PC 3000

Blocs Fonctions

CE

EUROTHERM
AUTOMATION

HA 174 415 - Indice 2

SOMMAIRE

Chapitre 1

PRESENTATION

Edition 1

Sommaire

INTRODUCTION

Un bloc fonction est un concept formalisé par la norme internationale IEC 1131/3 Langages de programmation pour automates programmables. Un bloc fonction intègre un algorithme que l'utilisateur ou le programmeur peut contrôler à l'aide d'une série de paramètres.

La bibliothèque de blocs fonctions PC3000 offre un ensemble de blocs fonctions qui peuvent être considérés comme des "instruments logiciels" de haut niveau. Ces instruments, semblables à des instruments discrets, peuvent être reliés les uns aux autres par voie logicielle pour créer une stratégie de contrôle pour une application particulière, comme un contrôleur de PID, une horloge, un compteur, etc.

La plupart des blocs fonctions ont des **entrées** qui peuvent être écrites à partir de la station de programmation ou de la station de l'opérateur, à partir du système de supervision ou du PC3000 lui-même par "câblage par soft" ou à partir d'instructions d'action dans un grafcet.

La plupart des blocs fonctions ont des **sorties** qui sont calculées par le bloc fonction.

De nombreux blocs fonctions ont des **Paramètres de configuration** qui sont réglés au démarrage et ne changent plus.

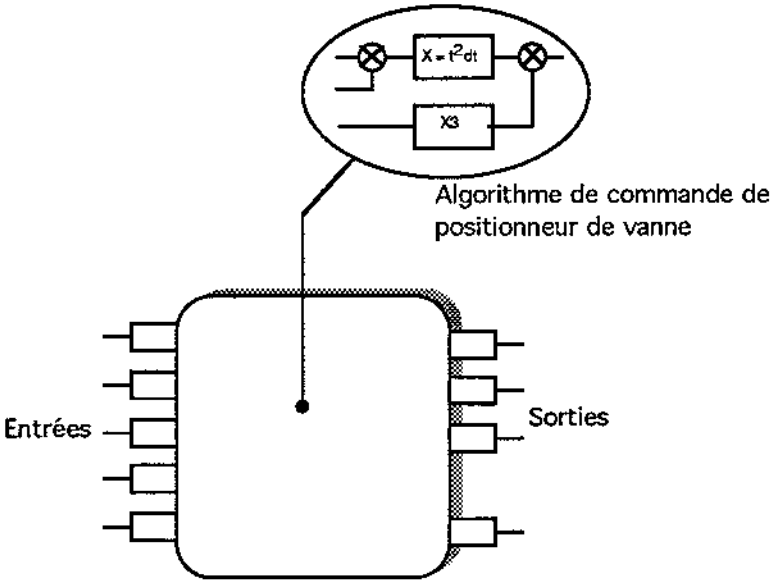


Figure 1-1- Bloc fonction

Les entrées d'un bloc fonction peuvent être initialisées à une valeur de "démarrage à froid" lors de la première exécution du bloc. Tous les paramètres ont un démarrage à froid par défaut qui peut être modifié par le programme utilisateur.

Les blocs fonctions peuvent être interconnectés par "câblage par soft" pour construire des stratégies de commande complexes. Le câblage prend la forme d'expressions texte qui déterminent la relation entre la sortie d'un bloc et l'entrée du suivant. Les connexions sont réalisées par des pas de langage construits au moyen du langage Texte structuré (TS) ou par câblage graphique. Ce câblage peut être point à point ou peut inclure des expressions logiques ou arithmétiques complexes.

Les fonctions numériques comme ABS_REAL son représentées sous une forme schématique semblable aux blocs fonctions dans la documentation PC3000. Toutefois, une fonction peut avoir une ou plusieurs entrées mais a une seule sortie. Au contraire, les blocs fonctions ont des entrées et des sorties multiples.

Il est important de noter les différences entre l'exécution des fonctions PC3000 et l'exécution des blocs fonctions. Dans le PC3000, l'exécution d'une fonction, numérique ou autre, est un événement transitoire. Une fonction est exécutée pendant l'évaluation d'une expression ; elle n'est plus exécutée avant d'apparaître de nouveau dans une expression.

Au contraire, les blocs fonctions et les pas de câblage associés sont exécutés en continu, à la vitesse fixée par la *tâche* à laquelle ils sont associés. Les aspects variables dans le temps comme la manipulation d'un paramètre de configuration sont gérés à l'aide du programme séquentiel. Un bloc fonction contient généralement une mémoire interne qui conserve les valeurs entre les exécutions. Dans la plupart des cas, ces valeurs sont conservées en cas de mise hors tension et permettent un démarrage à chaud du bloc fonction. Le *Manuel utilisateur du PC3000* contient une présentation du câblage du bloc fonction et du programme séquentiel.

Les blocs fonctions s'exécutent dans un ordre strict ; par exemple, dans le cas des blocs associés à des entrées physiques, les valeurs entrées sont collectées à un moment donné, le corps du bloc fonction est exécuté ensuite. Pour les blocs fonctions de sortie, le déroulement est dans l'ordre inverse : le corps est exécuté avant l'envoi de la valeur de sortie à la sortie physique.

Le câblage entre les blocs est effectué comme s'il appartenait au bloc fonction qui contient le paramètre de destination. La vitesse d'exécution ne dépend pas du type du paramètre source ou destination. Si un bloc fonction est lié à plusieurs sorties d'un autre bloc, le système PC3000 garantit une production de l'ensemble des sorties par la même exécution.

Pour avoir une description complète de l'exécution des blocs fonctions, de la latence de l'entrée/sortie et d'autres sujets, il faut se reporter à la Base du système d'exploitation temps réel PC3000 (HA022918).

FORME GENERALE

Selon le type de station de programmation, le bloc fonction peut être représenté sous forme de texte ou de graphique. Se reporter au manuel utilisateur applicable pour avoir des détails sur le type de représentation utilisé.

Dans ce texte, les blocs fonctions sont représentés par des diagrammes. Tous les diagrammes de blocs fonctions adoptent la convention suivante : les **entrées** arrivent par la gauche, les **sorties** partent par la droite. Le type de données pour chaque paramètre est indiqué à côté de la 'broche' entrée ou sortie. Certains paramètres d'entrée peuvent être modifiés par le bloc fonction lors de son exécution, ces paramètres apparaissent aussi sous forme de sorties et sont appelés paramètres d'**entrée/sortie.**

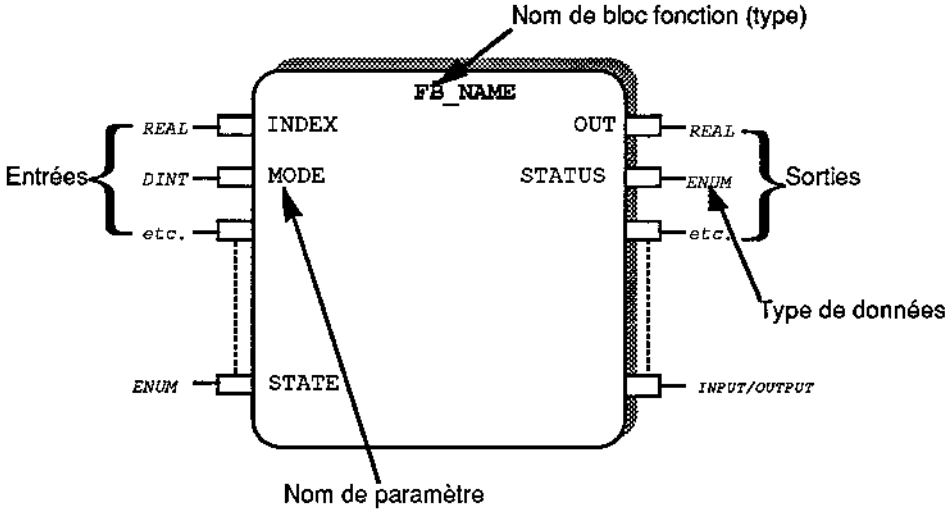


Figure 1-2 Forme générale

Noms des blocs fonctions sont limités à 8 caractères.

Pour avoir des détails relatifs à l'exécution des blocs fonctions et des entrées/sorties, il est conseillé de se reporter à la Base du système d'exploitation temps réel PC3000.

TERMINOLOGIE ET NOMENCLATURE :

Les termes suivants servent à décrire la hiérarchie des blocs fonctions de PC3000 :-

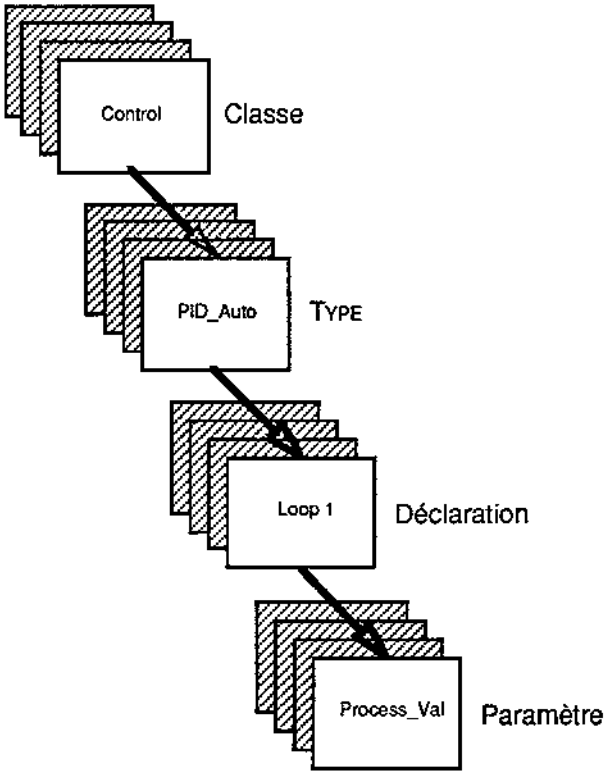


Figure 1-3 Hiérarchie des blocs fonctions

La nomenclature et les polices suivantes sont utilisées dans ce texte :

CONTROL	Indique la classe du bloc fonction
PID_Auto	Indique le type du bloc fonction
Loop1	Bloc fonction de type **PID_Auto** portant le nom déclaré de Loop1.
Loop1.PV	Paramètre déclaré du bloc Loop1 dont le mnémonique est PV.
REAL	Type de données IEC du paramètre.

Un nom de bloc de jonction dont le préfixe est 'x' indique un bloc obsolète. Ces blocs n'apparaissent normalement pas dans la liste de classes de la station de programmation mais si un programme développé dans une version antérieure est chargé, ces blocs sont accessibles.

Ce manuel donne uniquement des descriptions à des fins de référence.

ATTRIBUTS DES BLOCS FONCTIONS

Tous les blocs fonctions ont un ensemble d'attributs qui leur sont propres. Ces données sont regroupées dans un tableau pour chaque type de bloc.

Type	Nombre hexadécimal à quatre chiffres qui définit le type de bloc fonction. Il se rapporte au nom de fichier utilisé pour identifier les fichiers associés à ce bloc fonction installé sur la station de programmation. Ce nombre fixe également la classe et l'ordre dans lequel le bloc apparaît dans la liste de typesd sur la station de programmation. En utilisation normale, il n'est pas nécessaire de s'occuper de ce paramètre et il est uniquement donné à titre d'information.
Classe	Classe à laquelle est associé ce bloc, **CONTROL** par exemple.
Tâche par défaut	Une vitesse d'exécution par défaut est attribuée à chaque bloc fonction. La configuration par défaut PC3000 affecte les blocs fonctions à une des tâches Task_1 ou Task_2 (10ms et 100ms).
Liste récapitulative	Cet attribut définit les paramètres qui apparaissent dans le cadre d'une 'liste récapitulative' ou d'un résumé du bloc, lorsqu'ils sont affichés sur la station de programmation. La liste récapitulative offre une méthode d'affichage, sous forme compacte, des paramètres auxquels on accède le plus souvent, si bien que de nombreuses déclarations d'un même type de bloc fonction peuvent apparaître sur le même écran.
Durée d'exécution	Cet attribut peut servir à estimer le temps nécessaire au système temps réel PC3000 pour exécuter le bloc fonction. Dans le cas d'un bloc complexe comme PID_Auto, un certain nombre de durées est fourni et couvre toute une gamme de modes de fonctionnement. Les durées d'exécution des blocs banals qui ont un effet négligeable sur le fonctionnement du système ne sont pas données.
Besoin de mémoire pour déclaration	Indique la RAM nécessaire pour chaque 'déclaration' d'un type de bloc fonction donné. Les chiffres indiquent la mémoire utilisée en octets.

Tableau 1-1 Attributs des blocs fonctions

NOMS ET MNEMONIQUES DES PARAMETRES

Tous les noms de paramètres de blocs fonctions indiqués dans ce document utilisent la convention suivante :

Paramètre, par exemple **Preset_DT, Stop_Bits, Status**, où <Paramètre> est un nom texte pouvant avoir un maximum de 12 caractères.

La convention d'attribution des noms présente le paramètre de bloc fonction comme sur la station de programmation. En règle générale, les paramètres de blocs fonctions PC3000 ont des noms qui comportent des 'séparateurs' ayant la forme du trait de soulignement ; les noms commencent toujours par une majuscule.

Le soulignement est une partie valable du nom du paramètre.

Dans les sections consacrées à la description des paramètres ou, dans le cas de blocs relativement complexes, la description fonctionnelle, les noms de paramètres sont suivis de parenthèses ( ). Les parenthèses ( ) contiennent les mnémoniques utilisables comme abréviation pour référencer un paramètre dans un état de texte structuré : par exemple, on peut utiliser **SP** à la place du nom complet du paramètre **Setpoint (point de consigne).**

ATTRIBUTS DES PARAMETRES

Chaque paramètre de bloc fonction possède un ensemble d'attributs qui caractérisent son fonctionnement : entrée, sortie, type de données, etc.

Pour chaque bloc, les attributs sont présentés sous forme de tableau à la fin de chaque chapitre. Les atttributs sont les suivants :

Nom	Nom du paramètre sous forme de texte
Type	Type de données du paramètre. Cf. tableau 1-3, types de données.
Démarrage à froid	Valeur par défaut prise par le paramètre à la suite d'un 'démarrage à froid' du programme d'application. Il est possible de saisir d'autres valeurs avant de sauvegarder le programme d'application.
Accès en lecture	Niveau d'accès par défaut pour l'affichage des paramètres de la station de programmation (opérateur, superviseur ou configuration), désigné respectivement par Opér, Super and Config.
Accès en écriture	Niveau d'accès par défaut pour modifier le paramètre de la station de programmation (opérateur, superviseur ou configuration), désigné respectivement par Opér, Super et Config.
Limite supérieure	Limite supérieure utilisée pour les contrôles de plage dans le gestionnaire de communications EI Bisync. Il faut noter que la valeur peut être supérieure à cette limite si la mise à jour s'effectue depuis le programme d'application ; cette limite s'applique uniquement à l'interface de communications.
Limite inférieure	Limite inférieure de contrôles de plage dans le gestionnaire de communications EI Bisync.
Sens	Répertorie la gamme d'options associées à un paramètre énuméré ou les états d'un paramètre booléen, par exemple 'HAUT', 'BAS'.

Tableau 1-2 Attributs des paramètres

Conventions

Dans certains textes, le type de données standard IEC 1131-3 est également indiqué pour clarifier l'utilisation d'un type particulier de données. Dans ces cas, le type de données IEC figure entre parenthèses ; exemples : virgule flottante (REEL), durée (TEMPS).

Les valeurs hexadécimales sont indiquées avec un suffixe en minuscule 'h', par exemple 73h, 0Ah.

MOT-CLE	TYPE DE DONNEES	BITS	PLAGE
IO_ADDRESS	Adresse de canal	32	1:01:01 à 8:12:14
BOOL	Booléen	1	0 ou 1
REAL	Nombre réel	32	$\pm 10^{\pm 38}$
SINT	Entier court	8	-128 à 127
USINT	Entier court sans signe	8	0 à 255
INT	Entier	16	-32768 à 32767
UINT	Entier sans signe	16	0 à 65535
DINT	Entier double	32	-2147483648 à 2147483647
UDINT	Entier double sans signe	32	0 à 42944967296
TIME	Durée	32	Jusqu'à 49 jours
TIME_OF_DAY	Heure du jour	32	00:00:00 à 23:59:59
DATE	Date	32	01-Jan-1970 au 01-Jan-2136
DATE AND TIME	Date ET heure du jour	32	01-Jan-1970-00:00:00 au 01-Jan-2136-23:59:59
STRING	Chaîne de caractères de longueur variable		
ENUM	Elément d'une énumération	32	0 à 2^{32}

Table 1-3 Types de données des blocs fonctions

Chapitre 2

SYSTEME

Edition 1

Sommaire

PRESENTATION

Cette classe de blocs fonctions offre une interface entre le programme d'application de l'utilisateur et le système d'exploitation en temps réel PC3000. Les blocs qui offrent un accès aux paramètres qui commandent la mise en route du système, à l'horloge en temps réel et aux utilitaires d'affichage des messages de diagnostic sont inclus.

Se reporter à la base du système d'exploitation temps réel pour avoir plus d'informations sur l'utilisation de ces blocs fonctions.

BLOC FONCTION PcsSTATE

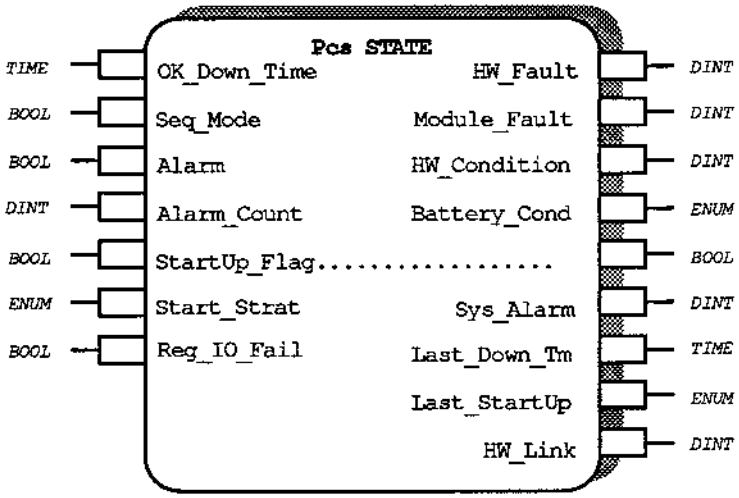


Figure 2-1 Diagramme du bloc fonction PcsState

Description fonctionnelle

Le bloc fonction PcsSTATE offre une interface avec le système temps réel PC 3000. Tous les programmes utilisateur ont automatiquement une déclaration de ce bloc. L'utilisateur n'a pas besoin d'en créer un. Ce bloc est implicite.

Attributs du bloc fonction

Type : 8 10

Classe : ..... SYSTEME

Catégorie: ..... système

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Alarm, Alarm_Count, ... Sys_Alarms

Description des paramètres

OK_Down_Time (ODT)

OK_Down_Time définit la durée maximale d'une coupure de courant acceptable par le système pour un démarrage à chaud. Cette durée doit inclure le temps nécessaire pour que le système effectue son auto-diagnostic de redémarrage (3 secondes environ). Le système accède uniquement à **OK_Down_Time** lorsque le paramètre stratégique de démarrage **Start_Strat** est positionné sur W_D_E_C (4) ou W_D_E_N (5).

Seq_Mode (SM)

Seq_Mode agit sur le mode du programme séquentiel. Si Seq_mode est positionné sur Hold (1), l'exécution du programme séquentiel est suspendue, c'est-à-dire que les étapes actives et les transitions ne sont pas exécutées. Si Seq_Mode est positionné sur Normal (0), le programme séquentiel peut tourner normalement.

N.B. : ce paramètre n'a aucune fonction dans les versions 2.09 et 2.27 du logiciel.

Alarm (AL)

Le paramètre Alarm n'a aucune fonction. Il a été inclus pour les extensions futures.

Alarm_Count (ALC)

Alarm_Count est affiché dans l'angle supérieur droit de l'écran de la station de programmation PC3000. Si Alarm_Count est égal à zéro, l'affichage indique "OK". Il est prévu pour être défini dans le programme utilisateur, soit par câblage soit dans les pas ST.

StartUp_Flag (STF)

Start Up_Flag peut servir à détecter si le PC3000 LCM a été mis hors tension. Il est positionné sur On (1), par le système, si le LCM est mis hors tension pendant qu'un programme utilisateur tourne. Il peut par conséquent être utilisé par le programme utilisateur pour détecter si le système a été mis hors tension et relancé.

Start_Strat (SUS)

Start_Strat indique la stratégie de mise en route que le système doit utiliser. Il possède six valeurs possibles :

Do_Not (0) :..... Le programme utilisateur ne peut pas démarrer.

Cold (1) : ..... Le programme utilisateur sera lancé à froid.

Warm (2) : ..... Le programme utilisateur sera lancé à chaud.

W_Els_C (3) : ..... Démarrage à chaud mais, en cas d'échec, démarrage à froid.

W_D_E_C (4) : .... Démarrage à chaud si le système est dans OK_Down_Time ; dans le cas contraire, démarrage à froid.

W_D_E_N (5) : .. Démarrage à chaud si le système est dans OK_Down_Time ; dans le cas contraire, pas de démarrage.

Reg_IO_Fail (RIF)

Lorsque Reg_IO_Fail est sur Yes (1), toutes les erreurs détectées au cours de la communication avec les modules d'entrée/sortie sont enregistrées dans le journal de consignation des erreurs système.

HW_Fault (HWF)

HW_Fault n'a aucune fonction. Il a été inclus pour les extensions futures.

Module_Fault (MF)

Module_Fault indique la défaillance d'un module matériel. Si Module_Fault est 0, aucune défaillance de module n'a été détectée. Si Module_Fault est différent de zéro, le numéro indique le module qui a eu une défaillance (ou le premier module s'il y a eu plusieurs défaillances), avec les chiffres répartis entre les champs R SS CC, où R est le numéro de rack, SS le numéro d'emplacement et CC le numéro de canal dans le module.

Par exemple, 30402 indique qu'il y a une défaillance dans le rack 3, module 4, canal 2.

HW_Condition (HWC)

HW_Condition n'a aucune fonction. Il a été inclus pour les extensions futures.

Battery_Cond (BAC)

Battery_Cond montre l'état de la batterie de la RAM LCM Il peut avoir quatre valeurs :

Good (0) : .....la batterie est en bon état.

Low (1) : ..... la tension de la batterie est faible et il faut remplacer la batterie.

Faulty (2) : ..... la tension de la batterie est insuffisante pour maintenir la mémoire.

Unknown (3) : . la tension de la batterie est non définie.

Sys_Alarms (SAC)

Sys_Alarms contient un décompte du nombre d'alarmes système qui se sont produites. Le détail des alarmes système est contenu dans le journal de consignation des erreurs.

Last_Down_Tm (LDT)

Last_Down_Tm enregistre la durée pendant laquelle le système a été arrêté depuis le dernier arrêt. Il comprend le temps nécessaire pour l'auto-diagnostic de mise en route du système. Last_Down_Tm enregistrera uniquement la durée d'arrêt si Start_Start est positionné sur W_D_E_C(4) ou W_D_E_N(5). Le paramètre enregistrera la durée de 0 à la valeur de OK_Down_Time. Si la durée d'arrêt est supérieure à cette durée, elle sera enregistrée comme étant égale à zéro.

Last_Start_Up (LSU)

Last_Start_Up indique le type de la dernière mise en route qui a été effectuée. Il peut avoir une des quatre valeurs suivantes :

None (0) : .....Le programme utilisateur n'a pas été lancé.

Cold (1) : .....Le programme utilisateur a été lancé à froid.

Warm (2) : .....Le programme utilisateur a été lancé à chaud.

Cold_WF (3) : Le programme utilisateur a été lancé à froid après un échec du démarrage à chaud.

HW_Links (LKS)

HW_Links indique la position des cavaliers d'adresse sur le LCM qui servent à paramétrer l'identificateur de groupe pour les communications Bisync Eurotherm. Sa configuration s'effectue à l'aide de cavaliers sur les LCM de la VERSION 1 et par un commutateur rotatif sur les modules de la VERSION 2. Dans l'un et l'autre cas, lHW_Links montre l'adresse actuelle.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres à chaque type	
Alarm	BOOL	No(0)	Oper	Super	Sens	Non(0) Oui (1)
Alarm Count	DINT	0	Oper	Super	Limite haute Limite basse	999 0
Battery Cond	ENUM		Oper		Sens	Bon (0) Faible (1) Défectueux (2) Inconnue (3)
HW Condition	DINT	0	Oper		Limite haute Limite basse	10,000 0
HW Fault	DINT	0	Oper		Limite haute Limite basse	10,000 0
HW Links	DINT		Oper		Limite haute Limite basse	1000 0
Last Down Tm	TIME	0	Oper			
Last StartUp	ENUM	Froid (0)	Oper		Sens	Néant(0) Froid(1) Chaud (2) WF froid (3)
Module Fault	DINT	0	Oper		Limite haute Limite basse	10,000 0
OK Down Time	TIME	0	Oper	Super		
Reg IO Fail	BOOL	Non(0)	Config	Config	Sens	No n(0) Oui(1)
Seq Mode	BOOL	Normal (0)	Oper	Super	Sens	Normal (0) Blocage (1)
StartUp Flag	BOOL	Off (0)	Oper	Super	Sens	Off (0) On (1)
Sys Alarms	DINT	0	Oper		Limite haute Limite basse	10,000 0

Tableau 2-1 Attributs des paramètres PcsState

BLOC FONCTION MESSAGES

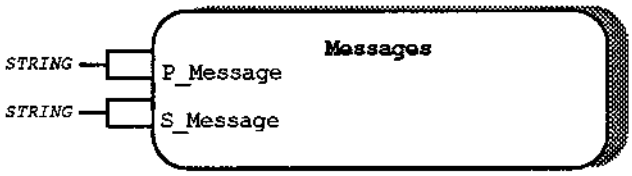


Figure 2-2 Diagramme du bloc fonction messages

Description fonctionnelle

En ligne, le bloc fonction messages permet l'affichage de deux chaînes sur la ligne d'affichage de messages en haut de l'ensemble des écrans de station de programmation PC3000. Les messages peuvent comporter un maximum de 40 caractères et sont affichés sur la gauche et la droite de la ligne d'affichage de messages. Ce bloc est créé implicitement dans tous les programmes utilisateur.

Attributs du bloc fonction

Type : 8 20

Classe : ..... SYSTEM

Tâche par défaut : ..... Task_2

Besoin de mémoire pour la déclaration : 84 octets

Description des paramètres

P_Message (PM)

Le paramètre P_Message contient le message primaire qui doit être affiché sur la moitié gauche de la ligne d'affichage. La chaîne d'entrée peut avoir une longueur maximale de 40 caractères.

S_Message (SM)

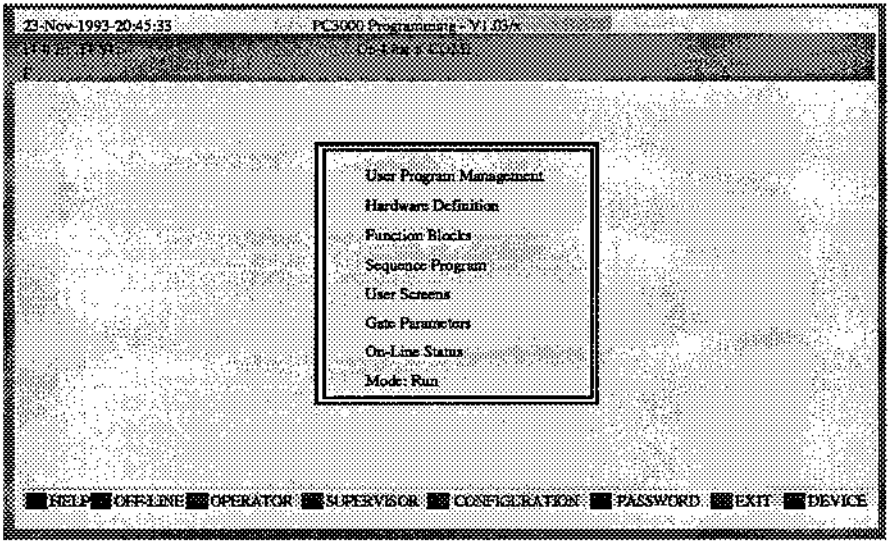
Le paramètre S_Message contient le message secondaire qui doit être affiché sur la moitié droite de la ligne d'affichage. La chaîne d'entrée peut avoir une longueur maximale de 40 caractères.

