutation est terminée.

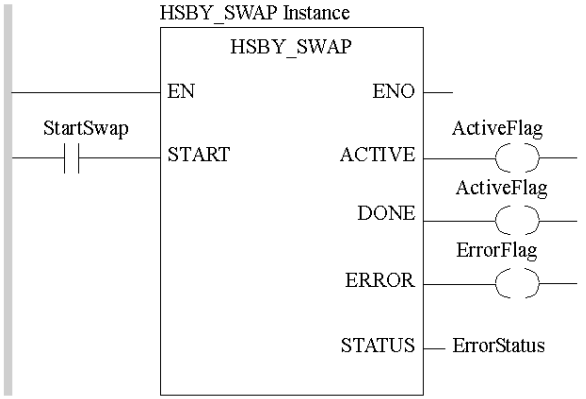
Représentation en FBD

Représentation



Représentation en LD

Représentation



Description des paramètres

Description du paramètre d'entrée :

Paramètre	Type de données	Signification
START	BOOL	START = 1 lance l'opération HSBY_SWAP. La valeur 1 doit être appliquée jusqu'à ce que l'opération soit terminée ou qu'une erreur intervienne.

Description des paramètres de sortie

Paramètre	Type de données	Signification
ACTIVE	BOOL	ACTIVE = 1 indique qu'une opération HSBY_SWAP est en cours.
DONE	BOOL	DONE = 1 indique que l'opération HSBY_SWAP a abouti.
ERROR	BOOL	ERROR = 1 indique qu'une erreur est apparue ou que l'opération HSBY_SWAP en cours a été abandonnée.
STATUS	INT	Le bloc HSBY_SWAP génère un code d'erreur (STATUS). Le tableau suivant dresse la liste de tous les codes d'erreur.

Tableau des codes d'erreur

Le tableau suivant explique les codes d'erreur :

Codes d'erreur	Description du défaut
0	OK
1	La fonction HSBY_SWAP a été abandonnée.
2	La redondance d'UC n'a pas été activée (%SW61.15=0).
3	L'unité redondante n'existe pas.
5	La permutation a échoué.

NOTE : les mots système %SW60 et %SW61 indiquent l'état de l'automate primaire et de l'automate redondant.

Basculement à l'aide du bit système %SW60.1 ou %SW60.2 du registre de commande

Une autre façon de forcer un basculement consiste à définir les bits du registre de commande. Pour ce faire, procédez comme suit :

Etape	Action
1	Ouvrez le fichier 1.
2	Connectez-vous au contrôleur primaire.
3	Identifiez le contrôleur primaire (A ou B).
4	Accès ● Bit système %SW60.1 du registre de commande Si le contrôleur connecté est le A. ● Bit système %SW60.2 du registre de commande Si le contrôleur connecté est le B.
5	Réglez le bit sur 0. **NOTE :** vérifiez que l'UC redondante est devenue l'UC primaire.
6	Ouvrez le fichier 2.
7	Connectez au nouveau contrôleur primaire.
8	Accédez au bit système du registre de commande, utilisé à l'étape 4.
9	Réglez le bit sur 1. **NOTE :** vérifiez que le contrôleur redondant est maintenant en mode connecté.
10	Vérifiez que les contrôleurs primaire et redondant sont en mode Run primaire et Run redondant.

Chapitre 31
HSBY_WR : écriture dans le registre de la commande de redondance d'UC

HSBY_WR

Description de la fonction

Cet EFB vous permet d'utiliser la fonction de redondance d'UC. Il parcourt (avec les autres EFB de redondance d'UC) la configuration des automates Quantum à la recherche des composants qui lui sont nécessaires.

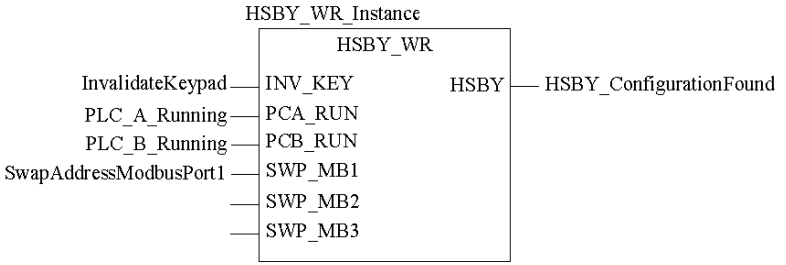
Ces composants correspondent à du matériel réellement connecté. Par conséquent, il est impossible de garantir que cet EFB se comportera comme convenu sur les simulateurs.

`HSBY_WR` permet de définir différents modes de redondance d'UC pour l'UC primaire. L'application de chacun de ces modes entraîne la modification du registre de commande %SW60 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* du système de redondance d'UC, cette opération étant réalisée automatiquement par le bloc fonction. En l'absence de configuration de redondance d'UC, la sortie `HSBY_ConfigurationFound` est mise à 0. Sinon, elle est mise à 1.

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

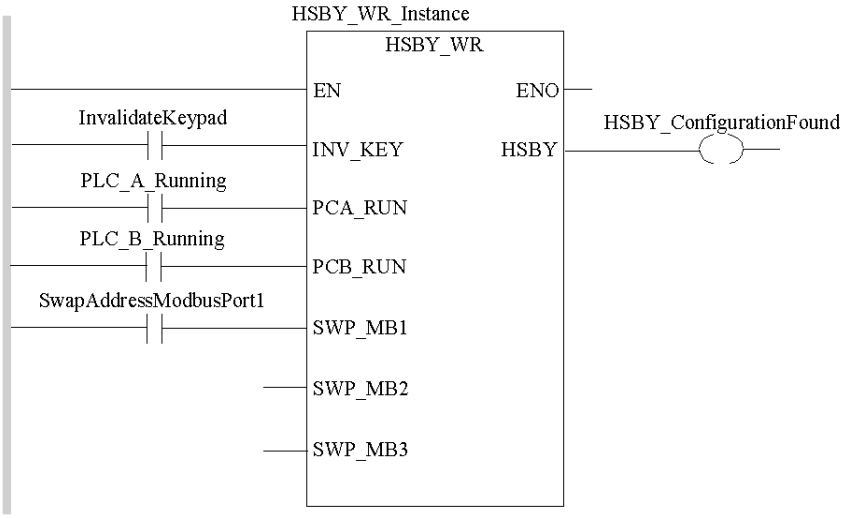
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL HSBY_WR_Instance (INV_KEY:=InvalidateKeypad,
         PCA_RUN:=PLC_A_Running, PCB_RUN:=PLC_B_Running,
         SWP_MB1:=SwapAddressModbusPort1,
         HSBY=>HSBY_ConfigurationFound)
```

Représentation en ST

Représentation :

```
HSBY_WR_Instance (INV_KEY:=InvalidateKeypad,
         PCA_RUN:=PLC_A_Running, PCB_RUN:=PLC_B_Running,
         SWP_MB1:=SwapAddressModbusPort1,
         HSBY=>HSBY_ConfigurationFound);
```

Description des paramètres

Description des paramètres d'entrée :

Paramètre	Type de données	Signification
INV_KEY	BOOL	Dans le sous-menu du bouton de l'automate du système de redondance d'UC : 1 = Les modifications ne sont pas autorisées. 0 = Les modifications sont autorisées.
PCA_RUN	BOOL	Sur un front descendant (1 -> 0), l'UC A du rack local est forcée en mode LOCAL. Sur un front montant (0 -> 1) et si le mode de bouton est RUN, l'UC A est forcée en mode RUN.
PCB_RUN	BOOL	Sur un front descendant (1 -> 0), l'UC B du rack local est forcée en mode LOCAL. Sur un front montant (0 -> 1) et si le mode de bouton est RUN, l'UC B est forcée en mode RUN.
SWP MB1	BOOL	En cas de basculement alors que le paramètre est à 0, l'adresse Modbus du port 1 du NOUVEL automate primaire change comme suit : ● Adresse du nouvel automate primaire = adresse de l'ancienne UC primaire ● Adresse du nouvel automate redondant = adresse de la nouvelle UC primaire + 128
		En cas de basculement alors que le paramètre est à 1, l'adresse Modbus du port 1 de l'automate reste inchangée : ● Adresse du nouvel automate primaire = adresse de l'ancienne UC redondante ● Adresse du nouvel automate redondant = adresse de l'ancienne UC primaire
SWP_MB2	BOOL	Non utilisé. Réservé.
SWP MB3	BOOL	Non utilisé. Réservé.

Description des paramètres de sortie :

Paramètre	Type de données	Signification
HSBY	BOOL	1 = Configuration de redondance d'UC trouvée 0 = Configuration de redondance d'UC introuvable

Chapitre 32
REV_XFER : écriture et lecture des deux registres de transfert inverse

REV_XFER

Description de la fonction

Cet EFB vous permet d'utiliser la fonction de redondance d'UC. Il parcourt (avec les autres EFB de redondance d'UC) la configuration des automates Quantum à la recherche des composants qui lui sont nécessaires. Ces composants correspondent à du matériel réellement connecté.

L'EFB `REV_XFER` permet d'envoyer deux registres (%SW62/63) de l'automate redondant vers l'automate primaire. Les deux registres EFB sont utilisés par le programme d'application (dans la première section) pour enregistrer les informations du diagnostic.

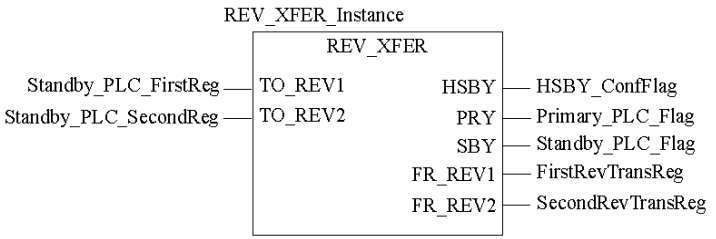
La fonction `REV_XFER` ne peut être utilisée que dans la première section exécutable du projet. Les adresses de paramètre `TO_REV1` et `TO_REV2` doivent se trouver dans la zone de non-transfert pour éviter d'être remplacées par l'UC primaire.

NOTE : lorsque l'automate est en mode local, aucune donnée n'est envoyée lors du transfert inverse, car la condition de redondance d'UC n'est plus valide.

`EN` et `ENO` sont projetés en tant que paramètres supplémentaires.

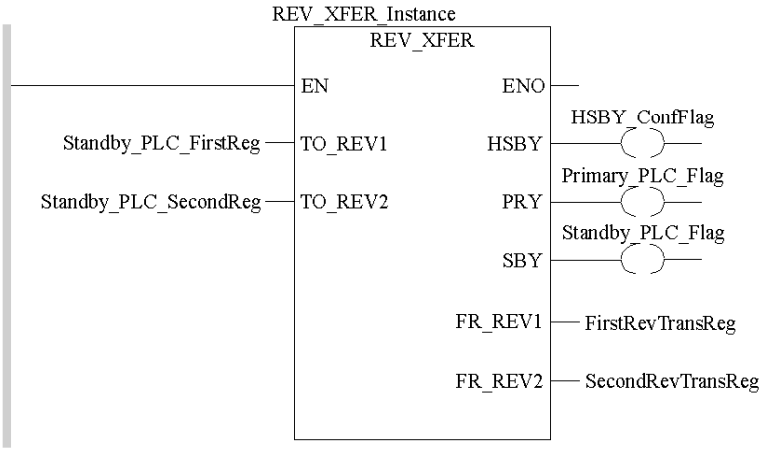
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL REV_XFER_Instance (TO_REV1:=Standby_PLC_FirstReg,
         TO_REV2:=Standby_PLC_SecondReg, HSBY=>HSBY_ConfFlag,
         PRY=>Primary_PLC_Flag, SBY=>Standby_PLC_Flag,
         FR_REV1=>FirstRevTransReg,
         FR_REV2=>SecondtRevTransReg)
```

Représentation en ST

Représentation :

```
REV_XFER_Instance (TO_REV1:=Standby_PLC_FirstReg,
         TO_REV2:=Standby_PLC_SecondReg, HSBY=>HSBY_ConfFlag,
         PRY=>Primary_PLC_Flag, SBY=>Standby_PLC_Flag,
         FR_REV1=>FirstRevTransReg,
         FR_REV2=>SecondtRevTransReg);
```

Description des paramètres

Description des paramètres d'entrée :

Paramètre	Type de données	Description
TO_REV1	INT	Décrit le premier registre de transfert inverse si l'automate est l'automate redondant. Les données de ce registre sont transférées de l'UC redondante vers l'UC primaire à chaque scrutation.
TO_REV2	INT	Décrit le second registre de transfert inverse si l'automate est l'UC redondante. Les données de ce registre sont transférées de l'UC redondante vers l'UC primaire à chaque scrutation.

Description des paramètres de sortie :

Paramètre	Type de données	Description
HSBY	BOOL	1 = Il s'agit d'une configuration de redondance d'UC. 0 = Il ne s'agit pas d'une configuration de redondance d'UC.
PRY	BOOL	1 = Cet automate est l'automate de l'UC primaire. 0 = Cet automate n'est pas l'automate de l'UC primaire.
SBY	BOOL	1 = Cet automate est l'automate de l'UC redondante. 0 = Cet automate n'est pas l'automate de l'UC redondante.
FR_REV1	INT	Contenu du premier registre de transfert inverse %SW62 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*. Sortie uniquement si HSBY = 1.
FR_REV2	INT	Contenu du second registre de transfert inverse %SW63 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*. Sortie uniquement si HSBY = 1.

Partie V
Gestion SFC

Aperçu

Cette partie décrit les fonctions de base et les blocs fonction de base de la famille `Gestion SFC`.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 33
CLEARCHART : réinitialisation de toutes les étapes actives

Description

Description de la fonction

Cette fonction sert à réinitialiser des graphes d'état.

⚠ AVERTISSEMENT

ERREUR SYSTEME INATTENDUE

N'utilisez pas `CLEARCHART` pour rechercher des erreurs sur des automates de machines-outils, de processus ou de systèmes de gestion des matières en cours de fonctionnement.

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

⚠ AVERTISSEMENT

COMPORTEMENT IMPREVU DE L'EQUIPEMENT

Evitez d'utilisez la commande `SFCCNTRL` avec les commandes `CLEARCHART`, `FREEZCHART`, `INITCHART`, `SETSTEP` et `RESETSTEP`.

Vérifiez soigneusement les interactions entre ces blocs fonction et analysez l'impact sur le SFC.

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

Un signal 1 à l'entrée `ClearSequence` permet d'arrêter le graphe d'état et de réinitialiser toutes les étapes. Tant que le signal 1 à l'entrée est présent, le graphe d'état reste dans ce mode. Cela signifie qu'il n'est pas possible de le démarrer.

Cet état est maintenu même si l'entrée `ClearSequence` revient à 0.

Un signal 1 à la sortie `ClearState` permet d'afficher l'état (graphe d'état réinitialisé).

Ce n'est que lorsqu'un signal 0 à l'entrée `ClearSequence` est à nouveau présent que le graphe d'état peut être démarré par l'intermédiaire du bloc fonction `SFCCNTRL` *(voir page 172)* (entrée `INIT`), de la fonction `INITCHART` *(voir page 161)* ou d'une autre commande d'activation d'étape externe.

Comme le graphe d'état est sans cesse réinitialisé tant qu'un signal 1 à l'entrée `ClearSequence` est présent, il est conseillé de prendre l'une des mesures suivantes afin d'empêcher un blocage continu du graphe d'état :

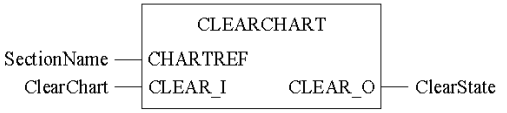
- Appel conditionnel de la fonction `CLEARCHART` par l'entrée `EN`.
- Appel conditionnel de la fonction `CLEARCHART`, par exemple par l'instruction `IF` dans le langage de programmation `ST`.
- Utilisation d'une détection de front (`R_TRIG`) à l'entrée `ClearSequence`.

NOTE : la sortie `ClearState` affiche l'état en cours du graphe d'état. Cela signifie que le bloc fonction `SFCCNTRL` *(voir page 172)* (entrée `CLEAR`), la fonction `INITCHART` *(voir page 161)*, la procédure `RESETSTEP` *(voir page 165)* (réinitialisation de la dernière/seule étape active) ou d'autres commandes externes peuvent également réinitialiser le graphe d'état.

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

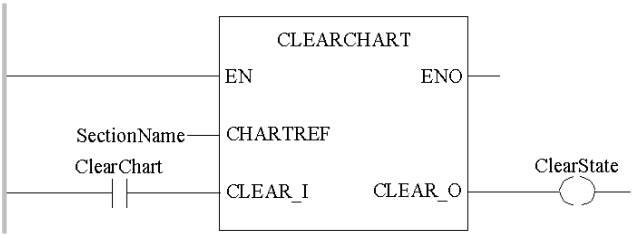
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD SectionName CLEARCHART ClearChart ST ClearState
```

Représentation en ST

Représentation :

```
ClearState := CLEARCHART (SectionName, ClearChart) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Description
CHARTREF	SFCCHART_STATE	Affectation de la section SFC à commander par l'intermédiaire du nom de la section. Lorsqu'une section SFC est créée, une variable du type de données SFCCHART_STATE lui est automatiquement affectée. La variable ainsi créée porte toujours le nom de la section SFC correspondante.
CLEAR_I	BOOL	0->1: Réinitialisation de toutes les étapes actives du graphe d'état.

Description des paramètres de sortie :

Paramètre	Type de données	Description
CLEAR_O	BOOL	1: Graphe d'état réinitialisé, ce qui signifie que le graphe d'état n'a aucune étape active.

Chapitre 34
FREEZECHART : gel d'un graphe d'état

Description

Description de la fonction

Cette fonction sert à « geler » des graphes d'état (l'évaluation des transitions est arrêtée).

⚠ AVERTISSEMENT

COMPORTEMENT INATTENDU DU SYSTEME

N'utilisez pas `FREEZECHART` pour rechercher des erreurs sur des automates de machines-outils, de processus ou de systèmes de gestion des matières en cours de fonctionnement.

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

⚠ AVERTISSEMENT

COMPORTEMENT INATTENDU DE L'EQUIPEMENT

Evitez d'utilisez la commande `SFCCNTRL` avec les commandes `CLEARCHART`, `FREEZCHART`, `INITCHART`, `SETSTEP` et `RESETSTEP`.

Vérifiez soigneusement les interactions entre ces blocs fonction et analysez l'impact sur le SFC.

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

En cas de signal 1 à l'entrée `FreezeSequence`, l'état en cours du graphe d'état est gelé. Les états des transitions ne sont plus évalués. De ce fait, un enchaînement n'est plus possible, même si la condition de la transition « active » est vraie.

Cette fonction peut être utilisée conjointement avec les fonctions destinées au traitement pas à pas (bloc fonction `SFCCNTRL` *(voir page 172)* (entrées `STEPUN` et `STEPDEP`) ou commandes SFC externes) en vue de corriger les erreurs.

La sortie `SequenceFreezed` est réglée sur 1 en cas de gel effectif du graphe d'état.

NOTE : la sortie affiche l'état en cours du graphe d'état. Cela signifie que le bloc fonction `SFCCNTRL` ou d'autres commandes SFC externes peuvent également geler le graphe d'état.

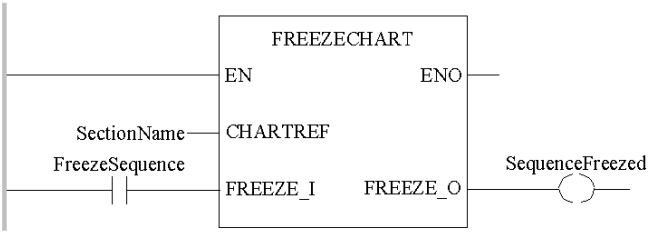
`EN` et `ENO` peuvent être configurés en tant que paramètres supplémentaires.

Représentation en FBD

Représentation :

Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD SectionName FREEZECHART SectionName ST SequenceFreezed
```

Représentation en ST

Représentation :

```
SequenceFreezed := FREEZECHART (SectionName, FreezeSequence) ;
```

Description des paramètres

Description du paramètre d'entrée :

Paramètres	Type de données	Description
`SectionName`	`SFCCHART_STATE`	Affectation de la section SFC à commander par l'intermédiaire du nom de la section. Lorsqu'une section SFC est créée, une variable du type de données `SFCCHART_STATE` lui est automatiquement affectée. La variable ainsi créée porte toujours le nom de la section SFC correspondante.
`FreezeSequence`	`BOOL`	1: Gel du graphe d'état (arrêt de l'évaluation des transitions)

Description du paramètre de sortie :

Paramètre	Type de données	Description
SequenceFreezed	BOOL	1: Graphe d'état gelé (évaluation des transitions arrêtée)

Chapitre 35
INITCHART : réinitialisation de toutes les étapes actives et démarrage normal du graphe d'état

Description

Description de la fonction

Cette fonction sert à réinitialiser et démarrer des graphes d'état normalement.

⚠ AVERTISSEMENT

ERREUR SYSTEME INATTENDUE

N'utilisez pas `INITCHART` pour rechercher des erreurs sur des automates de machines-outils, de processus ou de systèmes de gestion des matières en cours de fonctionnement.

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

⚠ AVERTISSEMENT

COMPORTEMENT IMPREVU DE L'EQUIPEMENT

Evitez d'utilisez la commande `SFCCNTRL` avec les commandes `CLEARCHART`, `FREEZCHART`, `INITCHART`, `SETSTEP` et `RESETSTEP`.

Vérifiez soigneusement les interactions entre ces blocs fonction et analysez l'impact sur le SFC.

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

De l'ensemble des fonctions et des blocs fonction de la commande des graphes d'état, c'est INITCHART qui a la priorité la plus élevée.

- Réinitialisation du graphe d'état
  Un signal 1 à l'entrée InitSequence permet d'arrêter le graphe d'état et de réinitialiser toutes les étapes. Tant que le signal 1 à l'entrée est présent, le graphe d'état reste dans ce mode. Cela signifie qu'il n'est pas possible de la démarrer.
  Comme le graphe d'état est sans cesse réinitialisé tant qu'un signal 1 à l'entrée InitSequence est présent, il est conseillé de prendre l'une des mesures suivantes afin d'empêcher un blocage continu du graphe d'état :
  - Appel conditionnel de la fonction INITCHART par l'entrée EN.
  - Appel conditionnel de la fonction INITCHART, par exemple par l'instruction IF dans le langage de programmation ST.
  - Utilisation d'une détection de front (R_TRIG) à l'entrée InitSequence.
- Démarrage normal du graphe d'état
  Un front 1->0 à l'entrée InitSequence permet de démarrer le graphe d'état normalement. Cela signifie que l'étape initiale est activée. Un signal 1 à la sortie InitState permet d'afficher cet état pendant un cycle.

NOTE : la sortie InitState affiche l'état en cours du graphe d'état. Cela signifie que le bloc fonction SFCCNTRL *(voir page 172)* (entrée INIT) ou d'autres commandes externes peuvent également démarrer le graphe d'état normalement.

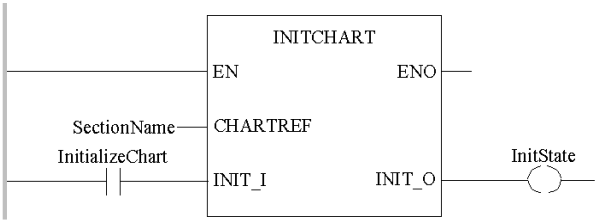
EN et ENO peuvent être configurés en tant que paramètres supplémentaires.

Représentation en FBD

Représentation :

Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD SectionName INITCHART InitializeChart ST InitState
```