Exemple:

Messages P_Message := 'Message principal';

Messages S_Message:= 'Autre message';

On obtient un écran de station de programmation (station de programmation sur DOS)



Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres à chaque type	
P_Message	STRING		Oper	Config	Longueur maximale de la chaîne	40 caractères
S_Message	STRING		Oper	Config	Longueur maximale de la chaîne	40 caractères

Tableau 2-2 Attributs des paramètres de messages

BLOC FONCTION TACHE

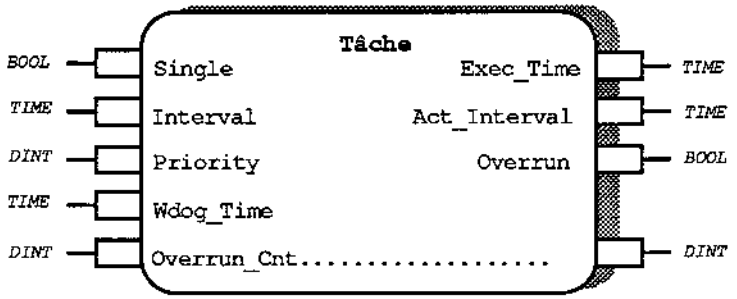


Figure 2-3 Diagramme de bloc fonction Tâches

Description fonctionnelle

Le bloc fonction Tâches offre au programmeur une interface utilisateur dans le système PC3000. Consulter le chapitre consacré au programmeur de tâches en temps réel, dans la base du système d'exploitation temps réel PC3000. Tous les programmes utilisateur ont deux blocs de tâches par défaut avec intervalles de 10ms et 100ms.

Attributs du bloc fonction

Type : ..... c 10

Classe : ..... SYSTEM

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Interval, Act_Interval,

..... Overrun_Cnt,

Besoin de mémoire pour la déclaration : 4396 octets

Description des paramètres

Single (SGL)

Le paramètre Single n'a aucune fonction Il a été inclus pour les extensions futures. Il doit normalement conserver sa valeur Non (0)

Interval (IV)

Le paramètre Interval définit la durée d'échantillonnage souhaitée entre les programmations de tâches. La tâche la plus rapide doit avoir un intervalle qui ne soit pas supérieur à 65 ms. En outre, chaque tâche (sauf la tâche la plus rapide) doit avoir un intervalle qui soit un multiple entier de la tâche qui lui est

immédiatement supérieure par la rapidité, c'est-à-dire que les intervalles de tâches doivent avoir un rapport monotone. Si la tâche la plus rapide dure plus de 65 ms ou si les intervalles n'ont pas un rapport monotone, le PC3000 ne pourra pas tourner et une erreur système sera enregistrée lorsque la commande RUN sera émise. Le PC3000 restera à l'état HALTED.

Priority (PRI)

Le paramètre Priority définit la priorité de la tâche que le programme anticipatif doit utiliser, 0 étant la priorité maximale et 7 la priorité minimale. La tâche la plus rapide doit avoir la priorité maximale.

Wdog_Time (WDT)

Wdog_Time définit la durée maximale attribuée au bloc fonction avant l'enregistrement d'un chien de garde. Si la tâche n'a pas fini de s'exécuter après un temps égal au Wdog_Time, un chien de garde utilisateur sera enregistré, provoquant la consignation d'une erreur système et une réinitialisation du PC3000.

Overrun_Cnt (OVC)

Le paramètre Overrun_Cnt indique le nombre de débordements de la tâche qui se sont produits depuis la dernière remise à zéro d'Overrun_Cnt. Pour remettre Overrun_Cnt à zéro, il faut lui affecter la valeur 0 ou relancer le programme utilisateur.

Exec_Time (EXT)

Exec_Time indique le temps qui a été nécessaire à la tâche lors de sa dernière exécution, elle inclut la durée d'exécution des programmes qu'elle appelle mais n'inclut pas la durée d'exécution d'autres tâches qui lui sont prioritaires.

Act_Interval (AIV)

Act_Interval indique le dernier intervalle entre les exécutions. Lorsque le système n'est pas débordé, cet intervalle est égal à **Interval.** Si un débordement se produit, Act_Interval indiquera l'intervalle réalisé.

Overrun (VRN)

Overrun indique si la durée d'exécution de la tâche est supérieure à Interval. En utilisation normale, **Act_Interval** est inférieur à **Interval,** Overrun sera donc sur Non (0). Si **Act_Interval** est supérieur à **Interval,** Overrun sera sur Oui (1).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Act_Interval	TIME	0ms	Oper			
Exec_Time	TIME	0ms	Oper			
Interval	TIME	500ms	Oper	Super	Limite haute Limite basse	5m 5ms
Overrun	BOOL	Non (0)	Oper		Sens	Non (0) Oui (1)
Overrun_Cnt	DINT	0	Oper	Super		
Priority	DINT		Oper	Super	Limite haute Limite basse	7 0
Single	BOOL	Non (0)	Oper	Super	Sens:	Non (0) Oui (1)
Wdog_Time	TIME	5s	Oper	Super	Limite haute Limite basse	10m 0ms

Tableau 2-3 Attributs des paramètres Tâches

BLOC FONCTION RT_CLOCK

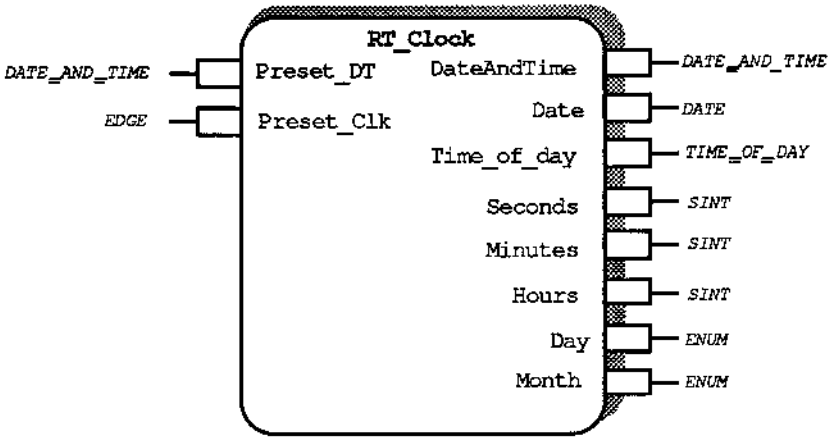


Figure 2-4 Diagramme du bloc fonction RT_Clock

Description fonctionnelle

Le bloc fonction RT_Clock permet de visualiser et de paramétrer l'horloge interne temps réel du PC3000. Pour paramétrer l'horloge, il faut saisir la date et l'heure correctes dans Preset_DT. Ces valeurs sont ensuite saisies dans le PC3000 par passage de Preset_Clk de Tock (0) à Tick (1), la réinitialisation de l'horloge, intervenant sur le front montant, c'est-à-dire lors du passage de 0 à 1. Une fois cette opération achevée, Preset_Clk doit être remis sur Tack(0).

Attributs du bloc fonction

Type : 8 48

Classe : ..... SYSTEM

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... DateAndTime

Besoin de mémoire pour la déclaration : 40 octets

Description des paramètres

Preset_DT (PDT)

Le paramètre Preset_DT sert uniquement à fixer la date et l'heure sur l'horloge du PC3000. La valeur de Preset_DT est saisie sur le front montant de Preset_Clk de Tock (0) à Tick (1)

Preset_Clk (PC)

Le paramètre Preset_Clk sert à saisir **Preset_DT** dans le PC3000. La saisie s'effectue sur le front montant **Preset_Clk** de Tock (0) à Tick (1).

DateAndTime (DAT)

La date et l'heure sont la date et l'heure actuelles. Elles sont issues de données en seconde à partir dur 1 janvier 1970, les dates antérieures ne sont donc pas valables.

Date (D)

La date est la date actuelle, telle qu'elle figure dans la mémoire du PC3000.

Time_Of_Day (TOD)

Time_Of_Day est l'heure actuelle, représentée sous la forme d'une heure sur 24 heures.

Seconds (SEC)

Le paramètre Seconds indique l'heure actuelle en secondes, entre 0 et 59.

Minutes (MIN)

Le paramètre Minutes indique l'heure actuelle en minutes, entre 0 et 59.

Hours (HR)

Le paramètre Hours indique l'heure actuelle en heures, entre 0 et 23.

Day (DAY)

Le paramètre Day indique la date actuelle en jours, de 0 à 6 (dimanche à samedi).

Month (M)

Le paramètre Month indique la date actuelle en mois, de 0 à 11 (janvier à décembre).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres à chaque type	
Date	DATE	1 janvier 1970	Oper	Bloc	Limite haute Limite basse	19 jan 2038 1 jan 1970
DateAndTime	DATE_AND_TIME	1 janvier 1970 00:00:00	Oper	Bloc	Limite haute Limite basse	19 jan 2038 03:14:06 1 jan 1970 00:00:00
Day	ENUM	Sun (0)	Oper	Bloc	Sens	Dim (0) Lun (1) Mar(2) Mer (3) Jeu (4) Ven (5) Sam(6)
Hours	SINT	0	Oper	Bloc	Limite haute Limite basse	23 0
Minutes	SINT	0	Oper	Bloc	Limite haute Limite basse	59 0
Month	ENUM	Jan (0)	Oper	Bloc	Valeurs énumées	Jan (0) Fév (1) Mars (2) Avril (3) Mai (4) Juin(5) Juil(6) Août (7) Sept (8) Oct (9) Nov (10) Déc (11)
Preset_Clk	BOOL	Tock (0)	Oper	Super	Sens	Tock (0) Tick (1)
Preset_DT	DATE_AND_TIME	Jan 1 1970 00:00:00	Super	Super	Limite haute Limite basse	19 jan 2038 03:14:06 1 jan 1970 00:00:00
Seconds	SINT	0	Oper	Bloc	Limite haute Limite basse	59 0
Time_Of_Day	TIME_OF_DAY	00:00:00	Oper	Bloc	Limite haute Limite basse	23:59:59 00:00:00

Tableau 2-4 Attributs des paramètres RT Clock

Chapitre 3

COMMUNICATIONS

Edition 1

Sommaire

Sommaire (suite)

Sommaire (suite)

PRESENTATION DES COMMUNICATIONS

INTRODUCTION

Objet

La présentation des communications décrit les principes de base des communications série, aide à choisir les protocoles de communications qui conviennent, définit la manière de configurer un système de communications, fournit un guide de diagnostic des défauts courants et donne des conseils permettant d'optimiser les communications.

Cette présentation des communications sera particulièrement utile aux utilisateurs pour la première configuration d'un système de communications série. On part du principe que le lecteur connaît un peu la terminologie des communications ; les explications de la terminologie plus technique seront fournies. L'annexe A contient un glossaire de termes.

But

Ce document contient les informations techniques permettant à un utilisateur de configurer la liaison de communications entre un PC3000 et d'autres périphériques à l'aide des communications série.

Les communications série peuvent servir à assurer l'interface avec :

la station de programmation PC3000

d'autres appareils Eurotherm (par exemple 905s).

les unités d'entraînement à moteur, les blocs de thyristors

les automates logiques programmables (utilisant par exemple le protocole Siemens 3964 (R))

les écrans opérateur comme Euro Panel ou les terminaux Xycom 12 et 9 pouces

les autres PC3000

les systèmes d'acquisition de données

les systèmes "cell control" comme le Production Orchestrator d'Eurotherm

les systèmes de supervision comme le progiciel ESP d'Eurotherm

les imprimantes pour la production des rapports

les divers matériels déposés comme les balances électroniques, les capteurs intelligents, etc.

Pour toute information sur l'utilisation de protocoles de communications spécifiques, l'utilisateur doit consulter le protocole qui s'y rapporte figurant dans ce chapitre.

PRINCIPES DE COMMUNICATIONS

Les communications série permettent les échanges d'informations entre plusieurs périphériques par codage, en une série de bits, des informations contenues dans un flux d'octets de données. La transmission des bits peut s'effectuer par un certain nombre de techniques dépendant du support utilisé pour la liaison série. Par exemple, avec des câbles à paires torsadées, les bits sont transmis par commutation de la différence de tension entre la paire de fils. En règle générale, le transfert des informations s'effectue par décomposition des informations en une série de messages encadrés par des caractères de contrôle spéciaux dépendant du protocole utilisé.

Le périphérique récepteur renvoie normalement un court message au périphérique émetteur pour accuser réception de chaque message. Il est détecté par l'émetteur et implique une réception correcte du dernier message et la possibilité que l'émetteur envoie le message suivant.

Pour garantir la détection des erreurs de transmission, la plupart des protocoles exigent l'envoi des messages à l'aide d'une structure donnée, c'est-à-dire que certains caractères de commande sont ajoutés au début et à la fin du message. Un octet spécial de contrôle de bloc est habituellement ajouté à la fin du message, selon la teneur du message. Par exemple, avec le protocole EI Bisync, le caractère de contrôle d'horloge est fonction de la parité longitudinale de chaque octet du message et est transmis comme dernier octet du message. Le logiciel du périphérique déporté peut alors vérifier que le message est correct en contrôlant les caractères de commande encadrants et le caractère de vérification de bloc. En cas de détection d'une erreur, le périphérique déporté renvoie normalement un message de commande spécial (caractère de commande 'Pas reçu', par exemple, pour indiquer à l'émetteur que le message doit être à nouveau transmis).

Fonctionnement maître-esclave

Avec la plupart des protocoles de communications série, il est possible de relier de nombreux périphériques à une liaison série en utilisant le mode de fonctionnement maître-esclave. Un périphérique est appelé le périphérique 'maître' et les autres sont appelés 'esclaves'.

Le périphérique maître est habituellement le seul périphérique de la liaison série qui peut envoyer des informations aux autres périphériques reliés à la liaison série ou leur demander des information. Les périphériques esclaves ont une fonction complémentaire : ils peuvent uniquement envoyer des informations au périphérique maître en réponse à une demande de celui-ci ou recevoir des informations qui leur sont envoyées par le périphérique maître. Avec ce mode de fonctionnement, un périphérique esclave ne peut ni envoyer directement des informations à un autre périphérique esclave ni envoyer des informations quand cela ne lui est pas demandé.

On utilise souvent ce mode pour relier le PC3000 à un certain nombre d'appareils discrets. Par exemple, avec un PC3000 configuré comme maître d'une liaison série EI Bisync, il est possible d'interroger un certain nombre d'appareils multipoints de la série 900 pour obtenir les valeurs de process de chaque appareil.

Même s'il n'y a que deux périphériques sur une liaison série point à point, il faut qu'un périphérique soit le maître et l'autre l'esclave pour de nombreux protocoles série.

Adresse des périphériques esclaves

Chaque esclave doit avoir une adresse de communications distincte afin que le périphérique maître puisse envoyer les messages à un périphérique donné. Par exemple, avec EI Bisync, chaque esclave reçoit une adresse composée d'une identité de groupe (GID) et d'une identité d'unité (UID). Etant donné que le schéma d'adressage varie d'un protocole à l'autre, il faut veiller à ce que chaque périphérique esclave ait une adresse de communications unique et valable pour le protocole utilisé.

Il faut aussi configurer les périphériques maîtres pour l'adressage d'un périphérique esclave donné, c'est-à-dire que les adresses des esclaves et les adresses que connaît le périphérique maître doivent correspondre. La configuration des adresses pour l'utilisation avec les ports PC3000 en modes maître ou esclave est décrite dans la suite de ce document. Pour les périphériques déportés particuliers comme un PLC utilisant JBus, la configuration des adresses de communications sera définie dans le manuel de communications relatif à chaque périphérique.

Fonctionnement égal à égal

L'utilisation de protocoles qui prennent en charge le mode de fonctionnement égal à égal permet à n'importe quel périphérique d'un réseau d'émettre une demande d'envoi d'informations à un autre périphérique ou de lire des informations d'un autre périphérique. En règle générale, ce mode de fonctionnement est associé à des réseaux à jetons ou Ethernet, bien que quelques protocoles série comme le protocole Siemens 3964 (R) permettent le fonctionnement égal à égal sur une seule liaison série point à point par une technique de maître mobile.

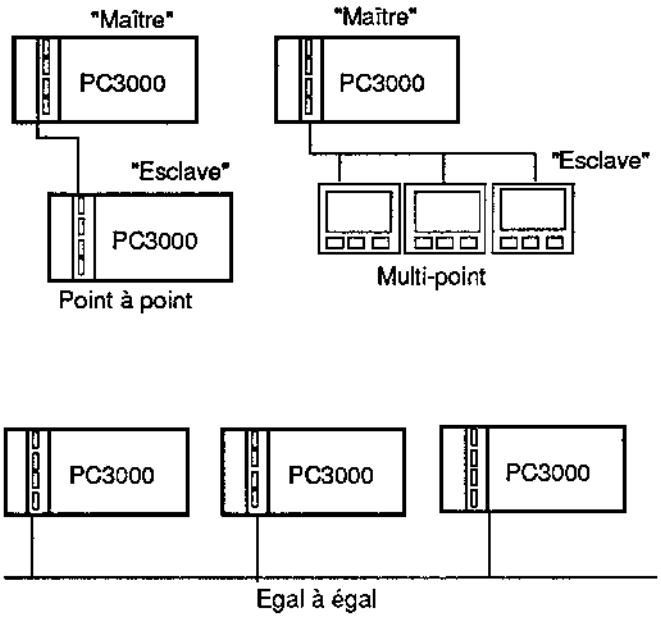


Figure 3-1 Configurations des communications

Adressage des paramètres

Pour adresser un élément, un paramètre ou une fonction donné(e) dans un périphérique, chaque protocole définit un schéma d'adressage, qui peut varier considérablement entre les protocoles mais permet généralement l'adressage d'éléments simples ou multiples.

Avec EI Bisync, on utilise normalement l'identité de groupe (GID) et l'identité d'unité (UID) pour adresser un périphérique esclave multi-points. Une identité de canal (CHID) peut être nécessaire en option pour adresser un sous-ensemble de paramètres. Un mnémonique à deux caractères est nécessaire pour adresser les paramètres souhaités. Pour plus de détails sur l'adressage des paramètres dans PC3000, cf. la description consacrée à EI Bisync dans ce chapitre.

PORTS DE COMMUNICATIONS

Normes

Le PC3000 offre un certain nombre de ports de communications sur des modules comme LCM et le module de communications intelligent (ICM). Chaque port utilise ou, dans le cas de l'ICM, peut être configuré pour utiliser l'une des normes de transmission suivantes :

RS232 Cette norme permet de relier deux périphériques par une faible longueur de câble (normalement moins de 15 m), un fil servant à l'émission des données et l'autre à la réception. Elle doit uniquement être utilisée dans les environnements avec peu de parasites et pour des vitesses d'émission inférieures à 20 kBaud. Elle ne permet pas les communications multi-points. Le fonctionnement semi-duplex et duplex intégral est possible.

RS422 Cette norme permet de réaliser une connexion multi-points d'un ou plusieurs périphériques esclaves à partir d'un seul périphérique maître. Il faut l'utiliser dans les environnements à niveau de parasites élevé, de préférence à la norme RS232, mais elle nécessite deux paires de câbles torsadés, une pour l'émission et une pour la réception des données. Contrairement à la norme RS232, elle offre une émission de signaux différentielle équilibrée et peut par conséquent être utilisée sur des distances et à des vitesses supérieures. Le fonctionnement semi-duplex et duplex intégral est possible, bien que le mode semi-duplex soit utilisé plus fréquemment.

RS485 Cette norme a des caractéristiques comparables à celles de RS422 mais elle permet aux périphériques de communiquer par des protocoles d'égal à égal. Une seule paire de câbles à paires torsadées est utilisée pour l'émission et la réception des données. Ce mode permet uniquement un fonctionnement en semi-duplex.

Les modes RS422 et RS485 permettent une longueur totale de liaison série pouvant atteindre 1000m sans répéteur de signaux avec des vitesses d'émission jusqu'à 100 kBaud et un câble à paires torsadées 24 AWG.

Le tableau ci-dessous donne une brève description des possibilités des ports de communications PC3000.

Module	Port	Norme	Modes	Vitesses (Baud)
LCM	A	RS422	Point à point uniquement*	75-38,4 k
	B	RS422	Point à point uniquement*	75-38,4 k
	C	RS422	Maître à esclave*	75-115,2 k
ICM	A	RS422 ou RS485	Maître ou esclave	300-115,2 k
	B	RS422 ou RS485	Maître ou esclave	300-115,2 k
	C	RS422 ou RS485	Maître ou esclave	300-115,2 k
	D	RS4232	Point à point	300-19,2 k

Les trois ports LCM peuvent fonctionner en mode maître ou en mode esclave. Les ports LCM A et B ne peuvent servir pour les configurations point à point que lorsqu'ils sont configurés comme esclaves mais ils peuvent être maîtres de liaisons série multi-points. Le port LCM C peut servir comme maître ou comme esclave sur les configurations multi-points.

Le port D du LCM peut uniquement servir pour les racks d'extension et ne doit être relié à aucun autre périphérique externe.

Les liaisons peuvent être configurées sur la carte-mère du LCM de manière à configurer l'adresse de l'esclave qui peut être utilisée pour les protocoles (dont EI Bisync et JBus). De même, il existe sur l'ICM un commutateur qui permet de configurer l'adresse de l'esclave par défaut. Les paramètres des adresses d'esclaves et les blocs fonctions du module esclave doivent recevoir des valeurs nulles pour que les adresses matérielles par défaut soient utilisables. Cf. 'Base du matériel PC3000' pour avoir des détails sur le paramétrage des adresses d'esclaves sur ces modules.

Installation de modules intelligents

Il faut utiliser les ICM lorsque des ports de communications supplémentaires sont nécessaires ou lorsqu'il faut décharger le LCM d'une partie des activités de communications. Cela est particulièrement illustré par les données de performances du PC 3000 comme les dépassements de capacité des tâches, consulter les chapitres relatifs aux tâches dans la 'Base du système d'exploitation temps réel du PC3000'. Pour obtenir des valeurs spécifiques au fonctionnement des blocs fonctions, consulter les données relatives aux blocs fonctions dans ce chapitre.

Utilisatn le LBus, les ICM ne peuvent être installés que dans les emplacements 1 à 5 du premier rack PC3000. Il est important de noter que seuls des modules LBus ou d'autres ICM peuvent prendre place entre un ICM et le LCM. En d'autres termes, les ICM doivent faire partie d'un ensemble contigu de modules LBus situés à côté du LCM.

Il faut se reporter au 'Manuel d'installation PC3000' pour avoir des détails supplémentaires sur le câblage et les possibilités des modules de communications.

Câblage entre les périphériques et les ports

Il existe un certain nombre de câbles qui permettent de relier les périphériques les plus couramment utilisés au PC3000. Il est conseillé de les utiliser lorsque cela est possible. Le 'Manuel d'installation PC 3000 HA 022231' en donne une liste complète.

Entre .. et le périphérique ..		Mode
LCM ou ICM	261	RS422
ICM Port D	Terminal couleur 12 pouces	RS23
261	Terminal couleur 12 pouces	RS23
LCM ou ICM	Terminal 9 pouces	RS422
LCM ou ICM	Euro-Panel	RS422
261	PC COM Port	RS232

Lors du câblage entre les périphériques maîtres et les périphériques esclaves en mode RS422, il faut relier la paire d'émission du maître Tx- et Tx+ à la paire de réception Rx- et Rx+ du esclave et la paire d'émission de ce dernier Tx- et Tx+ à la paire de réception du maître Rx- et Rx+. Les paires Tx-, Rx+ et Rx-, Rx+ des esclaves multiples sont couplées comme le montre la figure 'Câblage des périphériques esclaves multi-points'.

N.B. : l'ICM offre une liaison sur la carte qui permet de permuter les connexions Rx et Tx sur le port A.

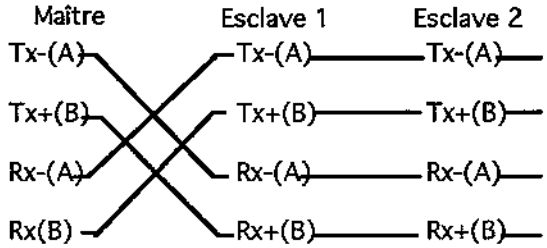


Figure 3-2 Câblage de périphériques esclaves multi-points à l'aide de RS422

N.B. : Raccordements donnés à titre indicatif pour des maîtres et esclaves utilisant la même convention de signe.

Liaison série multi-points

Lorsqu'on réalise une liaison multi-points entre un certain nombre de périphériques esclaves et un périphérique maître à l'aide de RS422, il est conseillé de réduire le plus possible la longueur des tronçons de câbles assurant la liaison avec les points de contact. Ces points de contact doivent être placés à proximité des groupes d'appareils, afin que les câbles de liaison de chaque appareil soient courts par rapport à la distance entre les points de contact et le port du périphérique maître.

En règle générale, des câbles peu coûteux non blindés conviennent pour des réseaux dont la longueur va jusqu'à 250 m environ, à des vitesses de transmission allant jusqu'à 19,2 kBauds ; pour les vitesses supérieures et les distances jusqu'à 1000 m, il est conseillé d'utiliser des câbles de bonne qualité à paires torsadées blindées. Lors de la détermination des besoins en câbles, il faut prendre en compte les critères comme l'environnement (parasites en particulier), la position des câbles, la vitesse de transmission et les taux d'erreur acceptables.

Un périphérique maître peut posséder une liaison multi-points avec un maximum de 10 périphériques esclaves sur une seule liaison série avec RS422 et avec un maximum de 31 périphériques avec RS485.

Le 'Manuel d'installation PC3000 HA 174383' donne des détails supplémentaires sur les besoins relatifs au câblage.

Conversion fonctionnement RS422 - fonctionnement RS232

Il est souvent nécessaire de relier au PC3000 un périphérique déporté comme un PC, qui ne possède que des ports RS232. En branchant un convertisseur RS422 - RS232 qui convient, on peut utiliser n'importe quel port RS422 PC3000.

Nous recommandons

l'interface série universelle Eurotherm 261

Référence :	261/230	- pour alimentation en 230V
	261/115	- pour alimentation en 115V

Même avec une interface 261, un port PC3000 configuré pour RS422 peut continuer à fonctionner en mode maître ou en mode esclave, lorsque ce mode est pris en charge.

Pour minimiser les interférences dues aux parasites, il est conseillé de placer l'interface 261 à proximité du périphérique déporté, afin que le câble RS232 entre l'interface 261 et le périphérique déporté soit le plus court possible.

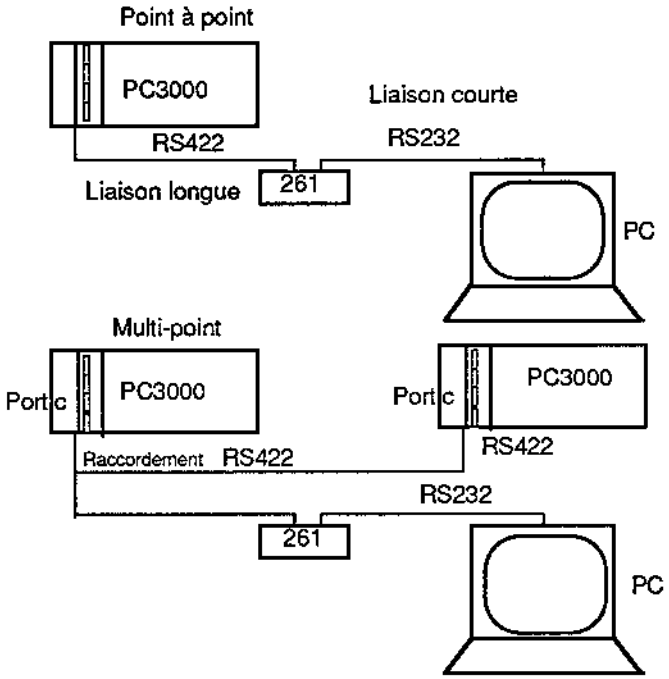


Figure 3-3 Utilisation des convertisseurs RS422/RS232

Le 'Manuel d'installation PC3000' donne des détails supplémentaires sur l'interface 261.

Si besoin est, l'interface 261 peut aussi servir à convertir un port PC3000 comme l'ICM port D qui prend uniquement en charge la conversion RS232 - RS422.

COMMUNICATIONS PAR DEFAUT

Prise en charge des ports

Les ports A, B et C prennent toujours le protocole EI Bisync comme protocole par défaut lors de la mise sous tension du PC3000, à condition qu'ils ne soient pas affectés à des blocs fonctions de communications spécifiques dans un programme utilisateur en marche. Cette caractéristique est destinée à garantir que la station de programmation PC3000 puisse toujours communiquer avec le PC3000 lorsqu'aucun programme utilisateur ne tourne ou n'est chargé. (L'utilisation des blocs fonctions de communications est décrite dans les chapitres suivants de ce document).

Par convention, le port LCM B sert normalement à la station de programmation mais il est possible d'utiliser les ports A ou C s'ils ne sont pas utilisés pour d'autres applications de communications.

Les communications série prises en charge sur les ports LCM A, B et C possèdent les caractéristiques suivantes :

Débit 9.6 kBauds

Mode esclave

Protocole EI

Adresse de l'esclave configurée par les liaisons LCM*

Un bit d'arrêt (norme EI Bisync)

*L'adresse de l'esclave par défaut (GID) est sélectionnable à l'aide de liaisons de la carte LCM parmi une des 16 adresses de la plage '0 à F'. L'adresse configurée en usine est '7'. Attention, la sélection d'adresse et les liaisons multi-points des PC3000 sont impossibles sur les LCM antérieurs à la version 2. L'adresse LCM est décrite en détail dans le bloc fonction 'Module esclave EI Bisync', dans ce chapitre. Se reporter aussi au 'Manuel d'installation PC3000' pour avoir des détails supplémentaires sur le LCM et sur la manière d'identifier le numéro de version.

Les communications par défaut sont essentiellement prévues pour être utilisées avec la station de programmation PC3000 mais elles peuvent aussi servir à communiquer avec le système de supervision Eurotherm (ESP) et le système "cell control" Production Orchestrator. Par exemple, un PC sur lequel tourne ESP peut être relié aux ports A, B ou C à l'aide du protocole par défaut EI Bisync. Toutefois, les communications par défaut peuvent être utilisées pour diverses applications 9.6 kBauds, EI Bisync et en PC3000 mode esclave, à condition que le partenaire déporté possède le protocole maître EI Bisync.

Le protocole EI Bisync fournit un mécanisme permettant à un maître déporté de lire et écrire une gamme importante de paramètres et de fonctions dans le PC3000,. Ce qui comprend la totalité des paramètres de blocs fonctions utilisés dans un programme utilisateur chargé qui tourne, les fonctions du système PC3000 et les paramètres des blocs fonctions de variables d'esclaves EI Bisync.

Le port LCM C permet d'avoir une liaison multi-points avec un certain nombre de PC3000 à partir d'une seule liaison série RS422. Par exemple, la station de programmation PC3000 peut être reliée, par l'intermédiaire d'un convertisseur 261, à un certain nombre de PC3000 avec liaisons multi-points à l'aide du port C sur les différents LCM ; il faut toutefois noter que chaque LCM doit être configuré à une adresse d'esclave différente (GID pour EI Bisync par exemple).

Adressage des paramètres

Toutes les fois qu'un programme utilisateur est créé sur la station de programmation PC3000 (PS), tous les paramètres de communications adressables des blocs fonctions reçoivent une identité unique. Pour les communications par défaut EI Bisync, chaque paramètre d'un programme utilisateur a une adresse unique composée de l'identité d'unité (UID), de l'identité de canal (CHID) et d'un mnémonique. La partie consacrée à l'adressage des paramètres dans la description du 'Module de périphérique esclave EI Bisync', dans ce chapitre, en donne une description détaillée.

Nous insistons sur le fait qu'il n'est pas nécessaire de configurer les blocs fonctions de communications ni de configurer des paramètres dans le programme utilisateur pour utiliser les communications par défaut. Lors de la création du programme utilisateur sur la station de programmation PC3000, l'utilisateur a la possibilité de produire deux fichiers '<nom>. CEL" et '<nom>.GAT', qui définissent les paramètres adressables dans le programme utilisateur. La partie <nom> du nom de fichier varie pour chaque programme utilisateur. Consulter le 'Guide utilisateur de la station de programmation PC3000' pour avoir des détails supplémentaires sur la manière de créer ces fichiers.

Les deux fichiers d'adresses contiennent des informations identiques mais avec une structure légèrement différente. Chacun contient une liste des paramètres de blocs fonctions qui existent dans un programme utilisateur, ainsi que leurs adresses et les types de données qu'ils contiennent, par exemple virgule flottante (REEL), booléenne (BOOL) etc..

Le fichier 'CEL' sert au Production Orchestrator pour la création d'une base de données destinée à l'adressage des paramètres PC3000. L'annexe donne une description de la structure des fichiers.

Le fichier 'GAT' est utilisé dans ESP et contient une liste des paramètres de blocs fonctions PC3000 définis sous forme de portes ESP. Pour avoir des informations supplémentaires sur la structure des fichiers 'GAT', consulter la documentation ESP.

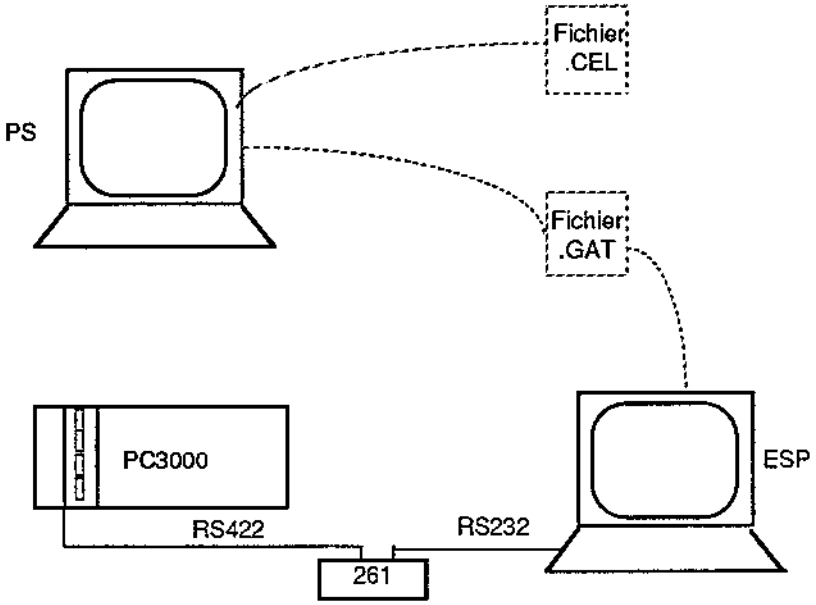


Figure 3-4 Communication avec ESP

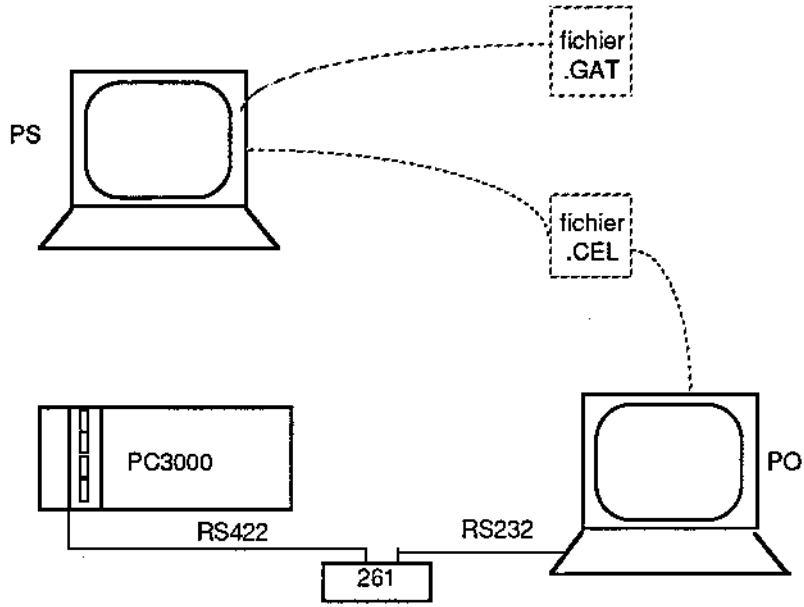


Figure 3-5 Communication avec le Production Orchestrator

Normalement, pour configurer les communications entre le PC3000 et ESP ou Production Orchestrator, il n'est pas nécessaire de connaître en détail le protocole EI Bisync ou la structure des fichiers 'GAT' ou 'CEL', il faut seulement :

sélectionner un port LCM libre : A, B ou C. Le port B est normalement réservé à la station de programmation.

si le PC (utilisé pour ESP ou Production Orchestrator) n'est pas équipé d'une interface RS422, utiliser des câbles standard (qui peuvent être commandés à Eurotherm) pour assurer la connexion entre le port et un convertisseur 261 et entre le convertisseur 261 et le PC.

produire soit un fichier GAT (pour ESP) soit un fichier CEL (pour Production Orchestrator) sur la station de programmation pour le programme utilisateur.

copier le fichier GAT sur ESP pour créer des écrans ESP ou copier le fichier CEL sur le jeu d'outils de construction de cellules de Production Orchestrator.

lorsque le programme utilisateur est chargé et tourne, ESP ou Production Orchestrator peut adresser les paramètres de blocs fonctions dans le programme utilisateur à l'aide des adresses fournies par le fichier d'adresses.

N.B. : en cas de modification du programme utilisateur, en particulier en cas d'ajout ou de suppression de blocs fonctions, les adresses des paramètres des blocs fonctions autres que les blocs 'esclaves' EI Bisync peuvent changer. Il est par conséquent conseillé de créer le nouveau fichier GAT ou CEL et de le copier dans ESP ou Production Orchestrator pour le réintégrer. L'utilisation des blocs fonctions de variables esclaves EI pour éviter cette opération est décrite dans une partie ultérieure.

Applications sur mesures

Dans certains cas, il peut être nécessaire de développer un programme d'application fonctionnant comme maître EI Bisync pour communiquer avec un PC3000, par exemple pour créer un programme d'édition personnalisé sur PC lisant des paramètres particuliers du PC3000. Il est possible de développer rapidement une application PC avec module maître EI Bisync intégré en utilisant une bibliothèque logicielle commercialisée par Eurotherm. Il est aussi possible de développer un module maître EI Bisync en se reportant au 'Manuel EI Bisync'.

BLOCS FONCTIONS DE COMMUNICATIONS

Il existe trois types de blocs fonctions associés aux communications. Avec la station de programmation, il faut créer et configurer ces blocs dans un programme utilisateur de la même manière que pour n'importe quel autre bloc fonction.

Blocs fonctions de modules de communications. Utilisés pour affecter un protocole et définir diverses caractéristiques pour les ports indiqués.

Blocs fonctions de variables esclaves. Utilisés, dans une communication où le PC3000 est esclave, comme "boîtes aux lettres" d'échange Les paramètres esclaves sont accessibles depuis les périphériques maîtres par un protocole désigné.

Blocs fonctions de variables déportées. Utilisés dans une communication où le PC3000 est maître les "boîtes aux lettres" d'échange permettent de rapatrier des données de provenance de périphériques déportés.

Blocs fonctions Protocole

Il est possible de désigner un port pour qu'il soit utilisé avec un protocole en créant un bloc fonction de module de communications propre à ce protocole.

Protocole	Type de bloc fonction	Commentaire
EI Bisync Master	EI_Bisync_M	
EI Bisync Slave	EI_Bisync_S	
JBus Master	JBus_M	Prend en charge JBus et Modbus Maître
JBus Slave	JBus_S	Prend en charge JBus et Modbus Esclave
Siemens 3964(R)	Siemens_M_S	Prend en charge le maître et l'esclave
Toshiba Master	Toshiba_M	Pour Toshiba EX250, EX500 et EX2000
Euro Panel	Euro_Panel	Prend en charge la connexion avec un affichage Euro_Panel
User defined	Raw_Comms	Permet un contrôle de niveau élémentaire du port série

Pour avoir une liste complète des protocoles de communications pour la version actuelle du PC3000, consulter le 'Récapitulatif technique du PC 3000 HA 174064'.

Affectation d'un bloc fonction de communications à un port

Chaque bloc fonction de communications a un paramètre Port qui sert à affecter le bloc fonction à un port sélectionné. L'affectation du port est définie sous la forme d'une chaîne de deux caractères, où le premier est un chiffre compris entre 0 et 5 représentant l'emplacement dans le rack du PC3000 et le deuxième caractère est une lettre représentant le port dans cet emplacement. Par exemple, '3A' affecte le port A du module dans l'emplacement 3 du rack principal PC3000.

L'utilisateur doit s'assurer que :

a) l'emplacement désigné contient un module matériel comme un ICM qui peut prendre en charge les communications série avant de faire tourner le programme utilisateur et

b) que le port considéré n'a pas été affecté à d'autres modules.

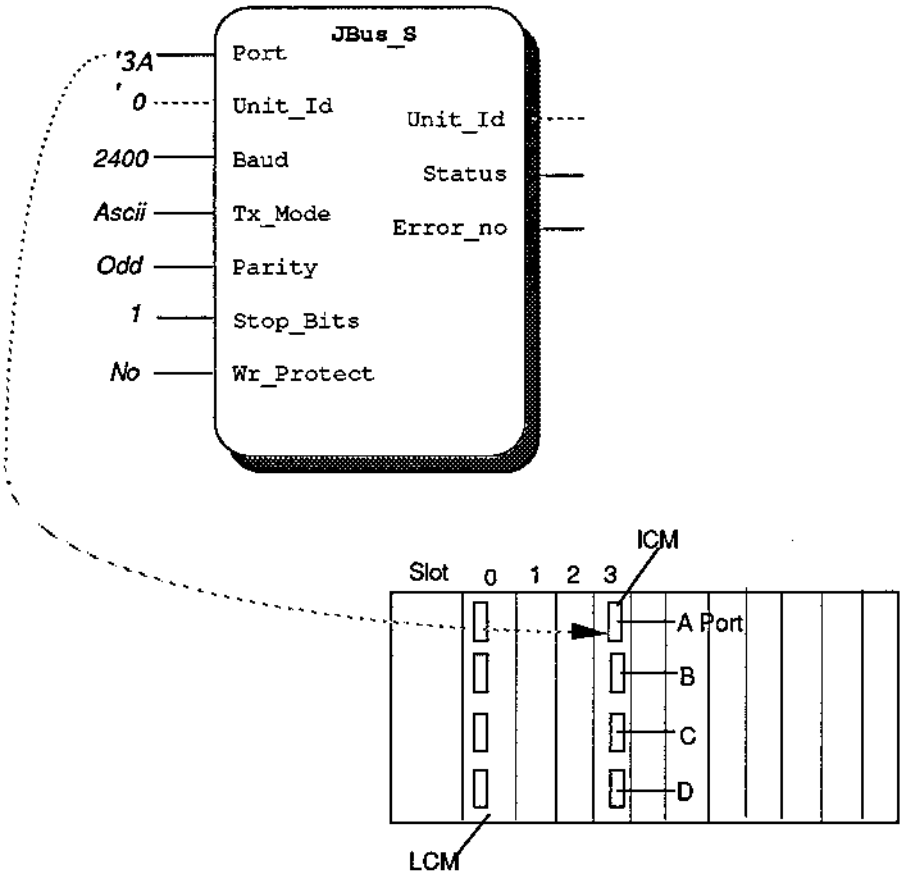


Figure 3-6 Affectation d'un bloc fonction de communications à un port

Modification des paramètres de blocs fonctions de communications

La plupart des blocs fonctions de communications ont des paramètres d'entrée qui peuvent servir à personnaliser la configuration du port affecté (modification du débit et de la parité, par exemple).

Bien que la modification de ces paramètres soit possible, ils ne sont pris en compte qu'au lancement du programme utilisateur . Les paramètres de cette catégorie sont par exemple :

le débit

les bits d'arrêt

la parité

Un programme utilisateur ne peut pas modifier dynamiquement la configuration d'un port de communications. Toutefois, il est possible de modifier temporairement la configuration pour réaliser des tests, en utilisant la station de programmation (cf. la partie suivante). Pour toute information sur les paramètres de configuration des modules de communications, consulter le module de communications concerné de ce chapitre.

Modification temporaire des paramètres de configuration

Certains paramètres de blocs fonctions de communications comme le numéro de port, la parité ou le débit d'un module de communications sont normalement définis lors de la création du bloc fonction. Ils peuvent être modifiés à n'importe quel moment à l'aide de la station de programmation lorsqu'elle est en ligne avec le PC3000 mais ce dernier ignore les nouvelles valeurs des paramètres de configuration des blocs fonctions de communications . Il ne prend en compte ces paramètres qu'à un lancement du programme

Pour modifier temporairement, à des fins de test, un paramètre comme le numéro des ports, il faut arrêter le programme utilisateur, modifier le port puis relancer le programme utilisateur. Pour une modification définitive, il faut intégrer cette modificatin dans le programme source application développée en mode "off line" et sauver, recompiler, télécharger à nouveau.

Détection d'erreurs des blocs fonctions de module de communications

Chaque bloc fonction de module de communications possède les paramètres de sortie suivants :

Status Lorsque le programme utilisateur tourne, ce paramètre passe de 'Go' à 'NoGo' si le module de communications détecte une erreur. Status peut rester sur 'NoGo' si certains paramètres de configuration sont incorrects (affectation de port erronée par exemple).

Error No Ce paramètre est un entier qui reçoit une valeur correspondant au type d'erreur détectée. La valeur utilisée varie en fonction des protocoles. Lorsque cela était possible, les codes d'erreur ont été utilisés de manière à correspondre à ceux utilisés normalement avec le protocole considéré. S'il

n'y a aucune erreur, il est positionné sur 0, c'est-à-dire 'OK'. Consulter l'annexe pour avoir une liste complète des codes d'erreur standard.

Si l'erreur est temporaire, **Status** et **Error-No** reviendront respectivement à 'Go' et 'OK' à la prochaine exécution du bloc fonction de communications, à condition que l'erreur ait été corrigée.

Blocs fonctions de variables esclaves

Les blocs fonctions de variables esclaves contiennent les valeurs de paramètres auxquels il est possible d'accéder à l'aide d'adresses spécifiques de communications par l'intermédiaire de ports qui prennent en charge un protocole esclave donné. Pour les protocoles incluant EI Bisync, ils permettent d'affecter aux paramètres une adresse propre au protocole. Les variables esclaves sont configurées pour utiliser un protocole désigné et sont par conséquent accessibles par n'importe quel port qui pilote ce protocole. Un nombre quelconque de variables peuvent être associées à un protocole mais il est impossible que des protocoles différents y accèdent simultanément.

Chaque variable esclave contient une valeur ou un ensemble de valeurs qui peuvent être lues et écrites par le programme utilisateur et sont accessibles par un protocole maître déporté pouvant effectuer une lecture et une écriture.

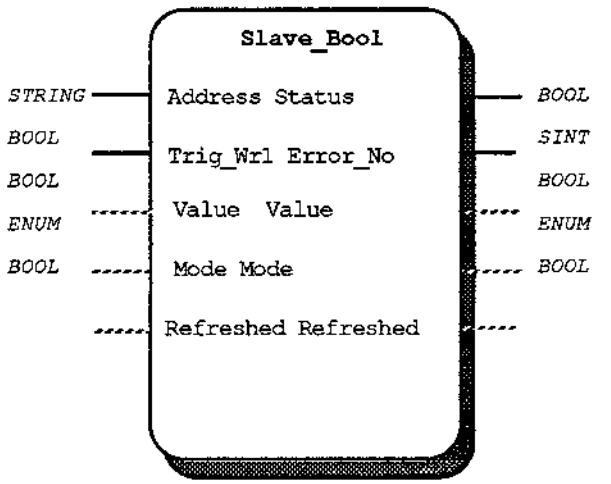


Figure 3-7 Exemple d'un bloc fonction de variable esclave

Les adresses des variables esclaves ne changent pas lors de la modification d'un programme utilisateur. Par opposition, les adresses normales EI Bisync utilisées pour les paramètres de blocs fonctions peuvent changer lors de la modification d'un programme utilisateur, en particulier lors de l'ajout ou de la suppression de blocs fonctions. Les variables esclaves sont par conséquent particulièrement utiles

lorsqu'il est nécessaire de minimiser les changements d'adresses de paramètres dans les périphériques maîtres déportés, cf. chapitre 3, partie 'Adressage des paramètres à l'aide des communications par défaut'.

Par exemple, en utilisant une variable esclave, il est possible de fournir un paramètre à virgule flottante (REEL) qui est accessible par n'importe quel port prenant en charge l'esclave EI Bisync. La variable esclave peut recevoir une adresse EI Bisync et un mnémonique comme par exemple : canal = '1', mnémonique = '80'.

Ce paramètre est toujours accessible depuis n'importe quel port prenant en charge l'esclave EI Bisync avec la même adresse et le même mnémonique.

Normalement, les variables esclaves servent à fournir des valeurs de paramètres qui peuvent être lues et écrites par des périphériques déportés et des systèmes de supervision, par exemple, les terminaux opérateur utilisant JBus ou l'Euro-Panel.

Il existe différentes variables esclaves pour stocker les paramètres possédant des types de données différents. Les types pris en charge sont les suivants :

Utilisation	Type de bloc fonction
Booléen simple	Slave_Bool
Réel simple	Slave_Real
Entier simple	Slave_Int
Temps simple	Slave_Time
Chaîne simple	Slave_Str
Ensemble de 8 booléens Ensemble de 64 booléens	Slv_Bool_8 Slv_Bool_64
Ensemble de 8 réels Ensemble de 64 réels	Slv_Real_8 Slv_Real_64
Ensemble de 8 entiers Ensemble de 64 entiers	Slv_Int_8 Slv_Int_64
Ensemble de 16 booléens lus à l'aide des communications sous la forme d'un entier simple à 16 bits (mot d'état)	Slave_SW

Affectation d'adresses et de protocoles à des variables esclaves

Chaque variable esclave a un paramètre de configuration qui définit le protocole et l'adresse de communications pour le paramètre. L'adresse est définie sous la forme d'une chaîne de caractères où les deux premiers caractères sélectionnent le

protocole associé au paramètre et le reste de la chaîne définit l'adresse propre au protocole.

Les caractères de sélection de protocole comprennent :

EB Eurotherm EI Bisync

EP Euro-Panel display

SI Siemens 3964(R)

JB JBus Slave

Consulter le dernier 'Récapitulatif technique PC3000 HA022230' pour avoir la liste complète.

Exemples de chaînes d'adresses pour les variables esclaves :

```
'EB060' - Paramètre à adresser par le protocole esclave EI
                Bisync avec une identité de canal = '0',
                mnémonique - '60'
Le GID et l'UID sont définis par le bloc fonction de module de
                communications.
'EBX1B' - Paramètre à adresser par le protocole esclave EI
                Bisync avec une identité de canal = 'X',
                mnémonique = '1B'
Le GID et l'UID sont définis par le bloc fonction de module de
                communications EI Bisync.
'JB0001' - Paramètre à adresser par le protocole esclave JBus
                sous la forme du registre 0001. L'identité
                de l'unité JBus est définie par le bloc
                fonction de module de communications JBus.
```

Se reporter aux descriptions des blocs fonctions de modules de communications spécifiques pour avoir des détails sur les conventions applicables aux adresses de protocoles. Il existe des cas où certaines adresses ne sont pas autorisées. Par exemple, avec EI Bisync, il doit toujours y avoir un canal, le premier caractère d'un mnémonique de variable esclave doit être un chiffre compris entre '0' et '9', les variables esclaves sont toujours adressées par un UID = '0' ou 'P' si elles sont adressées depuis ESP.

Il est impossible de modifier le paramètre Adresse pendant que le programme utilisateur tourne et, de ce point de vue, ce paramètre se comporte comme les autres paramètres de configuration des blocs fonctions de communications, cf. partie : 'Modification temporaire des paramètres de configuration'.

La figure 3-8 montre :

une variable esclave booléenne possédant une identité de canal = '0', mnémonique = '60',

une variable esclave entière avec une identité de canal = '0', mnémonique = '62',

- ces deux variables peuvent être adressées à l'aide de chaque module EI Bisync, par l'intermédiaire du 'OA' utilisant GID = '0', UID = '0' et par l'intermédiaire du port '1B' utilisant GID = '1', UID = '0'.

une variable esclave entière qui peut être adressée sous la forme du registre '0001' esclave 3 par un protocole esclave JBus par l'intermédiaire du port '0C'.

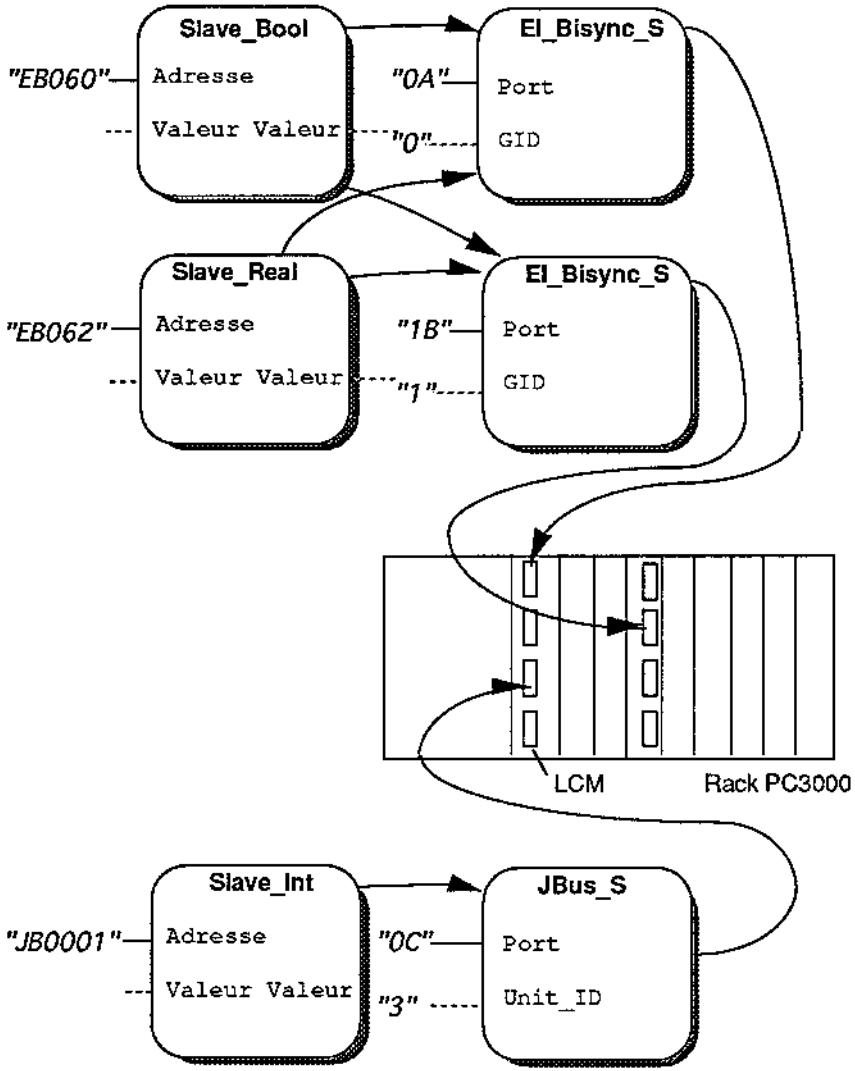


Figure 3-8 Exemple d'utilisation de variable esclave

Liste des adresses de variables esclaves

La station de programmation offre une possibilité de produire un fichier contenant une liste de variables esclaves possédant des adresses de communications pour un protocole sélectionné. Cela peut par exemple être utile lors de la configuration d'un terminal d'affichage comme un écran 9M (9 pouces monochrome) ou CRT12C (12 pouces couleur) utilisant JBus et où il y a un nombre élevé de variables esclave.

Le fichier produit s'appelle '<nom>.ads', où '<nom>' est différent selon les programmes utilisateur.

Contrôle des blocs fonctions de variables esclaves

Tous les blocs fonctions de variables esclaves possèdent un ensemble de paramètres qui comprend :

Mode Peut prendre les valeurs suivantes : Rd-Wr qui permet à la valeur esclave d'être lue ou écrite par les communications, Rd-Only, qui empêche les communications déportées d'écrire dans la variable esclave et Wr_Once qui permet à la variable esclave d'être écrite une fois mais empêche toute écriture ultérieure. Wr_Once sert à garantir qu'un périphérique déporté n'écrasera pas une valeur écrite dans une variable esclave avant que le programme utilisateur ait eu le temps de la traiter. Le mode passe de Wr_Once à Rd-Only lorsqu'une écriture dans la variable esclave est terminée.

Trig_Wrl Paramètre booléen qui, lorsqu'il passe de Off à On, fait prendre la valeur Wr_Once au paramètre Mode. Il ne doit être piloté que depuis une source dérivée par câblage par soft comme une entrée numérique I/O. Ce paramètre doit être repositionné sur Off pour pouvoir être réutilisé. Ce paramètre peut être utilisé comme partie d'un verrouillage pour réguler les valeurs écrites dans la variable esclave depuis un périphérique déporté.