Représentation en ST

Représentation :

```
InitState := INITCHART (SectionName, InitializeChart) ;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètres	Type de données	Description
CHARTREF	SFCCHART_STATE	Affectation de la section SFC à commander par l'intermédiaire du nom de la section. Lorsqu'une section SFC est créée, une variable du type de données SFCCHART_STATE lui est automatiquement affectée. La variable ainsi créée porte toujours le nom de la section SFC correspondante.
INIT_I	BOOL	0->1: Réinitialisation de toutes les étapes actives du graphe d'état. 1->0: Démarrer le graphe d'état normalement (définir étape initiale)

Description des paramètres de sortie :

Paramètre	Type de données	Description
INIT_O	BOOL	1: Graphe d'état démarré normalement (n'est actif que pendant un cycle).

Chapitre 36
RESETSTEP : réinitialisation d'une étape spécifique du graphe d'état

Description

Description de la fonction

Cette procédure sert à réinitialiser une étape dans un graphe d'état.

NOTE : la procédure n'est utilisable qu'en mode de fonctionnement à jetons multiples.

La procédure réinitialise l'étape indiquée.

Du fait que l'étape demeure réinitialisée pendant toute la durée d'exécution de cette procédure (`RESETSTEP`), laquelle s'exécute de façon cyclique, prenez l'une des mesures suivantes pour éviter une constante réinitialisation de l'étape :

- Appel conditionnel de la procédure `RESETSTEP` par l'entrée `EN`.
- Appel conditionnel de la procédure `RESETSTEP`, par exemple par l'instruction `IF` dans le langage de programmation `ST`.
- Utilisation d'une détection de front (`R_TRIG`) à l'entrée.

En cas de réinitialisation de la dernière/seule étape active, seuls le bloc fonction `SFCCNTRL` *(voir page 172)* (input `INIT`), la fonction `INITCHART` *(voir page 161)* ou la procédure `SETSTEP` *(voir page 169)* ou une autre commande d'activation d'étape externe permet de redémarrer le graphe d'état.

⚠ AVERTISSEMENT

ERREUR SYSTEME INATTENDUE

N'utilisez pas `RESETSTEP` pour rechercher des erreurs sur des automates de machines-outils, de processus ou de systèmes de gestion des matières en cours de fonctionnement.

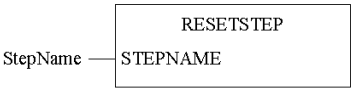
Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

⚠ AVERTISSEMENT
COMPORTEMENT IMPREVU DE L'EQUIPEMENT Evitez d'utilisez la commande `SFCCNTRL` avec les commandes `CLEARCHART`, `FREEZCHART`, `INITCHART`, `SETSTEP` et `RESETSTEP`. Vérifiez soigneusement les interactions entre ces blocs fonction et analysez l'impact sur le SFC. **Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.**

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

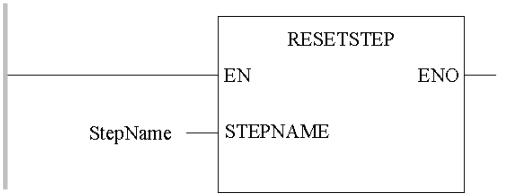
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD StepName RESETSTEP
```

Représentation en ST

Représentation :

```
RESETSTEP (StepName);
```

Description des paramètres

Description du paramètre d'entrée :

Paramètres	Type de données	Signification
`StepName`	`SFCSTEP_STATE`	Affectation de l'étape à réinitialiser par l'intermédiaire du nom de l'étape. (Lors de la création d'une étape SFC, une variable du type de données `SFCSTEP_STATE` lui est automatiquement affectée. La variable créée porte toujours le nom de l'étape SFC correspondante.)

Chapitre 37
SETSTEP : activation d'une étape spécifique du graphe d'état

Description

Description de la fonction

Cette procédure sert à activer une étape dans un graphe d'état.

NOTE : la procédure n'est utilisable qu'en mode de fonctionnement à jetons multiples.

La procédure active l'étape spécifiée, en plus des étapes déjà actives. Elle n'a pas d'effet sur ces dernières.

Comme l'étape demeure active pendant toute la durée d'exécution de cette procédure (`SETSTEP`), laquelle s'exécute de façon cyclique, prenez l'une des mesures suivantes pour éviter que l'étape soit constamment active :

- Appel conditionnel de la procédure `SETSTEP` par l'entrée `EN`.
- Appel conditionnel de la procédure `SETSTEP`, par exemple par l'instruction `IF` dans le langage de programmation `ST`.
- Utilisation d'une détection de front (`R_TRIG`) à l'entrée.

⚠ AVERTISSEMENT

ERREUR SYSTEME INATTENDUE

N'utilisez pas `SETSTEP` pour rechercher des erreurs sur des automates de machines-outils, de processus ou de systèmes de gestion des matières en cours de fonctionnement.

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

⚠ AVERTISSEMENT

COMPORTEMENT IMPREVU DE L'EQUIPEMENT

Evitez d'utilisez la commande `SFCCNTRL` avec les commandes `CLEARCHART`, `FREEZCHART`, `INITCHART`, `SETSTEP` et `RESETSTEP`.

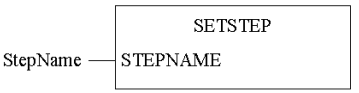
Vérifiez soigneusement les interactions entre ces blocs fonction et analysez l'impact sur le SFC.

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

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

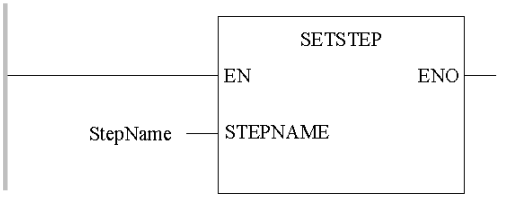
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD StepName SETSTEP
```

Représentation en ST

Représentation :

```
SETSTEP (StepName);
```

Description des paramètres

Description du paramètre d'entrée :

Paramètres	Type de données	Signification
`StepName`	`SFCSTEP_STATE`	Affectation de l'étape à activer par l'intermédiaire du nom de l'étape. (Lors de la création d'une étape SFC, une variable du type de données `SFCSTEP_STATE` lui est automatiquement affectée. La variable créée porte toujours le nom de l'étape SFC correspondante.)

Chapitre 38
SFCCNTRL : contrôle SFC

Introduction

Ce chapitre décrit le bloc `SFCCNTRL`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Description

Description de la fonction

Ce bloc fonction sert à contrôler les séquences d'exécution.

Il vous permet, par exemple, d'effectuer une exécution pas à pas, d'activer ou de désactiver les conditions de transition d'un traitement ou de réinitialiser un graphe d'état.

⚠ AVERTISSEMENT

ERREUR SYSTEME INATTENDUE

N'utilisez pas les paramètres `INIT`, `CLEAR`, `DISTRANS`, `DISACT`, `STEPUN` et `STEPDEP` pour rechercher des erreurs lors du contrôle de machines-outils, de processus ou de systèmes de gestion des matériaux en cours de fonctionnement.

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

⚠ AVERTISSEMENT

FONCTIONNEMENT D'EQUIPEMENT NON INTENTIONNEL

Evitez d'utiliser la commande `SFCCNTRL` avec les commandes `CLEARCHART`, `FREEZCHART`, `INITCHART`, `SETSTEP` et `RESETSTEP`.

Vérifiez soigneusement les interactions entre ces blocs fonction et analysez l'impact sur le SFC.

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

⚠ AVERTISSEMENT

FONCTIONNEMENT D'EQUIPEMENT NON INTENTIONNEL

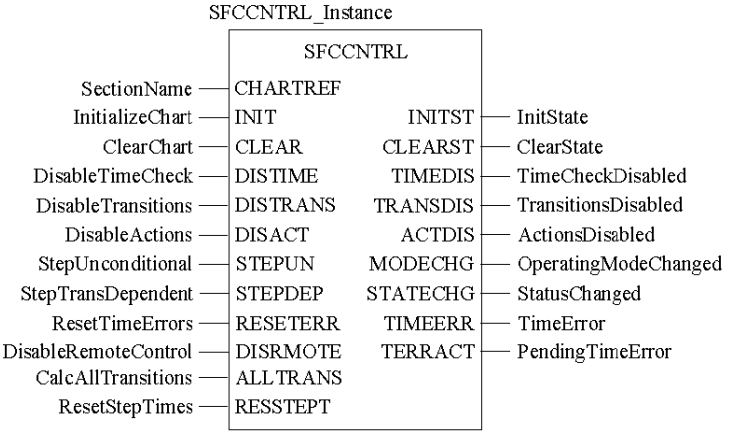
N'utilisez pas plus d'une instance de ce bloc dans chaque section SFC.

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

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

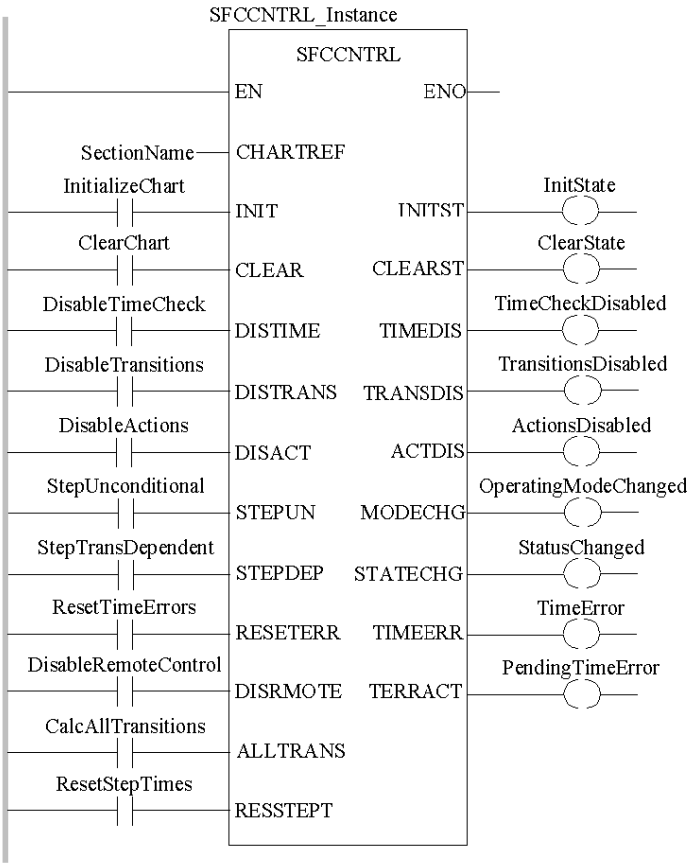
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL SFCCNTRL_Instance (CHARTREF:=SectionName, INIT:=InitializeChart,
CLEAR:=ClearChart, DISTIME:=DisableTimeCheck, DISTRANS:=DisableTran-
sitions, DISACT:=DisableActions, STEPUN:=StepUnconditional,
STEPDEP:=StepTransDependent, RESETERR:=ResetTimeErrors,
DISRMOTE:=DisableRemoteControl, ALLTRANS:=CalcAllTransitions,
RESSTEPT:=ResetStepTimes, INITST=>InitState, CLEARST=>ClearState,
TIMEDIS=>TimeCheckDisabled, TRANSDIS=>TransitionsDisabled,
ACTDIS=>ActionsDisabled, MODECHG=>OperatingModeChanged,
STATECHG=>StatusChanged, TIMEERR=>TimeError, TERRACT=>PendingTimeError)
```

Représentation en ST

Représentation :

```
SFCCNTRL_Instance (CHARTREF:=SectionName, INIT:=InitializeChart,
CLEAR:=ClearChart, DISTIME:=DisableTimeCheck, DISTRANS:=DisableTran-
sitions, DISACT:=DisableActions, STEPUN:=StepUnconditional,
STEPDEP:=StepTransDependent, RESETERR:=ResetTimeErrors,
DISRMOTE:=DisableRemoteControl, ALLTRANS:=CalcAllTransitions,
RESSTEPT:=ResetStepTimes, INITST=>InitState, CLEARST=>ClearState,
TIMEDIS=>TimeCheckDisabled, TRANSDIS=>TransitionsDisabled,
ACTDIS=>ActionsDisabled, MODECHG=>OperatingModeChanged,
STATECHG=>StatusChanged, TIMEERR=>TimeError, TERRACT=>PendingTimeError)
;
```

Description des paramètres

Description des paramètres d'entrée :

Paramètre	Type de données	Description
CHARTREF	SFCCHART_STATE	L'association avec la section SFC à contrôler est effectuée par le nom de la section. Lorsqu'une section SFC est créée, une variable du type de données SFCCHART_STATE lui est automatiquement affectée. La variable ainsi créée porte toujours le nom de la section SFC correspondante.
INIT	BOOL	0->1 : réinitialisation de toutes les étapes actives du graphe d'état. 1->0 : démarrage du graphe d'état standard (configuration de l'étape initiale).
CLEAR	BOOL	0->1 : réinitialisation de toutes les étapes actives du graphe d'état.
DISTIME	BOOL	1 : désactivation de la surveillance du temps.

Paramètre	Type de données	Description
DISTRANS	BOOL	1 : désactivation de l'évaluation des transitions (graphe d'état figé).
DISACT	BOOL	1 : désactivation du traitement des actions et remise à zéro de toutes les actions du graphe d'état.
STEPUN	BOOL	0->1 : activation de l'étape suivante, indépendamment de la condition de transition.
STEPDEP	BOOL	0->1 : activation de l'étape suivante en fonction de la condition de transition.
RESETERR	BOOL	0->1 : réinitialisation de l'erreur de surveillance du temps.
DISRMOTE	BOOL	1 : blocage du contrôle du graphe d'état à l'aide des paramètres du panneau d'animation en ligne.
ALLTRANS	BOOL	1 : calcul de toutes les sections de transition.
RESSTEPT	BOOL	0->1 : désactivation et réinitialisation du calcul du temps. 1->0 : redémarrage du calcul du temps. **Rappel :** si la fonction RESSTEPT est active, les étapes pour lesquelles un temps de retard est défini ne sont jamais activées (car RESSTEPT réinitialise constamment le temps actuel de l'étape, ce qui annule l'effet du réglage du temps de retard de celle-ci).

Description des paramètres de sortie :

Paramètre	Type de données	Description
INITST	BOOL	1 : graphe d'état démarré normalement (n'est actif que pendant un cycle).
CLEARST	BOOL	1 : graphe d'état réinitialisé, ce qui signifie qu'il n'a aucune étape active.
TIMEDIS	BOOL	1 : surveillance du temps désactivée.
TRANSDIS	BOOL	1 : évaluation des transitions désactivée.
ACTDIS	BOOL	1 : traitement des actions désactivé et toutes les actions du graphe d'état réinitialisées.
MODECHG	BOOL	1 : mode de fonctionnement du graphe d'état modifié (n'est actif que pendant un cycle).
STATECHG	BOOL	1 : état du graphe d'état modifié (n'est actif que pendant un cycle).
TIMEERR	BOOL	1 : erreur survenue pendant la surveillance du temps (n'est actif que pendant un cycle).
TERRACT	BOOL	1 : erreur dans la surveillance du temps.

Description des paramètres

Généralités

⚠ AVERTISSEMENT
ERREUR SYSTEME INATTENDUE N'utilisez pas les paramètres `INIT`, `CLEAR`, `DISTRANS`, `DISACT`, `STEPUN` et `STEPDEP` pour rechercher des erreurs lors du contrôle de machines-outils, de processus ou de systèmes de gestion des matériaux en cours de fonctionnement. **Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.**

CHARTREF

Association avec le SFC à contrôler

Lorsqu'une section SFC est créée, une variable du type de données `SFCCHART_STATE` lui est automatiquement affectée. La variable créée porte toujours le nom de la section SFC correspondante.

Cette variable sert à affecter le bloc fonction `SFCCNTRL` à la section SFC à contrôler.

NOTE : L'utilisateur ne peut pas créer le type de données `SFCCHART_STATE` et ne peut pas modifier sa valeur. Par conséquent, dans la table d'animation, cette variable ne peut pas être modifiée et le champ correspondant reste vide.

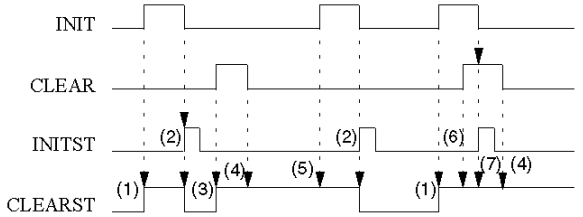
INIT

Réinitialise le graphe d'état et démarre normalement.

Cette entrée du bloc fonction a priorité sur toutes les autres.

- Réinitialisation du graphe d'état
  Lorsqu'un front 0->1 est détecté en entrée, le graphe d'état est arrêté et toutes les étapes sont réinitialisées. Aucun accès par opérateur n'est possible.
  Cet état est maintenu tant que l'entrée est 1. L'état (réinitialisation du graphe d'état) est indiqué par un signal 1 au niveau de la sortie `CLEARST`.
- Démarrage normal du graphe d'état
  Lorsqu'un front 1->0 est détecté en entrée et que `CLEAR` est sur 0, le graphe d'état démarre normalement (l'étape initiale est activée). Pendant un cycle, cela est indiqué par un signal 1 à la sortie `INITST`.

Interaction entre INIT, CLEAR, INITST et CLEARST :



Interaction entre INIT, CLEAR, INITST et CLEARST :

Phase	Description
1	Lorsqu'un front 0->1 est détecté au niveau de l'entrée INIT, toutes les étapes du graphe d'état sont réinitialisées et la sortie CLEARST est réglée sur 1.
2	Lorsqu'un front 1->0 est détecté au niveau de l'entrée INIT, l'étape initiale du graphe d'état est activée, la sortie INITST est réglée sur 1 pour un cycle et la sortie CLEARST est réglée sur 0.
3	Lorsqu'un front 0->1 est détecté au niveau de l'entrée INIT, toutes les étapes du graphe d'état sont réinitialisées et la sortie CLEARST est réglée sur 1.
4	Un front 1->0 au niveau de l'entrée CLEAR n'affecte ni l'état du graphe d'état, ni les sorties du bloc fonction.
5	Si le graphe d'état est déjà à l'état de réinitialisation, un front 0->1 au niveau de l'entrée INIT n'affecte ni l'état du graphe d'état, ni les sorties du bloc fonction.
6	Si le graphe d'état est déjà à l'état de réinitialisation, un front 0->1 au niveau de l'entrée CLEAR n'affecte ni l'état du graphe d'état, ni les sorties du bloc fonction.
7	Lorsque CLEAR est réglé sur 1, un front 1->0 au niveau de l'entrée INIT n'affecte ni l'état du graphe d'état (CLEARST), ni les sorties du bloc fonction.

CLEAR

Réinitialisation du graphe d'état

Lorsqu'un front 0->1 est détecté en entrée, le graphe d'état est arrêté et toutes les étapes sont réinitialisées. Aucun accès par opérateur n'est possible.

L'état (réinitialisation du graphe d'état) est indiqué par un signal 1 au niveau de la sortie CLEARST.

Cet état demeure même si l'entrée repasse à 0.

Le graphe d'état ne peut être démarré que par un front 1->0 au niveau de l'entrée INIT de la fonction INITCHART *(voir page 161)* ou par une autre commande (externe) d'activation des étapes, lorsque CLEAR est réglé sur 0.

DISTIME

Désactivation de la surveillance du temps

La surveillance du temps de l'étape active est inhibée par un signal 1 en entrée. Le système continue à calculer le temps d'exécution, mais ne signale aucune erreur si celui-ci dépasse la limite supérieure ou inférieure.

Les erreurs existantes ne sont pas affectées par cette fonction (animation ou sortie `TERRACT`).

DISTRANS

Désactivation de l'évaluation de la transition

L'état actuel du graphe d'état est gelé par un signal 1 en entrée. L'état des transitions n'est plus évalué. Par conséquent, il n'est plus possible de progresser dans le graphe d'état, même si la condition de transition de la transition « active » est vraie.

Cette fonction peut être utilisée avec `STEPUN` et `STEPDEP` pour la correction d'erreur.

DISACT

Désactivation du traitement des actions

Toutes les actions actives sont réinitialisées avec un signal 1 en entrée.

STEPUN

Activation de l'étape suivante, indépendamment de la condition de transition

Lorsqu'un front 0->1 est détecté en entrée, les étapes actives courantes sont désactivées et les étapes suivantes sont activées, et ce, indépendamment de l'état de la condition de transition. Cette situation survient en premier lieu après l'expiration du délai d'exécution de l'étape active (temps de l'étape).

En mode de branchement simultané, toutes les branches sont activées. En mode de branchement alternatif, la branche gauche est toujours activée.

L'entrée `STEPDEP` est utilisée pour activer les branches en fonction du processus.

STEPDEP

Activation de l'étape suivante en fonction de la condition de transition

Lorsqu'un front 0->1 est détecté en entrée et qu'une condition de transition est remplie, les étapes suivantes sont activées. Cette situation survient en premier lieu après l'expiration du délai d'exécution de l'étape active (temps de l'étape).

La commande de contrôle n'est sensible que lorsqu'un signal 1 est détecté au niveau de l'entrée `DISTRANS`.

En gelant les transitions (DISTRANS = 1), il est possible de traiter manuellement le graphe d'état (exécution pas à pas) à l'aide de cette commande de contrôle. De cette manière, les transitions dépendent des conditions de transition.

RESETERR

Réinitialisation de l'erreur de surveillance du temps

Lorsqu'un front 0->1 est détecté en entrée, l'affichage de toutes les erreurs de surveillance du temps avec l'animation de la section SFC est désactivé. Les erreurs de surveillance du temps actuellement affichées sont mises à jour. Les erreurs de surveillance du temps existantes sont de nouveau signalées. En l'absence d'erreur de surveillance du temps, la sortie TERRACT est réinitialisée.

DISRMOTE

Empêche le contrôle du diagramme à l'aide des paramètres de la commande de l'animation SFC.

Un signal 1 en entrée empêche le contrôle du diagramme SFC à l'aide des paramètres de la commande de l'animation SFC (initialisation, désactivation de la surveillance du temps, désactivation de l'évaluation des transitions, désactivation du traitement des actions). Malgré cela, le diagramme SFC peut encore être contrôlé à l'aide du bloc fonction SFCCNTRL.

Lorsque DISRMOTE est réglé sur 1, SFCNTRL est dominant. Lorsque DISRMOTE n'est pas défini ou est réglé sur 0, les entrées DISTRANS, DISACT, DISTIME et INIT du bloc fonction SFCNTRL, ainsi que les entrées correspondantes de la commande de l'animation SFC sont traitées comme formant une combinaison OR logique.

Lorsque DISRMOTE est réglé sur 1, les commandes de l'animation SFC sont désactivées.

Lorsque DISRMOTE est réglé sur 0, les commandes de l'animation SFC sont activées et tout autre composant de la variable de la section SFC (SFCCHART_STATE) reste inchangé.

Lorsque DISRMOTE est réglé sur 1 ou 0, la valeur est affectée à la variable de la section SFC du diagramme SFC contrôlé.

Toutefois, cela présente un inconvénient. En effet, lorsque le bloc fonction SFCCNTRL règle la variable sur 1 (et bloque la commande de l'animation SFC), l'instance du bloc fonction SFCCNTRL est supprimée. Dans ce cas, la valeur bloquante est conservée jusqu'à la prochaine activation de l'option **Régénérer tout** en mode hors ligne.