Refreshed Ce paramètre booléen est positionné sur Yes toutes les fois qu'une nouvelle valeur a été écrite dans la variable esclave par la communication ; il peut aussi être acquitté par le programme utilisateur. Il sert essentiellement à signaler à un programme utilisateur qu'une nouvelle valeur a été écrite dans la variable esclave par la communication.

Status Paramètre booléen de sortie qui peut être utilisé par Go ou NOGO pour indiquer qu'une erreur s'est produite lors du fonctionnement avec la variable esclave.

Error_No Le paramètre entier de sortie passe de 0 (OK) à une valeur de code d'erreur toutes les fois que l'état passe à NOGO. Etant donné que le code d'erreur est fixé par le module de communications de protocole associé, il faut consulter la description de bloc fonction de module de communications pour avoir une description détaillée des codes d'erreur. Une valeur de code d'erreur 255 indique que les variables esclaves n'ont pas été initialisées et peut impliquer que des caractères

de sélection de protocole incorrects ont été fournis dans le paramètre Adresse. Cf. annexe C pour avoir une liste détaillée des codes d'erreur standard.

Accès verrouillé

Dans certaines applications, on peut accepter que la valeur d'une variable esclave soit écrasée par la communication maître avant que la valeur ait été traitée par le programme utilisateur ou qu'une nouvelle valeur soit écrite localement .

Toutefois, il existe des situations où l'échange de données doit garantir que chaque valeur est transférée et reconnue par le programme utilisateur. On peut obtenir ce résultat en utilisant une variable esclave de drapeau comme verrouillage. La figure 3-9 montre une situation type où un périphérique déporté envoie un flux de valeurs au programme utilisateur par l'intermédiaire de la variable esclave Val1. En mode Wr_Once le programme utilisateur garantit qu'une seule écriture est possible avant le traitement de la nouvelle valeur écrite dans Val1. Le mode Wr_Once empêche le périphérique déporté d'écraser la valeur, il provoque aussi la signalisation d'une erreur de communications si une nouvelle erreur est écrite. Une meilleure solution consiste à utiliser une deuxième variable esclave comme verrouillage. Le programme utilisateur positionne le drapeau pour signaler qu'il est possible d'écrire une nouvelle valeur. Le périphérique déporté interroge ensuite le drapeau et, lorsqu'il est re-positionné, il peut envoyer la valeur suivante.

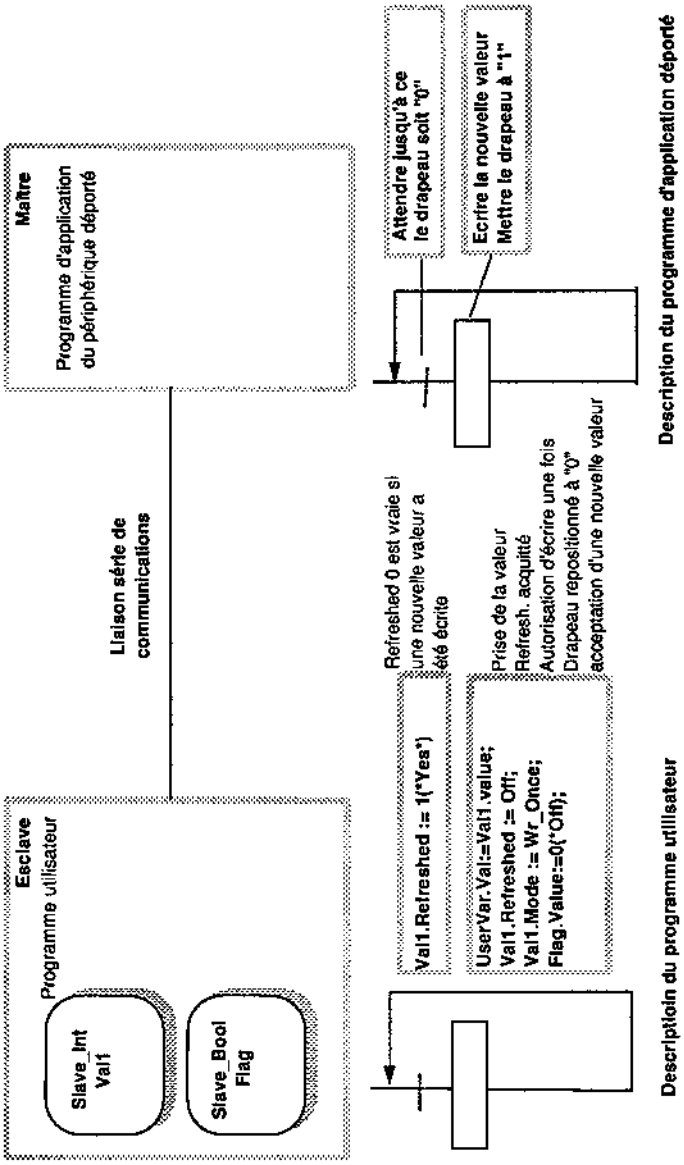


Figure 3-9 Exemple de disposition verrouillée

Contraintes de synchronisation des variables esclaves

Si le bloc fonction de module de communications est affecté à une tâche PC3000 plus rapide que le bloc fonction de la variable esclave, on peut observer certains effets secondaires.

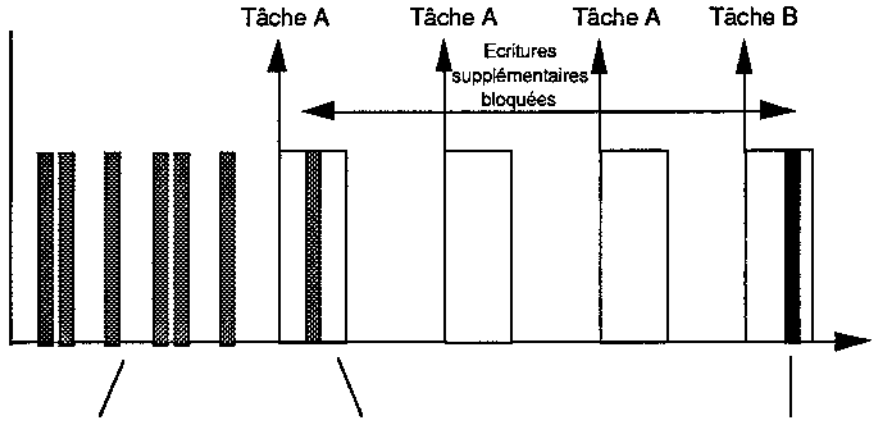
La situation suivante risque de se produire : lorsque la communication envoie des données à écrire dans une variable esclave, le module de communications écrit les données dans une mémoire tampon interne, au sein du bloc fonction de la variable esclave. Les données ne sont pas écrites dans le paramètre de sortie de la valeur de la variable esclave tant que la tâche à laquelle est associée la variable esclave n'est pas exécutée. Par conséquent, si un périphérique déporté à réponse rapide écrit dans une variable esclave puis essaie de lire immédiatement la valeur avant que la variable esclave ait été mise à jour, le périphérique déporté peut lire l'ancienne valeur de la variable esclave et non la valeur qui vient d'être écrite.

Les tentatives ultérieures d'écriture dans une variable esclave avant que la mémoire tampon interne ait été transférée à la sortie de la variable esclave seront bloquées et feront revenir le module de communications à un accusé de réception négatif (NAK) pour chaque tentative. Si la variable esclave est en mode Wr_Once, la mémoire tampon interne contient la première valeur écriture et ne peut pas être écrasée par des écritures ultérieures. Toutefois, la nouvelle valeur contenue dans la mémoire tampon interne ne sera pas copiée dans la sortie de la variable esclave avant que la tâche de l'esclave ait été achevée.

Synchronisation des variables esclaves dans le cas le plus défavorable

Le temps la plus défavorable entre la réception du message série provenant d'un périphérique déporté destiné à une variable esclave et son apparition sur la sortie d'une variable esclave est la suivante :

```
= temps de la tâche du bloc fonction e communications + temps
                de la tâche du bloc fonction de la variable
                esclave
```



N.B.: Le bloc fonction Comms. driver est associé à la tâche A
Le bloc fonction Variable esclave est associé à la tâche B

Figure 3-10 Temps de mise à jour d'une variable esclave

Par exemple, avec un module de communications et une variable esclave dans une tâche de 10ms, le temps le plus défavorable entre la réception du dernier caractère d'un message série et la valeur décodée apparaissant sur le paramètre de sortie de la valeur de la variable esclave sera de 20 ms.

Le temps le plus défavorable entre un programme utilisateur écrivant une nouvelle valeur dans une variable esclave et la valeur qui peut être lue par les communications depuis un périphérique déporté est la suivante :

= temps de tâche pour le bloc fonction de module de communications

Il est à noter que les durées de bout en bout les plus défavorables entre le programme d'application déporté et le programme utilisateur doivent aussi inclure les durées de transmission des communications série qui varient avec le débit et les temps d'attente entre les caractères et les messages du périphérique déporté.

Blocs fonctions de variables déportées

Un certain nombre de blocs fonctions de variables déportées sont fournis pour lire et écrire les paramètres ayant des données de différents types dans les périphériques déportés. Les variables déportées peuvent uniquement être utilisées avec les ports qui prennent en charge les protocoles qui fonctionnent en mode maître, comme EI-Bisync-M,JBus-M et Siemens-M-S.

Normalement, les variables déportées sont utilisées lorsque le PC3000 est le maître d'une liaison série sur laquelle sont branchés en multi-points un certain nombre de périphériques comme les appareils Eurotherm de la série 900. Dans ces cas, les variables déportées peuvent servir à lire et écrire des paramètres sélectionnés dans n'importe quel appareil.

Chaque variable déportée est configurée à l'aide d'une chaîne d'adresse qui définit le port à utiliser, l'adresse esclave du périphérique déporté qui est relié au port et l'adresse du paramètre dans le périphérique déporté.

Les possibilités offertes par chaque bloc fonction de variable déportée sont commandées à l'aide d'un jeu de paramètres standard et comprennent :

à la demande, une lecture simple d'un paramètre déporté

une lecture continue, c'est-à-dire l'interrogation d'un paramètre déporté ; il est possible de spécifier la durée entre lectures.

à la demande, une écriture simple dans un paramètre déporté

la mesure du temps écoulé entre les lectures ou écritures continues

un horodatage pour la dernière lecture ou écriture réalisée avec succès.

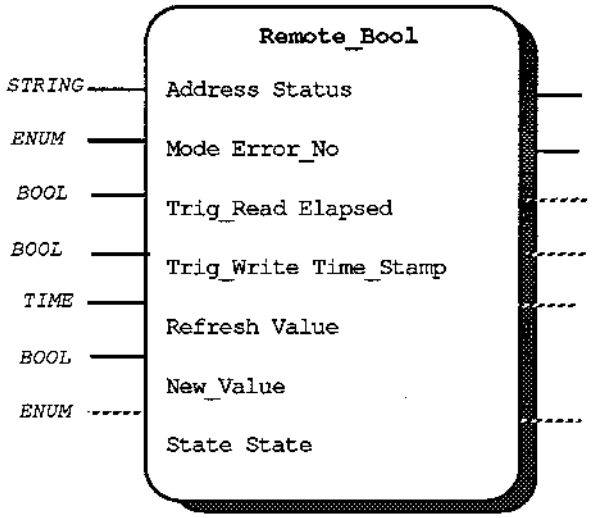


Figure 3-11 Exemple d'un bloc fonction de variable déporté

Il existe une grande variété de variables déportées différentes pouvant assurer l'interface avec les paramètres déportés de différents types de données. Les types de données pris en charge sont en particulier les suivants :

Utilisation	Type de bloc fonction
Booléen simple	Remote_Bool
Réel simple	Remote_Real
Entier simple	Remote_nt
Chaîne simple	Remote_Str
lecture par les communications sous forme d'un entier simple à 16 bits	
(mot d'état)	Remote-SW

Se reporter au dernier 'Récapitulatif technique PC3000 HA022230' pour avoir une liste détaillée.

Affectation d'adresses aux variables déportées

Il est possible d'associer une variable déportée à un paramètre dans un périphérique en définissant une valeur du paramètre Adresse. Il s'agit d'une chaîne dans laquelle les deux premiers caractères définissent le port PC3000 et le reste de la chaîne définit l'adresse propre au protocole. Dans certains cas, des caractères supplémentaires sont ajoutés à la fin de la chaîne d'adresse, pour fournir des informations supplémentaires comme le format utilisé pour la transmission du paramètre par la liaison série. La sélection du port utilise la même convention que celle des blocs fonctions de modules de communications, cf. partie 'Affectation d'un bloc fonction de communications à un port' où les premier et deuxième caractères définissent respectivement l'emplacement dans le rack et le port du PC3000.

Exemples :

```
'0A011PVf'
Accède au port A du LCM (emplacement 0), -
en supposant que le port est attribué aux communications
maîtres EI Bisync
Bloc fonction(EI-Bisync-M),
le  reste de l'adresse implique:
GID = 0, UID = 1, - définit l'adresse du périphérique esclave
Numéro du canal = 1, mnémonique = 'PV',
- définit l'adresse du paramètre
-format f - définit le format Bisync utilisé pour transmettre
la valeur.
```

```
'1B011234IR1'
Accède au port B d'un module de communications  dans
l'emplacement  1, -
en supposant que le port est attribué à des communications
maître JBus.
Bloc fonction (JBus-M),
Le reste de l'adresse implique:
Identité de l'unité  = 01 - définit l'adresse du périphérique
esclave JBus
Adresse du registre = 1234
I - définit l'espace de l'adresse d'entrée
format R1 - définit un format JBus
```

Pour avoir des détails sur la partie de l'adresse propre au protocole, se reporter à la description du bloc fonction maître du module de communications. La figure 3-11 montre :

une variable booléenne déportée accédant au port '0A' qui a été attribuée au protocole maître EI Bisync. L'adresse sélectionne un périphérique esclave avec GID=0, UID=1 et un paramètre avec un mnémonique = 'HD' à l'aide du format par défaut. Il n'y a aucune identité de canal.

une variable déportée avec virgule flottante (REEL) accédant au port '1B' qui est attribuée au protocole maître EI Bisync. L'adresse sélectionne un périphérique esclave avec GID=0, UID=0, un paramètre avec identité de canal = '2' et un mnémonique = 'PV' utilisant le format '1'.

une variable déportée entière accédant au port '0C' qui est attribuée au protocole maître JBus. L'adresse sélectionne un périphérique esclave avec une identité d'unité = '02' et un registre à l'adresse '1000' utilisant la structure 'R'.

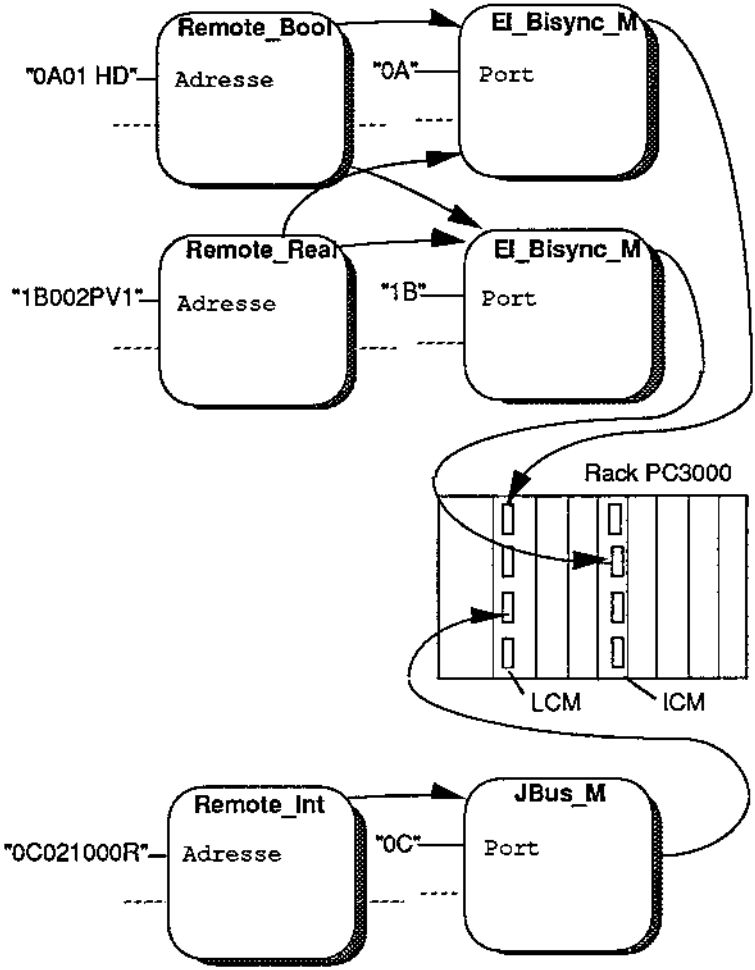


Figure 3-12 Exemple d'utilisation de variables déportées

Modification des adresses de variables déportées

Il est possible de modifier à tout moment la chaîne d'adresse d'une variable déportée à l'aide du programme utilisateur : la nouvelle adresse prend effet à la prochaine exécution de la variable déportée. On a ainsi un système très souple où il est possible d'utiliser une seule variable déportée pour accéder à différents paramètres possédant le même type de données sur différents ports et en utilisant différents protocoles.

Par exemple, en modifiant l'adresse, il est possible de lire la valeur de process PV depuis un nombre donné d'appareils reliés en multi-points sur un certain nombre de ports.

```
exemple : adresses pour lire la PV pour quatre appareils, 2 en
                  liaison multi-points pour chacun des deux
                  ports 1B et 1C.
'1B01 PV'
'1B02 PV'
'1C01 PV'
'1C02 PV'
```

Contrôle des blocs fonctions des variables déportées

En mode 'lecture', le paramètre **Valeur** contient la dernière valeur lue dans le paramètre, dans le périphérique déporté.

En mode 'écriture', la valeur contenue dans le paramètre de saisie **Nouvelle valeur** est écrite dans le paramètre déporté. Si l'opération d'écriture s'effectue correctement, le paramètre Valeur est mis à jour pour correspondre au paramètre **Nouvelle valeur.**

En règle générale, le paramètre **Valeur** contient la dernière valeur qui a été écrite ou lue correctement dans le paramètre déporté. Toutefois, si aucune opération de lecture ou d'écriture ne s'est produite, le paramètre **Valeur** ne doit pas être pris en compte.

Tous les paramètres de variables déportées ont un jeu de paramètres qui leur permet de contrôler les différents modes de fonctionnement :

Mode

Ce mode de fonctionnement de base spécifique peut être :

Mode 'Demand' - permet le déclenchement d'une lecture ou d'une écriture déportée par passage de State sur Ecriture ou par passage des paramètres Trig_Read ou Trig_Write de Off à On.

Mode 'R_Cont' - permet d'interroger en continu un paramètre déporté à intervalles réguliers définis par le paramètre de rafraîchissement TIME.

Mode 'W_Cont' - permet d'écrire en continu dans un paramètre, à intervalles réguliers définis par le paramètre de rafraîchissement TIME.

Mode 'Change' - provoque l'écriture dans le paramètre déporté lorsque la valeur du paramètre de saisie Nouvelle valeur ne correspond pas au paramètre Valeur. Ce mode peut servir à remplacer le mode d'écriture en continu lorsqu'on souhaite une diminution du nombre de transactions de communications, car on évite ainsi les écritures qui essaient d'envoyer la même valeur. La durée minimale entre les modifications transmises à un périphérique déporté est donnée par le temps de rafraîchissement.

Si une erreur se produit, l'écriture sera répétée à intervalles réguliers définis par le paramètre de rafraîchissement TIME.

Trig_Read, Trig_Write. Il faut utiliser ces paramètres lorsqu'il est nécessaire de déclencher une simple lecture ou écriture sur demande à partir d'une source issue du câblage soft. Par exemple, une entrée numérique I/0 peut être câblée par soft à l'un de ces paramètres pour déclencher une lecture ou une écriture à partir d'une impulsion numérique produite extérieurement. Ils ne doivent pas être écrits à l'aide de ST dans un grafcet (SFC) (cf. paramètre Etat).

Rafraîchissement

Ce paramètre TIME définit l'intervalle entre les transactions de communications pour les modes de lecture ou d'écriture continues. Cf. Mode pour avoir d'autres utilisations de ce paramètre.

Etat

Il s'agit d'un paramètre d'entrée/sortie qui peut avoir les valeurs suivantes : OK, En attente, Erreur, Ecriture, Lecture.

Le paramètre de sortie Etat peut prendre les valeurs suivantes : 'En attente' pendant qu'une transaction a démarré et attend d'être exécutée, en général pendant qu'elle attend la réponse d'un périphérique esclave déporté, 'Ok' lorsqu'une transaction s'est achevée de manière satisfaisante ou avant qu'une transaction ait été effectuée et 'Erreur' lorsqu'une transaction a échoué pour une raison quelconque (cf. Error-No).

Le paramètre d'entrée Etat peut être écrit dans le grafcet pour obliger la variable déportée à effectuer diverses fonctions. Si l'on positionne le paramètre Etat sur 'Ecriture', on déclenche une écriture simple de la valeur du paramètre actuel Nouvelle valeur du périphérique déporté, quel que soit le mode sélectionné à ce moment. Par exemple, si le mode est 'R Cont' (lecture continue), le positionnement d'Etat sur 'Ecriture' déclenchera une écriture unique entre les transactions de lecture. De même, le positionnement d'Etat sur 'Lecture' déclenche toujours une transaction de lecture unique. Le paramètre Etat restera en 'Ecriture' ou 'Lecture' jusqu'à la prochaine exécution du bloc fonction de variable déportée, lorsqu'il passera à 'En attente' si la transaction peut être démarrée avec succès ; dans le cas contraire, il passera sur 'Erreur'.

Le positionnement d'Etat sur 'Ok' provoque l'arrêt de toute transaction en cours. Par exemple, si une transaction d'écriture attend un accusé de réception d'un périphérique déporté, le fait de forcer le paramètre Etat à passer de 'En attente' à 'Ok' provoquera l'absence de prise en compte du message d'accusé de réception lorsqu'il sera reçu. Le fait de positionner Etat sur 'Erreur' peut servir à des fins de test lorsqu'aucune transaction n'est en attente, pour simuler un état d'erreur ; dans ce cas, Error No est positionné sur 255 pour la prochaine exécution du bloc fonction de variable déportée.

Error-No Il s'agit d'un entier qui identifie un état d'erreur, 0 correspondant à 'Ok' et les autres valeurs sont configurées si le paramètre Etat est 'Erreur'. L'annexe C donne une liste détaillée des codes d'erreur standard.

Ecoulé

Paramètre TIME qui indique le temps écoulé depuis les derniers modes lecture ou écriture ; dans les autres modes, sa valeur n'a aucune signification. Si le temps écoulé est significativement plus long que la période de rafraîchissement nécessaire, cela peut indiquer que le périphérique déporté, la liaison série ou le système PC3000 ne peut pas traiter les transactions de communications à la vitesse demandée. Si la transaction prend fin car aucune réponse n'a été reçue, le temps écoulé est fixé à l'achèvement de la transaction.

Time_Stamp Définit la date et l'heure auxquelles la dernière transaction de lecture ou d'écriture s'est achevée avec succès. Mis à jour lors de l'exécution du bloc fonction de variable déportée et lorsqu'il coïncide avec le passage du paramètre 'Etat' sur 'Ok'.

Etat

Paramètre booléen de sortie qui peut être GO ou NOGO pour indiquer qu'une erreur s'est produite lors du fonctionnement avec la variable déportée.

Exemple d'utilisation d'une variable déportée

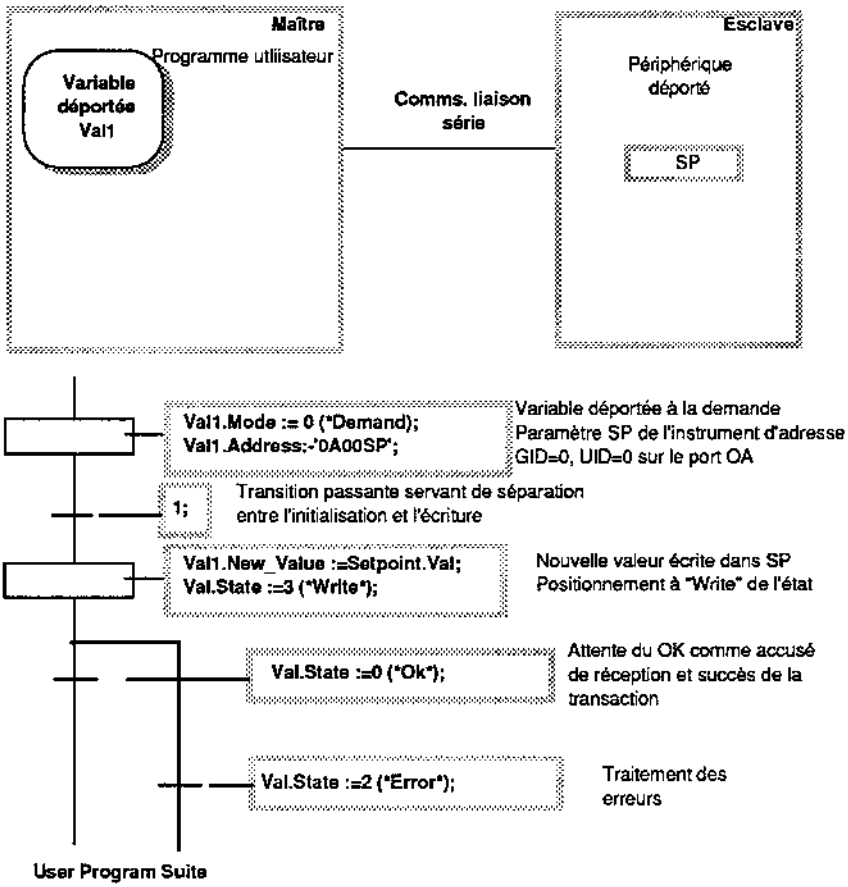


Figure 3-13 Exemple d'utilisation d'une variable déportée

La figure 3-13 montre dans les encadrés grisés une partie du grafcet nécessaire pour écrire une nouvelle valeur de point de consigne pour un appareil déporté. Il est à noter que seule l'adresse est propre à un protocole donné, le reste du code est identique pour les différents protocoles.

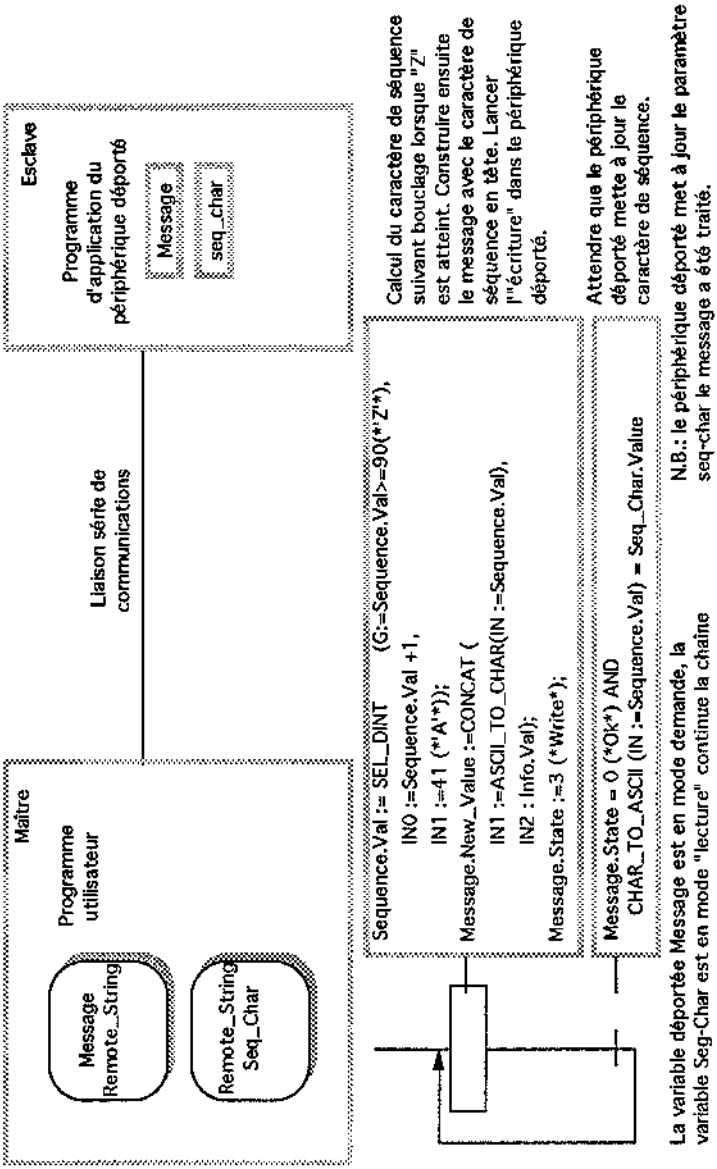


Figure 3-14 Exemple d'accès verrouillé à un périphérique déporté

Accès verrouillé à un périphérique déporté

Avec certaines applications, il est essentiel que chaque message soit reçu par un périphérique déporté, c'est-à-dire qu'il est impossible qu'un message soit écrasé avant d'avoir été traité par le périphérique déporté. La figure 3-14 montre une disposition de ce type : 'Exemple d'accès verrouillé à un périphérique déporté'. Les informations sont regroupées en une chaîne qui est écrite dans un paramètre au sein du périphérique déporté. Un caractère est ajouté en tête de la chaîne comme numéro d'ordre. Chaque nouvelle chaîne transmise a comme préfixe le caractère suivant dans un ordre défini, par exemple A,B,,,X,Y,Z,A,B etc. La série revient à A après avoir atteint Z.

Le PC3000 possède deux variables déportées, une pour écrire la chaîne d'informations et l'autre pour relire le caractère d'ordre mis à jour par le périphérique déporté lorsque le message a été traité. Le caractère d'ordre au sein du périphérique déporté est interrogé par le PC3000. Lorsqu'il correspond au caractère d'ordre utilisé dans la dernière chaîne, les nouvelles informations peuvent être transmises.

Synchronisation la plus défavorable de la variable déportée

En supposant qu'aucune autre transaction de variables déportées ne soit en attente, le temps le plus défavorable entre l'écriture dans la variable déportée, pour lancer une transaction de communications et la détection de l'achèvement de cette transaction est la suivante :

```
= intervalle de la tâche du bloc fonction de variable
                    déportée +
 intervalle de la tâche du bloc fonction de module de
                    communications du port +
durée de la transmission de la liaison série de communications
                    vers le périphérique déporté +
durée de process du périphérique déporté +
durée de la transmission retour de la liaison série  de
                    communication vers le PC3000 +
intervalle de la tâche du bloc fonction de module de
                    communications du port +
intervalle de la tâche du bloc fonction de la variable
                    déportée
```

Par exemple, pour lire une valeur de process (PV) provenant d'un appareil de la série 905 à l'aide d'une variable déportée affectée à une tâche de 20ms, un bloc fonction de communications maître EI Bisync affecté à une tâche de 10ms et utilisant une liaison série tournant à 9600 Bauds aura un temps d'attente le plus défavorable de :

= 20ms +
10ms +
10ms+

2ms +

8ms +

20ms +

10ms+

80ms

Mise en file d'attente de transactions de variables déportées

Il est possible d'amorcer un certain nombre de variables déportées pour effectuer des transactions sur le même port en une courte période de temps. Chaque bloc fonction de module de communications qui prend en charge le mode de fonctionnement maître possède une file d'attente interne qui contient les transactions des variables déportées qui sont en attente. Normalement, un module peut contenir un maximum de 100 transactions en attente mais il faut se reporter à la description propre au bloc fonction de module de communications pour avoir la taille exacte de la file d'attente. Le paramètre de sortie Queue_Space pour le bloc fonction de module de communications peut servir à contrôler la place disponible.

Les transactions sont mises en attente selon le principe "premier arrivé premier servi". Il est à noter qu'il n'existe aucun ordre spécifique de mise en file d'attente pour les transactions pour les variables déportées amorcées dans la même exécution de SFC, par exemple dans le même pas de SFC, et qu'il est possible de mettre en file d'attente une seule transaction à la fois pour une variable déportée donnée.

Les modules de communications série fournis couramment ne peuvent pas superposer simultanément des transactions, par conséquent chaque transaction doit être terminée (c'est-à-dire qu'un message de communications doit être reçu d'un périphérique déporté et transmis à la variable déportée associée) avant que la transaction suivant puisse être lancée ; l'espace de la file d'attente utilisé pour la transaction terminée est libéré.

Si des transactions sont déjà en file d'attente, le délai d'exécution pour une nouvelle transaction de variable déporté comprend le temps nécessaire pour achever toutes les transactions déjà dans la file d'attente.

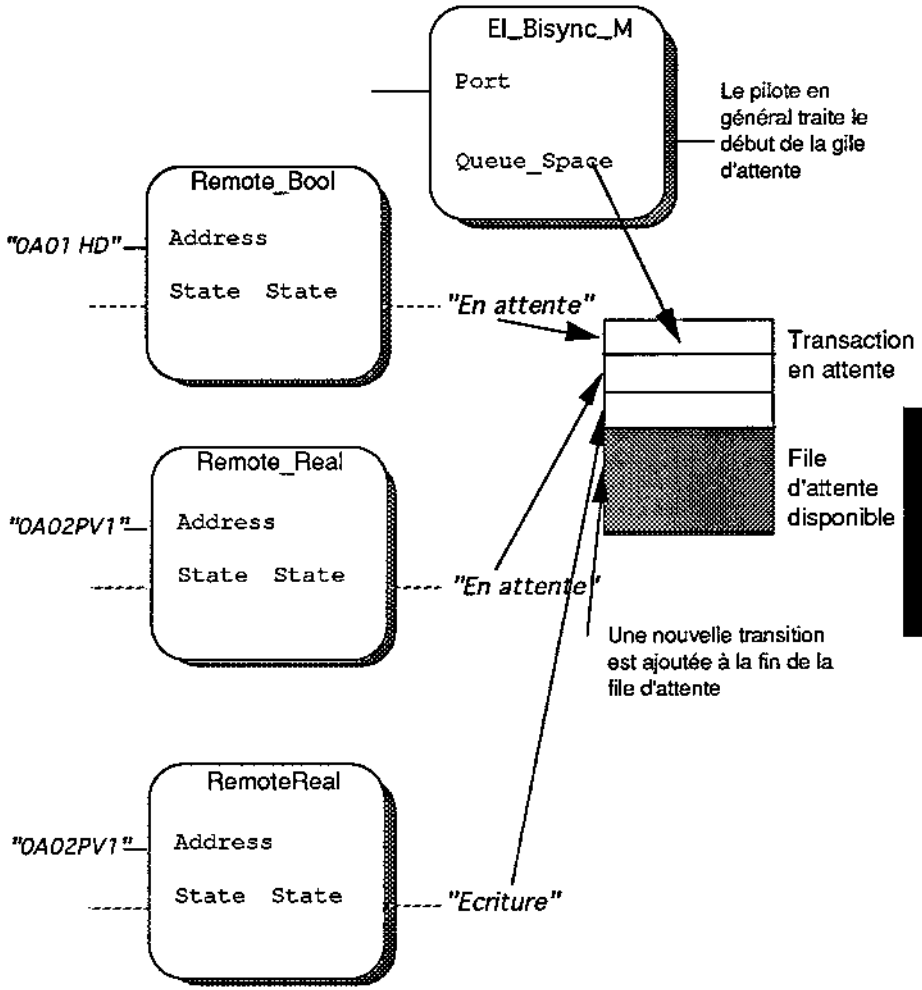


Figure 3-15 Mise en file d'attente des transactions provenant des variables déportées

Problème d'espace dans la file d'attente

S'il n'y a pas d'espace libre dans la file d'attente pour une nouvelle transaction de variable déportée, la variable déportée essaiera de manière répétée d'effectuer la

transaction jusqu'à ce qu'il y ait de l'espace disponible, ce qui augmente le délai d'exécution de la transaction.

Pour détecter les retards de communications occasionnés par cette situation, le programme utilisateur doit surveiller le paramètre Espace dans la file d'attente du module de communications et, s'il y a peu d'espace dans la file d'attente, il doit prendre des mesures pour diminuer le volume de communications sur le port associé, par exemple en allongeant la période de rafraîchissement de la variable déportée pour les lectures ou écritures en continu.

La valeur de la file d'attente peut servir à indiquer des problèmes sur une liaison série donnée ou un périphérique déporté donné. Si l'espace dans la file d'attente diminue rapidement, cela signifie que les transactions de communications prennent plus de temps que d'habitude pour se terminer. Cela peut être dû au fait que certaines erreurs de transmission des communications sont détectées et que certains messages doivent être retransmis ou qu'un périphérique donné ne peut pas répondre, dans ce cas les transactions avec ce périphérique peuvent être annulées par le mécanisme de "time out".

Bloc de communications brutes

Le bloc fonction de module Raw_Comms est destiné aux applications dans lesquelles il est nécessaire de contrôler le port de communications à un niveau élémentaire et qui nécessitent une souplesse permettant de construire ou d'analyser les messages exactement comme ils sont émis ou reçus par l'intermédiaire d'une liaison série. Compte tenu de l'absence de protocole formalisé, le module n'élabore aucun message ou format de message supplémentaire.

Comme pour les autres blocs fonctions de modules de communications, le bloc Raw_Comms peut être affecté à n'importe quel port série à l'aide du paramètre d'entrée, numéro de port (cf. partie 'Affectation d'un bloc fonction de communications à un port').

Raw-Comms offre une vaste gamme de fonctions de niveau élémentaire dont :

- l'accès direct au message émis ou reçu par la liaison série.
- un contrôle indépendant de l'émission et de la réception des messages, comprenant une sélection du débit sur les lignes d'émission et de réception séparée.
- un contrôle du flux de messages à l'aide de Prêt à émettre (Clear to Send, CTS) et Demande pour émettre (RTS). Cf. **Note 1.**
- sélection d'un écho des caractères reçus lorsque cela est nécessaire.
- sélection d'une séquence d'effacement pour la suppression des caractères dans la mémoire tampon de réception.

Le développement de programmes utilisateur destinés à fonctionner avec Raw_Comms est plus complexe que l'utilisation des blocs fonctions avec protocoles car le programme doit manipuler la structure des messages et la synchronisation.

N.B. : il est impossible d'utiliser les variables esclaves ou déportées avec Raw_Comms.

Pour avoir une description complète, se reporter au document 'Module de communications brutes'.

Utilisation de Raws_Comms

Les applications types du bloc fonction de module Raw_Comms sont par exemple :

les communications avec des périphériques utilisant des protocoles simple qui ne sont pas standard,

l'envoi de rapports à des imprimantes série ou à des imprimantes spécialisées, par exemple pour l'impression d'étiquettes.

les communications avec des terminaux orientés caractères comme DEC VT100 ou avec des périphériques d'affichage simples.

Note 1: ces fonctionnalités ne sont actuellement pas prises en charge par les ports LCM ou ICM

Bloc fonction de module Euro-Panel

Un bloc fonction spécial de module de communications esclave est prévu pour gérer l'interface avec un Euro-Panel qui est un afficheur de 2 x 40 caractères avec pavé numérique et touches de fonctions. Le bloc fonction Euro-Panel gère tout le dialogue par messages de communications de niveau élémentaire avec l'afficheur, ce qui permet de programmer facilement des messages propres à l'utilisateur, des champs d'affichage et des champs de saisie de données. Bien que l'Euro-Panel utilise RS422, il est impératif d'affecter un seul afficheur à chaque port RS422.

La structure des messages affichés sur l'Euro-Panel est définie par un jeu de paramètres formatés Il s'agit de paramètres d'entrées auxquels il est possible d'affecter des chaînes de caractères pour définir la structure du message, les champs de saisie, les affichages numériques, etc. à l'aide d'un langage d'affichage simple et structuré appelé Operator Interface Formatting Language (OIFL).

L'afficheur Euro-Panel est le maître de la liaison série RS422. Il est possible d'associer des variables esclaves à un ou plusieurs blocs fonctions Euro-Panel à l'aide des caractères de sélection de protocole 'EP' dans l'adresse esclave (cf. chapitre 4 'Affectation d'adresses et de protocoles aux variables esclaves'). Le reste de l'adresse esclave sert à attribuer un nom unique destiné à être utilisé par le module Euro-Panel dans les chaînes formatées, qui peuvent accepter n'importe quelle chaîne alpha-numérique (y compris le trait de soulignement '_'). La valeur d'une variable esclave affectée à un Euro-Panel est automatiquement affichée sur le tableau, si elle est référencée dans une chaîne formatée.

Exemples d'adresses de variables esclaves utilisées avec l'Euro-Panel

```
'EPpv1'
'EPmaxtemp'
EPpump'
```

Ces adresses peuvent être affichées à l'aide des chaînes de structure attribuées au module Euro-Panel. Exemples :

```
''PV 1 is:', pv1'
''Max temp: ', maxtemp:7.2'
'@10:1, 'Pump', pump (Off,Vacuum,Purge)'
```

Au cours de l'exécution du programme utilisateur, il est possible de modifier dynamiquement les chaînes d'affichage, si bien que des menus et des paramètres peuvent être amenés à l'affichage en réponse à l'utilisation des touches de fonctions du panel.

Consulter la description du bloc fonction 'Module Euro-Panel' pour avoir des détails sur les possibilités et la programmation de l'écran Euro_Panel.

PROBLEMES ET SOLUTIONS

Cette partie répertorie certains problèmes qui surviennent couramment avec les communications série et sont donnés à titre d'indication générale ; cette liste ne doit pas être considérée comme exhaustive. Pour commencer, il faut identifier le n° d'erreur des blocs fonctions de communications associés. Une liste complète des codes d'erreur est fournie à la fin de chaque description de bloc fonction de module de communications.

a) Problème : il est impossible d'établir des communications élémentaires entre un périphérique déporté et le PC3000.

Vérifier ce qui suit :

Le câble est correctement branché, vérifier la polarité des lignes de réception et d'émission. Avec RS422, vérifier que tous les points de branchement sont correctement reliés. Avec RS232, vérifier que le commun est branché et, si besoin est, que les signaux de contrôle de modem qui conviennent sont simulés pour le périphérique (Rebouclage par exemple de RTS sur CTS...).

Le débit et les bits d'arrêt définis pour le module de communications correspondent à ceux attendus par le périphérique déporté.

Le protocole du bloc fonction de communications correspond au périphérique déporté,

Vérifier qu'il existe un seul périphérique maître, pour les protocoles qui nécessitent un seul maître avec les esclaves, c'est-à-dire qui du PC3000 ou du périphérique déporté est le maître.

b) Problème : le périphérique déporté ne peut pas lire ou écrire une variable esclave.

Vérifier ce qui suit :

L'adresse esclave est affectée à un protocole connu

Un bloc fonction de communications pour le protocole esclave sélectionné a été créé,

Le bloc fonction de communications est affecté au même port que celui qui est relié au périphérique déporté,

L'adresse esclave correcte est utilisée par le périphérique maître déporté

Si une écriture dans l'esclave échoue, vérifier que le mode esclave permet les transactions d'écriture et que le bloc fonction de module de communications a un paramètre de protection en écriture comme **Wr_Protect** positionné sur No.

c) Problème : la variable déportée ne peut pas communiquer avec un périphérique déporté.

Vérifier ce qui suit :

La variable déportée accède à un port associé à un bloc fonction de module de communications qui prend en charge un protocole fonctionnant en mode maître,

Le port correct pour le périphérique déporté est adressé,

La chaîne d'adresse correcte donnée est pour le paramètre dans le périphérique déporté. (Avec les périphériques EI Bisync, vérifier que le caractère de canal est fourni, si le périphérique n'utilise pas les numéros de canal, vérifier qu'un caractère "espace" remplace le caractère canal).

Le bon caractère de format a été ajouté à l'adresse pour le paramètre du périphérique déporté.

Chaque périphérique esclave déporté a une adresse esclave unique, par exemple, avec EI Bisync, chaque appareil déporté a un GID différent.

d) Problème : un système de communications peut impliquer le branchement de nombreux appareils en multi-points sur le PC3000, sur une seule liaison série. Au cours de la mise en service, certains des appareils peuvent être absents ou hors tension. Il est fréquemment nécessaire que le programme utilise les variables déportées pour scruter les paramètres dans chacun des appareils. Toutefois, lorsque certains appareils ne sont pas présents sur la liaison série, les transactions avec les appareils manquants prendront beaucoup plus de temps pour s'effectuer car la communication attendra une temporisation puis réessaiera chaque transaction qui a échoué. Cela ralentit considérablement la transaction de communications avec les autres appareils et peut provoquer une diminution de capacité de la file d'attente.

Solution : le programme utilisateur doit tester **Status** et **Error _No** sur chaque variable déportée pour rechercher les "time-out". **Status** sera positionné sur NOGO et **Error_No** sera une valeur différente de zéro fonction du code d'erreur approprié. La valeur du code d'erreur dépend du type de communications utilisé.

En cas de détection d'une transaction de variable déportée qui a dépassé le temps imparti, le temps de **rafraîchissement** pour la répétition des transactions de lecture ou d'écriture avec les appareils manquants doit être allongé. Lorsque Status revient à Go, c'est-à-dire qu'il indique que l'appareil est à nouveau présent, le temps de **rafraîchissement** peut revenir à la valeur normale. On peut parvenir à ce résultat en utilisant le câblage par soft avec le bloc fonction de la variable déportée, comme :

```
remotePV.Refresh:= SEL_TIME (G:= remotePV.Status <> 1 (*
GO *),
                            IN0:= normRate.Val,
                            IN1:= failRate.Val) ;
```

Où remote PV est un bloc fonction de variable déportée et normRate et failRate sont des variables utilisateur contenant les temps de rafraîchissement en mode normal et défaillance.

COMMUNICATIONS

BLOC FONCTION EI_BISYNC_M

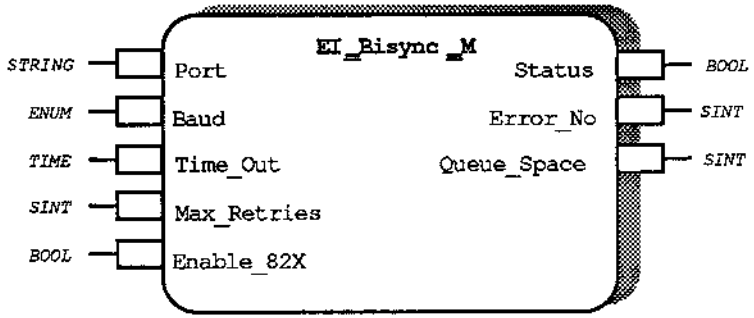


Figure 3-16 Diagramme EI_Bisync_M

Description fonctionnelle

Le bloc fonction EI_Bisync_M prend en charge les communications série sur un port de communications série désigné utilisant le protocole ASCII EI Bisync. Ce bloc fonction configure le port série pour qu'il fonctionne en mode maître. Il n'est normalement pas nécessaire de connaître en détail le protocole Bisync pour utiliser ce bloc fonction. Toutefois, en cas de besoin, il est possible de se reporter au Manuel de communications EI Bisync pour avoir des détails spécifiques. Avant de lire cette description, il est conseillé de se familiariser avec les grandes lignes du système de communications PC3000 en lisant la Présentation des communications PC3000.

Ce bloc fonction sera nécessaire pour concevoir ou programmer le PC3000 de telle façon qu'il utilise le protocole EI Bisync pour qu'il fonctionne en mode maître, c'est-à-dire pour un port de communications série PC3000 relié à un ou plusieurs périphériques esclaves EI Bisync Slave comme des contrôleurs discrets de la série 900.

Attributs du bloc fonction

Type : 8 40

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port Status Queue_Space

Besoin de capacité mémoire : . 2054 octets

Durée d'exécution : ..... 20 µsecs

Le bloc maître EI Bisync (EI_Bisync_M) traite les détails propres au protocole des communications EI Bisync et est pris en charge par les blocs fonctions génériques de variables déportées. Les blocs de variables déportées sont liés au module par une adresse propre au protocole et peuvent demander l'interrogation ou la mise à jours des paramètres d'un appareil de communication par l'intermédiaire d'un port de communications donné.

N.B. : seule la version ASCII du protocole EI Bisync est prise en charge.

Description des paramètres

Paramètres de configuration du module

Le bloc EI_Bisync_M possède plusieurs paramètres de saisie de configuration qui définissent divers aspects du module et doivent être configurés avant le lancement du programme utilisateur. La modification de ces paramètres pendant l'exécution du programme utilisateur n'aura aucun effet sur le module, sauf dans des circonstances particulières (cf. la partie 'Modification temporaire des paramètres de configuration' dans la partie 'Présentation des communications PC3000').

Port

Le paramètre Port est l'adresse à deux caractères du port sur lequel fonctionne le protocole EI Bisync. Le premier caractère est un chiffre compris entre 0 et 5 représentant l'emplacement dans le rack et le deuxième caractère représente le port de cet emplacement. Par exemple, '0C' est le port C sur le LCM et, s'il y avait un ICM dans la fente 3, '3A' pourrait désigner son port supérieur.

Baud

Le paramètre Baud offre un choix de 11 débits différents compris entre 75 Bauds et 115,2 kBauds (comme le représente le tableau 3-1), avec une valeur par défaut de 9600 Bauds. Il faut noter que tous les ports ne peuvent pas prendre en charge tous les débits. Dans ce cas, une erreur est indiquée lorsque le bloc fonction est mis en marche pour la première fois (cf. description d'Error_No).

Time_Out

Le paramètre Time_Out spécifie le temps pendant lequel le maître Bisync doit attendre un message-réponse à une demande émise. Une fois ce temps écoulé, le module estime qu'il y a eu une erreur de transmission et la demande peut être à nouveau émise ou une erreur peut être renvoyée au bloc fonction de la variable à distance qui a émis la demande. Time_Out prend par défaut la valeur 5 secondes.

Valeur d'énumération	Débit
0	75
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	115200

Tableau 3-1 Débits EI_Bisync_M

Max_Retries

Le paramètre Max_Retries spécifie le nombre de nouvelles tentatives d'émission d'une demande en cas de détection d'une erreur de transmission comme un dépassement du délai imparti ou une erreur de total de contrôle. En l'absence de réception d'une réponse valable après ce nombre de nouvelles tentatives, une erreur est renvoyée au bloc de paramètres déportés qui a émis la demande.

Max_Retries prend par défaut la valeur 2, une demande sera donc envoyée trois fois avant qu'une erreur soit signalée et la demande interrompue.

Enable_82X

Le paramètre Enable_82x est configuré pour permettre la communication avec la famille d'appareils 82x (825 par exemple).

Ces appareils nécessitent un bit de stop supplémentaire.

N.B. : ce paramètre doit uniquement être utilisé lorsque des appareils 82x sont reliés au port car il divise la vitesse des communications par 10 .

Paramètres d'état du module

L'état du module est indiqué par trois paramètres de sortie dans le bloc fonction EI_Bisync_M.

Etat

Le paramètre Etat est une indication booléenne de l'état de la liaison contrôlée par le module. Si la liaison n'a aucun problème, ce paramètre indique Go mais,

lorsqu'une erreur se produit, le paramètre Error_No indique la cause du problème.

Error_No

Le paramètre Error_No indique la cause des erreurs de la liaison. Si cette dernière fonctionne correctement, Error_No sera 0 (OK). Pour avoir des détails complets sur les codes d'erreur, consulter la partie consacrée à la signalisation des erreurs.

Queue_Space

Le paramètre Queue_Space indique l'espace restant dans la file d'attente pour les opérations avec les paramètres déportés. Si la valeur est zéro, cela indique que la largeur de bande de la liaison est insuffisante pour faire face au nombre de demandes de variable déportées effectuées et les données seront perdues. Si cette situation se produit, il faut diminuer les fréquences d'interrogation des paramètres.

Fonctionnement avec les variables à distance

Les demandes effectuées par le PC3000 sont contrôlées par un ou plusieurs blocs de variables déportées qui peuvent déclencher des demandes de lecture et d'écriture par l'intermédiaire d'un module qui prend en charge le mode de fonctionnement maître.

Le module EI_Bisync_M prend actuellement en charge les blocs de types Remote_Bool, Remote_Real, Remote_Int, Remote_Time et Remote_Str et Remote_SW.

Adressage

Il est nécessaire de configurer une adresse propre au protocole dans le bloc de la variable à distance, servant à accéder aux périphériques déportés. La figure 2 montre un exemple de structure d'adresse. Le champ du port est défini, comme dans le paramètre Port du bloc fonction, sous la forme d'un numéro d'emplacement suivi d'une lettre pour le port situé dans cet emplacement.

La partie de l'adresse propre au protocole commence par une identité esclave (ID) qui indique l'adresse du périphérique esclave qui doit répondre à une demande. Cette identité est composée de deux caractères, l'identificateur de groupe (Gid) et l'identificateur d'unité (Uid). Le Gid et l'Uid sont compris entre '0' (30h) et 'o' (6Fh).

Ces caractères sont suivis de l'identificateur de canal (Chid), compris entre ESPACE (20h) et ' ' (7Eh). S'il a la valeur ESPACE, cela indique qu'aucun Chid ne doit être envoyé à l'appareil esclave. Vient ensuite un mnémonique à deux caractères (Mn0 Mn1) dont chacun peut être compris entre '!' (21h) et ' ' (7Eh).

Enfin, zéro, un ou plusieurs caractères de structure indiquent la manière dont le champ de données doit être codé. Cela est nécessaire car le standard Ei Bisync

offre un certain nombre de codages différents pour un même type de données. Les formats-caractères multiples sont uniquement utilisés lorsque la variable déportée comporte des éléments multiples ; dans ce cas, tous les champs d'un même type sont codés de la même manière. S'il n'y a aucun format-caractère, c'est le codage par défaut qui est utilisé. Les formats sont définies dans la partie ci-dessous.

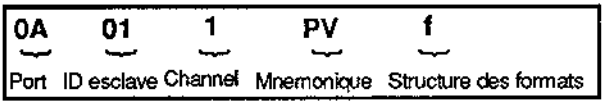


Figure 3-17 Exemple d'adresse de variable déportée

Il est possible de modifier à tout moment le paramètre Adresse d'un bloc fonction de variable à distance, ce qui permet de réutiliser le bloc pour communiquer avec différents appareils et différents paramètres. Il est déconseillé de modifier ce paramètre lorsque le paramètre **Etat** du bloc fonction est 'En attente', c'est-à-dire lorsqu'une demande est en attente.

Formats des données

Cette partie décrit les différents codages de données qui peuvent être sélectionnés pour chaque type élémentaire de données. Le caractère de format qui fait partie de la chaîne d'adresse du bloc de la variable déportée, indique au bloc, le codage qu'il doit utiliser.

Virgule flottante (REEL)

Le bloc fonction Remote_Real donne accès à un paramètre à virgule flottante unique, sur un appareil esclave.

Structure	Description	
P	24-bit IEEE condensé	VALEUR PAR DEFAUT
p	32-bit IEEE condensé†	
F	ASCII libre, 8 caractères maxi.	
f	ASCII libre , 6 caract ères maxi.	
J	ASCII libre, 6 caractères maxi.	Structure flottante TCS, 1er caractère différent de '.' et '-'
0	Structure fixe TCS, XXXX. ou XXXX-	
1	Structure fixe TCS, XXX.X ou XXX-X	
2	Structure fixe TCS, XX.XX ou XX-XX	
3	Structure fixe TCS, X.XXX ou X-XXX	
4	Structure fixe TCS, .XXXX ou -XXXX	
Q	Structure SSD 64 bit s	

Tableau 3-2 Formats à virgule flottante EI_Bisync_M

N.B.: -† Le format 32 bits n'est pas pris en charge actuellement.

Entier

Le bloc fonction Remote_Int donne accès à un paramètre Entier avec signe ou Mot d'état unique sur un appareil esclave. Le bloc fonction Remote_SW comprime/décomprime la valeur provenant de ou destinée à seize paramètre booléens.

Structure	Description	
Z	libres 0 à 8 caractères Hex (32 bits maximum)	VARIABLE PAR DEFAUT
B	8 bits, 2 caractères Hex	
X	16 bits, 4 caractères Hex	
Y	32 bits, 8 caractères Hex	

Tableau 3-3 Structures des entiers EI_Bisync_M

Bool

Le bloc fonction Remote_Bool donne accès à un paramètre booléen unique sur un appareil esclave.

Il n'existe aucun format sélectionnable par l'utilisateur pour ce type. Il est envoyé sous la forme d'un entier de 'format libre' possédant une valeur 0 pour faux et 1 pour vrai.

Time

Le bloc fonction Remote_Time donne accès à un paramètre Durée unique (TIME) sur un appareil esclave.