Ce problème peut survenir dans les situations suivantes :

- La variable de la section SFC affectée à l'entrée CHARTREF a été remplacée en mode connecté (cela signifie que l'instance du bloc fonction SFCCNTRL a été affectée à une autre section SFC).
- L'instance du bloc fonction SFCNTRL a été supprimée.
- La section comportant l'instance du bloc fonction SFCNTRL a été supprimée.

Autres possibilités en mode connecté : créez une nouvelle instance du bloc fonction SFCCNTRL, affectez un nom de section SFC approprié à l'entrée CHARTREF et réglez DISRMOTE sur 0. Cela débloquera la commande de l'animation SFC. Vous pouvez maintenant supprimer l'instance du bloc fonction SFCCNTRL.

Comment éviter le problème ? Pour éviter ce problème, assurez-vous toujours que DISRMOTE est réglé sur 0 avant de supprimer l'instance d'un bloc fonction SFCCNTRL.

NOTE : pour plus d'informations, consultez la section relative au bloc fonction SFC_RESTORE (*Interaction avec le bloc fonction SFCCNTRL, page 202*).

ALLTRANS

Calcul de toutes les sections de transition

Avec un signal 1 en entrée, toutes les sections de transition sont calculées (même si l'étape correspondante n'est pas active). Pour le traitement du graphe d'état, seules les transitions actives continuent d'être évaluées. Cette fonction sert uniquement à afficher simultanément l'animation de tous les états de transition.

NOTE : En raison du traitement supplémentaire de toutes les sections de transition dont l'étape est inactive, le temps de cycle du programme peut être allongé de manière considérable.

RESSTEPT

Désactivation du calcul du temps

Avec un signal 0->1, toutes les erreurs de surveillance du temps (temps de surveillance minimal et maximal), le temps écoulé depuis l'activation de l'étape et la sortie TERRACT sont réinitialisés. De la même manière, toutes les erreurs de temps du diagnostic sont annulées et la surveillance du temps de l'étape est arrêtée. Cet état est valide tant que le signal 1 est présent.

Lorsqu'un front 1->0 est détecté, tous les temps (en commençant par 0) sont recalculés et la surveillance du temps est activée.

NOTE : Lorsque la fonction RESSTEPT est active, les étapes ayant un temps de retard défini ne sont jamais activées. (Avec RESSTEPT, le temps de l'étape actuelle est sans cesse réinitialisé si bien que le temps de retard défini ne peut jamais être atteint.)

INITST

Démarrage normal du graphe d'état

Un front 1->0 à l'entrée INIT permet de démarrer le graphe d'état normalement. Cela signifie que l'étape initiale est activée. Pendant un cycle, cela est indiqué par un signal 1 à la sortie INITST.

voir aussi *INIT, page 177*

NOTE : La sortie indique l'état actuel du graphe d'état. Le graphe d'état peut également être démarré normalement à l'aide de la fonction INITCHART *(voir page 161)* ou d'une autre commande de contrôle externe.

CLEARST

Graphe d'état réinitialisé

Lorsqu'un front 0->1 est détecté au niveau de l'entrée INIT ou CLEAR, le graphe d'état est arrêté et toutes les étapes sont réinitialisées.

Cet état est maintenu jusqu'à ce qu'un front 1->0 soit détecté au niveau de l'entrée INIT.

voir aussi *INIT, page 177*

NOTE : La sortie indique l'état actuel du graphe d'état. Le graphe d'état peut également être réinitialisé à l'aide de la fonction INITCHART *(voir page 161)*, CLEARCHART *(voir page 153)* ou RESETSTEP *(voir page 165)* (réinitialisation de la dernière/seule étape active) ou d'une autre commande de contrôle externe.

TIMEDIS

Surveillance du temps désactivée

La sortie est réglée sur 1 lorsque l'affichage des erreurs de temps a été désactivé, que cet affichage ait été désactivé ou non à l'aide du bloc fonction lui-même (entrée DISTIME) ou via les commandes de contrôle SFC.

TRANSDIS

Evaluation des transitions désactivée

La sortie est à 1 si l'évaluation des transitions a été arrêtée.

NOTE : la sortie affiche l'état en cours du graphe d'état. Cela signifie que la fonction FREEZECHART *(voir page 157)* ou d'autres commandes SFC externes peuvent également geler le graphe d'état.

ACTDIS

Le traitement des actions a été désactivé et toutes les actions du graphe d'état ont été réinitialisées.

La sortie est réglée sur 1 si l'affichage des actions a été arrêté, que cet affichage ait été arrêté ou non à l'aide du bloc fonction lui-même (entrée DISACT) ou via les commandes de contrôle SFC.

MODECHG

Mode de marche du graphe d'état modifié

La sortie est réglée sur 1 pour un cycle si un ou plusieurs modes de marche du graphe d'état ont été modifiés, que cette modification ait été effectuée ou non à l'aide du bloc fonction lui-même (activation ou réinitialisation des entrées INIT, CLEAR *(voir page 178)*, DISTIME, DISACT ou DISTRANS) ou via les commandes de contrôle SFC externes.

STATECHG

Etat du graphe d'état modifié

La sortie est à 1 pour un cycle si l'état du graphe d'état a été modifié, que la modification ait été déclenchée par le traitement du graphe d'état ou ait été effectuée à l'aide du bloc fonction lui-même ou via les commandes de contrôle SFC externes.

TIMEERR

Présence d'une erreur lors de la surveillance du temps

La sortie est à 1 pour un cycle si une ou plusieurs erreurs de surveillance du temps sont survenues.

TERRACT

Une erreur s'est produite lors de la surveillance du temps.

La sortie reste à 1 en présence d'une ou de plusieurs erreurs de surveillance du temps.

Chapitre 39
SFC_RESTORE : Sauvegarde et restitution de SFC

Introduction

Ce chapitre décrit le bloc SFC_RESTORE.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Description

Description de la fonction

Ce bloc fonction permet de redémarrer tous les diagrammes SFC d'une application avec un ensemble d'étapes actives à un état donné, tel que sauvegardé avant une défaillance d'UC.

En situation de restitution, il est possible de définir un ensemble d'étapes qui représentent un état donné dans le processus et de continuer à partir de ce point.

Les diagrammes se comportent de la même manière que si `INITCHART` et `SETSTEP` étaient utilisés : la durée d'étape est remise à 0 et les actions P/P1 sont exécutées.

Pour plus d'informations, consultez *Stratégie de sauvegarde/restitution, page 194*.

⚠ AVERTISSEMENT

FONCTIONNEMENT D'EQUIPEMENT NON INTENTIONNEL

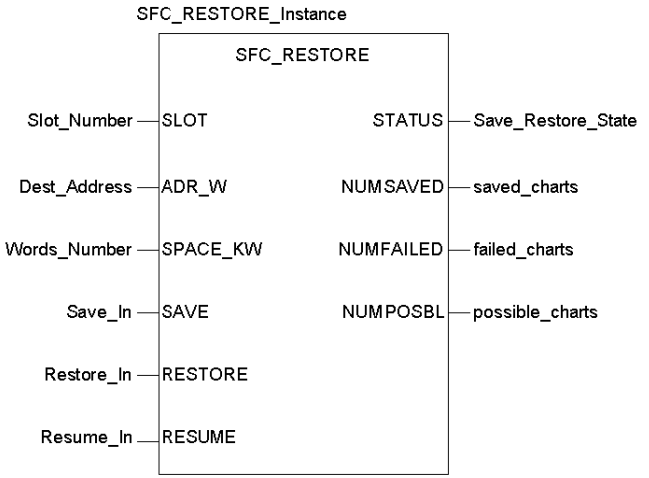
Vérifiez l'état de tous les diagrammes SFC avant de redémarrer l'application du processus.

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

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

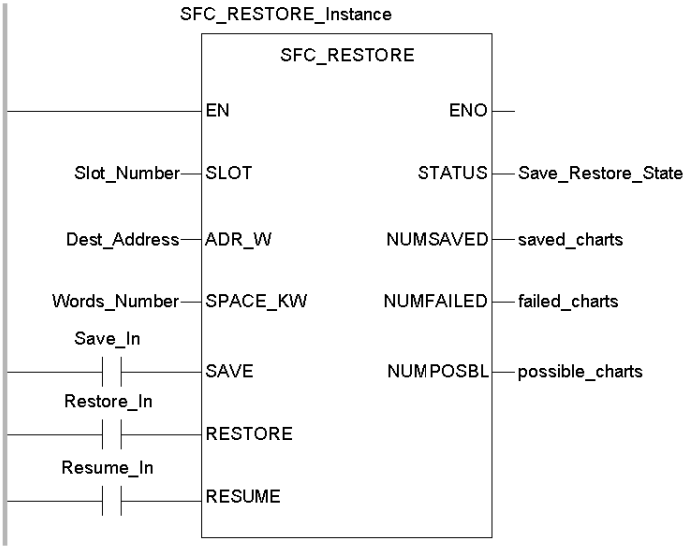
Représentation en FBD

Représentation



Représentation en LD

Représentation



Représentation en IL

Représentation :

```
CAL SFC_RESTORE_Instance (SLOT:=Slot_Number,
    ADR_W:=Dest_Address, SPACE_KW:=Words_Number,
    SAVE:=Save_In, RESTORE:=Restore_In, RESUME:=Resume_In,
    STATUS=>Save_Restore_State, NUMSAVED=>saved_charts,
    NUMFAILED=>failed_charts, NUMPOSBL=>possible_charts)
```

Représentation en ST

Représentation :

```
SFC_RESTORE_Instance (SLOT:=Slot_Number,
    ADR_W:=Dest_Address, SPACE_KW:=Words_Number,
    SAVE:=Save_In, RESTORE:=Restore_In,
    RESUME:=Resume_In,
    STATUS=>Save_Restore_State, NUMSAVED=>saved_charts,
    NUMFAILED=>failed_charts, NUMPOSBL=>possible_charts);
```

Description des paramètres

Le bloc fonction SFC_RESTORE permet d'accéder à la mémoire (SLOT, ADR_W, SPACE_KW) et de sélectionner les modes possibles (SAVE, RESTORE, RESUME).

Il indique également l'état de l'opération (STATUS).

La quantité de mémoire, en tant que paramètre d'entrée (SPACE_KW), détermine l'espace pouvant être utilisé par l'interpréteur SFC.

Cette zone mémoire démarre à l'adresse donnée (ADR_W) et est considérée comme réservée à l'usage exclusif du système SFC.

Description des paramètres d'entrée

Paramètre	Type de données	Description
SLOT	INT	Emplacement de la carte PCMCIA ● 0 : emplacement supérieur (par défaut) ● 1 : emplacement inférieur
ADR_W	DINT	adresse de la zone d'archivage dans laquelle sont stockées les données SFC sur la carte mémoire (décalage de mot)
SPACE_KW	INT	Ce nombre détermine, en kilo-mots, la taille de la mémoire que l'interpréteur SFC est autorisé à utiliser. L'interpréteur SFC calcule le nombre de sections SFC pouvant être enregistrées dans cet espace. Le résultat est visualisé au niveau de la sortie NUMPOSBL. **NOTE :** Le nombre de sections SFC enregistrées est soumis à une limite système. Tout espace réservé au-delà de la limite système n'est pas utilisé.
SAVE	BOOL	● 0 : mode sauvegarde non sélectionné ● 1 : mode sauvegarde sélectionné actif uniquement si ❍ RESTORE = 0 et ❍ RESUME = 0 ● 1->0 front descendant = sauvegarde arrêtée Cet événement est signalé au buffer de diagnostic.

Paramètre	Type de données	Description
RESTORE	BOOL	• 0 : mode restitution non sélectionné • 0->1 front montant = étapes sauvegardées restituées de la zone de stockage dans chaque section SFC Cet événement est signalé au buffer de diagnostic. Toutes les sections SFC passent dans les modes `Inhiber transitions` et `Inhiber actions`. • 1 : pas de `SAVE` / pas de `RESTORE` / pas de `RESUME` • 1->0 front descendant = pas d'activité
RESUME	BOOL	• 0 : mode reprise non sélectionné • 0->1 front montant = réactivation de tous les diagrammes SFC en mode normal si `RESTORE` = 0 Cet événement est signalé au buffer de diagnostic. • 1 : pas d'activité, mais empêche l'exécution de `SAVE` • 1->0 front descendant = pas d'activité

Description des paramètres de sortie

Paramètre	Type de données	Description
STATUS	INT	0 = OK message d'erreur `STATUS` (voir *Messages d'erreur STATUS, page 203*)
NUMSAVED	INT	nombre de sections SFC ayant fait l'objet d'une sauvegarde au cours du dernier cycle
NUMFAILED	INT	nombre de sections SFC n'ayant pas pu être sauvegardées au cours du dernier cycle
NUMPOSBL	INT	nombre de sections SFC pouvant être enregistrées dans l'espace mémoire donné

Configuration requise et restrictions

Général

L'analyseur vérifie les règles suivantes et, en cas de non-respect dans l'application, des erreurs sont générées :

- L'EFB ne doit être utilisé qu'une seule fois dans toute l'application.
- Une carte PCMCIA de type `DATA` doit se trouver à l'emplacement approprié.
- La taille des données doit correspondre aux limites de la mémoire.

Un seul `SFC_RESTORE`

Un seul bloc fonction `SFC_RESTORE` est utilisé pour contrôler la sauvegarde et la restauration du SFC.

NOTE : `SFC_RESTORE` ne peut être utilisé qu'une seule fois dans une application (instance à contrôle unique).

NOTE : Le nombre de sections SFC enregistrées est limité à 250.

Emplacement de `SFC_RESTORE`

Il est recommandé de situer l'utilisation de `SFC_RESTORE` dans une section exécutée avant la première section SFC.

Une modification lors des entrées a donc une incidence sur toutes les sections SFC de la même scrutation.

Il est également vivement recommandé de ne pas désactiver cet EFB ou la section dans laquelle il se trouve, sinon les données sauvegardées ne sont plus cohérentes avec les diagrammes SFC.

`SFC_RESTORE` désactivé ou supprimé

Si le bloc fonction `SFC_RESTORE` est désactivé ou supprimé, l'interpréteur SFC n'est plus notifié.

Ce comportement est détecté après quelques scrutations et la fonction d'enregistrement et de restauration du SFC est alors arrêtée. Plus aucune sauvegarde n'est effectuée.

Les données de la carte mémoire PCMCIA sont conservées et un avertissement est enregistré dans le tampon de diagnostic.

Mode actif actuel

L'action ou le mode actif actuel est calculé conformément au tableau ci-après.

Mode/Action	Valeur `Save`	Valeur `Restore`	Valeur `Resume`
inactif	0	0	0
sauvegarde	1	0	0
restauration	0 ou 1	0 -> 1 (front montant)	0 ou 1
reprise	0 ou 1	0	0 -> 1 (front montant)

Carte mémoire PCMCIA

Le bloc fonction `SFC_RESTORE` peut uniquement être utilisé dans les UC avec un emplacement de carte mémoire PCMCIA.

Lors de l'analyse, la présence d'une carte mémoire PCMCIA adaptée est vérifiée dans les données de configuration.

Une erreur est signalée si aucun emplacement ou un emplacement erroné est sélectionné.

Si la valeur correspond à une constante, une vérification supplémentaire est effectuée pour s'assurer que la zone de mémoire est adaptée à la taille de la carte mémoire à partir de l'adresse donnée.

Néanmoins, une erreur d'exécution s'affiche sur la sortie `STATUS` en cas d'absence de la carte (`WRITE_PCMCIA`, par exemple. `16#0201 : aucune zone de fichiers dans la carte mémoire`).

Estimation de la mémoire

Pour estimer la quantité de mémoire nécessaire pour enregistrer des sections SFC, utilisez la formule suivante :

octets $_{\Sigma}$ = 570 + n*210

570 octets nécessaires pour la gestion des données
n nombre de sections SFC
210 octets nécessaires par section SFC

NOTE : Ces valeurs peuvent changer lors de la mise à jour de l'OS.

Zone mémoire

A partir de l'adresse donnée par l'entrée `ADR_W`, l'interpréteur SFC suppose une zone mémoire de `SPACE_KW` * 2 Ko utilisables par la fonction de sauvegarde et de restauration du SFC.

Une somme de contrôle (checksum) effectuée sur chaque bloc vérifie l'intégrité de cette mémoire.

En cas de détection d'une erreur (par exemple `Erreur de checksum`), le bloc concerné est marqué comme étant non valide et une erreur est signalée.

Dans l'analyse suivante, SFC tentera de sauvegarder de nouveau les étapes actives de sorte que l'impossibilité d'effectuer une restauration peut n'être que temporaire.

⚠ AVERTISSEMENT

FONCTIONNEMENT D'EQUIPEMENT NON INTENTIONNEL

Prévoyez une mémoire suffisante sur la carte PCMCIA pour la fonction SFC de sauvegarde et de restauration.

Réservez cette zone de mémoire aux seules fonctions SFC de sauvegarde et de restauration pour éviter tout chevauchement avec les autres parties de votre application.

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

NOTE : Ne perdez pas de vue que même les blocs `READ_PCMCIA` et `WRITE_PCMCIA` peuvent gérer la carte mémoire PCMCIA.

Nombre de sections SFC sauvegardées

Selon l'entrée `SPACE_KW`, une certaine quantité de mémoire est utilisée. Lors de la génération, aucun contrôle n'est effectué pour vérifier que la quantité de mémoire convient.

Si le nombre de sections SFC détectées au moment de l'exécution est trop important pour la mémoire, une erreur est signalée.

La restauration des diagrammes SFC qui ont été enregistrés reste possible.

Si le nombre de sections SFC actives augmente (activation de sections, ajout avec la modification en ligne), ce type d'erreur peut être spontanément détecté au moment de l'exécution.

Trois sorties disponibles donnent des indications sur le nombre de sections SFC dans l'application :

- `NUMSAVED` : nombre de sections SFC sauvegardées
- `NUMFAILED` : nombre de sections SFC non sauvegardées
- `NUMPOSBL` : nombre maximum de sections SFC pouvant être sauvegardées dans le scénario actuel

NOTE : S'il n'est pas nécessaire d'économiser la mémoire, il est recommandé d'ajouter un peu d'espace mémoire pour les futures extensions. Le nombre de sections SFC enregistrées est limité à 250.

Mémoire non effacée

Selon une génération/modification en ligne, de nouveaux paramètres seront pris en compte dans l'EFB.

En cas de changement d'adresse, la nouvelle mémoire sera utilisée.

NOTE : L'ancienne mémoire ne sera pas effacée.

Une modification de `SPACE_KW` se traduira par l'augmentation ou la diminution de la mémoire utilisée.

Il est supposé que cette mémoire est présente et disponible de façon exclusive.

Nouvelle sauvegarde en cas de génération

En cas de génération/modification d'une partie de l'application changée, les données sauvegardées ne sont plus valides.

En effet, la signature de l'application, utilisée pour identifier les pairs application/stockage valides, est modifiée.

NOTE : Pour pouvoir procéder à une nouvelle restauration, une nouvelle sauvegarde doit obligatoirement être effectuée après une génération/modification.

Stratégie de sauvegarde/restitution

Stratégie de sauvegarde

La stratégie de sauvegarde est basée sur une sauvegarde des étapes actives dans une zone mémoire externe à l'UC (carte mémoire PCMCIA).

Dans l'environnement de sections individuelles, il est possible de savoir si des étapes actives ont changé. Ce n'est que dans un tel cas (ou pour une reprise après erreur) que la sauvegarde est effectuée.

NOTE : Si l'état d'un diagramme n'a pas changé, les données ne sont pas mises à jour dans la cartouche.

Pour éviter une utilisation excessive de la mémoire et gagner du temps, seuls les StepID (identificateurs internes) des étapes actives sont sauvegardés.

Chaque section SFC est sauvegardée séparément selon son contexte.

Stratégie de restitution

En cas de restitution, le système lit les StepID dans la zone de la carte mémoire PCMCIA et les active dans les diagrammes.

Ces StepID doivent être cohérents : au moment de la sauvegarde, toutes les étapes doivent être présentes et identifier les mêmes objets dans les sections SFC.

NOTE : A cet effet, l'application ne doit être ni modifiée ni compilée entre les étapes de sauvegarde et de restitution.

Sauvegarde des étapes actives

Vue d'ensemble

En mode `Sauvegarde`, une sauvegarde est effectuée à l'issue du traitement de chaque diagramme.

Si les étapes actives d'un diagramme sont modifiées, les nouveaux StepID (identificateurs internes) sont stockés dans la mémoire et un état est mémorisé.

Cela se répète pour toutes les sections SFC suivantes du cycle.

Lors de la prochaine exécution de l'EFB, l'état cumulé de l'opération de sauvegarde est récupéré et affiché au niveau de la sortie.

Si plusieurs erreurs sont détectées, la plus significative (voir *Messages d'erreur* `STATUS`, *page 203*) est affichée.

Effets d'une modification de l'application

Si l'application est modifiée (ajout, suppression, déplacement ou désactivation de sections SFC), cela se répercute sur les données sauvegardées.

Modification	Effet
ajout d'une section SFC	Les étapes actives sont sauvegardées dès que la section est exécutée.
suppression d'une section SFC	Aucune sauvegarde ultérieure ; les données en mémoire sont effacées.
déplacement d'une section SFC	Les données sont stockées dans un ordre différent.
désactivation d'une section SFC	Aucune sauvegarde ultérieure, mais les données restent valides. Les données sont à nouveau sauvegardées si la section est réactivée.

Somme de contrôle

Le système vérifie si le volume de données concorde avec l'espace mémoire disponible suivant les paramètres utilisateur (nombre de sections SFC et nombre d'étapes actives par section SFC).

Chaque section fait l'objet d'une somme de contrôle, laquelle permet de vérifier dans une certaine mesure que le contenu n'est pas endommagé.

NOTE : Si le contrôle échoue, les données concernées sont marquées comme invalides.

Mise à jour des données enregistrées

- Si certaines sections SFC ne sont pas exécutées parce qu'une condition d'exécution n'est pas vérifiée, les StepID (identificateurs internes) de la dernière exécution restent disponibles.
- En cas de nouvelle exécution, les StepID sont mis à jour.
- Les sections SFC récemment ajoutées enregistrent leur StepID dès qu'elles sont exécutées.
- Les données enregistrées des sections SFC supprimées sont effacées immédiatement.

En général, après une compilation/modification, toutes les données sont invalidées. En effet, elles ne sont plus utilisables car la signature de l'application a changé. Cela évite que la mémoire soit inutilement occupée par des données obsolètes.

Il est fréquent que, dans cet environnement, une restitution effectuée après une défaillance d'UC réunisse des données mixtes, provenant des diagrammes de deux cycles, car une telle défaillance se produit généralement à un point ou un autre du cycle.

NOTE : Dans de tels cas, les états des diagrammes peuvent être incohérents.

Restitution des étapes en vue d'une activation

Vue d'ensemble

Si un front montant (0->1) de RESTORE est détecté, le première section SFC exécutée applique une restitution à toutes les sections.

La simultanéité des reprises est nécessaire pour assurer que chaque section SFC récupère les StepID (identificateurs internes) enregistrés, même ceux qui sont désactivés.

Si un signal élevé est actif au niveau de l'entrée RESTORE, aucune opération SAVE ni RESUME n'est effectuée, même si ces entrées sont actives.

Cela est nécessaire pour empêcher l'écrasement de la zone de sauvegarde et maintenir une claire distinction entre les modes de fonctionnement.

CLEARCHART / SETSTEP

Pour effectuer une opération RESTORE, un CLEARCHART est exécuté.

Cela réinitialise toutes les étapes et actions, et efface toutes les erreurs SFC du buffer de diagnostic.