Il n'existe aucun format sélectionnable par l'utilisateur pour ce type. Il est envoyé sous la forme d'un entier de 'structure libre'. La valeur représente le nombre de millisecondes.

Chaîne

Le bloc fonction Remote_Str donne accès à un paramètre Chaîne unique sur un appareil esclave.

Structure	Description	
S	Standard	VALEUR PAR DEFAUT
r	Brut	

Tableau 3-4 Structures de chaînes EI_Bisync_M

Le codage standard est composé de la chaîne précédée d'une apostrophe (60h). Tous les caractères non imprimables sont remplacés par le caractère "escape" de code (1Bh) suivi d'un nombre hexadécimal à deux chiffres qui représente le code ASCII du caractère. Un caractère n'est pas imprimable si son code ASCII est inférieur à 20h ou supérieur à 7eh.

Le codage brut implique l'absence de codage. La chaîne est envoyée sans modification. Il s'agit de la structure normalement utilisée uniquement pour la mise au point des communications mais elle peut servir à accéder aux paramètres d'un appareil non standard ou pour envoyer des paramètres de données composites pour lesquels il n'existe aucun type de bloc fonction de variable déportée type.

Données composites

Le module peut traiter les paramètres de données composites mais, pour l'instant, aucun bloc de variable déportée ne les prend en charge.

Messages multi-blocs

Les messages multi-blocs ne sont pris en charge que de manière limitée dans ce module.

Un message reçu est accepté s'il a la structure multi-blocs mais un seul bloc est accepté. Dans ce cas, le message commence par le caractère SOH ('05h') et finit par le caractère ETX ('03h).

Aucun message émis n'est multi-blocs.

Exemple

Cet exemple montre les blocs fonctions nécessaires pour accéder aux paramètres Variable de process PV et Mot d'état SW de trois appareils, deux étant reliés au port A du LCM et l'autre au port B du LCM.

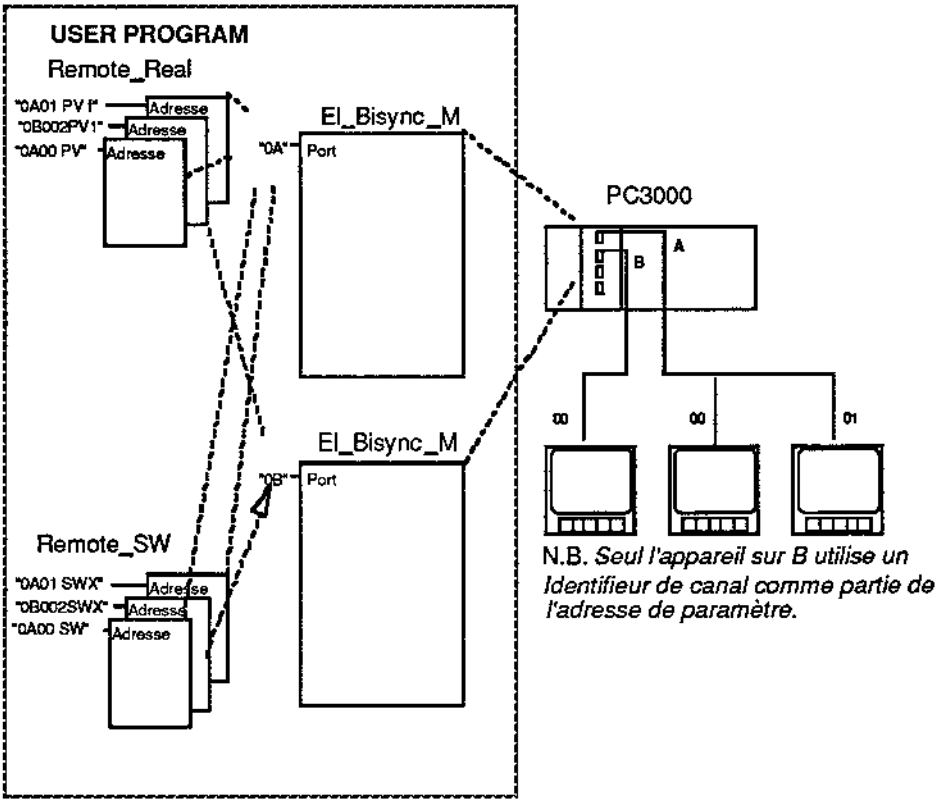


Figure 3-18 Exemple d'utilisation EI_Bisync_M

Erreurs

S'il y a une erreur propre au bloc fonction EI_Bisync_M ou au driver EI Bisync, l'erreur est signalée par l'intermédiaire du paramètre **Error_No** du bloc fonction EI_Bisync_M. Les erreurs liées à une variable déportée donnée sont signalées par l'intermédiaire du bloc de variable déportée associé.

Signalisation des erreurs

Erreurs de bloc fonction

Les erreurs suivantes sont signalées par le bloc EI_Bisync_M, par l'intermédiaire du paramètre Error_No.

Error_No	Description de l"erreur
1	ERREUR DE PORT : ABSENCE D'ADRESSE Il y a moins de deux caractères dans le paramètre Port du bloc fonction.
5,6	ERREUR DE PORT : DEBIT PAS DISPONIBLE Le débit demandé n'est pas disponible sur ce port série.
11	ERREUR DE PORT : D'EMPLCAMENT IINTERDIT Le numéro d'emplacement sélectionné n'est pas autorisé. Le numéro d'emplacement est le premier caractère du paramètre Port et doit être compris entre '0' et '5'.
12	ERREUR DE PORT : PORT INTERDIT Le n° de port sélectionné n'est pas autorisé. Le 2ème caractère du paramètre Port doit être compris entre "A" et "C" pour 1 LCM ou entre A et D pour ICM.
17	ERREUR DE PORT : PORT UTILISE Le port sélectionné est déjà utilisé par un autre bloc de comm.

Tableau 3-5 Codes d'erreur EI_Bisync_M

Codes d'erreur de variables déportées

Ces erreurs sont signalées par l'intermédiaire d'**Error_No** du bloc de variables déportées.

Error_No	Description de l'erreur
11	ERREUR D'ADRESSE : EMPLACEMENT INTERDIT Le premier caractère du paramètre Adresse n'est pas compris dans la plage valable entre '0' et '5'.
12	ERREUR D'ADRESSE : PORT INTERDIT Le deuxième caractère du paramètre Adresse n'est pas compris dans la plage valable de 'A' à 'C' pour un port LCM ou de 'A' à 'D' pour un port ICM.
16	ERREUR D'ADRESSE : AUCUN PARAMETRE DEPORTE EN SERVICE Aucun bloc de comm. maître correct n'est affecté au port figurant dans le paramètre Adresse.
50	ERREUR DE MODULE : INITIALISATION IMPOSSIBLE L'initialisation du module de communications est impossible, le débit demandé étant par exemple pas disponible sur le port sélectionné. Le paramètre Error_No du bloc El_Bisync_M donne le motif de cette impossibilité.
60	ERREUR DE COMMUNICATIONS : ERREUR DE CHECKSOMME La réponse à une demande de lecture de paramètre a eu une erreur de checksomme.
61	ERREUR DE COMMUNICATIONS : TIME-OUT Aucune réponse à une demande n'a été reçue dans la période de time-out. L'appareil esclave a eu une défaillance ou les demandes sont tellement altérées que l'esclave ne reconnaît pas sa propre adresse.
62	ERREUR DE DONNEES : NOMBRE DE CARACTERES TROP IMPORTANT La réponse à une demande de lecture de paramètre contenait trop de caractères. Cela peut être dû à une erreur de transmission.
63	ERREUR D'ADRESSE : CHAINE TROP COURTE La chaîne d'adresse contenue dans le paramètre Adresse est trop courte pour être valable.
64	ERREUR DE DONNEES : HEX TROP LONG La réponse à une demande de lecture d'un paramètre entier avait une valeur codée avec plus de 8 caractères hexadécimaux.
65	ERREUR DE DONNEES : CARACTERE PAS HEXADECIMAL La réponse à une demande de lecture d'un paramètre entier avait une valeur codée avec un caractère pas hexadécimal.
66	ERREUR DE DRIVER: TYPE DE PARAMETRE PAS VALABLE Le type de paramètre/champ utilisé par le bloc de variables déportées n'est pas encore pris en charge par le driver.

Tableau 3-6 Codes d'erreur des variables à distance

Error_No	Description de l'erreur
67	ERREUR DE DONNEES : VALEUR NON BOOLEENNE La réponse à une demande de lecture d'un paramètre booléen avait une valeur différente de 0 et de 1.
68	ERREUR DE DONNEES : CHAMPS TROP NOMBREUX La réponse à une demande de lecture d'un paramètre multi-éléments contenait un nombre de champs supérieur à ce qui était attendu.
69	ERREUR DE DONNEES : VALEUR TROP GRANDE Lors du codage d'une valeur pour sa transmission, il est apparu que cette valeur était trop grande pour être codée à l'aide de la structure spécifiée. Par exemple, une valeur REELLE supérieure à 9999,0 ne peut pas être représentée à l'aide d'une des structures TCS.
70	ERREUR DE DONNEES : MESSAGE TROP LONG Lors du codage d'un message pour sa transmission, le module a dépassé la taille maximale de la mémoire tampon. Cela peut uniquement se produire avec les paramètres multi-éléments.
71	ERREUR DE DONNEES : CHAINE TROP LONGUE La réponse à une demande de lecture d'un paramètre CHAINE contenait un nombre de caractères supérieur aux capacités du bloc de variable déportées.
72	ERREUR DE DONNEES : CARACTERE PAS VALABLE La réponse à une demande de lecture d'un paramètre CHAINE contenait un caractère non imprimable qui n'avait pas été converti en caractère "escape".
73	ERREUR DE DONNEES : LONGUEUR DE CHAMP PAS VALABLE La réponse à une demande de lecture de paramètre contenait des données d'une longueur incorrecte. Par exemple, un REEL codé à l'aide de la structure SSD ('Q') doit contenir exactement 16 caractères hexadécimaux.
100	ERREUR DE DONNEES NAK La réponse à une demande d'écriture de paramètre était un accusé de réception négatif. Cela peut être dû à des mnémoniques de paramètres incorrects, à une écriture sur un paramètre en lecture seule ou à des données situées en dehors de la plage normale. La valeur du paramètre EE de l'appareil esclave doit indiquer le motif.
101	ERREUR DE COMMUNICATIONS : NI ACK NI NAK La réponse à une demande d'écriture de paramètre n'était ni un accusé de réception positif ni un accusé de réception négatif. Cela indique normalement que la réponse a été altérée, peut-être par deux esclaves ayant la même adresse.
103	ERREUR DE DONNEES : EOT La réponse à une demande de lecture de paramètre était un caractère EOT. Ce caractère indique que l'esclave a rejeté la demande. Cela peut être dû à des mnémoniques de paramètre incorrects ou à la lecture d'un paramètre en écriture seule. La valeur du paramètre EE de l'appareil esclave doit indiquer le motif.

Tableau 3-6 Codes d'erreur des variables déportées (suite)

Error_No	Description de l'erreur
104	ERREUR DE COMMUNICATIONS : VALEUR DIFFERENTE DE STX La réponse à une demande de lecture d'un paramètre ne commençait pas par un caractère STX. Cela indique normalement que la réponse a été altérée, peut-être par deux esclaves qui possédaient la même adresse.
105	ERREUR D'ADRESSE : GID PAS VALABLE Le caractère d'identification de groupe dans la chaîne d'adresse ne se situe pas dans la plage valable (3ème caractère).
106	ERREUR D'ADRESSE : UID PAS VALABLE Le caractère d'identification d'unité dans la chaîne d'adresse ne se situe pas dans la plage valable (4ème caractère).
107	ERREUR D'ADRESSE : CHID PAS VALABLE Le caractère d'identification de canal dans la chaîne d'adresse ne se situe pas dans la plage valable (5ème caractère).
108	ERREUR D'ADRESSE : MNEMONIQUE PAS VALABLE Les caractères de mnémonique dans la chaîne d'adresse ne se situent pas dans la plage valable (6ème et 7ème caractères).
109	ERREUR DE DONNEES : CARACTERE NON CONDENSE La réponse à une demande de lecture d'un paramètre REEL codé à l'aide de la structure condensée IEEE ('P') contenait un caractère pas valable.
110	ERREUR DE DONNEES : VALEUR DIFFERENTE DE RS La réponse à une demande de lecture pour un paramètre multi-éléments à 2 niveaux n'avait pas de caractère RS au début des données.
111	ERREUR DE DONNEES : STRUCTURE TROP PROFONDE Le module prend actuellement en charge des paramètres multi-éléments sur un maximum de 2 niveaux.
112	ERREUR INTERNE : STRUCTURE VIDE Le bloc de la variable déportée n'a demandé aucune transmission !
113	ERREUR DE COMMUNICATIONS : DISCORDANCE DE MNEMONIQUE Les mnémoniques restitués dans la réponse à une demande de paramètre de lecture sont différents de ceux contenus dans la demande.
114	ERREUR DE DONNEES : GT ATTENDU La réponse à une demande de lecture d'un paramètre Entier ou Mot d'état ne commençait pas par un '>' au début des données.
115	ERREUR DE DONNEES : APOSTROPHE ATTENDUE La réponse à une demande de lecture de paramètre Chaîne à l'aide du codage standard n'avait pas d'apostrophe au début des données.

Tableau 3-6 Codes d'erreur des variables déportées (suite)

Error_No	Description de l'erreur
201	ERREUR DE COMMUNICATIONS : SURCHARGE DE RECEPTION Une erreur de surcharge a été détectée sur un caractère reçu.
202	ERREUR DE COMMUNICATIONS : PARITE DE RECEPTION Une erreur de parité a été détectée sur un caractère reçu.
203	ERREUR DE COMMUNICATIONS : PARITE ET SURCHARGE Une erreur de parité et de surcharge a été détectée au cours de la réception.
204	ERREUR DE COMMUNICATIONS : ERREUR DE TRAME Une erreur de trame été détectée sur un caractère reçu.
205	ERREUR DE COMMUNICATIONS : TRAME ET SURCHARGE Une erreur de trame et de surcharge a été détectée au cours de la réception.
206	ERREUR DE COMMUNICATIONS : TRAME ET PARITE DE RECEPTION Une erreur de trame, de parité a été détectée au cours de la réception.
207	ERREUR DE COMMUNICATIONS : TRAME, SURCHARGE ET PARITE DE RECEPTION Une erreur de trame de parité et de surcharge de capacité a été détectée au cours de la réception.
208-215	MODIFICATION DE L'ETAT D'INTERRUPTION Un état d'interruption a été détecté ou supprimé La ligne a été déconnectée Le débit du périphérique à déporté est trop faible

Tableau 3-6 Codes d'erreur des variables déportées (suite)

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	**STR ING**	'0A'	Oper	Config		
Baud	**ENUM**	_9600	Oper	Config	Numération	75(0) 300(1) 600(2) 1200(3) 2400(4) 4800(5) 9600(6) 19200(7) 38400(8) 57600(9) 115200(10)
Time_Out	**TIME**	5s	Oper	Super	Lim. haute Lim. basse	24jours 100ms
Max_Retries	**SINT**	2	Oper	Super	Lim. haute Lim. basse	1000 0
Enable_82X	**BOOL**	No	Config	Config	Sens	Non (0) Oui(1)
Status	**BOOL**	NoGo	Oper.	Bloc	Sens	NOGO(0) Go(1)
Error_No	**SINT**	0	Oper	Bloc	Lim. haute Lim. basse	255 0
Queue_Space	**SINT**	0	Oper	Bloc	Lim. haute Lim. basse	255 0

Tableau 3-7 Attributs des paramètres EI_Bisync_M

BLOC FONCTION EI_BISYNC_S

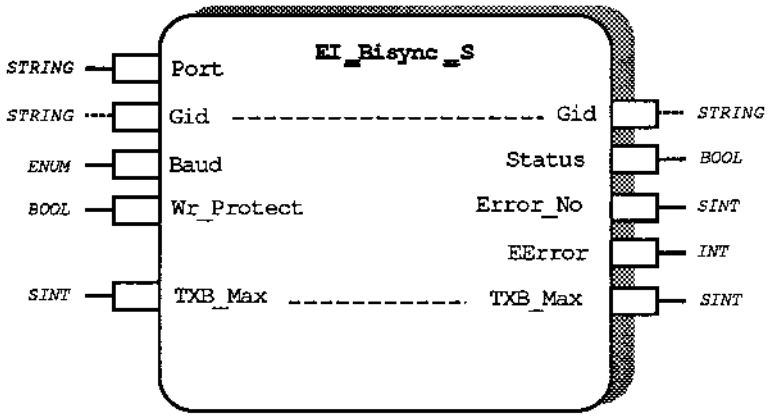


Figure 3-19 Diagramme de bloc fonction EI_Bisync_S

Description fonctionnelle

Le bloc fonction EI_Bisync_S prend en charge les communications sur un port de communications série désigné avec le protocole ASCII EI Bisync et configure le port série pour qu'il fonctionne en mode esclave. Il n'est normalement pas nécessaire de connaître en détail le protocole Bisync pour utiliser ce bloc fonction. Toutefois, il est possible de se reporter au manuel de communications EI Bisync pour avoir des détails spécifiques si besoin est. Avant de lire cette description, il est conseillé de se familiariser dans les grandes lignes avec le système de communications PC3000 en lisant la présentation générale des communications PC3000.

L'utilisation des variables esclaves avec le bloc fonction EI_Bisync_S et l'adressage associé sont décrits en détail dans la description de ce bloc fonction. Etant donné que le protocole par défaut adopté pour les communications PC3000 se comporte comme un esclave EI Bisync, les détails du protocole par défaut sont également définis.

Ce bloc fonction est nécessaire lorsqu'on conçoit ou programme le PC3000 pour qu'il utilise une interface EI Bisync fonctionnant en mode esclave, c'est-à-dire lorsque le PC3000 est relié à un périphérique maître EI Bisync par l'intermédiaire d'une liaison série. Les périphériques maîtres types sont le système de supervision (ESP) et le Production Orchestrator (PO) d'Eurotherm.

> **N.B. :** quelques caractéristiques nouvelles décrites dans ce document ne sont pas prises en charge dans la version actuelle de ce bloc fonction ; elles sont énumérées dans l'annexe.

Le protocole de communications bisynchrones Eurotherm (EI Bisync) est utilisé sur le PC3000, avec le PC3000 agissant comme esclave, à l'aide d'un bloc fonction de périphérique appelé EI_Bisync_S.

Le bloc de périphérique traite des détails des communications propres au protocole et est pris en charge par les blocs fonctions génériques de variables esclaves. Ces blocs sont liés au périphérique par une adresse propre au protocole et définissent les valeurs qui peuvent être lues ou écrites par un périphérique déporté à l'aide du type de protocole spécifié dans le paramètre Adresse, dans le cas présent `EB' pour EI Bisync.

Outre la fourniture d'un accès aux variables esclaves, le module offre un accès direct aux paramètres système et aux paramètres de bloc fonction à l'aide des adresses de communications réservées.

Les adresses des paramètres du bloc fonction peuvent changer à chaque fois que le programme utilisateur PC3000 est reconstruit, de telle sorte que les périphériques maîtres EI Bisync puissent importer un fichier d'adresses de paramètres de bloc fonction pour accélérer les modifications des tables d'adressage des paramètres dans le périphérique maître. ESP est un exemple de périphériques maîtres EI Bisync qui utilisent les adresses de paramètres de blocs fonctions automatiquement produites par la station de programmation dans les fichiers d'adresses. Lorsqu'un programme utilisateur est élaboré, l'utilisation peut en option créer un fichier .GAT qui contient les paramètres de bloc fonction sous la forme d'une liste de portes ESP. Pour comprendre la manière dont ces fichiers d'adresses sont utilisés dans ces produits, consulter les manuels ESP.

> **N.B. :** les ports LCM A,B et C fournissent un protocole esclave EI Bisync par défaut, si aucun bloc fonction de périphérique de communications ne leur est affecté.

Bien que le protocole par défaut puisse être utilisé pour les communications EI Bisync avec un périphérique maître, il est conseillé d'affecter un bloc fonction esclave EI_Bisync_S à ces ports lorsqu'ils sont utilisés. Cela inclut le port A qui est normalement utilisé pour la station de programmation PC3000.

L'utilisation d'un bloc fonction de module de communications EI_Bisync_S à la place des communications par défaut présente les avantages suivants :

Le bloc fonction EI_Bisync_S fournit des informations comprenant les codes d'erreur.

Le listing (ST) du programme utilisateur peut montrer l'affectation de l'ensemble des ports en répertoriant les blocs fonctions de modules de communications.

Il est possible d'utiliser les variables esclaves. Les communications par défaut ne peuvent accéder à aucune variable esclave EI Bisync.

Les caractéristiques des communications comme le **débit** peuvent être modifiées à l'aide du bloc fonction de module de communications.

> **N.B. :** un bloc fonction de module de communications EI_Bisync_S n'exerce un contrôle sur un port affecté que lorsque le programme utilisateur est en cours d'exécution, c'est-à-dire lorsque la station de programmation PC3000 affiche le mode PC3000 comme étant 'En marche'. Dans tous les autres modes, le PC3000 revient toujours aux communications par défaut qui offrent les protocoles esclaves EI Bisync sur les ports A,B et C du LCM. Par exemple, un bloc fonction de communications EI_Bisync_S affecté au port C pourrait fixer le débit sur 19200 Bauds. Le port fonctionne à 19200 Bauds uniquement lorsque le programme utilisateur s'exécute ; toutes les fois que le programme utilisateur cesse son exécution, le port revient au protocole esclave par défaut EI Bisync fonctionnant à 9600 Bauds.

Attributs du bloc fonction

Durée d'exécution : ..... 12 µsec

Type : 8 50

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... état du port Wr_Protect Gid

Besoins de capacités mémoire : ..... 2466 octets

Description des paramètres

Paramètres de configuration du bloc de communication

Le bloc EI_Bisync_S possède plusieurs paramètres d'entrée qui définissent différents aspects du module et doivent être configurés avant le lancement du programme utilisateur. La modification de ces paramètres pendant l'exécution du programme utilisateur n'a aucun effet sur le bloc, sauf dans des circonstances particulières : cf. la partie 'Modification temporaire des paramètres de configuration' dans 'Présentation des communications PC3000'. La seule entrée qui peut agir sur le bloc lorsqu'il est en marche est **Wr_Protect.**

Port

Le paramètre Port est l'adresse à deux caractères du port sur lequel doit fonctionner le protocole Bisync. Le premier caractère est un chiffre compris entre 0 et 5 qui représente l'emplacement dans le rack et le deuxième caractère

est une lettre représentant le port dans cet emplacement, c'est-à-dire que '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' pourrait désigner son propre port.

Baud

Le paramètre Baud offre un choix de 11 débits différents compris entre 75 Bauds et 115,2 kBauds (comme l'indique le tableau 3-9) avec une valeur par défaut de 9600 Bauds. Il faut noter que tous les ports ne peuvent pas prendre en charge tous les débits. Si un port ne peut pas prendre en charge un débit, une erreur sera indiquée lors de la première activation du bloc fonction (cf. la description d'**Error_No**).

Valeur énumérée	Débit
0	75
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	115200

Tableau 3-8 Débits El_Bisync_S

Gid

Le paramètre Gid indique l'identificateur de groupe Bisync auquel le pc3000 répondra par l'intermédiaire de la liaison de communications. Le PC3000 répondra à tous les identificateurs d'unité dans ce groupe.

La gamme de Gid est comprise entre '0' (30h) et 'o' (6Fh).

Si le Gid est vide (chaîne nulle), un identificateur sélectionné par le matériel est utilisé. Il est défini par des cavaliers sur le LCM ou par un commutateur rotatif sur un ICM dépendant du module sur lequel se trouve le port.

LCM version 1

Liaisons LCM (LK12 LK11 LK2 LK1)	Commutateur ICM	GID
1000	0	'0'
1001	1	'1'
1010	2	'2'
1011	3	'3'
1100	4	'4'
1101	5	'5'
1110	6	'6'
1111	7	'7'
0000	8	'8'
0001	9	'9'
0010		'A'
0011		'B'
0100		'C'
0101		'D'
0110		'E'
0111		'F'

Tableau 3-9 GID sélectionnés par le matériel

N.B. : les LCM sont livrés avec l'ensemble des liaisons en place. Le GID par défaut paramétré en usine est égal à '7'. Sur le LCM version 2, le commutateur rotatif correspond directement au GID.

Wr_Protect

Lorsqu'il est configuré, le paramètre Wr_Protect inhibe toutes les écritures arrivant par ce port. Si une opération d'écriture est tentée alors que la protection en écriture est active, le module émet une réponse indiquant une erreur.

TXB_Max

Le paramètre TXB_Max contrôle la longueur maximale d'un bloc de transmission. S'il est positionné sur zéro, la fonction multi-blocs est désactivée et les messages dont la longueur de champ de données est supérieure à 255 octets peuvent être transmis. S'il est différent de zéro, tout message qui dépasserait le maximum serait scindé en plusieurs blocs.

La valeur peut être lue ou écrite par l'intermédiaire de la liaison de communications avec le paramètre BL.

Paramètres d'état du module

L'état du module est indiqué par trois paramètres de sortie dans le bloc fonction EI_Bisync_S.

Etat

Le paramètre Etat est une indication booléenne de la liaison contrôlée par le module. En l'absence de problème avec la liaison, ce paramètre indique Go mais, en cas d'erreur, il passe sur NOGO. Si l'état est sur NOGO, le paramètre **Error_No** indique la cause du problème.

Error_No

Le paramètre Error_No indique la cause des erreurs survenues sur la liaison. Si le paramètre Status est sur Go, Error_No est 0 (OK). Pour avoir des détails complets sur les codes d'erreur, consulter la partie relative à la signalisation des erreurs.

EError

Le paramètre EError indique normalement la cause du rejet de la dernière transaction arrivée. Du fait que le PC3000 fonctionne comme esclave, il indique les transactions de lecture arrivées rejetées en renvoyant un caractère EOT ou NAK pour une transaction d'écriture rejetée. Toutefois, dans certains cas, le PC3000 ne répond pas lorsqu'il reçoit une transaction altérée avec des erreurs de parité par exemple et le paramètre EError n'est pas modifié.

La valeur est accessible par la liaison de communications, par lecture du paramètre système standard EE.

La valeur reste identique jusqu'au rejet d'une autre transaction.

Pour avoir des détails complets sur les codes d'erreur EE, consulter la partie relative à la signalisation des erreurs.

Adressage des paramètres

Le Gid sélectionne un PC3000 donné sur une liaison série multi-points ; dans un PC3000, il y a trois domaines de paramètres différents qui sont accessibles à l'aide de l'Uid, de l'identité de canal et du mnémonique EI Bisync. Ces domaines sont les paramètres système, les variables esclaves et les paramètres de blocs fonctions.

Une adresse de paramètre EI Bisync est composée d'une identité d'unité (Uid), d'une identité de canal (Chid) et d'un mnémonique à deux caractères (Mn0 Mn1). Le tableau 11 montre les composants d'adresse qui définissent le mappage vers les différents domaines :

Uid	Mn0	Domaine de paramètre
Valeur quelconque	Différente '0' à '9'	Système
'0'	'0' à'9'	Esclave
Différente de'0'	'0' à '9'	Bloc fonction

Tableau 3-10 Mappage de domaine

Les mnémoniques système ne nécessitent pas une Uid particulière ; un périphérique maître externe peut utiliser n'importe quelle valeur. Tous les mnémoniques système ont un caractère alphabétique pour le premier caractère, 'EE','MN' par exemple. La partie Mnémoniques système Bisync PC3000 donne une liste complète.

Les variables esclaves reçoivent toujours une Uid '0' et doivent recevoir des mnémoniques dans l'adresse de variable esclave commençant par un caractère numérique, '1A', '4B', '90' par exemple.

La station de programmation PC3000 attribue aux paramètres de blocs fonctions une Uid, des identités de canaux et des mnémoniques lorsqu'un programme utilisateur est créé. L'Uid et l'identité de canal servent à accéder à un emplacement de bloc fonction donné, le mnémonique définit le paramètre dans l'emplacement. Les affectations sont données dans les fichiers .CEL et .GAT qui peuvent être créés lors de la consitution d'un programme utilisateur, pour faciliter l'intégration ESP.

Adressage pour ESP

Toutes les fois qu'un PC3000 reçoit une adresse avec une Uid dans la plage '0' (30h) à 'O' (4Fh), les données de paramètres sont codées ou décodées comme étant normales. Toutefois, si le caractère Uid est décalé par ajout de 20h à sa position dans les codes ASCII, de telle sorte qu'il soit dans la plage 'P' (50h) à 'o'(6Fh), les données sont codées/décodées pour l'ensemble de supervision Eurotherm (ESP). Par exemple, si l'Uid normale='1', le fait d'utiliser Uid=Q' provoque l'utilisation du codage/décodage des données ESP.

Toutes les valeurs d'Uid des définitions de porte ESP données dans le fichier .GAT produit par la station de programmation PC3000 sont décalées de 20h pour garantir l'utilisation du codage approprié.

Identités de canaux

Une identité de canal est normalement nécessaire pour toutes les transactions, sauf lors de la lecture de paramètres système, lorsqu'une identité de canal est facultative et n'est pas prise en compte, ce qui signifie qu'il est possible de lire un mnémonique système avec ou sans identité de canal. Toutefois, il faut noter qu'une identité de canal est nécessaire lors de l'écriture dans les paramètres système.

Variables esclaves

L'utilisateur peut associer des emplacements de blocs fonctions de variables esclaves à ce module en faisant commencer la chaîne d'adresse d'un emplacement par 'EB'. Tout emplacement d'un bloc EI_Bisync_S peut ensuite lire ou écrire le paramètre Valeur du bloc de variables esclaves et est donc accessible par plusieurs ports de communications.

Ces paramètres ne sont accessibles que lorsque le PC3000 fonctionne en mode MARCHE.

L'adresse est saisie sous la forme 'EB<Chid> < Mn0 Mn1>'. Lorsque Chid peut être compris entre '!' (21h) et ' ~' (7Eh), Mn0 peut être compris entre '0' (30h) et '9' (39h) et Mn1 est compris entre '0' (30h) et ' ~' (7Eh). Il faut noter que Mn0 est limité à des caractères numériques pour éviter que l'adresse entre en conflit avec des paramètres système prédéfinis. Les listes de caractères acceptables pour les adresses de variables esclaves figurent dans la partie Plages d'adresses des variables esclaves.

L'identité de canal et Mn1 peuvent utiliser des caractères numériques et alphabétiques mais les plages peuvent être étendues à certains caractères spéciaux si besoin est.

Exemples d'adresses de variables esclaves valables :

'EB110'

'EB111'

'EBA6B'

'EBZ99'

> **N.B. :** les variables esclaves affectées au protocole esclave EI Bisync utilisant le sélecteur de protocole 'EB' sont accessibles depuis n'importe quel port auquel a été affecté un bloc fonction de module de communications EI_Bisync_S. Toutefois, bien que les ports A, B et C du LCM fonctionnent par défaut comme esclaves EI Bisync lorsqu'aucun bloc fonction de module de communications ne leur est attribué, l'accès aux variables esclaves à l'aide du protocole par défaut n'est pas possible.

La 'Présentation des communications CP3000' fournit une description complète de l'utilisation des variables esclaves'.

Multi-éléments

Si le bloc de la variable esclave est multi-éléments, par exemple Slave_Real_8, l'adresse désigne un paramètre composite contenant la totalité des éléments. Pour accéder aux différents éléments, on applique un décalage à Mn1. Par

exemple, pour Slave_Real_8 avec une adresse 'EBx90', le PC3000 répond à 18 adresses indiquées ci-après :

GID	Valeur	Codage
GID 0x90	paramètre composite	Codage standard
GID 0x91	Value_1	Codage standard
GID 0x92	Value_2	Codage standard
GID 0x93	Value_3	Codage standard
GID 0x94	Value_4	Codage standard
GID 0x95	Value_5	Codage standard
GID 0x96	Value_6	Codage standard
GID 0x97	Value_7	Codage standard
GID 0x98	Value_8	Codage standard
GID Px90	paramètre composite	Codage ESP
GID Px91	Value_1	Codage ESP
GID Px92	Value_2	Codage ESP
GID Px93	Value_3	Codage ESP
GID Px94	Value_4	Codage ESP
GID Px95	Value_5	Codage ESP
GID Px96	Value_6	Codage ESP
GID Px97	Value_7	Codage ESP
GID Px98	Value_8	Codage ESP

Tableau 3-11 Adressage multi-éléments

Plusieurs variables esclaves peuvent être lues sous la forme d'un paramètre composite si l'on spécifie un Mn1 '*' par l'intermédiaire de la liaison de communications. Par exemple, s'il y avait trois variables esclaves 'EBy11', 'EBy12' et 'EBy15', elles pourraient être lues ou écrites sous la forme de données composées à l'aide de l'adresse GID 0y1* ou GID Py1*. Si un esclave multi-éléments est inclus, un paramètre composite à deux niveaux est produit.

Paramètres de blocs fonctions

Il est possible d'accéder à n'importe quel paramètre de bloc fonction si son adresse est connue. Cette adresse est normalement lue à partir des fichiers .CEL ou .GAT produits par la station de programmation.

Les paramètres de blocs fonctions sont uniquement accessibles lorsque le PC3000 contient un programme utilisateur valable.

Un paramètre dans le programme utilisateur fait toujours partie d'un emplacement d'un bloc fonction. Les paramètres dans un emplacement sont

numérotés de 1 à 790 et les emplacements dans un programme utilisateur sont numérotés de 1 à 2914.

Mappage standard d'adresse

Pour le codage standard de données, le mappage du numéro d'emplacement de bloc fonction et du numéro de paramètre dans l'emplacement vers les adresses EI Bisync est le suivant :

UID = (emplacement / 5Eh) + 31h

CHID = (emplacement % 5Eh) + 21h

Mn0 = (paramètre / 4Fh) + 30h

Mn1 = (paramètre % 4Fh) + 30h

où % représente (l'opérateur Modulo)?

On obtient ainsi la plage :

UID ..... '1' (30h) à 'O' (4Fh)

CHID ..... '!' (21h) à ' ' (7Eh)

Mn0 ..... '0' (30h) à '9' (39h)

Mn1 ..... '0' (30h) à ' ' (7Eh)

Mappage ESP d'adresse

Pour le codage ESP de données, le mappage du numéro d'emplacement et du numéro de paramètre dans l'emplacement vers les adresses EI Bisync est identique à celui du codage standard, à la différence près que l'on ajoute 20h à l'Uid, c'est-à-dire :

UID = (emplacement / 5Eh) + 51h

On obtient ainsi la gamme :

UID 'R' (30h) à 'o' (4Fh)

Multi-paramètres

Pour accéder à l'ensemble des paramètres d'un emplacement de bloc fonction sous la forme d'un paramètre composite unique, on utilise le mnémoniquec Mn0 Mn1 '00'.

Paramètres système

Tous les PC3000 ont un jeu de paramètres intégré, qui comprennent les paramètres standard Identification d'appareil (II) et Mode (MN). D'autres paramètres prennent par exemple en charge le téléchargement des programmes

utilisateur et la surveillance de l'état des racks d'entrée/sortie. Les détails de l'utilisation des paramètres système ne figurent pas dans ce document, bien que la partie Mnémoniques système EI PC3000 donne une liste complète des paramètres système et des fonctions accessibles par les mnémoniques EI Bisync.

Lors de la lecture d'un paramètre, il est possible de spécifier n'importe quelle Uid valable. Certains mnémoniques système nécessitent aussi une identité de canal Chid. Il est aussi possible de donner une Chid lors de la lecture des mnémoniques qui n'en ont pas nécessairement besoin ; dans ce cas, elle n'est pas prise en compte à condition qu'elle se situe dans la plage EI Bisync valable. Dans tous les cas, si l'on utilise une Chid, elle est renvoyée dans le message réponse.

Codage des données

Le contenu du message de communications est décrit succinctement dans cette partie. Il existe deux structures pour chaque type de données, chacune étant sélectionnée automatiquement en fonction de l'Uid utilisée lors de la sélection du PC3000.

La première est un sous-ensemble de structures spécifiées dans le 'Manuel de communications EI Bisync(HP022047)' et 'Extensions de protocole de communications EI Bisync (HP024113)' ; elle a été choisie de manière à être la plus efficace possible pour le transfert et le traitement de données de chaque type élémentaire. Elle est utilisée si l'Uid est comprise entre '0' (30h) et 'O' (4Fh).

La deuxième est adaptée aux besoins du système de supervision Eurotherm (ESP) qui ne prend en charge qu'un nombre limité de types de données. Elle est utilisée si l'Uid est comprise entre 'P' (50h) et 'o' (6Fh).

Spécification de la structure des variables esclaves

Pour certains types de données, la structure EI Bisync utilisée lors de l'accès aux variables esclaves peut être sélectionnée à l'aide d'un caractère de structure qui est ajouté à la fin de l'adresse de la variable esclave.

Exemple pour Slave_Real :

Adresse = 'EB060' : implique la structure de précision 24 bits par défaut ;
adresse = 'EB060p' : sélectionne la structure de précision 32 bits

(Cf. le codage REEL pour avoir des détails supplémentaires sur ces structures)

Codage standard

On utilise le codage suivant pour l'accès EI_Bisync normal.

Virgule flottante (REEL)

Les valeurs à virgule flottante (réels) sont émises et reçues avec la structure condensée simple IEEE. Il s'agit d'une structure de longueur variable qui

nécessite uniquement la transmission des données significatives. Le caractère de ce type de structure est '@'.

	Bits	Nbre. maxi de caractères envoyés	Exemple d'adresse	
P	24	4	'EBx99' ou 'EBx99P'	VALEUR PAR DEFAUT
p	32	6	'EBx99p'	Pas pris en charge pour l'instant

Tableau 3-12 Structures à virgule flottante EI_Bisync_S

Pour diminuer le nombre de caractères transmis par le PC3000, il est possible de limiter la résolution de ce nombre. Par défaut, 24 bits au maximum de la valeur de 32 bits sont transmis mais il est possible d'envoyer la résolution totale en cas de besoin, dans la chaîne d'adresse, à l'aide du caractère de structure 'p'. Ce caractère 'P' peut servir à sélectionner explicitement la précision 24 bits.

Le tableau ci-dessous donne des exemples de valeurs à virgule flottante et les caractères ASCII associés utilisés pour coder les valeurs en cas de transmission par EI Bisync avec la précision 24 bits.

Valeur REELLE	Codage en caractères ASCII
0.0	@
2.0	@P
-2.0	@p
0.5	@Op
-0.5	@op
1000.0	@QGh
10000.0	@Qap
100000.0	@Q1Mp

Tableau 3-13 Valeurs codées à virgule flottante EI_Bisync_S

Un nombre réel possédant la structure IEEE se décompose dans les champs suivants :

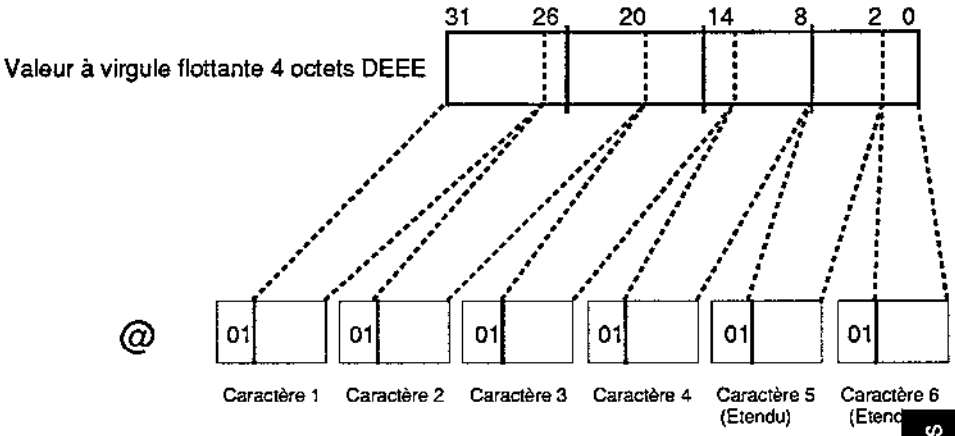


Figure 3-20 Codage avec virgule flottante

Les extensions de protocole de communications EI Bisync (HP024113) donnent davantage de détails sur cette structure.

Entier

Les valeurs entières sont émises et reçues sous forme hexadécimale ASCII. Il s'agit d'une structure de longueur variable qui nécessite uniquement la transmission des données significatives. Le code de type de structure est ' et la longueur du contenu peut être comprise entre 0 et 8 caractères pour représenter une valeur 32 bits. Les valeurs négatives nécessitent la transmission de la totalité des 8 caractères.

Le tableau ci-après montre les valeurs entières et les codes ASCI associés.

Valeur entière	Codes ASCII
0	>
1	>1
16	>10
256	>100
4096	>1000
65536	>10000
1048576	>100000
16777216	>1000000
268435456	>10000000
2147483647	>7FFFFFFF
-1	>FFFFFFFF

Tableau 3-14 Valeurs entières EI_Bisync_S

BOOL

Les valeurs booléennes sont envoyées de la même manière que les valeurs entières.

Valeur booléenne	Codes ASCII
0	>
1	>1

Tableau 3-15 Codes booléens EI_Bisync_S.

DUREE

Les valeurs de durée sont envoyées sous forme d'un nombre entier de millisecondes sans signe. Le tableau 3-17 montre des exemples de valeurs de durée et les codes ASCII associés.

Valeur de DUREE	Codes ASCII
0ms	>
1ms	>1
16ms	>10
256ms	>100
4s96ms	>1000
1m5s536ms	>10000
17m28s576ms	>100000
4h39m37s216ms	>1000000
3d2h33m55s456ms	>10000000
49d17h2m47s295ms	>FFFFFFFF

Tableau 3-16 Codes de durée El_Bisync_S.

DATE

Les valeurs de date sont envoyées sous forme d'un nombre entier sans signe de jours depuis le 01/01/1970.

Valeur de DATE	Codes ASCII
1970-01-01	>
1970-01-02	>1
1991-04-04	>1E53
2020-12-31	>48C3

Tableau 3-17 Codes de date El_Bisync_S.

DATE_AND_TIME

Les valeurs Date et heure sont envoyées sous la forme d'un nombre entier de millisecondes depuis le 01/01/1970 00:00:00. Le tableau 3-19 montre des exemples de valeurs DATE_AND_TIME et les codes ASCII associés.

Valeur DATE_AND_TIME	Codes ASCII
1970-01-01-00:00:00	>
1970-01-01-00:00:01	>1
1991-04-04-16:14:32	>27FB50E8
2038-01-19-03:14:07	>7FFFFFFF

Tableau 3-18 Codes Date et heure EI_Bisync_S.

TIME_OF_DAY

Les valeurs Heures du jour sont envoyées sous la forme d'un nombre entier de secondes depuis 00:00:00.

Le tableau 20 montre des exemples de valeurs TIME_OF_DAY et les codes ASCII associés.

Valeur TIME_OF_DAY	Codes ASCII
00:00:00	>
00:00:01	>1
00:00:16	>10
00:04:16	>100
01:08:16	>1000
18:12:16	>10000
23:59:59	>1517F

Tableau 3-19 Codes Heure du jour EI_Bisync_S.

STRING

Les valeurs Chaîne sont envoyées par apposition d'une apostrophe (27h) comme préfixe et expansion de tous les caractères non imprimables en une suite "escape" à trois caractères. Cette suite est composée du caractère "ESCAPE" (1Bh) suivi de deux chiffres hexadécimaux représentant le code ASCII de ce caractère.

Un caractère est dit non imprimable si son code ASCII est inférieur à 20h ou supérieur à 7Eh.

Données composites

Un maximum de trois niveaux de structures imbriquées sont pris en charge.

Quatre caractères servent de séparateurs entre les champs, dans un ordre hiérarchique.

Abréviation	Code hexadécimal		
FS	1C	Démarrage niveau 3	
GS	1D	Démarrage niveau 2	Séparateur de 3ème niveau
RS	1E	Démarrage niveau 1	Séparateur de 2ème niveau
US	1F		Séparateur de 1er niveau

Tableau 3-20 Codage des données composites EI_Bisync_S

Le premier caractère indique le nombre de niveaux de la structure. Les champs situés dans la structure sont séparés par des caractères séparateurs de niveau inférieur.

La présence d'un séparateur de niveau supérieur implique la présence simultanée de tous les séparateurs de niveau inférieur.

Exemple :

RS **>1234** *US* **@Pqr**

est une structure de niveau 1 à 2 champs.

GS **>1234** *US* **@Pqr** *RS* **>1**

est une structure à 2 niveaux avec 2 champs au niveau supérieur, le premier étant une structure contenant 2 champs.

Lors de l'écriture dans le paramètre, les champs qui ne doivent pas être modifiés peuvent être oubliés. Par conséquent, pour écrire dans le dernier champ du niveau supérieur dans l'exemple précédent, on aurait :

GSRS **>1**

Toujours lors de l'écriture, un niveau peut être interrompu prématurément s'il ne reste pas de données à envoyer. Par conséquent, pour écrire uniquement dans le premier champ de la structure inférieure de l'exemple précédent, on aurait :

GS **>1234**

Codage du système de supervision Eurotherm (ESP)

Les codes suivants sont utilisés pour accéder aux paramètres à l'aide de l'Uid normale décalée de 20h. Ils sont prévus pour les communications avec ESP qui nécessitent un codage de remplacement.

Virgule flottante (REEL)

Codage identique au codage standard, sauf avec le champ de données de longueur fixe.

Exemples (résolution 24 bits) :

Valeur REELLE	Codes ASCII
0.0	@@@@@
2.0	@P@@@
-2.0	@p@@@
0.5	@Op@@
-0.5	@op@@
1000.0	@QGh@
10000.0	@Qap@
100000.0	@Q1Mp

Tableau 3-21 Codage à virgule flottante (REEL).

ENTIER

Converti en REEL.

BOOL

Codage identique au codage standard, sauf avec le champ de données de longueur fixe.

Valeur booléenne	Codes ASCII
0	>00
1	>01

Tableau 3-22 Codage booléen EI_Bisync_S pour ESP.

HEURE

Envoyé sous la forme d'un nombre de secondes avec virgule flottante (REEL).

DATE

Envoyé sous la forme d'un nombre de jours avec virgule flottante (REEL) depuis le 01/01/1970.

DATE_AND_TIME

Envoyé sous la forme d'un nombre de secondes avec virgule flottante (REEL) depuis le 01/01/1970-00:00:00.

TIME_OF_DAY

Envoyé sous la forme d'un nombre de secondes avec virgule flottante (REEL) depuis 00:00:00.

CHAINE

Il n'existe aucun codage spécial ESP pour les valeurs Chaîne, elles sont donc envoyées à l'aide du codage standard.

Données composites

Les séparateurs de champs et les règles sont identiques à ceux du codage standard mais les données sont codées de la manière définie dans cette section pour ESP.

Protocole Link Layer

Nous donnons un bref récapitulatif du protocole EI Bisync Link Layer (structures d'échanges de messages de base). Pour avoir une description détaillée, se reporter au 'Manuel de communications EI Bisync (HP022047)'.

Adressage des appareils

On considère le PC3000 comme adressé s'il reçoit le caractère EOT (04h) suivi de sa Gid indiquée deux fois et d'une Uid valable indiquée deux fois.

Exemple :

EOT **0011**

adresse un PC3000 possédant une Gid '0'. L'Uid '1' est utilisée comme composant de l'adresse d'un paramètre dans le PC3000.

Le PC3000 reste adressé jusqu'à la réception d'un EOT ou jusqu'à une mise sous tension ou une réinitialisation.

Lecture de paramètres

Pour lire un paramètre, il faut adresser le PC3000 en ajoutant les champs d'adresse de paramètre CHID Mn0 Mn1 et un ENQ (05h). Par exemple :

EOT **0011123***ENQ*

Si le paramètre est valable, le PC3000 répond par un STX (02h), le CHID Mn0 Mn1 tel qu'il a été envoyé, les données, un ETX (03h) et le caractère de contrôle. Ce caractère de contrôle est la somme binaire (identique au OU exclusif) de chaque caractère suivant le STX jusqu'à l'ETX inclus. Exemple :

STX **123>456***ETX* :

où ':' est le caractère de contrôle.

Si le paramètre n'est pas valable, le PC3000 répond par un EOT (04h) et le paramètre système EE contient un code indiquant le motif pour lequel le paramètre n'est pas valable.

Ecriture de paramètres

Pour écrire un paramètre, il faut adresser le PC3000 comme ci-dessus, puis lui envoyer un STX (02h), la CHID Mn0 Mn1 des paramètres, les données, un ETX (03h) et un caractère de contrôle (comme pour la lecture). Exemple :

EOT **0011***STX* **123>456***ETX* :

Si le paramètre est valable, le PC3000 répond par un ACK (06h).

Si le paramètre n'est pas valable, le PC3000 répond par un NAK (15h) et le paramètre système EE contient un code indiquant le motif pour lequel le paramètre n'est pas valable.

Exemple

Cet exemple montre les blocs fonctions nécessaires pour avoir accès à trois variables esclaves à virgule flottante (REEL) et à trois variables à mot d'état (SW) par les ports A et C du LCM.

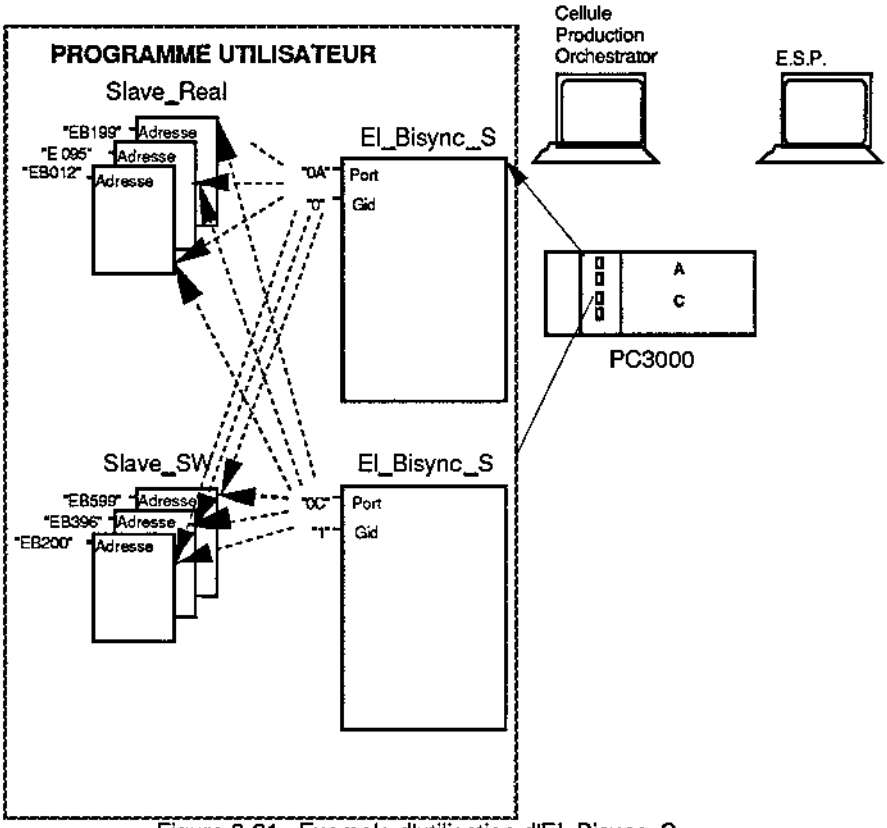


Figure 3-21 Exemple d'utilisation d'EI_Bisync_S.

N.B. : tous les blocs fonctions de modules de communications EI_Bisync_S ont accès aux variables esclaves qui ont une adresse contenant le sélecteur de protocole 'EB'.

Signalisation des erreurs

En cas d'erreur spécifique de bloc fonction EI_Bisync_S ou de module EI Bisync, l'erreur est signalée par l'intermédiaire du paramètre **Error_No** du bloc fonction EI_Bisync_S. Les erreurs concernant une variable esclave donnée sont signalées par l'intermédiaire du bloc de variable esclave associé.

Erreurs de bloc fonction

Error_No	Description de l'erreur
1	ERREUR DE PORT : ABSENCE D'ADRESSE •Le paramètre Port du bloc fonction contient moins de deux caractères.
5/6	ERREUR DE PORT : DEBIT PAS DISPONIBLE •Le débit demandé n'est pas disponible sur ce port série.
11	ERREUR DE PORT : EMPLACEMENT INTERDIT •Le numéro d'emplacement sélectionné n'est pas autorisé. Le numéro d'emplacement est le premier caractère du paramètre Port.
12	ERREUR DE PORT : PORT INTERDIT •Le numéro de port sélectionné n'est pas autorisé. Le numéro de port est le deuxième caractère du paramètre Port.
17	ERREUR DE PORT : PORT UTILISE •Le port sélectionné est déjà utilisé pour un autre module.
18	ERREUR : PARAMETRES ESCLAVES TROP NOMBREUX •Un trop grand nombre de variables esclaves ont été déclarées dans le système.
19	ERREUR DE PORT : TYPES DE MODULES TROP NOMBREUX •Un trop grand nombre de types de modules esclaves ont été déclarés dans le système.

Tableau 3-23 Codes d'erreur EI_Bisync_S.

Codes d'erreur de variables esclaves

Ces erreurs sont signalées par l'intermédiaire du paramètre Error_No du bloc de la variable esclave.

Error_No	Description de l'erreur
1	CHAINE D'ADRESSE TROP COURTE •La chaîne d'adresse contenue dans le paramètre Adresse est trop courte pour être valable.
11	ERREUR D'ADRESSE : EMPLACEMENT INTERDIT •Le premier caractère du paramètre Adresse ne se situe pas dans la plage valable entre '0' et '5'.
12	ERREUR D'ADRESSE : PORT INTERDIT •Le deuxième caractère du paramètre Adresse ne se situe pas dans la plage valable entre 'A' et 'C' pour un port LCM ou entre 'A' et 'D' pour un port ICM.
100	CANAL OU MNEMONIQUE D'ADRESSE PAS VALABLE •L'identité ou le mnémonique de canal ne se situe pas dans la plage autorisée.
255	MODULE PAS TROUVE •Aucun module de port prenant en charge cette variable esclave n'a été trouvé. Il n'existe aucun module prenant en charge le protocole sélectionné (EB dans ce cas) ou le module n'a pas pu être initialisé. Le paramètre Error_No du bloc EI_Bisync_S donne le motif de cette erreur.

Tableau 3-24 Codes d'erreur de variables esclaves.

Codes d'erreur EE

Les erreurs sont indiquées par le paramètre EError du bloc EI_Bisync_S et par le paramètre système EE lu par l'intermédiaire de la liaison série. Les valeurs décimales de ces codes seront utilisées par le paramètre EError lors de l'affichage sur la station de programmation PC3000. La valeur hexadécimale est utilisée lorsque l'erreur est lue depuis le PC3000 à l'aide du mnémonique EI Bisync EE.

EError	Description de l'erreur
0 (0000h)	EERROR : NEANT •Aucune demande n'a échoué depuis la remise à zéro du système PC3000.
503 (01F7h)	EERROR : CANAL OU MNEMONIQUE PAS VALABLE •L'identité ou le mnémonique de canal dans la demande n'est pas reconnu(e).
754 (02F2h)	EERROR : BCC PAS VALABLE •Le total de contrôle (BCC) d'une demande d'écriture ne correspondait pas au contenu du message.
1528 (05F8h)	EERROR : LECTURE UNIQUEMENT •Il était impossible d'écrire sur le paramètre désigné dans une demande d'écriture. Cela peut être dû au fait que Write_Protect est positionné sur le bloc EI_Bisync_S ou sur le bloc de la variable déportée.
2040 (07F8h)	EERROR : DONNEES PAS VALABLES •Le contenu d'une demande d'écriture n'était pas valable. Par exemple, un caractère non imprimable ou un caractère de type de paramètre manquant.
2296 (08F8h)	EERROR : DONNEES HORS PLAGE •Le contenu d'une demande d'écriture dépassait la plage admise pour le paramètre.
8443 (20FBh)	EERROR : TYPE DE PARAMETRE PAS PRIS EN CHARGE •Il y a eu une tentative d'accès à une variable esclave qui possède une valeur d'un type pas encore pris en charge par le module EI_Bisync_S.
8699 (21FBh)	EERROR : CODAGE IMPOSSIBLE •Le paramètre en cours de lecture n'a pas pu être codé en raison d'une limitation interne.
8955 (22FBh)	EERROR : ACCES COMPOSITE PAS ENCORE PRIS EN CHARGE •La lecture ou l'écriture de tous les paramètres d'un bloc fonction sous la forme d'un seul paramètre multi-éléments n'est pas encore prise en charge.
9211 (23FBh)	EERROR : RS ATTENDU •Le caractère RS (1Eh) était attendu comme premier caractère de la demande d'écriture. Le paramètre est une structure mono-niveau
9467 (24FBh)	EERROR : ELEMENTS TROP NOMBREUX •La demande d'écriture pour un paramètre multi-éléments contenait un trop grand nombre de champs.
9723 (25FBh)	EERROR : STRUCTURE TROP PROFONDE •Il y a eu une tentative d'accès à une variable esclave ayant une structure qui n'est pas mono-niveau. Ce type de structure n'est pas pris en charge actuellement.