Ensuite, une opération SETSTEP est effectuée pour chaque StepID extrait de la zone de stockage.

Contrôle général

Avant le démarrage de l'opération RESTORE, un contrôle général vérifie l'intégrité des données de la carte mémoire.

L'ID de génération de l'application doit concorder et la version de celle-ci doit être comprise dans l'intervalle valide.

Si les contrôles ne réussissent pas tous, la restitution est abandonnée.

Une erreur (INIT effectué en raison d'une erreur de restitution) est signalée à l'EFB et au buffer de diagnostic.

NOTE : L'automate ne passe pas à l'état HALT. Tous les diagrammes SFC commencent par les étapes INIT.

Contrôles individuels

Les sections sont également contrôlées individuellement :

- L'opération de sauvegarde doit être effectuée.
- La somme de contrôle doit concorder.
- L'identificateur StepID doit être valide.

Si une erreur survient au cours de la reprise de certaines sections, la section SFC en cours revient à son état initial.

Un avertissement est émis pour indiquer que les diagrammes SFC n'ont pas pu être tous restitués/repris/initialisés.

La détection de défaillances de ce type n'empêche pas le mécanisme de restitution de restituer le plus grand nombre d'étapes possible.

Reprise des sections SFC

Vue d'ensemble

Après avoir vérifié que les sections SFC sont dans l'état approprié, utilisez l'entrée `RESUME` pour définir tous les diagrammes SFC sur le mode d'action et de transition activé.

Pour connaître le comportement en cas d'interaction avec le bloc fonction SFCCNTRL, reportez-vous à la section *Interaction avec le bloc fonction SFCCNTRL, page 202*.

Vérification des diagrammes SFC

⚠ AVERTISSEMENT
FONCTIONNEMENT D'EQUIPEMENT NON INTENTIONNEL Vérifiez l'état de tous les diagrammes SFC avant de redémarrer l'application du processus. **Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.**

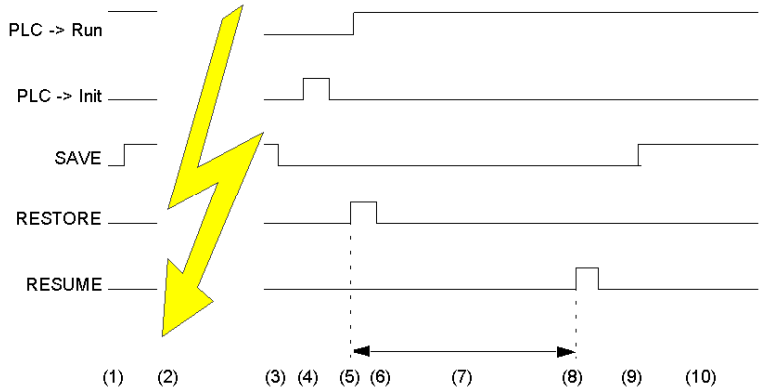
Exécution d'un cycle de restitution complet

Pour exécuter un cycle de restitution complet des diagrammes SFC, procédez comme suit :

Etape	Action	Commentaire
1	Chargez l'application.	Utilisez la même application que celle employée pour sauvegarder les étapes. Selon les situations, cette action peut être facultative.
2	Réinitialisez l'entrée `SAVE`.	Cela empêche l'écrasement des données sauvegardées.
3	Réinitialisez l'entrée `RESUME`.	-
4	Sélectionnez **Automate → Init**.	Déjà exécuté si l'application vient d'être chargée. Cette action peut être facultative.
5	Définissez l'entrée `RESTORE`.	Le front montant lance la restitution de tous les diagrammes de la prochaine section SFC exécutée. Doit être défini avant l'exécution de RUN pour faire passer le SFC en mode `désactivé`.
6	Exécutez l'application.	-
7	Réinitialisez l'entrée `RESTORE`.	-

Etape	Action	Commentaire
8	Vérifiez les diagrammes SFC.	Le front montant de l'entrée `RESTORE` désactive les actions et les transitions de tous les diagrammes SFC. **Remarque :** Veillez à éviter tout risque de blessure avant d'activer les diagrammes SFC. Consultez le message de sécurité *Vérification des diagrammes SFC, page 199* ci-dessus.
9	Activez chaque diagramme séparément avec **Commande de l'animation SFC**.	Selon les situations, cette action peut être facultative.
10	Définissez l'entrée `RESUME`.	Le front montant de l'entrée `RESUME` active toutes les transitions et actions SFC. **Remarque :** Cela supprime les indicateurs `Inhiber transitions` et `Inhiber actions` de toutes les sections SFC.
11	Réinitialisez l'entrée `RESUME`.	-
12	Définissez l'entrée `SAVE`.	Une fois le système redevenu opérationnel, vous pouvez à nouveau activer pour sauvegarder les étapes actives.

Schéma temporel



1 Marche normale.
2 Défaillance UC.
3 Réinitialisez l'entrée SAVE.
4 Sélectionnez PLC -> Init.
5 Définissez l'entrée RESTORE et sélectionnez PLC -> Run.
6 Réinitialisez l'entrée RESTORE.
7 Cela entraîne la désactivation de toutes les actions/transitions SFC (au nombre de 5). Il est maintenant possible d'activer les actions/transitions SFC à l'aide du SFC Animation Panel.
8 Définissez et réinitialisez l'entrée RESUME.
9 Définissez l'entrée SAVE.
10 Marche normale.

Interaction avec le bloc fonction SFCCNTRL

Présentation

Certaines sections SFC peuvent également être contrôlées par un ou plusieurs blocs fonction SFCCNTRL.

Si les entrées DISTRANS et/ou DISACT du bloc fonction SFCCNTRL sont utilisées, les entrées RESTORE et RESUME du bloc fonction SFC_RESTORE adoptent le comportement décrit ci-après.

DISRMOTE = 1

Si l'entrée DISRMOTE est à 1, le bloc fonction SFCCNTRL est maître.

Après l'exécution d'une opération RESTORE ou RESUME avec le bloc fonction SFC_RESTORE, la valeur présente aux entrées DISTRANS / DISACT du bloc fonction SFCCNTRL est active.

DISRMOTE = 0

Si l'entrée DISRMOTE est à 0, le bloc fonction SFCCNTRL et la commande à distance (**Commande de l'animation SFC** ou bloc fonction SFC_RESTORE) ont le même niveau de priorité.

Il fonctionne comme un circuit de contacts shunté (« wired OR ») : une valeur TRUE provenant d'une source quelconque annule la transition/l'action.

Pour obtenir une valeur FALSE, la source qui la règle sur TRUE doit la supprimer.

Dans ce cas, la commande provenant du bloc fonction SFC_RESTORE produit le même résultat que **Commande de l'animation SFC**, c'est-à-dire qu'elle est considérée comme la même source.

Après l'exécution d'une opération RESTORE, le schéma passe en mode désactivé, quelle que soit la valeur active au niveau du bloc fonction SFCCNTRL.

Le bloc fonction SFCCNTRL pourrait le maintenir désactivé, mais pour l'activer, vous devez utiliser **Commande de l'animation SFC** ou l'entrée RESUME du bloc fonction SFC_RESTORE.

Messages d'erreur STATUS

Gestion des erreurs

L'automate ne passe pas à l'état PAUSE en raison d'un incident lié au stockage, mais fournit un avertissement en deux étapes :

- Un message d'erreur est fourni avec la sortie STATUS de l'EFB.
- Un message d'erreur est envoyé au tampon de diagnostic.

Messages liés à l'accès à la mémoire

Messages relatifs à l'accès à la carte mémoire PCMCIA

Code	Description
16#0000	Action effectuée correctement
16#0102	ADR_W + SPACE_KW - 1 est supérieur au nombre maximal de mots déclarés dans l'automate
16#0104	Aucune application, ni aucun mot valide dans l'automate
16#0201	Aucune zone de fichiers dans la carte mémoire
16#0202	Erreur de carte mémoire
16#0204	Carte mémoire protégée en écriture
16#0241	ADR_W < 0
16#0242	ADR_W + SPACE_KW -1 est supérieur à l'adresse la plus élevée de la carte mémoire
16#0401	SPACE_KW = 0
16#0402	Slot_Number est différent de 0 et 1
16#0501	Service non pris en charge

Messages spécifiques à SFC

De nouveaux messages/avertissements spécifiques à SFC ont été définis en plus des messages d'accès à la carte mémoire PCMCIA.

Code	Description
16#a4F4	Fin du mode SAVE
16#a5F4	RESTORE initié
16#a6F4	RESUME initié
16#a7F4	Aucun élément sauvegardé. Aucun élément restauré, aucun ayant été sauvegardé
16#a8F4	Perte de la communication avec l'EFB SFC_RESTORE

Code	Description
16#a9F4	Nombre trop élevé de sections SFC pour SFC_Restore. Le paramètre SPACE_KW de l'EFB SFC_RESTORE dépasse les limites du système **NOTE :** Cet avertissement indique que l'espace maximal de la carte, visualisé dans la sortie NUMPOSBL, peut-être réglé en réduisant le paramètre SPACE_KW.
16#aaF4	Données sauvegardées non valides
16#abF4	Il a été impossible de restaurer/reprendre/initialiser tous les diagrammes
16#acF4	Espace mémoire trop faible
16#adF4	L'application a été modifiée
16#aeF4	Une procédure INIT a été effectuée en raison d'une erreur de restauration

NOTE : Si plusieurs statuts ont été signalés, seul le plus significatif apparaît dans la sortie STATUS de l'EFB. L'importance accroît avec les nombres plus élevés (notification hexadécimale ou non signée). L'avertissement INIT (16#aeF4) est le plus significatif.

NOTE : Si l'erreur 16#a0F4 (commande SFC non définie) apparaît dans la sortie STATUS de l'EFB, le système d'exploitation de l'automate ne prend pas en charge la fonction SFC_Restore (par exemple, version inadéquate du système d'exploitation).

Rapports dans le tampon de diagnostic

Les événements suivants sont signalés dans le tampon de diagnostic, mais pas dans l'EFB.

Code	Description
16#a4F4	Fin du mode SAVE
16#a5F4	RESTORE initié
16#a6F4	RESUME initié
16#a9F4	Nombre trop élevé de sections SFC pour SFC_Restore. Le paramètre SPACE_KW de l'EFB SFC_RESTORE dépasse les limites du système **NOTE :** Cet avertissement indique que l'espace maximal de la carte, visualisé dans la sortie NUMPOSBL, peut-être réglé en réduisant le paramètre SPACE_KW.

Partie VI
Horloge système

Aperçu

Cette partie décrit les fonctions de base et les blocs fonction de base de la famille `Horloge système`.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 40
FREERUN : temporisateur libre

Description

Description de la fonction

Cette fonction met en œuvre un compteur libre qui peut être utilisé pour mesurer la durée d'exécution des sections et des programmes utilisateurs.

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

Déterminer la durée de fonctionnement d'une section

Déterminer la durée de fonctionnement d'une section :

Etape	Action
1	Placez une fonction `FREERUN` au début et une autre à la fin de la section.
2	Assurez-vous que l'ordre d'exécution des fonctions est tel que la première fonction `FREERUN` est exécutée comme première fonction de la section et que la fonction FREERUN de fin est exécutée comme dernière fonction de la section.
3	Calculez la différence des deux valeurs déterminées. Ce Delta représente le temps enveloppe de la section en microsecondes.

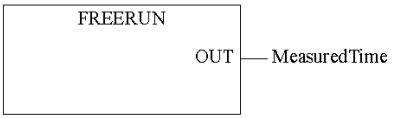
Déterminer la durée de fonctionnement d'un programme

Déterminer la durée de fonctionnement d'un programme :

Etape	Action
1	Placez une fonction `FREERUN` au début de la première section du programme et une autre à la fin de la dernière section.
2	Assurez-vous que l'ordre d'exécution des fonctions est tel que la première fonction `FREERUN` est exécutée comme première fonction de la section et que la fonction FREERUN de fin est exécutée comme dernière fonction de la section.
3	Calculez la différence des deux valeurs déterminées. Ce Delta représente le temps enveloppe du programme en microsecondes.

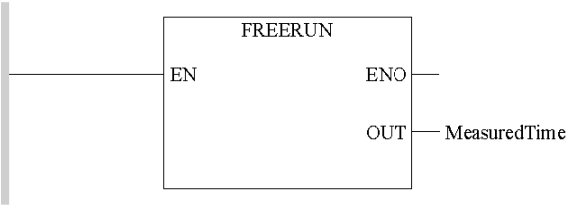
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
FREERUN
ST MeasuredTime
```

Représentation en ST

Représentation :

```
MeasuredTime := FREERUN () ;
```

Description des paramètres

Description des paramètres de sortie :

Paramètres	Type de données	Signification
`MeasuredTime`	`DINT`	Affiche le temps mesuré depuis le lancement du programme en microsecondes.

Chapitre 41
GET_TS_EVT_M : Lecture de la mémoire tampon des événements horodatés M340 et M580

Description

Description de la fonction

Le bloc fonction GET_TS_EVT_M obtient les données horodatées dans :

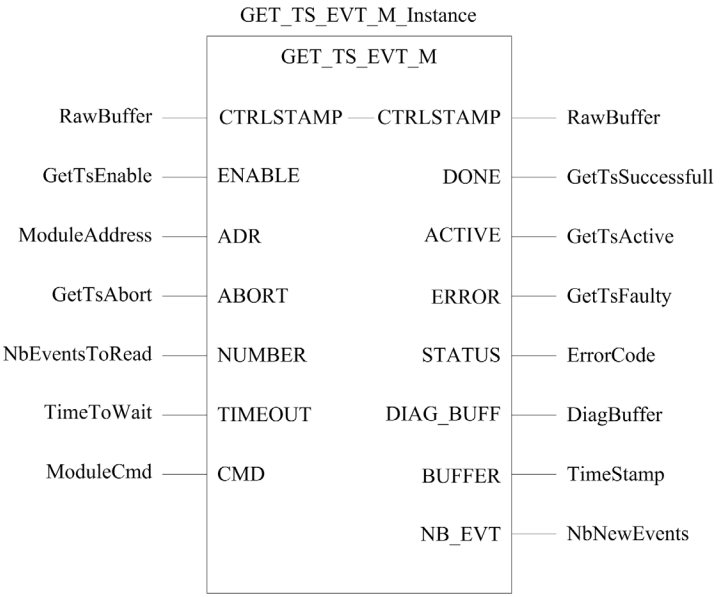
- BMX ERT 1604T dans une station locale Modicon M340,
- BMX ERT 1604T dans des stations locales et distantes Modicon M580,
- UC BME P58 xxx Modicon M580,
- BMX CRA 312 10 d'un Modicon M580,
- BME CRA 312 10 d'un Modicon M580.

Cette fonction permet de lire la mémoire tampon des événements afin de les rendre accessibles à l'application automate.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

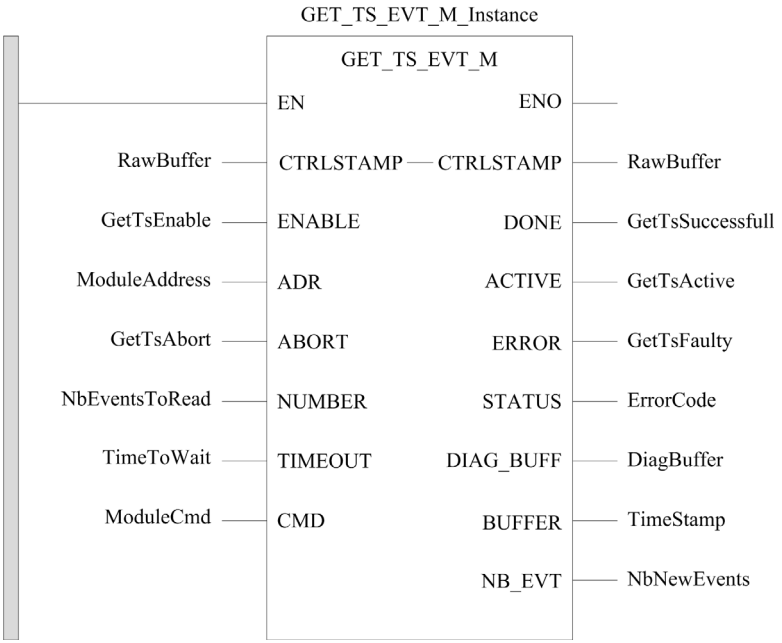
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL GET_TS_EVT_M_Instance (CTRLSTAMP:=RawBuffer, ENABLE:=GetTsEnable,
ADR:=ModuleAddress, ABORT:=GetTsAbort, NUMBER:=NbEventsToRead,
TIMEOUT:=TimeToWait, CMD:=ModuleCmd, DONE=>GetTsSuccessfull,
ACTIVE=>GetTsActive, ERROR=>GetTsFaulty, STATUS=>ErrorCode,
DIAG_BUFF=>DiagBuffer, BUFFER=>TimeStamp, NB_EVT=>NbNewEvents)
```

Représentation en ST

Représentation :

```
GET_TS_EVT_M_Instance (CTRLSTAMP:=RawBuffer, ENABLE:=GetTsEnable,
ADR:=ModuleAddress, ABORT:=GetTsAbort, NUMBER:=NbEventsToRead,
TIMEOUT:=TimeToWait, CMD:=ModuleCmd, DONE=>GetTsSuccessfull,
ACTIVE=>GetTsActive, ERROR=>GetTsFaulty, STATUS=>ErrorCode,
DIAG_BUFF=>DiagBuffer, BUFFER=>TimeStamp, NB_EVT=>NbNewEvents)
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
ENABLE	BOOL	Réglé sur 1 pour envoyer la requête au module concerné.
ADR	ANY_ARRAY_INT	Tableau contenant l'adresse de l'esclave Modbus, le résultat de la fonction ADDM *(voir Unity Pro, Communication, Bibliothèque de blocs)* ou ADDMX *(voir Unity Pro, Communication, Bibliothèque de blocs)*.
ABORT	BOOL	Réglé sur 1 pour abandonner l'opération en cours.
NUMBER	INT	Nombre maximum d'événements à lire dans le tampon local du module.
TIMEOUT	INT	Délai d'attente maximum de la réponse de la station. La base de temps pour ce paramètre est de 100 ms. **NOTE :** TIMEOUT = 0 correspond à un délai d'attente infini.
CMD	INT	Réglé sur: ● 0 : pour lire le tampon du module ● 1 : réinitialiser le tampon du module

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
DONE	BOOL	Exécution terminée du bloc fonction. Réglé sur 1 lorsque l'exécution du bloc fonction a abouti.
ACTIVE	BOOL	Bloc fonction en cours d'exécution. Réglé sur 1 lorsque l'exécution du bloc fonction est en cours.
ERROR	BOOL	Réglé sur 1 si une erreur est détectée par le bloc fonction.
STATUS	WORD	Code fournissant un compte rendu de communication et d'opération *(voir page 214)*. ● Octet 0 : compte rendu de communication ● Octet 1 : compte rendu d'opération

Paramètre	Type	Commentaire
DIAG_BUFF	ANY_ARRAY_INT	Description: ● Octet 0 : pourcentage de saturation du tampon ● Octet 1 : indicateurs de diagnostic: ❍ Bit 0 = 1 : l'heure est OK et synchronisée ❍ Bit 1 = 1 : le tampon est saturé, des événements peuvent être perdus Cela se produit en mode **Ecrasement**, lorsque certains ou la totalité des événements ont été écrasés depuis la dernière réponse. Le mode **Ecrasement** n'est disponible que si le BM• CRA 312 10 est en mode **Application** et il n'est pas pris en charge par le BMX ERT 1604T. ❍ Bit 2 = 1 : le tampon est saturé et l'horodatage s'est arrêté Cela se produit en mode **Arrêt en cas de tampon saturé**, lequel est pris en charge par le BMX ERT 1604T et le BM• CRA 312 10 en mode **Système**. Tout nouvel événement détecté avant la lecture du tampon d'événements est perdu. ❍ Bit 3 = 1 : la requête de lecture est la première car le module 140 CRA 312 10 ou BM• CRA 312 10 était alimenté (le BMX ERT 1604T n'utilise pas ce bit).
BUFFER	ANY_ARRAY_INT	Tampon brut contenant des entrées horodatées d'événement : ● Mot 0 : ❍ Octet 0 : Réservé ❍ Octet 1 : valeur de la variable après détection de la modification ● Mot 1 : ID de l'événement ● Mot 2 : nombre de secondes depuis le 01/01/1970 minuit (bits 15 à 0) ● Mot 3 : nombre de secondes depuis le 01/01/1970 minuit (bits 31 à 16) ● Mot 4 : fraction de seconde (bits 15 à 0) ● Mot 5 : ❍ Octet 0 : fraction de seconde (bits 23 à 16) ❍ Octet 1 : qualité de l'heure **NOTE :** la taille du paramètre BUFFER doit être un multiple de 6. Sinon, l'erreur 16#0900 est générée.
NB_EVENT	INT	Nombre de nouveaux événements lus dans le tampon local du module BMX ERT 1604T.

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

Paramètre	Type	Commentaire
CTRLSTAMP	DDT CTRL_STAMP	Spécifie le tampon d'enregistrements bruts de l'UC : ● index de DEBUT : INT ● index de FIN : INT ● Mode de fonctionnement (une description détaillée du mode du mot de fonctionnement *(voir page 215)* est fournie ci-dessous) : WORD.

Description du paramètre STATUS

Le tableau suivant décrit le paramètre STATUS :

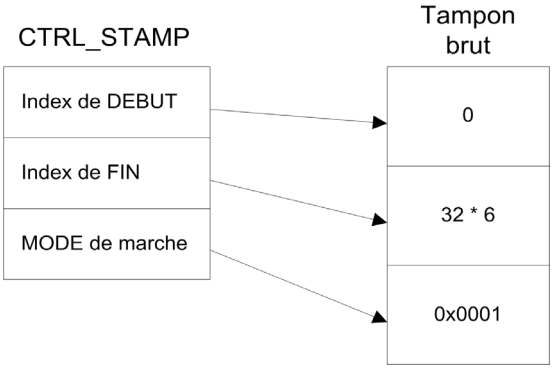
Compte rendu de communication (octet 0)		Compte rendu d'opération (octet 1)	
Valeur	**Description**	**Valeur**	**Description**
00 hex	Echange correct (requête traitée correctement)	00 hex	Résultat positif
		01 hex	Le nombre d'événements dans le tampon de l'automate a atteint la valeur maximale.
		02 hex	Le tampon est saturé et des événements ont été écrasés depuis le dernier échange.
		04 hex	Le tampon est saturé et l'enregistrement est interrompu.
01 hex	Echange arrêté suite à un timeout	00 hex	Valeur par défaut
02 hex	Echange arrêté à la demande de l'utilisateur (CANCEL)	00 hex	Valeur par défaut
03 hex	Format d'adresse incorrect	00 hex	Valeur par défaut
04 hex	Adresse cible incorrecte	00 hex	Valeur par défaut
06 hex	Paramètres spécifiques incorrects	01 hex	Paramètre CMD non valide.
		02 hex	Des paramètres utilisateur ont été modifiés entre deux invocations pendant l'exécution de l'EFB.
07 hex	Problème lors de l'envoi à la cible	00 hex	Valeur par défaut
09 hex	Taille insuffisante du tampon de réception (<1 EVT) ou taille du tampon ne correspondant pas à un multiple de 6 entiers	00 hex	Valeur par défaut
0B hex	Processeur sans ressources système	00 hex	Valeur par défaut
FF hex	Echange incorrect (échec du traitement de la demande)	FF hex	Erreur de communication générale

NOTE : si un code d'erreur ne figure pas dans le tableau ci-dessus, consultez les autres codes d'erreur.

NOTE : le paramètre ENO est réglé sur 1 en cas d'échange correct.

Description du paramètre CTRLSTAMP

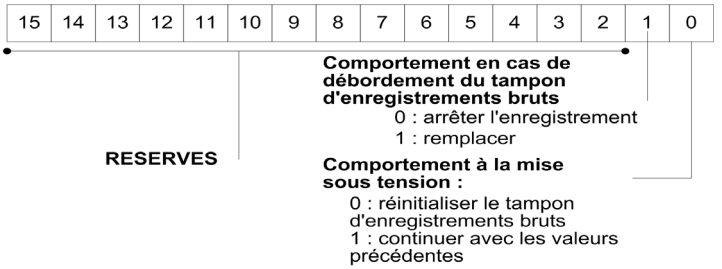
Exemple de structure de DDT CTRL_STAMP et lien avec le tampon de l'automate :



L'exemple ci-dessus montre le contenu de CTRL_STAMP après l'écriture de 32 événements (1 entrée d'événement compte 6 mots) dans le tampon d'automate configuré comme suit :

- Le tampon de l'automate est localisé et il y a 32 événements à écrire.
- Arrêtez l'enregistrement lorsque le tampon est saturé et continuez avec la valeur précédente à la mise sous tension.

Structure du mot de MODE de marche :



Niveau du tampon :

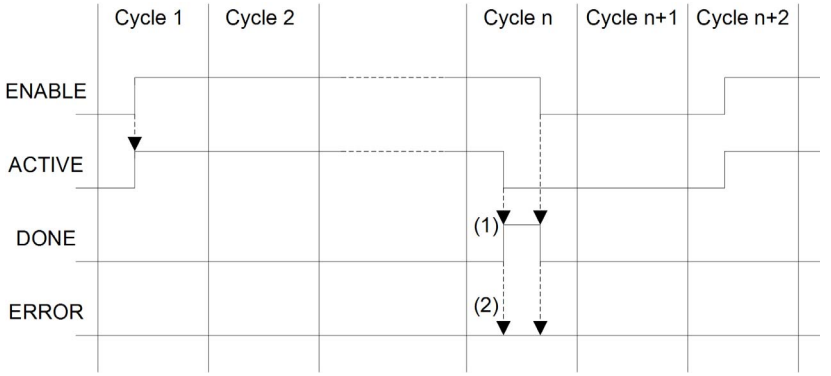
- Si index de DEBUT = index de FIN, le tampon est vide
- Si index de FIN + 6 = index de DEBUT, le tampon est saturé (dans l'équation précédente, 6 représente la taille d'un événement). Le tampon est saturé lorsqu'il reste de l'espace pour un seul événement (6 x INT).

Le comportement de l'EFB en cas de tampon saturé dépend de la valeur du bit 1 du paramètre `MODE de marche` :

- Si le bit de **comportement en cas de débordement du tampon d'enregistrements bruts** est réglé sur 0 (arrêter l'enregistrement), le tampon n'est pas alimenté en nouvelles données.
- Si le bit de **comportement en cas de débordement du tampon d'enregistrements bruts** est réglé sur 1 (remplacer le tampon), les données anciennes sont remplacées par les nouvelles. Dans ce cas, l'EFB met à jour à la fois l'index de DEBUT et l'index de FIN.

Mode de fonctionnement des paramètres Enable, Active, Done et Error

Les paramètres `ENABLE`, `ACTIVE`, `DONE` (ou `SUCCESS`) et `ERROR` fonctionnent de la manière suivante :



(1) `DONE` = 1 si aucune erreur, DONE = 0 si erreur
(2) `ERROR` = 0 si aucune erreur, ERROR = 1 si erreur

Le paramètre `ENABLE` est écrit par l'application.

Les paramètres `ACTIVE`, `DONE` et `ERROR` sont lus par l'application.

Pour ne lancer la fonction de communication qu'une seule fois, le signal `ENABLE` doit être remis à 0 dès que le paramètre `ACTIVE` est réglé sur 0. Si le paramètre `ENABLE` est maintenu à 1 lorsque le paramètre `ACTIVE` est réglé sur 0, la fonction de communication est relancée et le paramètre `ACTIVE` sera réglé sur 1 lors du cycle suivant.

Chapitre 42
GET_TS_EVT_Q : lecture du tampon des événements horodatés Quantum

Description

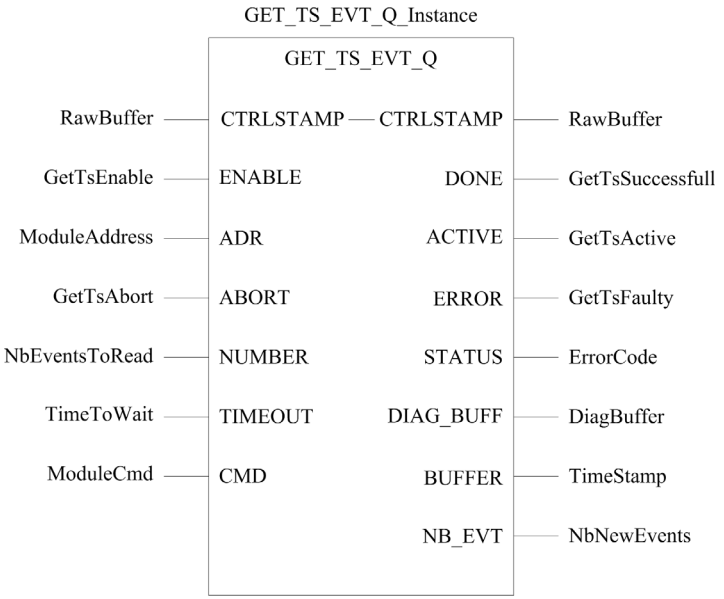
Description de la fonction

Le bloc fonction GET_TS_EVT_Q obtient les données horodatées d'un module d'E/S distantes Ethernet BMX CRA 312 10 ou d'un module BMX ERT 1604T dans une station d'E/S distantes (architecture Quantum). Il permet de lire le tampon d'événements du module d'E/S horodatées source (BMX CRA 312 10 ou BMX ERT 1604T) et de le rendre disponible à l'application automate Quantum.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

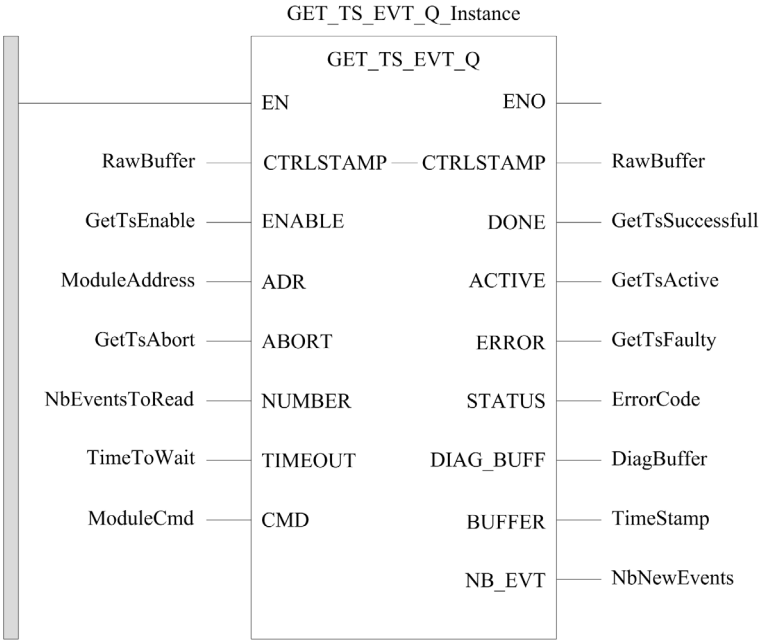
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
CAL GET_TS_EVT_Q_Instance (CTRLSTAMP:=RawBuffer, ENABLE:=GetTsEnable,
ADR:=ModuleAddress, ABORT:=GetTsAbort, NUMBER:=NbEventsToRead,
TIMEOUT:=TimeToWait, CMD:=ModuleCmd, DONE=>GetTsSuccessfull,
ACTIVE=>GetTsActive, ERROR=>GetTsFaulty, STATUS=>ErrorCode,
DIAG_BUFF=>DiagBuffer, BUFFER=>TimeStamp, NB_EVT=>NbNewEvents)
```

Représentation en ST

Représentation :

```
GET_TS_EVT_Q_Instance (CTRLSTAMP:=RawBuffer, ENABLE:=GetTsEnable,
ADR:=ModuleAddress, ABORT:=GetTsAbort, NUMBER:=NbEventsToRead,
TIMEOUT:=TimeToWait, CMD:=ModuleCmd, DONE=>GetTsSuccessfull,
ACTIVE=>GetTsActive, ERROR=>GetTsFaulty, STATUS=>ErrorCode,
DIAG_BUFF=>DiagBuffer, BUFFER=>TimeStamp, NB_EVT=>NbNewEvents)
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
ENABLE	BOOL	Réglé sur 1 pour envoyer la requête au module concerné.
ADR	ANY_ARRAY_INT	Tableau contenant l'adresse de l'esclave Modbus, c'est-à-dire le résultat de la fonction ADDMX *(voir Unity Pro, Communication, Bibliothèque de blocs)*.
ABORT	BOOL	Réglé sur 1 pour abandonner l'opération en cours.
NUMBER	INT	Nombre maximum d'événements à lire dans le tampon local du module. **NOTE :** Lorsque le numéro de l'élément demandé est 0, une erreur de paramètre incorrect apparaît (compte rendu de communication (octet 0) = 06 hex) et le tampon de diagnostic n'est pas mis à jour.
TIMEOUT	INT	Délai d'attente maximum de la réponse de la station. La base de temps pour ce paramètre est de 100 ms. **NOTE :** TIMEOUT = 0 correspond à un délai d'attente infini.
CMD	INT	Réglé sur: ● 0 : lire le tampon du module ● 1 : réinitialiser le tampon du module

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
DONE	BOOL	Exécution terminée du bloc fonction. Réglé sur 1 lorsque l'exécution du bloc fonction a abouti.
ACTIVE	BOOL	Bloc fonction en cours d'exécution. Réglé sur 1 lorsque l'exécution du bloc fonction est en cours.
ERROR	BOOL	Réglé sur 1 si une erreur est détectée par le bloc fonction.
STATUS	WORD	Code fournissant un compte rendu de communication et d'opération. *(voir page 221)* ● Octet 0 : compte rendu de communication ● Octet 1 : compte rendu d'opération

Paramètre	Type	Commentaire
DIAG_BUFF	ANY_ARRAY_INT	Description: ● Octet 0 : pourcentage de saturation du tampon ● Octet 1 : indicateurs de diagnostic : ❍ Bit 0 = 1 : l'heure est OK et synchronisée ❍ Bit 1 = 1 : le tampon est saturé, des événements peuvent être perdus Cela se produit en mode **Ecrasement**, lorsque certains ou la totalité des événements ont été écrasés depuis la dernière réponse. Le mode **Ecrasement** n'est disponible que si le BM• CRA 312 10 est en mode **Application** et il n'est pas pris en charge par le BMX ERT 1604T. ❍ Bit 2 = 1 : le tampon est saturé et l'horodatage s'est arrêté Cela se produit en mode **Arrêt en cas de tampon saturé**, lequel est pris en charge par le BMX ERT 1604T et le BM• CRA 312 10 en mode **Système**. Tout nouvel événement détecté avant la lecture du tampon d'événements est perdu. ❍ Bit 3 = 1 : la requête de lecture est la première car le module 140 CRA 312 10 ou BM• CRA 312 10 était alimenté (le BMX ERT 1604T n'utilise pas ce bit).
BUFFER	ANY_ARRAY_INT	Tampon brut contenant des entrées horodatées d'événement : ● Mot 0 : ❍ Octet 0 : Réservé ❍ Octet 1 : valeur de la variable après détection de la modification ● Mot 1 : ID de l'événement ● Mot 2 : nombre de secondes depuis le 01/01/1970 minuit (bits 15 à 0) ● Mot 3 : nombre de secondes depuis le 01/01/1970 minuit (bits 31 à 16) ● Mot 4 : fraction de seconde (bits 15 à 0) ● Mot 5 : ❍ Octet 0 : fraction de seconde (bits 23 à 16) ❍ Octet 1 : qualité de l'heure *(voir Horodatage système, Guide de l'utilisateur)* **NOTE :** la taille du paramètre BUFFER doit être un multiple de 6. Sinon, l'erreur 16#0900 est générée.
NB_EVENT	INT	Nombre de nouveaux événements lus dans le tampon local du module BMX CRA 312 10 ou BMX ERT 1604T (station d'E/S distantes Ethernet).

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

Paramètre	Type	Commentaire
CTRLSTAMP	DDT CTRL_STAMP	Spécifie le tampon d'enregistrements bruts de l'UC : ● index de DEBUT : INT ● index de FIN : INT ● Mode de fonctionnement (une description détaillée du mode du mot de fonctionnement *(voir page 222)* est fournie ci-dessous) : WORD.

Description du paramètre STATUS

Le tableau suivant décrit le paramètre STATUS :

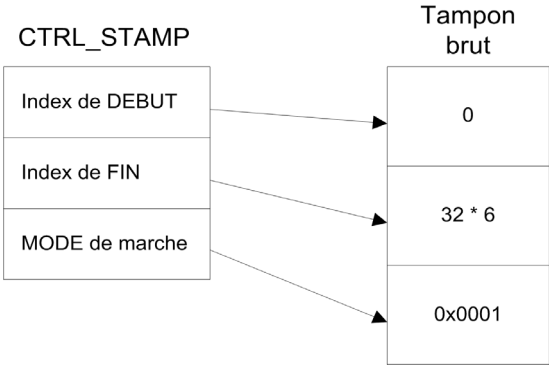
Compte rendu de communication (octet 0)		Compte rendu d'opération (octet 1)	
Valeur	**Description**	**Valeur**	**Description**
00 hex	Echange correct (requête traitée correctement)	00 hex	Résultat positif
		01 hex	Le nombre d'événements dans le tampon de l'automate a atteint la valeur maximale.
		02 hex	Le tampon est saturé et des événements ont été écrasés depuis le dernier échange.
		04 hex	Le tampon est saturé et l'enregistrement est interrompu.
01 hex	Echange arrêté suite à un timeout	00 hex	Valeur par défaut
02 hex	Echange arrêté à la demande de l'utilisateur (CANCEL)	00 hex	Valeur par défaut
03 hex	Format d'adresse incorrect	00 hex	Valeur par défaut
04 hex	Adresse cible incorrecte	00 hex	Valeur par défaut
06 hex	Paramètres spécifiques incorrects	00 hex	Valeur par défaut ou numéro d'événement réglé sur 0
		01 hex	Paramètre CMD non valide
		02 hex	Des paramètres utilisateur ont été modifiés entre deux invocations pendant l'exécution de l'EFB
09 hex	Taille insuffisante du tampon de réception (<1 EVT) ou taille du tampon ne correspondant pas à un multiple de 6 entiers	00 hex	Valeur par défaut
0B hex	Processeur sans ressources système	00 hex	Valeur par défaut
FF hex	Echange incorrect (échec du traitement de la requête)	FF hex	Erreur de communication générale détectée

NOTE : si un code d'erreur ne figure pas dans le tableau ci-dessus, consultez les codes d'erreur génériques *(voir page 267)*.

NOTE : le paramètre ENO est réglé sur 1 en cas d'échange correct.

Description du paramètre CTRLSTAMP

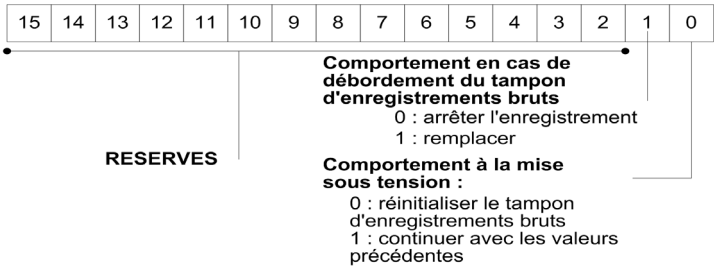
Exemple de structure de DDT CTRL_STAMP et lien avec le tampon de l'automate :



L'exemple ci-dessus montre le contenu de CTRL_STAMP après l'écriture de 32 événements (1 entrée d'événement compte 6 mots) dans le tampon d'automate configuré comme suit :

- Le tampon de l'automate est localisé et il y a 32 événements à écrire.
- Arrêtez l'enregistrement lorsque le tampon est saturé et continuez avec la valeur précédente à la mise sous tension.

Structure du mot de MODE de marche :



Niveau du tampon :

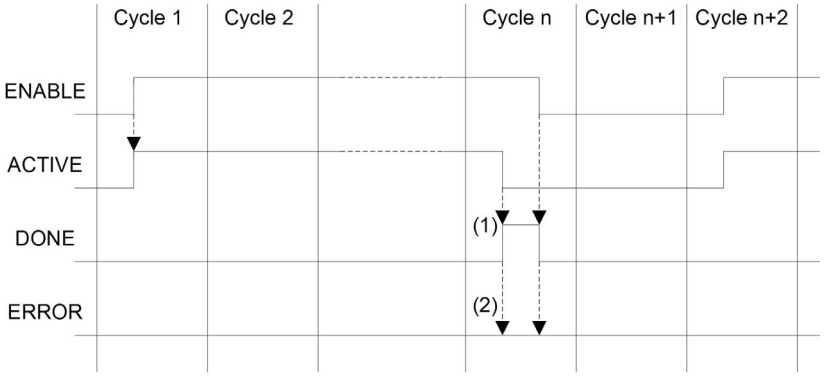
- Si index de DEBUT = index de FIN, le tampon est vide
- Si index de FIN + 6 = index de DEBUT, le tampon est saturé (dans l'équation précédente, 6 représente la taille d'un événement). Le tampon est saturé lorsqu'il reste de l'espace pour un seul événement (6 x INT).

Le comportement de l'EFB en cas de tampon saturé dépend de la valeur du bit 1 du paramètre `MODE de marche` :

- Si le bit de **comportement en cas de débordement du tampon d'enregistrements bruts** est réglé sur 0 (arrêter l'enregistrement), le tampon n'est pas alimenté en nouvelles données.
- Si le bit de **comportement en cas de débordement du tampon d'enregistrements bruts** est réglé sur 1 (remplacer le tampon), les données anciennes sont remplacées par les nouvelles. Dans ce cas, l'EFB met à jour à la fois l'index de DEBUT et l'index de FIN.

Mode de fonctionnement des paramètres Enable, Active, Done et Error

Les paramètres `ENABLE`, `ACTIVE`, `DONE` (ou `SUCCESS`) et `ERROR` fonctionnent de la manière suivante :



(1) `DONE` = 1 si aucune erreur, DONE = 0 si erreur
(2) `ERROR` = 0 si aucune erreur, ERROR = 1 si erreur

Le paramètre `ENABLE` est écrit par l'application.

Les paramètres `ACTIVE`, `DONE` et `ERROR` sont lus par l'application.

Pour ne lancer la fonction de communication qu'une seule fois, le signal `ENABLE` doit être remis à 0 dès que le paramètre `ACTIVE` est réglé sur 0. Si le paramètre `ENABLE` est maintenu à 1 lorsque le paramètre `ACTIVE` est réglé sur 0, la fonction de communication est relancée et le paramètre `ACTIVE` sera réglé sur 1 lors du cycle suivant.

Chapitre 43
PTC : lecture de la date et du code d'arrêt

Description

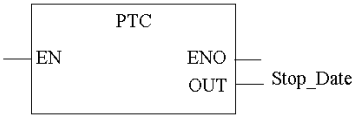
Description de la fonction

La fonction `PTC` lit la date et le code de l'arrêt d'automate le plus récent, et enregistre ces informations dans un tableau d'entiers.

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

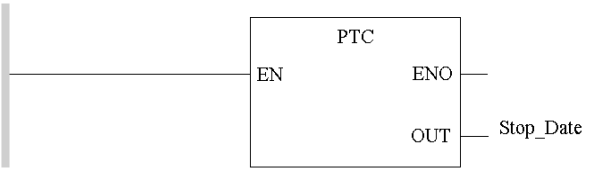
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
PTC
ST Stop_Date
```

Représentation en ST

Représentation :

```
PTC (Stop_Date);
```

Description des paramètres

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Stop_Date	ARRAY [0..4] OF INT	Tableau de 5 entiers contenant la date dans les quatre premiers mots (équivalent de %SW54 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* à %SW57 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*) et le code d'erreur dans le dernier mot. Le code d'erreur est celui indiqué dans le mot système %SW58 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* : ● 1 = passage de l'état RUN à l'état STOP par le terminal, ● 2 = arrêt suite à un défaut logiciel (débordement de la tâche de l'automate), ● 4 = coupure de courant secteur, ● 5 = arrêt suite à une défaillance matérielle, ● 6 = arrêt suite à une instruction HALT. **Exemple** : arrêt à 22:53:10 le 8 janvier 2001. Le contenu de Stop_Date était le suivant : Stop_Date[0]=16#1000 Stop_Date[1]=16#2253 Stop_Date[2]=16#0108 Stop_Date[3]=16#2001 Stop_Date[4]=16#0006

Chapitre 44
RRTC_DT : lecture de la date système

Description

Description de la fonction

La fonction RRTC_DT stocke la date courante de l'horodateur de l'automate. Il s'agit de la valeur fournie dans Heure locale.

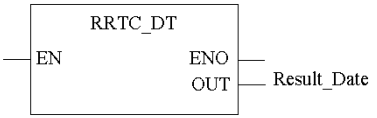
Vous devez configurer au moins une fois l'heure de l'automate avec Unity Pro dans l'écran Automate (onglet Horodateur).

NOTE : Pour les automates M580, les mots système %SW49 à %SW53 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* indiquent l'heure courante en UTC (Coordinated Universal Time, temps universel coordonné).

Les paramètres supplémentaires EN et ENO peuvent être configurés.

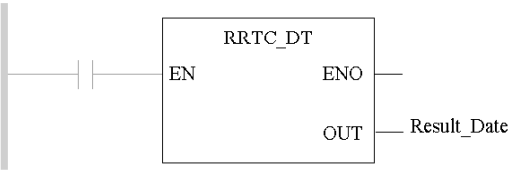
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
RRTC_DT
ST Result_Date
```

Représentation en ST

Représentation :

```
RRTC_DT(Result_Date);
```

Description des paramètres

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
`Result_Date`	`DT`	`Result_Date` contient la date courante au format DT.

Chapitre 45
RRTC_DT_MS : fonction d'horodateur du réseau

Description

Description de la fonction

La fonction `RRTC_DT_MS` permet de récupérer la date et l'heure de l'automate en temps UTC (Coordinated Universal Time, temps universel coordonné) avec une précision de 10 ms. Le résultat est renvoyé par deux sorties présentant un format différent :

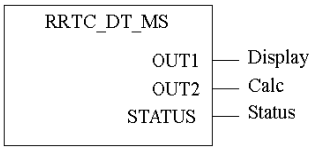
- format d'affichage,
- format de calcul.

NOTE : sur les automates M580, les mots système %SW49 à %SW53 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* indiquent également la date et l'heure courantes en UTC (Coordinated Universal Time, temps universel coordonné). Pour obtenir l'heure locale, utilisez la fonction `RRTC_DT` *(voir page 227)*.

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

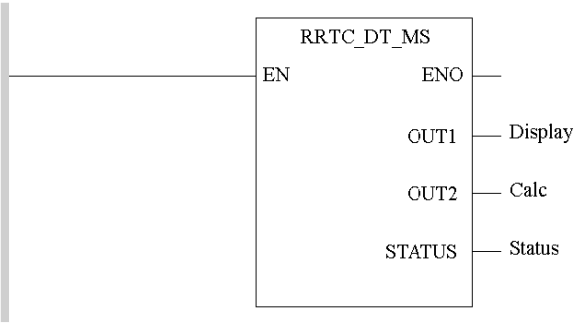
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
RRTC_DT_MS(Display, Calc, Status)
```

Représentation en ST

Représentation :

```
RRTC_DT_MS(Display, Calc, Status);
```

Description des paramètres

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Affichage	`Display_NTPC`	La sortie `Display` contient la date et l'heure de l'automate. Le type `Display_NTPC` est une structure prédéfinie comprenant un élément de type `DT` et un élément de type `INT`. Ce qui nous donne : • `Display.DT_value` contenant la date, • `Display.Milisecond` contenant le nombre de millisecondes de cette date, car la seconde est l'unité de mesure minimale du format `DT`.

Paramètre	Type	Commentaire
Calc	Calc_NTPC	La sortie Calc contient les informations de date et d'heure provenant d'un serveur NTP (comme dans une variable Display, mais dans un format différent). Le type Calc_NTPC est une structure prédéfinie comprenant un élément de type UDINT et un élément de type INT. Ce qui nous donne : ● Calc.Seconds contenant le nombre de secondes écoulées depuis le 1er janvier 1980 à 00:00. ● Calc.Fraction_Second contenant le nombre de millisecondes à ajouter pour obtenir un résultat avec une précision de l'ordre de la milliseconde.
STATUS	INT	L'octet de poids faible est contrôlé par l'UC. Lorsque cet octet est réglé sur 0 : ● la valeur d'horloge n'est pas disponible ; ● la date et l'heure ne sont pas mises à jour au cours des deux dernières minutes. Lorsque cet octet est réglé sur 1 : ● la date et l'heure sont mises à jour au cours des deux dernières minutes ; ● la date et l'heure sont acceptables.
		Si le processeur est client NTP, l'octet de poids fort est géré par le module Ethernet. Lorsque cet octet est réglé sur 0, la valeur d'horloge transmise à l'UC n'est pas acceptable. Lorsque cet octet est réglé sur 1, la date/heure mise à jour reçue du serveur et envoyée au module est : ● comprise dans un intervalle de deux minutes, ● acceptable (décalage de 10 ms maximum).
	Pour que l'heure valide de l'UC soit valide, l'octet de poids fort et l'octet de poids faible du paramètre STATUS doivent être réglés sur 1. Dans le cas contraire, une erreur d'exécution est générée (voir ci-dessous).	

Erreurs d'exécution

Si l'octet de poids faible ou de poids fort de la sortie STATUS est défini sur 0, le bit **%S18** *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* est défini sur 1 par le système de l'automate.

Chapitre 46
R_NTPC : fonction d'horodateur du réseau

Description

Description de la fonction

La fonction R_NTPC est disponible pour les automates M340, M580 Premium et Quantum. Elle permet de récupérer la date et l'heure d'un serveur NTP dans deux formats :

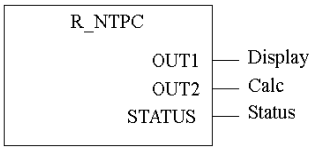
- format d'affichage,
- format de calcul.

NOTE : cette fonction nécessite une connexion à un réseau Ethernet permettant d'accéder à un serveur NTP *(voir Modicon Quantum with Control Expert, Ethernet Network Modules, User Manual)*.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

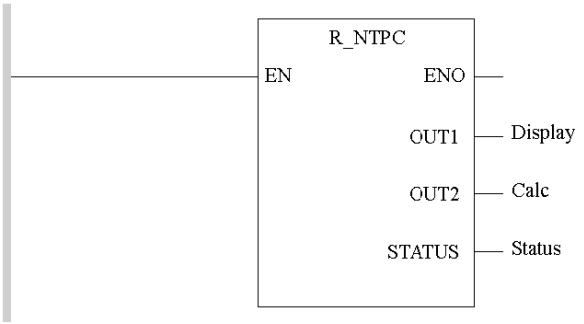
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
R_NTPC(Display, Calc, Status)
```

Représentation en ST

Représentation :

```
R_NTPC(Display, Calc, Status);
```

Description des paramètres

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Affichage	`Display_NTPC`	La sortie `Display` contient les informations de date et d'heure provenant d'un serveur NTP. Le type `Display_NTPC` est une structure prédéfinie comprenant un élément de type `DT` et un élément de type `INT`. Ce qui nous donne : • `Display.DT_value` contenant la date, • `Display.Milisecond` contenant le nombre de millisecondes de cette date, car la seconde est l'unité de mesure minimale du format `DT`.
Calc	`Calc_NTPC`	La sortie `Calc` contient les informations de date et d'heure provenant d'un serveur NTP (comme pour la variable `Display`), mais dans un format différent. Le type `Calc_NTPC` est une structure prédéfinie comprenant un élément de type `UDINT` et un élément de type `INT`. Ce qui nous donne : • `Calc.Seconds` contenant le nombre de secondes écoulées depuis le 1er janvier 1980 à 00:00. • Calc.Fraction_Second contenant le nombre de millisecondes à ajouter pour obtenir un résultat avec une précision de l'ordre de la milliseconde.

Paramètre	Type	Commentaire
STATUS	INT	L'octet de poids faible est contrôlé par l'UC. Lorsque cet octet est réglé sur 0 : ● la valeur d'horloge n'est pas disponible ; ● la date et l'heure ne sont pas mises à jour au cours des deux dernières minutes. Lorsque cet octet est réglé sur 1 : ● la date et l'heure sont mises à jour au cours des deux dernières minutes ; ● la date et l'heure sont acceptables.
		Si le processeur est client NTP, l'octet de poids fort est géré par le module Ethernet. Lorsque cet octet est réglé sur 0, la valeur d'horloge transmise à l'UC n'est pas acceptable. Lorsque cet octet est réglé sur 1, la date/heure mise à jour reçue du serveur et envoyée au module est : ● comprise dans un intervalle de deux minutes, ● acceptable (décalage de 10 ms maximum).
	Pour que l'heure valide de l'UC soit valide, l'octet de poids fort et l'octet de poids faible du paramètre STATUS doivent être réglés sur 1. Sinon, une erreur d'exécution est générée (voir l'explication ci-dessous).	

Erreurs d'exécution

Si l'octet de poids faible ou de poids fort de la sortie STATUS est défini sur 0, le bit **%S18** *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)* est défini sur 1 par le système PLC.

Chapitre 47
SCHEDULE : fonction d'horodateur

Description

Description de la fonction

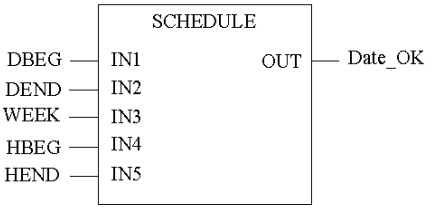
La fonction SCHEDULE permet de commander des actions à des horaires et des dates prédéfinis ou calculés.

Elle règle sur 1 la sortie OUT si la date fournie par l'horloge de l'automate au moment de l'appel de la fonction appartient à la période programmée dans les paramètres d'entrée.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

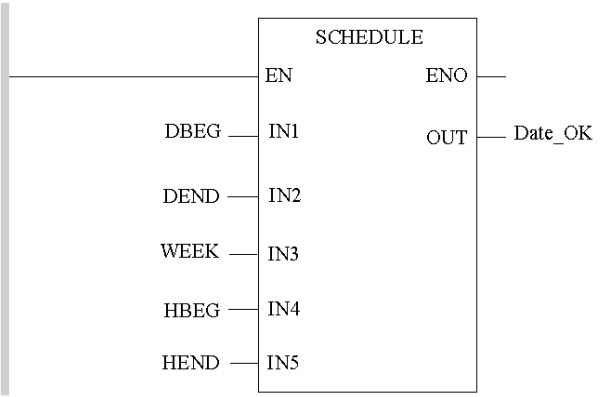
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD DBEG
SCHEDULE DEND, WEEK, HBEG, HEND, Date_OK
```

Représentation en ST

Représentation :

```
SCHEDULE(DBEG, DEND, WEEK, HBEG, HEND, Date_OK);
```

Description des paramètres

Le tableau ci-après décrit les paramètres d'entrée.

Paramètre	Type	Commentaire
DBEG	INT	Entier codant la date de début de la période (mois-jour) en BCD (valeurs limites : 01-01 à 12-31).
DEND	INT	Entier codant la date de fin de la période (mois-jour) en BCD (valeurs limites : 01-01 à 12-31).
WEEK	INT	Entier codant le ou les jours de la semaine pris en compte dans la période définie par les paramètres DBEG et DEND. Les sept bits de poids faible représentent les sept jours de la semaine : bit 6 = Lundi, bit 5 = Mardi, etc., bit 0 = Dimanche.

Paramètre	Type	Commentaire
HBEG	BYTE	Entier double codant l'heure de début de la période dans la journée (heures-minutes-secondes) en BCD format heure du jour. Valeurs limites : 00:00:00, 23:59:59.
HEND	BYTE	Entier double codant l'heure de fin de la période dans la journée (heures-minutes-secondes) en BCD format heure du jour. Valeurs limites : 00:00:00, 23:59:59.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Date_OK	EBOOL	La sortie Date_OK est réglée sur 1 si la date fournie par l'horloge de l'automate au moment de l'appel de la fonction appartient à la période programmée dans les paramètres d'entrée.

NOTE :

- Les deux paramètres DBEG et DEND définissent une plage de jours dans l'année. Cette plage peut s'étendre sur deux années civiles. Exemple : du 10 octobre au 7 avril. Le 29 février peut être utilisé dans cette période, il sera ignoré les années non bissextiles.
- Les deux paramètres HBEG et HEND définissent une plage d'heures dans la journée. Cette plage peut s'étendre sur deux jours. Exemple : de 22:00 à 06:10:20.
- Si l'une des dates DBEG et DEND ou l'une des heures HBEG et HEND est erronée, (c'est-à-dire qu'elle ne correspond pas à une date ou une heure réelle), la sortie Date_OK est réglée sur 0 et le bit **%S18** sur 1.
- Lorsque la précision est secondaire, il est possible d'alléger la charge du processeur de l'automate en cadençant les appels à la fonction SCHEDULE par le bit système **%S6** ou **%S7**.

Exemples

Programmation de deux plages horaires non continues :

SCHEDULE	(16#0501, 16#1031, 2#0000000001111100, 16#08300000, 16#12000000, Date1_OK);	(*date de début : 1er mai*) (*date de fin : 31 octobre*) (*lundi à vendredi*) (*heure de début : 08:30*) (*heure de fin : 12:00*) (*résultat dans Date1_OK*)
SCHEDULE	(16#0501, 16#1031, 2#0000000001111100, 16#14000000, 16#18000000, Date2_OK);	(*date de début : 1er mai*) (*date de fin : 31 octobre*) (*lundi à vendredi*) (*heure de début : 14:00*) (*heure de fin : 18:00*) (*résultat dans Date2_OK*)
%Q0.0	:=Date1_OK OR Date2_OK;	(*affectation de la sortie sur Date1_OK ou Date2_OK*)

Chapitre 48
WRTC_DT : mise à jour de la date système

Description

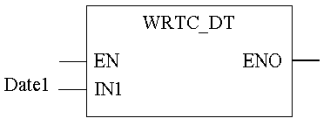
Description de la fonction

La fonction `WRTC_DT` met à jour la date courante dans l'horloge temps réel de l'automate.

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

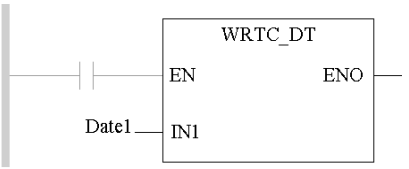
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Date1
WRTC_DT
```

Représentation en ST

Représentation :

```
WRTC_DT(Date1);
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Date1	DT	Date1 doit contenir la valeur courante de la date au format DT. Le contenu de cette variable doit être affecté par le programme avant de lancer la fonction.

Partie VII
Particularités système

Vue d'ensemble

Cette section décrit les fonctions et blocs fonction élémentaires des `particularités système`.

Contenu de cette partie

Cette partie contient les chapitres suivants :

Chapitre 49
IS_PAR_CON : paramètre connecté ?

Introduction

Ce chapitre décrit le bloc fonction `IS_PAR_CON`.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Description

Description de la fonction

Cette fonction est utilisée dans un code DFB afin de savoir si un paramètre d'entrée (broche) ou un paramètre d'E/S (broche) est connecté ou relié à une variable lors de l'utilisation d'une instance du DFB.

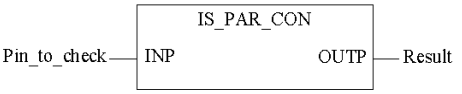
NOTE : cette fonction ne peut être utilisée que dans un code DFB. Le recours à la fonction `IS_PAR_CON` ailleurs que dans un code DFB n'est pas autorisé.

Dans le code DFB, vous devez saisir le nom des paramètres d'entrée ou d'E/S à vérifier. Il est impossible de saisir le numéro de broche. Pour plus d'informations, voir *Exemple, page 248*.

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

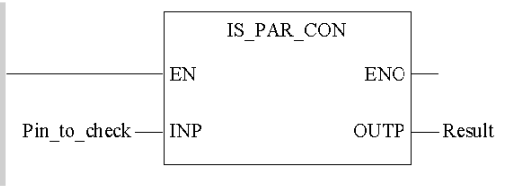
Représentation en FBD

Représentation :



Représentation en LD

Représentation :



Représentation en IL

Représentation :

```
LD Pin_to_check IS_PAR_CON ST Result
```

Représentation en ST

Représentation :

```
Result := IS_PAR_CON (Pin_to_check) ;
```

Description des paramètres

Description du paramètre d'entrée :

Paramètres	Type de données	Signification
INP	Any	nom du paramètre (broche DFB) à vérifier

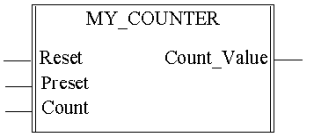
Description du paramètre de sortie :

Paramètres	Type de données	Signification
OUTP	BOOL	1 : Paramètre (broche DFB) connecté 0 : Paramètre (broche DFB) non connecté

Utilisation

Exemple

Dans cet exemple, les paramètres d'entrée (broches) du DFB suivant doivent être vérifiés pour déterminer s'ils sont connectés ou non.



Dans ce cas, le code DFB apparaît comme suit

```
(* Functional code of MY_COUNTER *)
IF RE (Reset) THEN
    internal_value:=0;
END_IF;
IF RE (Count) THEN
    internal_value:=internal_value+1;
END_IF;IF(internal_value>=Preset) THEN
    SET (Done);
ELSE
    RESET (Done);
END_IF;

(* Check if parameters are connected *)
ResetConnected := IS_PAR_CON (Reset) ;
PresetConnected := IS_PAR_CON (Preset) ;
CountConnected := IS_PAR_CON (Count) ;
```

Chapitre 50
IS_BIT_FORCED: bloc fonction

Description

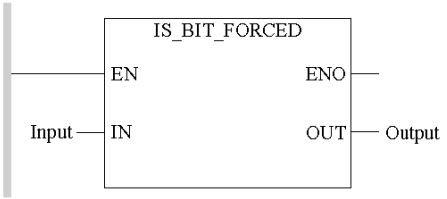
Description de la fonction

La fonction `IS_BIT_FORCED` teste si une variable `EBOOL` localisée est efforcée.

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

Représentation en LD

Représentation :



Représentation en FBD

Représentation :

Représentation en IL

Représentation :

```
CAL IS_BIT_FORCED (IN := Input)
Sortie ST
```

Représentation en ST

Représentation :

```
Sortie := IS_BIT_FORCED(IN := Input);
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
`Input`	`EBOOL`	Variable à tester pour savoir si elle est forcée et correspondant à une variable `EBOOL` localisée.

Le tableau suivant décrit le paramètre de sortie :

Paramètre	Type	Commentaire
`Output`	`BOOL`	Résultat : ● Vrai si `Input` est forcé ● Faux si `Input` n'est pas forcé

Chapitre 51
UNFORCE_BIT : bloc fonction

Description

Description de la fonction

La fonction UNFORCE_BIT lève le forçage d'une variable de type EBOOL localisée. Si la variable n'est pas forcée, cet EF n'a aucun effet.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

NOTE : le comportement de la fonction UNFORCE_BIT dépend de l'état du bit système %S79 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*.

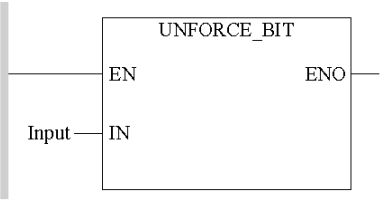
NOTE :
Le comportement des sorties forcées (%M) est différent dans Modsoft/NxT/Concept et dans Unity Pro.

- Avec Modsoft/NxT/Concept, vous **ne pouvez pas** forcer les sorties lorsque le commutateur de protection mémoire de l'UC Quantum est sur ON.
  Dans Unity Pro, vous **pouvez** forcer les sorties même lorsque le commutateur de protection mémoire de l'UC Quantum est sur ON.
- Avec Modsoft/NxT/Concept, les sorties forcées **conservent** leurs valeurs après un démarrage à froid.
  Avec Unity Pro, les sorties forcées **perdent** leurs valeurs après un démarrage à froid.

⚠ AVERTISSEMENT
COMPORTEMENT INATTENDU DES VARIABLES Vérifiez que le comportement des sorties forcées est cohérent avec l'application. **Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.**

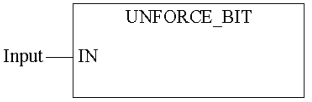
Représentation en LD

Représentation :



Représentation en FBD

Représentation :



Représentation en IL

Représentation :

```
CAL UNFORCE_BIT (IN := Input)
```

Représentation en ST

Représentation :

```
UNFORCE_BIT (IN := Input);
```

Description des paramètres

Le tableau suivant décrit le paramètre d'entrée :

Paramètre	Type	Commentaire
`Input`	`EBOOL`	La variable pour laquelle sera levé le forçage doit correspondre à une variable localisée de type `EBOOL`.

Chapitre 52
FORCE_BIT : bloc fonction

Description

Description de la fonction

La fonction FORCE_BIT permet de forcer une variable booléenne de type EBOOL sur 0 ou sur 1. Si la variable est déjà forcée sur 0 (ou sur 1), cet EF peut remplacer par 1 (ou 0) la valeur forcée sans appeler auparavant l'EF UNFORCE_BIT.

Les paramètres supplémentaires EN et ENO peuvent être configurés.

NOTE : le comportement de la fonction FORCE_BIT dépend de l'état du bit système %S79 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*.

NOTE :
Le comportement des sorties forcées (%M) est différent dans Modsoft/NxT/Concept et dans Unity Pro.

- Avec Modsoft/NxT/Concept, vous **ne pouvez pas** forcer les sorties lorsque le commutateur de protection mémoire de l'UC Quantum est sur ON.
  Dans Unity Pro, vous **pouvez** forcer les sorties même lorsque le commutateur de protection mémoire de l'UC Quantum est sur ON.
- Avec Modsoft/NxT/Concept, les sorties forcées **conservent** leurs valeurs après un démarrage à froid.
  Avec Unity Pro, les sorties forcées **perdent** leurs valeurs après un démarrage à froid.

⚠ AVERTISSEMENT

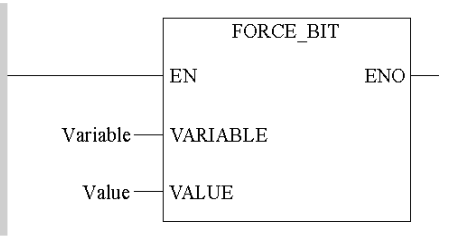
COMPORTEMENT INATTENDU DES VARIABLES

Vérifiez que le comportement des sorties forcées est cohérent avec l'application.

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

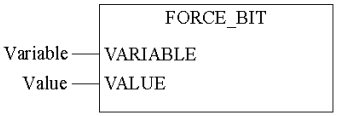
Représentation en LD

Représentation :



Représentation en FBD

Représentation :



Représentation en IL

Représentation :

```
CAL FORCE_BIT ( VARIABLE := Variable,VALUE := Value)
```

Représentation en ST

Représentation :

```
FORCE_BIT ( VARIABLE := Variable,VALUE := Value)
```

Description des paramètres

Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
VARIABLE	`EBOOL`	La variable à forcer doit correspondre à une variable localisée de type `EBOOL`.
VALUE	`BOOL`	Valeur de forçage

Chapitre 53
SIG_WRITE : écriture d'une signature

SIG_WRITE : écriture d'une signature dans la carte mémoire

Description de la fonction

La fonction SIG_WRITE permet d'écrire une signature spécifique dans une carte mémoire BMX RM •••. Une application incorporant la fonction SIG_CHECK EF ne s'exécute que sur une carte mémoire contenant une signature attendue.

La fonction SIG_WRITE écrit une signature sur la carte mémoire insérée dans une UC BMX P34 ••••. La signature est constituée de 8 mots (16 octets).

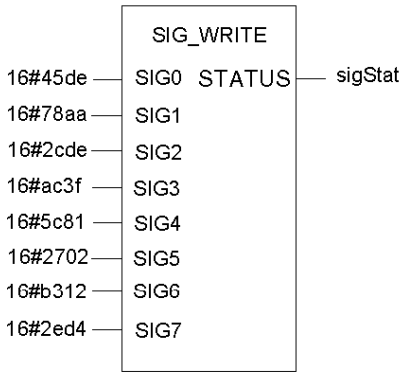
Cette fonction EF pouvant prendre plusieurs dizaines de millisecondes pour effectuer une écriture physique sur la carte mémoire, vérifiez que l'application prend en charge ce temps supplémentaire.

NOTE : %S62 est réglé sur 1 lorsqu'une carte contient une signature, quelle que soit la valeur des 8 mots inscrits.

NOTE : Cette fonction EF ne peut être utilisée que sur une BMX P34 ••••, Version 2.2 ou ultérieure.

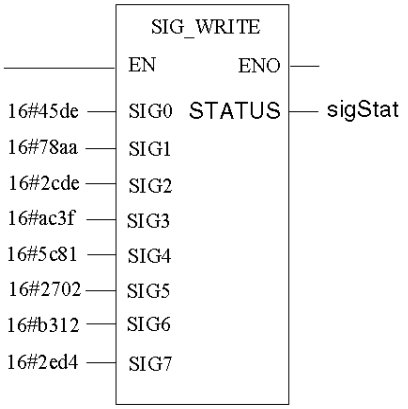
Représentation en FBD

La représentation en FBD de la fonction SIG_WRITE (avec des données exemples) est la suivante :



Représentation en Ladder

La représentation en Ladder de la fonction `SIG_WRITE` (avec des données exemples) est la suivante :



Représentation en IL

La représentation en IL de la fonction `SIG_WRITE` (avec des données exemples) est la suivante :

```
SIG_WRITE (
 SIG0 := 16#45de,
 SIG1 := 16#78aa,
 SIG2 := 16#2cde,
 SIG3 := 16#ac3f,
 SIG4 := 16#5c81,
 SIG5 := 16#2702,
 SIG6 := 16#b312,
 SIG7 := 16#2ed4,
 )
ST sigStat
```

Représentation en ST

La représentation en ST de la fonction `SIG_WRITE` est la suivante :

```
STATUS:=SIG_WRITE(SIG0, SIG1, SIG2, SIG3, SIG4, SIG5, SIG6, SIG7);
```

Description des paramètres

Le tableau ci-après décrit les paramètres d'**entrée** de `SIG_WRITE` :

Paramètre	Type	Commentaire
`SIG0..SIG7`	WORD	8 mots contenant la signature à écrire sur la carte mémoire

Le tableau ci-après décrit le paramètre de **sortie** de `SIG_WRITE` :

Paramètre	Type	Commentaire
STATUS	WORD	Résultat de l'opération d'écriture : **1.** Aucune erreur détectée **2.** Erreur détectée (par exemple, aucune carte mémoire ou carte protégée en écriture)

Chapitre 54
SIG_CHECK : vérification d'une signature

SIG_CHECK : vérification de la signature dans la carte mémoire

Description de la fonction

La fonction SIG_CHECK empêche une application de s'exécuter lorsque la carte SD n'a pas la signature attendue.

La signature compte 8 mots (16 octets). Utilisez la fonction SIG_WRITE *(voir page 255)* pour écrire la signature sur une carte SD.

Cette fonction élémentaire vérifie une signature fournie comme paramètre, en la comparant à celle qui est stockée dans la carte mémoire BMX RM ••• insérée dans l'UC.

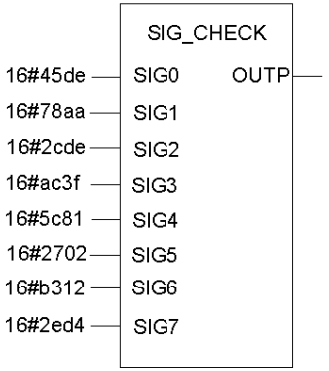
Si la signature dans la carte mémoire est différente, l'UC passe à l'état HALT avec le code d'erreur 0002 dans %SW125.

NOTE : %S62 est mis à 1 lorsqu'une carte contient une signature, quelle que soit la valeur des 8 mots écrits.

NOTE : cette fonction élémentaire n'est utilisable que sur une carte BMX P34 ••••, Version 2.2 minimum.

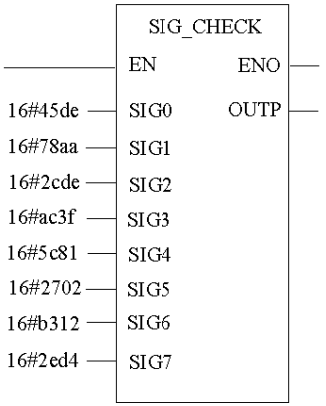
Représentation en FBD

La représentation en FBD de la fonction SIG_CHECK (avec des exemples de données) est la suivante :



Représentation en LD

La représentation en `LD` de la fonction `SIG_CHECK` (avec des exemples de données) est la suivante :



Représentation en IL

La représentation en `IL` de la fonction `SIG_CHECK` (avec des exemples de données) est la suivante :

```
SIG_CHECK (
 SIG0 := 16#45de,
 SIG1 := 16#78aa,
 SIG2 := 16#2cde,
 SIG3 := 16#ac3f,
 SIG4 := 16#5c81,
 SIG5 := 16#2702,
 SIG6 := 16#b312,
 SIG7 := 16#2ed4,
 )
ST %M0
```

Représentation en ST

La représentation en `ST` de la fonction `SIG_CHECK` est la suivante : `SIG_CHECK(SIG0, SIG1, SIG2, SIG3, SIG4, SIG5, SIG6, SIG7);`

Description des paramètres

Le tableau suivant décrit les paramètres d'**entrée** de la fonction `SIG_CHECK` :

Paramètre	Type	Commentaire
`SIG0 à SIG7`	WORD	Mots contenant la signature à vérifier

Le tableau suivant décrit les paramètres de `sortie` de la fonction **SIG_CHECK** :

Paramètre	Type	Commentaire
OUTP	BOOL	`TRUE` si la signature de la carte SD est égale aux paramètres d'entrée de la fonction, et `FALSE` dans le cas contraire.
ENO	BOOL	Toujours `TRUE`

Annexes

Annexe A
Valeurs et codes d'erreur des EFB

Introduction

Les tableaux suivants répertorient les valeurs et les codes d'erreur créés pour les EFB de la bibliothèque.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Erreurs courantes relatives aux valeurs à virgule flottante

Introduction

Le tableau suivant répertorie les codes d'erreur et les valeurs générés par des erreurs relatives aux valeurs à virgule flottante. Ces informations s'affichent dans la fenêtre Visualisation du diagnostic, tandis que les valeurs de code d'erreur sont écrites dans %SW125 *(voir EcoStruxure™ Control Expert, System Bits and Words, Reference Manual)*.

Erreurs courantes relatives aux valeurs à virgule flottante

Tableau des erreurs courantes relatives aux valeurs à virgule flottante

Codes d'erreur	Valeur d'erreur (format décimal)	Valeur d'erreur (format hexadécimal)	Description de l'erreur
FP_ERROR	-30150	16#8A3A	Valeur de base (n'apparaît pas comme une valeur d'erreur)
E_FP_STATUS_FAILED_IE	-30151	16#8A39	Opération sur valeur à virgule flottante interdite
E_FP_STATUS_FAILED_DE	-30152	16#8A38	L'opérande n'est pas un nombre de type REAL valide
E_FP_STATUS_FAILED_ZE	-30154	16#8A36	Division par zéro interdite
E_FP_STATUS_FAILED_ZE_IE	-30155	16#8A35	Opération sur valeur à virgule flottante/Division par zéro interdite
E_FP_STATUS_FAILED_OE	-30158	16#8A32	Dépassement sur valeur à virgule flottante
E_FP_STATUS_FAILED_OE_IE	-30159	16#8A31	Opération sur valeur à virgule flottante/Dépassement interdit
E_FP_STATUS_FAILED_OE_ZE	-30162	16#8A2E	Dépassement sur valeur à virgule flottante/Division par zéro
E_FP_STATUS_FAILED_OE_ZE_IE	-30163	16#8A2D	Opération sur valeur à virgule flottante/Dépassement/Division par zéro interdit
E_FP_NOT_COMPARABLE	-30166	16#8A2A	Erreur interne

Codes d'erreur des EFB avec le paramètre STATUS

Forme du code d'erreur de fonction

Les codes d'erreur des paramètres `STATUS` se présentent sous la forme **Mmss**, où :

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

Codes d'erreur courants

Description des codes d'erreur hexadécimaux :

Code d'erreur hexadécimal	Description
1001	Abandon par l'utilisateur.
1002	Abandon consécutif à un démarrage à chaud.
2001	Un type d'opération non pris en charge a été spécifié dans le bloc de commande.
2002	Un ou plusieurs paramètres du bloc de contrôle ont été modifiés pendant que l'élément `MSTR` était actif (cela ne s'applique qu'aux opérations qui nécessitent plusieurs cycles d'exécution). Les paramètres du bloc de contrôle ne peuvent être modifiés que dans les composants `MSTR` inactifs.
2003	Valeur incorrecte dans le champ de longueur du bloc de commande.
2004	Valeur incorrecte dans le champ d'offset du bloc de commande.
2005	Valeur incorrecte dans les champs de longueur et d'offset du bloc de commande.
2006	Champ de données non autorisé sur l'esclave.
2007	Champ de réseau non autorisé sur l'esclave.
2008	Chemin de routage réseau non autorisé sur l'esclave.
2009	Chemins de routage équivalent à leur propre adresse.
200A	Tentative d'obtention de plus de mots Global Data que ceux qui sont disponibles.
200B	Conflit de diffusion d'E/S sur écriture/lecture de données globales.
200C	Motif incorrect de la requête de changement d'adresse.
200D	Adresse incorrecte de la requête de changement d'adresse.
200E	Le bloc de contrôle ou le tampon de données n'est pas affecté, ou des éléments du bloc de contrôle ou du tampon de données sont hors de la plage %MW (4x).
200F	Espace de réponse trop petit dans le tampon de données.
2010	Longueur du tampon de commande incorrecte.
2011	Paramètre incorrect.
2012	Erreur de syntaxe dans la chaîne « rack.emplacement.voie ».
2013	Module manquant, non détecté ou non configuré.

Code d'erreur hexadécimal	Description
2015	Aucune donnée sur la voie (voie hors limites).
2016	Annulation en cas de timeout.
2017	Contexte de tâche incorrecte.
2018	Erreur de service du système de sécurité Ethernet.
2019	Données de réponse incorrectes (les données reçues ne correspondent pas à la réponse attendue).
201A	Somme de contrôle incorrecte de la réponse.
201B	Problème de compatibilité (par exemple, version EF ou DDT incompatible avec la version du micrologiciel).
30ss	Réponse exceptionnelle de l'esclave Modbus avec code d'exception **ss** *(voir page 269)* spécifique.
31ss	Réponse exceptionnelle de l'esclave Modbus à une erreur de protocole Unity avec code d'erreur **ss** *(voir page 270)* spécifique.
32ss	Acquittement exceptionnel par l'esclave Modbus d'une erreur de requête d'E/S du protocole Unity avec code d'erreur **ss** *(voir page 271)* spécifique.
33ss	Rapport UNI-TE.
34ss	Rapport de communication générique *(voir page 272)* (correspond au champ Rapport de communication des paramètres de gestion des EF Premium/M340).
35ss	Rapport d'opération générique en cas d'échange correct *(voir page 272)* (correspond au champ Rapport d'opération des paramètres de gestion des EF Premium/M340 lorsque Rapport de communication = 16#00).
36ss	Rapport d'opération générique en cas de message refusé *(voir page 272)* (correspond au champ Rapport d'opération des paramètres de gestion des EF Premium/M340 lorsque Rapport de communication = 16#FF).
37ss	Code d'état général CIP. *(voir page 274)*
4001	Réponse incohérente de l'esclave Modbus.
4002	Réponse Modbus Umas incohérente.
4003	Réponse UNI-TE incohérente (dépend du module).
4004	Requête de lecture des mots d'état refusée par la voie du module.
4005	Paramètres de commande refusés par la voie du module.
4006	Paramètres de réglage refusés par le module.
4007	Code d'abandon SDO (4 octets) pouvant figurer dans le champ de données si celui-ci est disponible.
5mss	Codes d'erreur Ethernet TCP/IP *(voir page 277)*.
6mss	Erreur de chemin de routage Modbus Plus *(voir page 281)*. Le sous-champ m indique l'emplacement de l'erreur (0 pour le nœud local, 2 pour le deuxième équipement du chemin, etc.).

Code d'erreur hexadécimal	Description
7mss	Codes d'erreur SY/MAX *(voir page 282)*.
8mss	Codes d'erreur détectée EtherNet/IP *(voir page 284)*.
F001	Nœud cible erroné indiqué pour l'opération MSTR. Option S985 référencée absente ou en mode de réinitialisation.
F002	Composant partiellement initialisé.

Codes de fonction d'exception Modbus (30ss)

Ce tableau indique la valeur hexadécimale ss dans les codes d'erreur 30ss :

Code d'erreur hexadécimal	Description
3001	L'esclave ne prend pas en charge l'opération demandée.
3002	Les registres d'esclave demandés n'existent pas.
3003	Une valeur de données non autorisée a été demandée.
3004	Erreur irrécupérable détectée dans l'esclave.
3005	L'esclave a accepté une commande de programme longue.
3006	La fonction ne peut pas être exécutée actuellement : une commande longue est en cours d'exécution.
3007	L'esclave a rejeté une commande de programme longue.
300A	Passerelle incapable d'allouer un chemin de communication interne.
300B	Aucune réponse de l'équipement cible.
30FF	Exception Modbus étendue. Données supplémentaires disponibles dans le champ de données (s'il est fourni) : ● Longueur de l'exception : représente la longueur de la réponse d'exception étendue, à l'exception de ces 2 octets. ● Données de l'exception : informations sur l'erreur correspondant au code fonction concerné.

La valeur ss correspond au code d'exception Modbus renvoyé par l'équipement esclave Modbus en cas d'erreur (deuxième octet du PDU d'exception Modbus) :

- code_exception-fonction = code fonction de la requête + 0x80 : 1 octet
- code_exception : 1 octet (renvoyé sous la forme ss dans le code d'erreur 16#30ss)

Détail des codes d'erreur STATUS 31ss à 37ss

Codes d'erreur propres au protocole Unity (31ss)

Ce tableau indique la valeur hexadécimale ss dans les codes d'erreur 31ss :

Code d'erreur hexadécimal	Description
3100	Erreur générique de protocole Unity
3180	Erreur de communication générique.
3181	Automate réservé par quelqu'un d'autre.
3182	Vous devez réserver l'automate.
3183	Requête ou sous-code inconnu(e).
3184	Objet inconnu (par exemple : %Z non implémenté).
3185	Génération de la réponse impossible
3186	La requête comprend des paramètres non valides (par exemple : mal structurée, trop de paramètres ou commande Csa erronée).
3187	Séquence incorrecte (par exemple, EndDownload avant BeginDownload).
3188	Taille de la réponse supérieure à celle du tampon disponible.
3189	Module non configuré (adresse potentiellement incorrecte).
318 A	Action non autorisée sur cet objet.
318B	Etat occupé : l'opération précédente est toujours en cours, toutes les ressources internes sont occupées pour la requête d'E/S ou le chargement en parallèle est trop lourd, etc.
3190	Erreur générique : une erreur s'est produite dans l'application.
3191	Violation d'accès : écriture dans un bloc ou une variable en lecture seule, tentative de téléchargement alors que la mémoire est protégée, etc.
3192	Objet inaccessible car en cours d'utilisation.
3193	Dépassement des limites : hors de la plage %MW, trop de points d'arrêt, pile d'appels trop importante, etc.
3194	Longueur incorrecte.
3195	Référence à une ressource ou une tâche inexistante, adresse de variable absente de la zone de données du DFB, etc.
3196	Objet ou ressource déjà défini(e). Par exemple : tentative de démarrage d'un élément déjà démarré, ID de point d'arrêt déjà utilisé, etc.
3197	Données incohérentes ou dans un état non autorisé. Par exemple : données incorrectes ou valeur erronée lors de l'écriture d'un objet.
3198	Objet existant, mais non initialisé.
3199	Voie hors limites dans une requête d'E/S.
319 A	Requête non encore implémentée.

Code d'erreur hexadécimal	Description
31A0	Application incompatible, cible ou plate-forme incorrecte.
31A1	Echec de la vérification de signature
31A2	Configuration de la mémoire PCMCIA incorrecte.
31B0	Automate dans un mode incorrect : téléchargement avec automate en mode RUN ou débogage avec automate en mode NOCONF, tentative de contournement d'une tâche, absence de point d'arrêt, chargement annulé par un téléchargement ou une modification en ligne, etc.
31B1	Impossible de modifier le mode : une E/S force l'automate à s'arrêter.
31B2	Timeout interne.
31B3	Délai du chien de garde écoulé.
31FF	Erreur générale non définie

Erreurs de requête d'E/S pour le protocole Unity (32ss)

Ce tableau indique la valeur hexadécimale ss dans les codes d'erreur 32ss :

Code d'erreur hexadécimal	Description
3202	Erreur lors de l'échange.
3207	Autre échange explicite en cours.
3209	Opération impossible.
320A	Données refusées par le bloc d'E/S.
320B	Ecriture non autorisée.
320C	Nombre maximum d'échanges.
3284	Objet inconnu.
3286	Tampon de lecture non valide.
328A	Action inconnue ou non valide.
328B	Tous les tampons sont utilisés.
3293	Objet hors limites.
3297	Valeur d'objet interdite (opérations d'écriture uniquement).
3299	Voie hors plage.

Rapport de communication générique (34ss)

Ce tableau indique la valeur hexadécimale ss dans les codes d'erreur 34ss :

Code d'erreur hexadécimal	Description
3401	Echange interrompu suite à un timeout
3402	Echange arrêté à la demande de l'utilisateur (CANCEL).
3403	Format d'adresse incorrect.
3404	Adresse cible incorrecte
3405	Format du paramètre de gestion incorrect.
3406	Paramètres spécifiques incorrects.
3407	Erreur détectée lors de l'envoi vers la destination.
3409	Réservé.
340 A	Taille du tampon de réception insuffisante.
340B	Processeur sans ressources système.
340C	Numéro d'échange incorrect.
340D	Télégramme non reçu.
340E	Longueur incorrecte.
340F	Service de télégramme non configuré.
3410	Module réseau manquant.
3411	Requête manquante.
3412	Serveur d'application déjà actif.
3413	Numéro de transaction UNI-TE V2 incorrect.

La valeur ss correspond au code de rapport de communication *(voir Unity Pro, Communication, Bibliothèque de blocs)* renvoyé par les fonctions élémentaires de communication sur les plates-formes Premium/Atrium/Mxxx.

Rapport d'opération générique (35ss et 36ss)

Cet octet de rapport est propre à chaque fonction et indique le résultat de l'opération sur l'application distante.

Rapport d'opération générique lorsque l'échange correct est codé sous la forme 16#35ss. Si l'équipement distant refuse le message, le rapport a la forme 16#36ss.

Ce tableau indique la valeur hexadécimale ss dans les codes d'erreur 35ss :

Code d'erreur hexadécimal	Description
3501	Requête non traitée.
3502	Réponse incorrecte.

La valeur ss dans les codes 35ss correspond au champ du rapport d'opération des paramètres de gestion des fonctions élémentaires Premium/M340 lorsque Rapport de communication *(voir Unity Pro, Communication, Bibliothèque de blocs)* = 16#00.

Ce tableau indique la valeur hexadécimale ss dans les codes d'erreur 36ss :

Code d'erreur hexadécimal	Description
3601	Pas de ressources vers le processeur.
3602	Pas de ressources de ligne.
3603	Aucun équipement ou équipement sans ressource.
3604	Erreur de ligne.
3605	Erreur de longueur.
3606	Voie de communication défectueuse.
3607	Erreur d'adressage.
3608	Erreur d'application.
360B	Absence de ressource système.
360C	Fonction de communication non active.
360D	Destinataire absent.
360F	Problème de routage intrastation ou voie non configurée.
3611	Format d'adresse non pris en charge.
3612	Aucune ressource cible.
3614	Connexion non opérationnelle (exemple : TCP/IP Ethernet).
3615	Aucune ressource sur la voie locale.
3616	Accès non autorisé (exemple : TCP/IP Ethernet).
3617	Configuration incohérente du réseau (exemple : TCP/IP Ethernet).
3618	Connexion temporairement indisponible.
3621	Serveur d'application arrêté.
3630	Erreur d'émission.

La valeur ss dans les codes 36ss correspond au champ du rapport d'opération des paramètres de gestion des fonctions élémentaires Premium/M340 lorsque Rapport de communication *(voir Unity Pro, Communication, Bibliothèque de blocs)* = 16#FF.

Code d'état général CIP (37ss)

Le tableau ci-dessous répertorie les codes d'état que vous pouvez rencontrer dans le champ de code d'état général d'un message de réponse à une erreur CIP détectée :

Code d'état général (hexadécimal)	Nom de l'état	Description de l'état
3701	Echec de la connexion	Echec d'un service lié à la connexion dans le chemin de connexion.
3702	Ressource indisponible	Les ressources nécessaires pour que l'objet exécute le service demandé n'étaient pas disponibles.
3703	Valeur de paramètre incorrecte	Reportez-vous au code d'état 0x20, la valeur à utiliser dans ce cas de figure.
3704	Erreur de segment de chemin	Le nœud de traitement n'a pas compris l'identifiant du segment de chemin ou la syntaxe du segment. Le traitement du chemin est interrompu lorsqu'une erreur de segment de chemin est détectée.
3705	Destination du chemin inconnue	Le chemin fait référence à une classe d'objets, une instance ou un élément de structure inconnu ou absent du nœud de traitement. Le traitement du chemin est interrompu lorsqu'une erreur de destination de chemin inconnue est détectée.
3706	Transfert partiel	Seule une partie des données attendues a été transférée.
3707	Connexion perdue	La connexion de messagerie a été perdue.
3708	Service non pris en charge	Le service demandé n'a pas été mis en œuvre ou défini pour cette instance/classe d'objets.
3709	Valeur d'attribut incorrecte	Attribut incorrect détecté.
370A	Erreur de liste d'attributs	Un attribut dans la réponse `Get_Attribute_List` ou `Set_Attribute_List` a un état non nul.
370B	Déjà en mode/état demandé	L'objet est déjà dans le mode/l'état demandé par le service.
370C	Conflit d'état d'objet	L'objet ne peut pas exécuter le service demandé dans son mode/état actuel.
370D	Objet déjà existant	L'instance demandée de l'objet à créer existe déjà.
370E	Attribut non configurable	Une requête de modification d'un attribut non modifiable a été reçue.
370F	Violation de privilège	Une vérification de droit d'accès/privilège a échoué.
3710	Conflit d'état d'équipement	Le mode/l'état de l'équipement interdit l'exécution du service demandé.
3711	Données de la réponse trop volumineuses	Les données à transmettre dans le tampon de réponse sont trop volumineuses pour la taille allouée au tampon.

Code d'état général (hexadécimal)	Nom de l'état	Description de l'état
3712	Fragmentation d'une valeur primitive	Le service a spécifié une opération qui va fragmenter une valeur de données primitive (soit la moitié d'un type de données REAL).
3713	Données insuffisantes	Le service n'a pas fourni suffisamment de données pour effectuer l'opération spécifiée.
3714	Attribut non pris en charge	L'attribut spécifié dans la requête n'est pas pris en charge.
3715	Trop de données	Le service a fourni plus de données que prévu.
3716	Objet inexistant	L'objet spécifié n'existe pas dans l'équipement.
3717	Séquence de fragmentation du service inactive	La séquence de fragmentation de ce service est désactivée pour ces données.
3718	Attributs non stockés	Les attributs de cet objet n'ont pas été enregistrés avant le service demandé.
3719	Echec de l'opération de stockage	Suite à une tentative infructueuse, les attributs de cet objet n'ont pas été enregistrés.
371A	Echec du routage, paquet de requête trop volumineux	La requête du service était trop volumineuse pour être transmise sur un réseau à l'emplacement cible. L'équipement de routage a dû annuler l'exécution du service.
371B	Echec du routage, paquet de réponse trop volumineux	Le paquet de réponse du service était trop volumineux pour être transmis sur un réseau à l'emplacement cible. L'équipement de routage a dû annuler l'exécution du service.
371C	Liste d'attributs manquante	La liste d'attributs fournie par le service ne contenait pas un attribut requis par ce même service pour effectuer l'opération demandée.
371D	Liste de valeurs d'attribut incorrecte	Le service renvoie la liste d'attributs contenant des informations d'état qui sont incorrectes pour ces attributs.
371E	Erreur de service intégré	Un service intégré a généré une erreur détectée.
371F	Erreur propre à un fournisseur	Une erreur propre à un fournisseur a été détectée. Le champ de code supplémentaire de la réponse définit l'erreur rencontrée. Utilisez ce code d'erreur général quand aucun de ceux figurant dans ce tableau ou dans une définition de classe d'objets ne correspond à l'erreur détectée.
3720	Paramètre incorrect	Un paramètre associé à la requête était incorrect. Ce code est utilisé lorsqu'un paramètre ne répond pas aux critères de cette spécification et/ou aux critères définis dans une spécification d'objet d'application.
3721	Valeur à écriture unique ou support déjà gravé	Le système a détecté une tentative d'écriture sur un support non réinscriptible (par exemple, disque WORM, PROM) déjà gravé ou une tentative de modification d'une valeur non modifiable.

Code d'état général (hexadécimal)	Nom de l'état	Description de l'état
3722	Réponse incorrecte reçue	Une réponse incorrecte est reçue (par exemple, le code du service de réponse ne correspond pas au code du service de requête ou le message de réponse est plus court que la taille minimale attendue). Ce code d'état peut être utilisé pour d'autres causes de réponse incorrecte.
3723	Saturation du tampon	Le message reçu dépasse la capacité du tampon de réception. Le message a été entièrement rejeté.
3724	Erreur de format du message	Le serveur ne prend pas en charge le format du message reçu.
3725	Clé défectueuse dans le chemin	Le segment de clé défini comme premier segment du chemin ne correspond pas au module cible. L'état de l'objet indique la partie défectueuse du contrôle de la clé.
3726	Taille de chemin incorrecte	La taille du chemin envoyé avec la requête de service est trop petite pour acheminer la requête à un objet ou comprenait trop de données de routage.
3727	Attribut inattendu dans la liste	La tentative de configuration concernait un attribut non modifiable pour l'instant.
3728	ID de membre incorrect	L'ID de membre spécifié dans la requête n'existe pas dans la classe, l'instance ou l'attribut spécifié.
3729	Membre non configurable	Une requête de modification d'un membre non modifiable a été reçue.
372A	Serveur de groupe 2 uniquement – Erreur générale	Ce code d'erreur détectée n'est signalé que par des serveurs DeviceNet de groupe 2 dotés d'au maximum 4 Ko d'espace de code, et uniquement à la place d'un service non pris en charge ou d'un attribut non pris en charge ou non configurable.
372B	Erreur Modbus inconnue	Un convertisseur CIP/Modbus a reçu un code d'exception Modbus.
372C	Attribut inaccessible	Une requête de lecture d'un attribut non lisible a été reçue.
372D à 37CF	–	Réservé par CIP pour les futures extensions.
37D0 à 37FF	Réservé pour les erreurs de classe d'objets et de service	Cette plage de codes d'erreur détectée permet d'indiquer des erreurs détectées correspondant à des classes d'objets. Ne l'utilisez que si aucun des codes d'erreur figurant dans ce tableau ne correspond exactement à l'erreur détectée.

NOTE : Extrait autorisé de *The CIP Networks Library, Volume 1*, Common Industrial Protocol (CIP™), Edition 3.6, avril 2009.

Détails des codes d'erreur Ethernet TCP/IP des EFB5mss

Codes d'erreur de réseau Ethernet TCP/IP (5mss)

NOTE : Les erreurs Ethernet sont gérées par les modules Ethernet ou le coprocesseur Ethernet, à l'exception du code d'erreur 5050 (hex).

Code d'erreur hexadécimal	Signification
5001	Réponse incohérente du réseau
5004	Appel système interrompu
5005	Erreur d'E/S
5006	Adresse inexistante
5009	Descripteur de socket incorrect
500C	Mémoire insuffisante
500D	Autorisation refusée
5011	Entrée existante
5016	Argument incorrect
5017	Espace insuffisant dans la table interne
5020	Connexion perdue
5023	Opération bloquée et socket non bloquant
5024	Socket non bloquant et impossible de fermer la connexion
5025	Socket non bloquant et échec d'une précédente tentative de connexion
5026	Opération socket sur un non-socket
5027	Adresse cible non valide
5028	Message trop long
5029	Type de protocole incorrect pour le socket
502A	Protocole non disponible
502B	Protocole non pris en charge
502C	Type de socket non pris en charge
502D	Opération non prise en charge sur un socket
502E	Famille de protocoles non prise en charge
502F	Famille d'adresses non prise en charge
5030	Adresse déjà utilisée
5031	Adresse non disponible
5032	Réseau hors service
5033	Réseau inaccessible

Code d'erreur hexadécimal	Signification
5034	Connexion réseau perdue lors de la réinitialisation
5035	Connexion abandonnée par l'homologue
5036	Connexion réinitialisée par l'homologue
5037	Mémoire tampon interne requise, mais impossible à affecter
5038	Socket déjà connecté
5039	Socket non connecté
503A	Emission impossible après l'arrêt du socket
503B	Trop de références : liaison impossible
503C	Expiration de la connexion (voir remarque ci-dessous)
503D	Connexion refusée
5040	Hôte hors service
5041	Hôte cible inaccessible depuis ce nœud
5042	Répertoire non vide
5046	« -1 » renvoyé par NI_INIT
5047	MTU non valide
5048	Longueur matérielle non valide
5049	Chemin indiqué introuvable
504A	Collision dans l'appel de sélection : ces conditions ont déjà été sélectionnées par une autre tâche
504B	ID de tâche incorrect
5050	Aucune ressource réseau
5051	Erreur de longueur
5052	Erreur d'adressage
5053	Erreur d'application
5054	Client incapable de traiter la requête
5055	Aucune ressource réseau
5056	Connexion TCP non opérationnelle
5057	Configuration incohérente
51ss	Codes d'erreur du service SMTP *(voir page 279)*
53ss	Codes d'erreur du service client Modbus *(voir page 280)*

NOTE :

- Code d'erreur 5055 (hex) pouvant survenir avant un code d'erreur 503C (hex).
- Aucun équipement distant n'a priorité sur un timeout.

Codes d'erreur du service SMTP (51ss)

Codes d'erreur hexadécimaux du service SMTP :

Code d'erreur hexadécimal	Signification
5100	Erreur interne
5101	Composant SMTP non opérationnel
5102	En-tête de message non configuré
5103	Valeur non valide dans l'en-tête de message
5104	Connexion au serveur SMTP impossible
5105	Erreur lors de la transmission du corps du message électronique au serveur SMTP
5106	Erreur lors de la fermeture de la connexion SMTP au serveur
5107	Echec de la requête SMTP **HELO**
5108	Echec de la requête SMTP **MAIL**. Authentification potentiellement requise par le serveur SMTP
5109	Echec de la requête SMTP **RCPT**
510A	Aucun destinataire accepté par le serveur SMTP
510B	Echec de la requête SMTP **DATA**
510C	Longueur incorrecte de la requête d'envoi de message électronique
510D	Echec d'authentification
510E	Réception d'une requête de réinitialisation de composant pendant une connexion ouverte

Codes d'erreur du service client Modbus (53ss)

Codes d'erreur hexadécimaux du service client Modbus :

Code d'erreur hexadécimal	Signification
5300	Inutilisé (réservé pour un usage ultérieur)
5301	Aucune ressource disponible pour le composant
5302	Adresse IP fournie inappropriée Par exemple : 0.0.0.0, adresse de diffusion, adresse de multidiffusion, etc.
5303	Délai de la transaction expiré. Requête acceptée par le serveur distant, mais aucune réponse fournie dans les 2 minutes
5304	Connexions actuellement toutes utilisées
5305	Accès refusé
5306	Réseau inaccessible
5307	Hôte arrêté
5308	Connexion réseau perdue lors de la réinitialisation
5309	Réseau arrêté
530A	Connexion refusée
530B	Connexion expirée
530C	En-tête MBAP erroné

Détails des codes d'erreur Modbus Plus des EFB 6mss

Codes d'erreur propres à Modbus Plus (6mss)

NOTE : le champ **m** du code d'erreur **6mss** est un `index` (dans les informations de routage) qui précise l'endroit où une erreur a été détectée. **m** = 0 signifie que l'erreur a été détectée sur le nœud local, **m** = 2 signifie que l'erreur a été détectée sur le deuxième équipement du chemin, etc.

Code d'erreur hexadécimal	Description
6m01	Pas de réception de réponse.
6m02	Accès au programme refusé.
6m03	Nœud hors service et incapable de communiquer.
6m04	Réponse reçue inhabituelle.
6m05	Chemin de données du nœud du routeur occupé.
6m06	Esclave hors service.
6m07	Adresse cible incorrecte.
6m08	Type de nœud non autorisé dans le chemin de routage.
6m10	L'esclave a rejeté la commande.
6m20	L'esclave a perdu une transaction active.
6m40	Chemin de sortie maître non attendu reçu.
6m80	Réponse reçue non attendue.
F001	Nœud cible erroné indiqué pour l'opération `MSTR`.

Codes d'erreur SY/MAX dans les EFB Quantum

Codes d'erreur propres à SY/MAX

Si vous utilisez Ethernet SY/MAX, trois types d'erreur supplémentaires peuvent apparaître dans le registre `CONTROL[1]` du bloc de commande.

Ces codes d'erreur ont la signification suivante :

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

Codes d'erreur hexadécimaux propres à SY/MAX

Les codes d'erreur hexadécimaux spécifiques à SY/MAX sont décrits ci-après :

Code d'erreur hexadécimal	Description
7101	Code opérande non valide détecté par l'équipement distant SY/MAX
7103	Adresse non valide détectée par l'équipement distant SY/MAX
7109	Essai d'écriture d'un registre protégé en écriture détecté par l'équipement distant SY/MAX
F710	Débordement récepteur détecté par l'équipement distant SY/MAX
7110	Longueur non valide détectée par l'équipement distant SY/MAX
7111	Equipement distant non actif, pas de liaison (se produit lorsque toutes les tentatives et temporisations ont été épuisées), détecté par l'équipement distant SY/MAX
7113	Paramètre non valide détecté par l'équipement distant SY/MAX dans une opération de lecture
711D	Itinéraire non valide détecté par l'équipement distant SY/MAX
7149	Paramètre non valide détecté par l'équipement distant SY/MAX dans une opération d'écriture
714B	Numéro de station non valide détecté par l'équipement distant SY/MAX
7101	Code opérande non valide détecté par le serveur SY/MAX
7203	Adresse non valide détectée par le serveur SY/MAX
7209	Essai d'écriture dans un registre protégé en écriture détecté par le serveur SY/MAX
F720	Débordement récepteur détecté par le serveur SY/MAX
7210	Longueur non valide détectée par le serveur SY/MAX
7211	Equipement distant non actif, pas de liaison (se produit lorsque toutes les tentatives et temporisations ont été épuisées), détecté par le serveur SY/MAX

Code d'erreur hexadécimal	Description
7213	Paramètre non valide détecté par le serveur SY/MAX dans une opération de lecture
721D	Itinéraire non valide détecté par le serveur SY/MAX
7249	Paramètre non valide détecté par le serveur SY/MAX dans une opération d'écriture
724B	Numéro de station non valide détecté par le serveur SY/MAX
7301	Code opérande non valide dans une requête de bloc `MSTR` en provenance du compilateur Quantum
7303	Etat du module QSE Lecture/Ecriture (adresse de routage 200 hors limites)
7309	Essai d'écriture dans un registre protégé en écriture, lorsqu'une écriture d'état est en cours d'exécution (Routage 200)
731D	Itinéraire non valide détecté par le compilateur Quantum. Itinéraires valides : ● dest_drop, 0xFF ● 200, dest_drop, 0xFF ● 100+drop, dest_drop, 0xFF ● Toutes les autres valeurs de routage entraînent une erreur.
734B	L'une des erreurs suivantes est survenue : ● Absence de configuration de table CTE. ● Aucune entrée de table CTE n'a été créée pour le numéro d'emplacement du module QSE. ● Aucune station valide n'a été précisée. ● Le module QSE n'a pas été réinitialisé après la création de la table CTE. Remarque : après écriture et configuration de la table CTE et son chargement dans le module QSE, vous devez réinitialiser le module QSE pour que les modifications prennent effet. ● Lors de l'utilisation d'une instruction `MSTR`, aucun emplacement ni station valide n'a été indiqué(e).

Codes d'erreur détectée EtherNet/IP

Codes d'erreur détectée EtherNet/IP

Les codes hexadécimaux d'erreur détectée EtherNet/IP sont les suivants :

Code d'erreur détectée (hex)	Description
800D	Timeout sur la requête de message explicite
8012	Equipement incorrect
8015	Soit : ● pas de ressources pour traiter le message, ou ● Evénement interne : pas de tampon disponible, pas de liaison disponible, envoi à la tâche TCP impossible.
8018	Soit : ● un autre message explicite est en cours pour cet équipement, ou ● une session de connexion ou d'encapsulation TCP est en cours.
8030	Timeout sur la requête Forward_Open
Remarque : les événements 81xx ci-après sont des codes d'erreur détectée de réponse Forward_Open, générés sur la cible distante et reçus par le biais de la connexion CIP.	
8100	Connexion utilisée ou Forward_Open en double
8103	Classe de transport et déclenchement de combinaison non pris en charge
8106	Conflit de propriété
8107	Connexion cible introuvable
8108	Paramètre de connexion réseau incorrect
8109	Taille de connexion incorrecte
8110	Cible de connexion non configurée
8111	Intervalle de trame demandé (RPI) non pris en charge
8113	Hors connexion
8114	ID du vendeur ou code produit différent
8115	Type de produit non concordant
8116	Révision non concordante
8117	Chemin d'application créé ou utilisé incorrect
8118	Chemin d'application de configuration incorrect ou incohérent
8119	Connexion Non-Listen Only non ouverte
811A	Objet cible hors connexion
811B	Intervalle de trame demandé (RPI) plus petit que la durée d'inhibition de production
8123	Connexion expirée
8124	Expiration de la requête non connectée

Code d'erreur détectée (hex)	Description
8125	Erreur détectée de paramètre dans une requête et un service non connectés
8126	Message trop grand pour le service unconnected_send
8127	Acquittement non connecté sans réponse
8131	Pas de mémoire tampon disponible
8132	Bande passante réseau non disponible pour les données
8133	Aucun filtre d'ID de connexion consommée disponible
8134	Non configuré pour l'envoi de données prioritaires programmées
8135	Signature de programmation non concordante
8136	Validation de la signature de programmation impossible
8141	Port non disponible
8142	Adresse de liaison non valide
8145	Segment invalide dans le chemin de connexion
8146	Erreur détectée dans le chemin de connexion du service Forward_Close
8147	Planification non spécifiée
8148	Adresse de liaison vers soi-même non valide
8149	Ressources secondaires non disponibles
814A	Connexion de rack déjà établie
814B	Connexion de module déjà établie
814C	Divers
814D	Connexion redondante différente
814E	Plus aucune ressource consommatrice de liaison configurable par l'utilisateur : le nombre configuré de ressources pour une application productrice a atteint la limite
814F	Plus aucune ressource consommatrice de liaison configurable par l'utilisateur : aucun consommateur configuré utilisable par une application productrice
8160	Propre au fournisseur
8170	Aucune donnée d'application cible disponible
8171	Aucune donnée d'application source disponible
8173	Non configuré pour la multidiffusion hors du sous-réseau
81A0	Erreur détectée dans l'affectation des données
81B0	Erreur détectée d'état d'objet facultatif
81C0	Erreur détectée d'état d'équipement facultatif

Code d'erreur détectée (hex)	Description
Remarque : toutes les erreurs détectées #82xx sont des erreurs détectées de réponse de session de registre.	
8200	Ressources insuffisantes de l'équipement cible
8208	En-tête d'encapsulation du message non reconnu par l'équipement cible
820F	Erreur détectée réservée ou inconnue de la cible

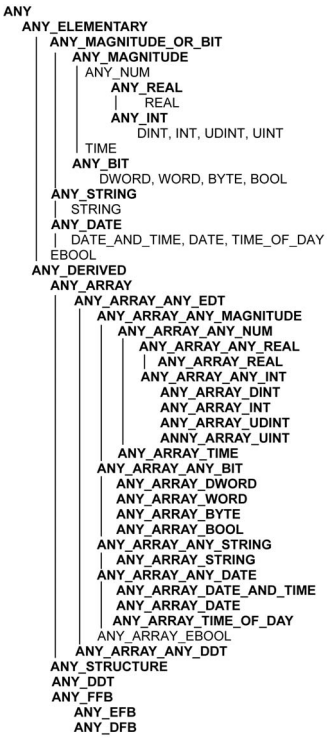
Glossaire

A

ANY

Une hiérarchie existe entre les différents types de données. Dans les DFB, il est parfois possible de déclarer les variables pouvant contenir plusieurs types de valeurs. On utilise alors les types `ANY_xxx`.

La figure suivante décrit cette structure hiérarchisée :



B

BOOL

BOOL est l'abréviation du type booléen. Il s'agit du type de données de base en informatique. Une variable de type BOOL peut avoir l'une des deux valeurs suivantes : 0 (FALSE) ou 1 (TRUE).

Un bit extrait d'un mot est de type BOOL, par exemple :%MW10.4

BYTE

8 bits constituent un octet (BYTE). La saisie d'un BYTE s'effectue soit en mode binaire, soit en base 8.

Le type BYTE est codé dans un format 8 bits qui, au format hexadécimal, s'étend de 16#00 à 16#FF.

D

DATE_AND_TIME

Voir DT.

DINT

DINT est l'abréviation du format Double INTeger (entier double codé sur 32 bits).

Les limites supérieure/inférieure sont les suivantes : - (2 puissance 31) à (2 puissance 31) - 1.

Exemple :

-2147483648, 2147483647, 16#FFFFFFFF.

DT

DT est l'acronyme de « Date and Time » (date et heure).

Le type DT, codé en BCD dans un format 64 bits, contient les informations suivantes :

- l'année codée dans un champ de 16 bits ;
- le mois codé dans un champ de 8 bits ;
- le jour codé dans un champ de 8 bits ;
- l'heure codée dans un champ de 8 bits ;
- les minutes codées dans un champ de 8 bits ;
- les secondes codées dans un champ de 8 bits.

NOTE : les 8 bits de poids faible ne sont pas utilisés.

Le type DT doit être saisi comme suit :

DT#<Année>**-**<Mois>**-**<Jour>**-**<Heure>**:**<Minutes>**:**<Secondes>

Le tableau ci-après donne les limites inférieure/supérieure de chaque élément :

Champ	Limites	Commentaire
Année	[1990,2099]	Année

Champ	Limites	Commentaire
Mois	[01,12]	Le 0 initial est toujours affiché ; il peut être omis lors de la saisie.
Jour	[01,31]	Pour les mois 01/03/05/07/08/10/12
	[01,30]	Pour les mois 04/06/09/11
	[01,29]	Pour le mois 02 (années bissextiles)
	[01,28]	Pour le mois 02 (années non bissextiles)
Heure	[00,23]	Le 0 initial est toujours affiché ; il peut être omis lors de la saisie.
Minute	[00,59]	Le 0 initial est toujours affiché ; il peut être omis lors de la saisie.
Seconde	[00,59]	Le 0 initial est toujours affiché ; il peut être omis lors de la saisie.

E

EBOOL

`EBOOL` est l'acronyme du type Extended BOOLean (booléen étendu). Un type `EBOOL` possède une valeur (0 pour `FALSE` ou 1 pour `TRUE`), mais également des fronts montants ou descendants et des fonctions de forçage.

Une variable `EBOOL` occupe un octet de mémoire.

L'octet contient les informations suivantes :

- un bit pour la valeur ;
- un bit pour l'historique (chaque fois que l'objet change d'état, la valeur est copiée dans ce bit) ;
- un bit pour le forçage (égal à 0 si l'objet n'est pas forcé, égal à 1 s'il est forcé).

La valeur par défaut de chaque bit est 0 (`FALSE`).

EN

`EN` correspond à **EN**able (activer) ; il s'agit d'une entrée de bloc facultative. Quand l'entrée `EN` est activée, une sortie `ENO` est automatiquement définie.

Si `EN` = 0, le bloc n'est pas activé, son programme interne n'est pas exécuté et `ENO` est réglé sur 0.

Si `EN` = 1, le programme interne du bloc est exécuté et `ENO` est réglé sur 1. Si une erreur survient, `ENO` reprend la valeur 0.

Si l'entrée `EN` n'est pas connectée, elle est automatiquement réglée sur 1.

ENO

ENO signifie **E**rror **NO**tification (notification d'erreur). C'est la sortie associée à l'entrée facultative EN.

Si ENO est réglé sur 0 (car EN = 0 ou en cas d'erreur d'exécution) :

- les sorties du bloc fonction restent dans l'état qui était le leur lors du dernier cycle de scrutation exécuté correctement ;
- la ou les sorties de la fonction, ainsi que les procédures, sont réglées sur 0.

F

FBD

FBD est l'acronyme de « Function Block Diagram » (langage en blocs fonction).

FBD est un langage de programmation graphique qui fonctionne comme un logigramme. Par l'ajout de blocs logiques simples (AND, OR, etc.), chaque fonction ou bloc fonction du programme est représenté(e) sous cette forme graphique. Pour chaque bloc, les entrées se situent à gauche et les sorties à droite. Les sorties des blocs peuvent être liées aux entrées d'autres blocs afin de former des expressions complexes.

I

IL

IL est l'acronyme de « Instruction List » (liste d'instructions).

Ce langage est une suite d'instructions simples.

Il est très proche du langage d'assemblage utilisé pour programmer les processeurs.

Chaque instruction est composée d'un code instruction et d'un opérande.

INT

INT est l'abréviation du format single INTeger (entier simple codé sur 16 bits).

Les limites supérieure/inférieure sont les suivantes : - (2 puissance 15) à (2 puissance 15) - 1.

Exemple :

-32768, 32767, 2#1111110001001001, 16#9FA4.

L

LD

LD est l'acronyme de « Ladder Diagram » (langage à contacts).

LD est un langage de programmation représentant les instructions à exécuter sous forme de schémas graphiques très proches d'un schéma électrique (contacts, bobines, etc.).

M

Multijeton

Mode de fonctionnement d'un SFC. En mode multijeton, le SFC peut posséder plusieurs étapes actives simultanément.

S

ST

ST est l'acronyme de « Structured Text » (langage littéral structuré).

Le langage littéral structuré est un langage élaboré proche des langages de programmation informatiques. Il permet de structurer des suites d'instructions.

STRING

Une variable de type `STRING` est une chaîne de caractères ASCII. La longueur maximale d'une chaîne est de 65 534 caractères.

T

TIME

Le type `TIME` exprime une durée en millisecondes. Codé sur 32 bits, ce type permet d'obtenir des durées de 0 à 2^{32}-1 millisecondes.

Le type `TIME` présente les unités suivantes : jours (`d`), heures (`h`), minutes (`m`), secondes (`s`) et millisecondes (`ms`). Une valeur littérale de type `TIME` est représentée par une combinaison des types précédents associés au préfixe `T#`, `t#`, `TIME#` ou `time#`.

Exemples : `T#25h15m`, `t#14,7S`, `TIME#5d10h23m45s3ms`

U

UDINT

`UDINT` est l'acronyme du format « Unsigned Double INTeger » (entier double non signé) (codé sur 32 bits). Les limites inférieure et supérieure sont les suivantes : 0 à (2 puissance 32) - 1.

Exemple :

`0`, `4294967295`, `2#11111111111111111111111111111111`, `8#37777777777`, `16#FFFFFFFF`.

UINT

UINT est l'abréviation du format « Unsigned INTeger » (entier non signé codé sur 16 bits). Les limites inférieure et supérieure sont les suivantes : 0 à (2 puissance 16) - 1.

Exemple :

0, 65535, 2#1111111111111111, 8#177777, 16#FFFF.

W

WORD

Le type WORD est codé dans un format 16 bits et sert à effectuer des traitements sur des chaînes de bits.

Le tableau ci-dessous donne les limites inférieure/supérieure des bases qui peuvent être utilisées :

Base	Limite inférieure	Limite supérieure
Hexadécimale	16#0	16#FFFF
Octale	8#0	8#177777
Binaire	2#0	2#1111111111111111

Exemples de représentation

Données	Représentation dans l'une des bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index

H

I

L

M

O

P

R

S

T

U

V

W