Tableau 3-25 Codes d'erreur EE.

EError	Description de l'erreur
9979 (26FBh)	EERROR : DEFAILLANCE INTERNE •Une défaillance s'est produite dans le module EI_Bisync_S au cours du traitement de la demande. Ce type d'erreur ne doit jamais se produire.
33275 (81FBh)	EERROR : TROP NOMBREUX FICHIERS OUVERTS •Un fichier est déjà ouvert sur ce port ou le nombre total de fichier ouverts est le maximum admis.
33276 (82F8h)	EERROR : SYSTEME DE FICHIERS PAS VALABLE •Le système de fichiers sélectionné n'est pas pris en charge.
33278 (84F8h)	EERROR : MODE D'OUVERTURE DE FICHIER PAS VALABLE •Le mode utilisé lors de l'ouverture d'un fichier n'est pas valable. Seuls Création, Lecture, Ecriture et Ajout sont pris en charge.
33279 (85F8h)	EERROR : STRUCTURE DE FICHIER INCORRECTE •Le contenu de la chaîne de structure est incorrect.
33280 (86F8h) and 33281 (87F8h)	EERROR : ESPACE INSUFFISANT POUR LE FICHIER •Mémoire insuffisance lors du formatage du système de fichiers.
33282 (88F8h)	EERROR : LE FICHIER N'EXISTE PAS •Le fichier n'existe pas dans le système de fichiers et, par exemple, ne peut pas être supprimé.
33283 (89F8h)	EERROR : LE FICHIER EXISTE DEJA •Le fichier existe déjà dans le système de fichiers et, par exemple, il est impossible de créer un nouveau fichier portant le même nom.
33284 (8AFBh)	EERROR : NOMBRE DE FICHIERS MAXIMAL •La mémoire fichier contient déjà le nombre maximal de fichiers autorisé.
33285 (8BFBh)	EERROR : FICHIER DEJA OUVERT POUR LA LECTURE •Le fichier est déjà ouvert pour la lecture et ne peut donc pas être ouvert pour l'écriture.
33286 (8CFBh)	EERROR : FICHIER DEJA OUVERT POUR L'ECRITURE •Le fichier est déjà ouvert pour l'écriture et ne peut donc pas être ouvert pour la lecture.
33287 (8DFBh)	EERROR : FICHIER PAS OUVERT POUR LA LECTURE •Le fichier n'est pas ouvert pour la lecture et ne peut donc pas être lu.
33288 (8EFBh)	EERROR : FICHIER PAS OUVERT POUR L'ECRITURE •Le fichier n'est pas ouvert pour l'écriture et ne peut donc pas être écrit.

Tableau 3-25 Codes d'erreur EE (suite)

EError	Description de l'erreur
33289 (8FFBh)	EERROR : PLUS DE MEMOIRE FICHIER •La mémoire fichier n'a plus de place pour conserver le fichier.
33290 (90FBh)	EERROR : MEMOIRE FICHIER NON FORMATEE •La mémoire fichier n'a pas été formatée.
33291 (91FBh)	EERROR : NOM DE FICHIER PAS VALABLE •Le nom de fichier contient des caractères pas valables (caractères de contrôle).

Tableau 3-25 Codes d'erreur EE (suite)

Mnémoniques système PC3000 EI Bisync

Les mnémoniques suivants sont destinés à prendre en charge le protocole EI Bisync ou à accéder aux fonctions système dans le PC3000. Les mnémoniques système sont pris en charge sur les ports LCM A, B ou C en cas de fonctionnement en mode communications par défaut ou lorsque le système est configuré comme pour le fonctionnement en mode esclave EI Bisync à l'aide du bloc fonction de module de communications (EI_Bisync_S). Bien que les fonctions système soient fournies comme élément de l'interface avec la station de programmation PC3000, elles peuvent être utilisées pour certaines applications spéciales.

Dans le tableau ci-dessous, les champs sans objet sont repérés par NA. Il faut noter que les données restituées lors de la lecture d'un mnémonique contiennent parfois un certain nombre de champs. Dans certains cas, l'écriture dans un mnémonique provoque l'exécution d'une fonction particulière.

Mnémonique	Description de lecture	Description d'écriture
EE	Dernier code d'erreur Bisync	NA
FF	Lecture de la structure du système de fichiers	Formatage du système de fichiers
FO	NA	Ouverture d'un fichier LCM
FC	NA	Fermeture d'un fichier LCM
FR	Nouvelle lecture du bloc du fichier LCM	NA
Fr	Nouvelle lecture du bloc du fichier LCM	NA
FW	NA	Ecriture d'un bloc dans un fichier LCM
FD	Lecture de la première entrée du répertoire	NA
Fd	Lecture de la prochaine entrée du répertoire	NA
FK	NA	Suppression d'un fichier LCM
FY	NA	Copie d'un fichier LCM
I	Identificateur d'appareil	NA
V0	Numéro de version	NA
CI	Informations de configuration	NA
MN	Lecture du mode PC3000	Demande d'un changement de mode PC3000

Tableau 3-26 Mnémoniques système PC3000

Les mnémoniques système suivants ne sont pas pris en charge et font l'objet de différences entre les différentes versions du PC3000. Ils sont uniquement indiqués à des fins de diagnostic.

Mnémoniques système PC3000

Mnémonique	Description de lecture	Description d'écriture
ss	Lit la stratégite de démarrage du PC3000	NA
se	Lit l'enregistrement d'erreurs système rack	NA
	(Chid sélectionne l'entrée)	
sc	Lit le décompte des erreurs système	Effacement du journal d'erreurs système
r1	Lit les informations du module d'emplacement 1	NA
	(CHID sélectionne l'emplacement)	
r2	Lit les informations du module d'emplacement 2	NA
	(CHID sélectionne l'emplacement)	
r3	Lit les informations du module d'emplacement 3	NA
	(CHID sélectionne l'emplacement)	
r4	Lit les informations du module d'emplacement 4	NA
	(CHID sélectionne l'emplacement)	
r5	Lit les informations du module d'emplacement 5	NA
	(CHID sélectionne l'emplacement)	
r6	Lit les informations du module d'emplacement 6	NA
	(CHID sélectionne l'emplacement)	
r7	Lit les informations du module d'emplacement 7	NA
	(CHID sélectionne l'emplacement)	
r8	Lit les informations du module d'emplacement 8	NA
	(CHID sélectionne l'emplacement)	
dl	Téléchargement du bloc du code source	Téléchargement du bloc du code source
ds	Lecture des informations d'état du code source	NA
db	Lecture du numéro de bloc du code source	Sélection du bloc du code source
ms	Lecture des détails de la zone du programme utilisateur sélectionnée	Sélection de la zone du programme utilisateur pour la lecture/écriture
mw	Lecture de la zone du programme utilisateur sélectionnée par 'ms'	Ecriture de la zone du programme utilisateur sélectionnée par 'ms'

Tableau 3-27 Mnémoniques système PC3000 pas pris en charge

Mnémonique	Description de lecture	Description d'écriture
xe	Lecture des informations d'état du programme utilisateur	NA
mi	Lecture des informations d'identité du programme utilisateur	NA
mv	Lecture de l'identité unique du programme utilisateur	NA
xs	Lecture des détails de la zone du programme utilisateur sélectionnée	Sélection de la zone du programme utilisateur pour la lecture/écriture
xw	Lecture de la zone du programme utilisateur sélectionnée par'xs'	Ecriture de la zone du programme utilisateur sélectionnée par 'xs'
xe	Lecture des informations d'état du programme utilisateur	NA
xi	Lecture des informations d'identité du programme utilisateur	NA
xv	Lecture de l'identité unique du programme utilisateur	NA

Tableau 3-27 Mnémoniques système PC3000 pas pris en charge (suite)

Plages d'adresses de variables esclaves

Les tableaux ci-après montrent les caractères acceptables par les adresses de variables esclaves, c'est-à-dire utilisables pour une identité de canal esclave.

Tableau ASCII de caractères hexadécimaux							
	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	

Tableau 3-28 Caractères d'adresses esclaves acceptables

Les caractères valables utilisables pour le premier caractère de mnémonique Mn0 d'une adresse de variable esclave sont indiqués dans le tableau 3-29.

Tableau ASCII de caractères hexadécimaux							
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9						

Tableau 3-29 Caractères acceptables pour le mnémonique Mn0

Les caractères valables utilisables pour le deuxième caractère de mnémonique Mn1 d'une adresse de variable esclave sont fournis dans le tableau 3-30.

Tableau ASCII de caractères hexadécimaux							
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	

Tableau 3-30 Caractères acceptables pour le mnémonique Mn1

Le tableau de codes ASCII est fourni à titre de référence générale.

Tableau ASCII de caractères hexadécimaux							
00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL

Tableau 3-31 Codes ASCII

Fonctions qui ne sont pas prises en charge

Cette partie décrit le fonctionnement du bloc fonction EI_Bisync_S inclus dans le logiciel à partir de la version 2.00, à l'exception des fonctions ci-dessous qui n'ont pas encore été mises en oeuvre.

- transfert de messages multi-blocs.
- accès à l'ensemble des paramètres d'un emplacement de bloc fonction sous la forme d'un seul paramètre composite.
- valeurs condensées à virgule flottante IEEE 32 bits (seule la structure 24 bits pour les valeurs à virgule flottante est prise en charge)

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	STRING	'0A'	Oper	Config		
Baud	ENUM	_9600	Oper	Config	Valeurs énumérées	_75(0) _300(1) _600(2) _1200(3) _2400(4) _4800(5) _9600(6) _19200(7) _38400(8) _57600(9) _115200(10)
Wr_Protect	BOOL	No	Oper	Super	Détect.	No(0) Yes(1)
TXB_Max	SINT	0	Oper	Config	Cf. remarque 1	
Gid	STRING	' '	Oper	Config		
Status	BOOL	NOGO	Oper	Bloc	Détect.	NOGO(0) Go(1)
Error_No	SINT	0	Oper	Block	Lim. haute Lim. basse	255 0

Tableau 3-32 Attributs des paramètres EI_Bisync_S

BLOC FONCTION RAW_COMMS

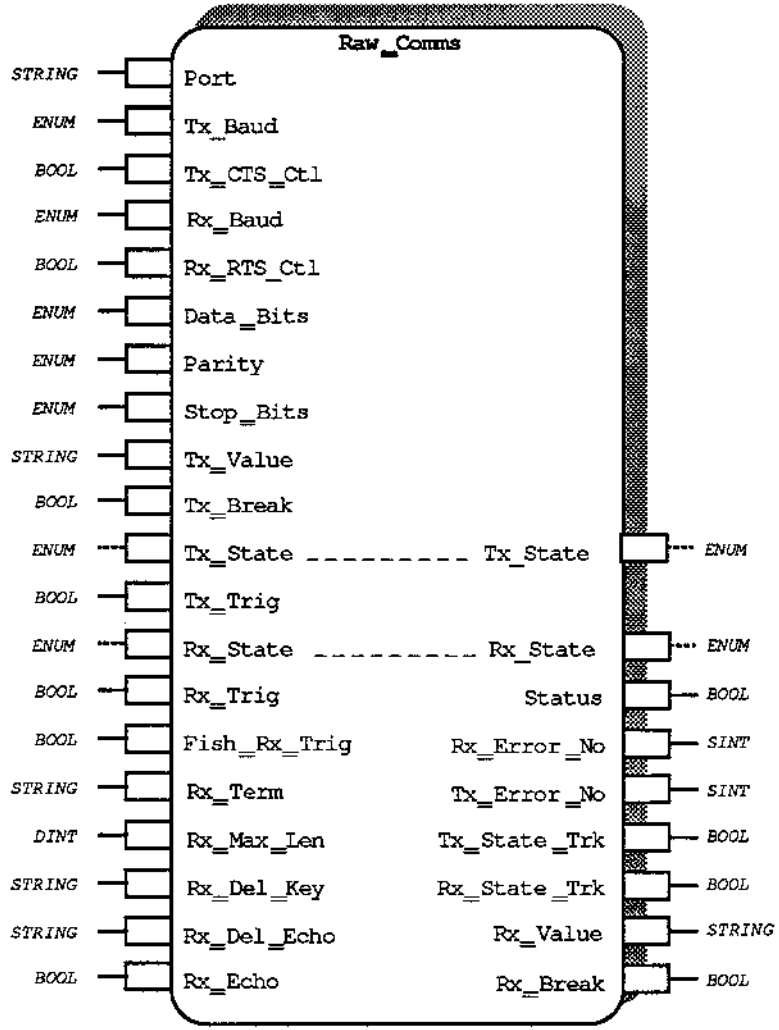


Figure 3-22 Diagramme Raw_Comms

Description fonctionnelle

Le bloc fonction Raw_Comms offre des possibilités des contrôler directement l'émission et la réception de messages par une liaison série. Avant de lire cette description, vous avez intérêt à acquérir des connaissances générales au sujet du système de communications PC3000 en lisant la présentation générale des communications PC3000.

Le bloc fonction Raw_Comms est utilisable pour les applications où il est nécessaire d'avoir un contrôle du port de communications série à un niveau élémentaire et une souplesse suffisante pour construire ou analyser des messages exactement comme ils sont émis ou reçus sur une liaison série. Etant donné qu'il n'y a aucun protocole, le module ne crée aucun message et aucune structure de message supplémentaire.

Le bloc Raw_Comms peut être affecté à n'importe quel port série à l'aide du paramètre de saisie Port (cf. 'Présentation générale des communications PC3000', partie 'Affectation d'un bloc fonction de communications à un port').

Raw_Comms offre une large gamme de fonctions de niveau élémentaire dont :

- un accès direct aux messages tels qu'ils sont émis ou reçus par l'intermédiaire de la liaison série.
- un contrôle indépendant de l'émission et de la réception des messages, comprenant une sélection séparée du débit sur les lignes d'émission et de réception.
- un contrôle du flux de messages à l'aide de Prêt à émettre (CTS) et Demande pour émettre. Ce contrôle n'est actuellement pas pris en charge sur les ports LCM et ICM.
- un écho sélectionnable des caractères reçus lorsque cela est nécessaire.
- un ordre de suppression, sélectionnable par l'utilisateur, pour la suppression de caractères dans la mémoire tampon de réception.

Le développement de programmes utilisateur destinés à fonctionner avec Raw_Comms est plus complexe que l'utilisation des blocs fonctions de module de protocole car la structure des messages et la synchronisation doivent être gérées par le programme.

N.B. : contrairement au cas des blocs fonctions de module de communications propres au protocole, il est impossible d'utiliser les variables esclaves ou déportées avec Raw_Comms.

Utilisation de Raw_Comms

Les applications types pour le bloc fonction de module Raw_Comms comprennent :

- les communications avec les périphériques utilisant des protocoles simples non standard,

•l'envoi des rapports à des imprimantes série en ligne ou à des imprimantes spécialisées, par exemple pour la production d'étiquettes.

•les communications avec les terminaux orientés caractères comme DEC VT100 ou avec les périphériques à affichage simple.

Le bloc fonction Raw_Comms offre à l'utilisateur le niveau minimal d'interface avec les ports série sur le PC3000. L'ensemble des paramètres de codage des données série comme le débit et la parité sont programmables par l'utilisateur. Le bloc effectue le minimum de traitement sur les caractères reçus et envoyés afin de conserver une souplesse maximale. Certaines fonctions simples pour l'écho, la suppression et l'achèvement de la chaîne de réception ont été incluses dans le bloc pour faciliter la programmation mais elles peuvent être désactivées si elles ne sont pas nécessaires.

Attributs du bloc fonction

Type: 8 60

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port Status

Capacité mémoire nécessaire : 864 octets

Description des paramètres

Le bloc Raw_Comms a un total de 25 paramètres. Ces paramètres se répartissent en sept groupes : configuration, émission, contrôle d'émission, réception, contrôle de réception, écho/suppression et état.

Paramètres de configuration du module

Le bloc Raw_Comms possède un certain nombre d'entrées qui configurent différents aspects du module. La modification de ces paramètres pendant l'exécution du programme utilisateur n'a aucun effet sur le module, sauf dans des circonstances particulières. Cf. 'Présentation générale des communications PC3000', partie 'Modification temporaire des paramètres de configuration'.

Port

Le paramètre Port est l'adresse à deux caractères du port auquel se rapporte le bloc. Le premier caractère est un chiffre compris entre 0 et 5, qui représente l'emplacement dans le rack et le deuxième caractère est une lettre représentant le port dans cet emplacement. Par exemple, '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' désignerait son port supérieur.

Tx_Baud

Le paramètre Tx_Baud offre un choix de 11 débits d'émission différents, compris entre 75 Bauds et 115,2 kBauds.

Valeur énumérée	Débit
0	75
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	115200

Tableau 3-33 Débit d'émission/réception de Raw_Comms

N.B. : tous les ports ne peuvent pas prendre en charge la totalité des débits. Si le port choisi ne prend pas en charge le débit donné, une erreur sera indiquée lors du premier lancement du bloc fonction (cf. la description de Rx/Tx_Error_No). Pour avoir des détails sur les débits pris en charge par des ports ou des modules donnés, consulter la documentation relative aux modules.

Tx_CTS_Ctl

Si le paramètre Tx_CTS_Ctl est sur On, l'émetteur n'enverra des données que lorsque la ligne CTS sera active. Si ce paramètre est sur Off, l'émetteur ne répondra pas à la ligne CTS mais il émettra toujours lorsque cela lui sera demandé. Si le contrôle de débit d'émission CTS n'est pas disponible sur le port indiqué, ce paramètre doit être positionné sur Off.

Si l'utilitaire n'est pas disponible et si le paramètre est sur On, une erreur sera indiquée lors du premier lancement du bloc fonction (cf. la description de Rx/Tx_Error_No). Pour avoir des détails sur les ports et modules qui prennent en charge le contrôle de débit CTS, consulter la documentation relative aux modules.

Rx_Baud

Le paramètre Rx_Baud est identique à Tx_Baud, à la différence près qu'il se rapporte au débit de réception.

Tx_RTS_Ctl

Le paramètre Tx_RTS_Ctl sert à demander un contrôle de débit pour le récepteur. Si Tx_RTS_Ctl est sur On, la sortie RTS sera uniquement active si le récepteur est prêt à recevoir des caractères. Si ce paramètre est sur Off, RTS restera actif en permanence. Si le contrôle de débit RTS n'est pas pris en charge par le port, Tx_RTS_Ctl doit être sur Off, sinon une erreur sera indiquée lors du premier lancement du bloc fonction (cf. la description de Rx/Tx_Error_No). Pour avoir des détails sur les ports et modules qui prennent en charge le contrôle de débit RTS, consulter la documentation relative aux modules.

Data_Bits

Le paramètre Data_Bits fixe le nombre de bits par caractère pour l'émission et pour la réception. S'il est inférieur à 8, les bits les plus significatifs (8 -Data_Bits) ne seront pas pris en compte lors de l'émission et seront considérés comme égaux à zéro lors de la réception. Le nombre de bits par caractère disponible est de 5, 6, 7 ou 8 et est représenté par les valeurs énumérées _5, _6, _7 et _8 .

Parité

Le paramètre Parité peut être positionné sur PAIR, IMPAIR, ESPACE, MARQUE ou NEANT. Si la parité est NEANT, seuls les bits de démarrage, de données et d'arrêt sont envoyés. Si la parité est PAIRE, IMPAIRE, ESPACE ou MARQUE, le bit supplémentaire sera envoyé après les bits de données. Lors de la réception, ce bit sera contrôlé et une erreur affichée s'il ne correspond pas à la configuration du paramètre Parité.

Stop_Bits

Le paramètre Stop_Bits fixe le nombre de bits d'arrêt attendus par le récepteur et envoyés par l'émetteur. Il peut y avoir 1, 1,5 ou 2 bits d'arrêt, représentés par les valeurs énumérées _1, _1_5 et _2 .

Paramètres d'émission

Tx_Value

Le paramètre Tx_Value sert à contenir la chaîne à transmettre, dont la longueur peut aller jusqu'à 128 caractères. Lorsque le paramètre **Data_Bits** a été positionné sur une valeur inférieure à _8, seul le nombre spécifié de bits les moins significatifs sera envoyé pour chaque caractère. Un, deux ou trois bits les

moins significatifs du caractère ne seront pas pris en compte, selon le nombre de Data_Bits.

Tx_Break

Lorsque le paramètre booléen Tx_Break est sur On, toute émission en cours est arrêtée et un état d'interruption sera produit sur la liaison série. Si des caractères sont perdus, Tx_Error_No l'indique. Il faut noter que l'émission de caractères est impossible tant que Tx_Break est sur On.

Paramètres de contrôle d'émission

Deux paramètres de contrôle d'émission permettent de lancer l'émission de Tx_Value. Ces deux paramètres peuvent servir à lancer l'émission mais l'un convient mieux au contrôle par câblage et l'autre au contrôle par programme séquentiel. Le paramètre Tx_State indique également l'état de l'émetteur.

Tx_State

Le paramètre Tx_State peut avoir des valeurs OK, EN ATTENTE, ERREUR et ECRITURE qui indiquent l'état actuel de l'émetteur. L'affectation d'Ecriture à ce paramètre depuis le programme séquentiel peut servir à démarrer l'émission de Tx_Value. Pour avoir des détails complets sur le fonctionnement de l'émetteur, cf. les chaînes d'émission page 3-108.

Tx_Trig

Le paramètre Tx_Trig est destiné à faciliter le contrôle du bloc fonction Raw_Comms par câblage. Lorsque Tx_Trig passe de Off à On, l'émission est lancée conformément à ce qui est décrit dans les chaînes d'émission page 3-108.

Paramètres de réception

Rx_Value

La chaîne Rx_Value contient les caractères reçus par l'intermédiaire du port série. Pour avoir des détails sur la manière dont la réception est contrôlée ou mise en mémoire tampon, cf. les chaînes de réception page 3-109.

Rx_Break

Si le paramètre Rx_Break est sur On, il indique qu'un état d'interruption a été détecté sur la ligne de réception. Lorsque l'état d'interruption disparaît, le paramètre Rx_Break revient sur Off .

Paramètres de contrôle de réception

Le bloc fonction Raw_Comms comporte cinq paramètres qui contrôlent la saisie des chaînes de caractères. Ces paramètres sont présentés ci-dessous et les

chaînes de réception, page 3-109, contiennent des détails complets sur la réception.

Rx_State

Le paramètre Rx_State est identique au paramètre Tx_State, avec des valeurs possibles OK, EN ATTENTE, ERREUR, LECTURE et VIDAGE. Comme pour le paramètre Tx_State, il est possible de lire le paramètre Rx_State pour voir l'état actuel du récepteur et de l'écrire pour le contrôler. Pour avoir des détails sur la manière dont le récepteur est contrôlé, cf. les chaînes de réception page 3-109.

Rx_Trig

Rx_Trig, comme Tx_Trig, est conçu de manière à faciliter le contrôle de la réception à l'aide du câblage PC3000. Lorsque Rx_Trig passe de Off à On, la réception est lancée, comme le décrivent les chaînes de réception page 3-109.

Flsh_Rx_Trig

Flush_Rx_Trig est comme Rx_Trig, à la différence près que ce paramètre vide la mémoire tampon de réception au lieu de lancer la réception. Là aussi, les chaînes de réception page 3-109 contiennent une description détaillée du fonctionnement de Flush_Rx_Trig dans les chaînes de réception page 3-109.

Rx_Term

Le paramètre Rx_Term est une chaîne d'un caractère qui sert à identifier la fin d'une ligne de saisie (cf. la partie Méthode **REF rx). Si Rx_Term reste vide, la saisie sera uniquement lue depuis la mémoire tampon de réception dans le paramètre Rx_Value du bloc fonction lorsque les caractères Rx_Max_Len ont été reçus.

Rx_Max_Len

Rx_Max_Len indique le nombre maximal de caractères qui seront contenus dans la mémoire tampon de réception avant qu'ils soient transmis à Rx_Value. Rx_Max_Len doit être compris entre 1 et 128, 128 étant la longueur maximale de la chaîne Rx_Value.

Paramètres d'écho et de suppression

Une fonction simple d'écho et de suppression est offerte par le bloc Raw_Comms pour faciliter la mise en oeuvre de la saisie de données sur un périphérique qui n'a pas d'écho local. Le fonctionnement de la fonction d'écho est expliqué en détail dans la partie Echo, page 3-1.

Rx_Echo

Ce paramètre booléen positionne la fonction d'écho sur On ou Off.

Rx_Del_Key

Ce paramètre de chaîne à un caractère précise le caractère qu'il faut utiliser pour la suppression, par exemple `$7F'. S'il reste vide, la suppression ne sera pas effectuée.

Rx_Del_Echo

Pour supprimer un caractère sur certains écrans, il est nécessaire de renvoyer l'écho d'un certain nombre de caractères, par exemple '<BS><espace><BS>'. Le paramètre Rx_Del_Echo sert à indiquer cette chaîne de caractères.

Paramètres d'état

Le groupe final de paramètres sont les paramètres d'état qui indiquent les erreurs liées à l'émission et à la réception.

Etat

Le paramètre Etat est un paramètre booléen qui prend la valeur Go ou NoGo pour indiquer si les communications fonctionnent sans problème. Si une erreur se produit, le paramètre Status passe à NoGo et un numéro d'erreur différent de zéro apparaîtra en conséquence dans Rx_Error_No ou Tx_Error_No.

Rx_Error_No

Rx_Error_No indique si une erreur s'est produite lors de la réception des données. Cela peut aller de la demande d'un débit de réception que le port série ne pouvait pas prendre en charge aux erreurs d'encadrement au cours de la réception. Si Rx_Error_No est égal à 0, cela indique qu'aucune erreur ne s'est produite. Pour avoir des détails complets sur les codes d'erreur, se reporter à la partie Codes d'erreur.

Tx_Error_No

Tx_Error_No agit comme Rx_Error_No, à la différence près qu'il est réservé exclusivement aux erreurs liées à la réception. Pour avoir des détails complets sur les codes d'erreur, se reporter à la partie Codes d'erreur.

Configuration

Avant que l'émission ou la réception commence, il est nécessaire de fixer la configuration du port. Les paramètres de configuration, tels qu'ils sont décrits dans la partie Paramètres de configuration du module, doivent être choisis avant le lancement du programme utilisateur, car la configuration est fixée lors de la première exécution du programme utilisateur.

Emission des chaînes

Après configuration du bloc fonction Raw_Comms, il est possible d'émettre une chaîne en plaçant sa valeur dans le paramètre **Tx_Value** puis en déclenchant

l'envoi de cette chaîne à l'aide du paramètre **Tx_State** ou **Tx_Trig**. Si le contrôle de l'émission doit être effectué depuis un programme séquentiel, il est conseillé d'utiliser le paramètre **Tx_State** de la manière indiquée ci-après :

```
Port.Tx_Value := 'The quick brown fox jumps  over the lazy dog.';
Port.Tx_State := 3 (*Write*) ;
```

Le positionnement du paramètre **Tx_State** sur Ecriture provoque le démarrage de la transmission de la chaîne. Au cours de la transmission de la chaîne, le paramètre **Tx_State** passe à En attente puis, une fois l'opération terminée, revient à Ok. Si une erreur se produit à un stade quelconque, **Tx_State** passe sur Erreur, **Tx_Status** passe à NoGo et un code d'erreur différent de zéro apparaît dans **Tx_Error_No**. Si l'on essaie une nouvelle émission avant le passage intégral de la dernière chaîne, l'ancienne chaîne sera arrêtée, la nouvelle sera entamée et une erreur sera consignée dans le paramètre **Tx_Error**. Pour cette raison, en temps normal, la transition qui suit une paire d'affectations de ce type contient une vérification de la transmission en cours d'achèvement, comme par exemple :

```
( Port.Tx_State = 0 (*Ok*) )
```

S'il est nécessaire de déclencher l'émission de la chaîne par câblage, il faut utiliser le paramètre **Tx_Trig**. Un passage de ce paramètre de Off à On provoque l'envoi de **Tx_Value**. Le déclenchement de l'émission à l'aide de **Tx_Trig** a le même effet que l'affectation d'Ecriture au paramètre Etat, l'état passant à En attente jusqu'à ce que **Tx_Value** ait été envoyé.

> **N.B. :** les caractères de contrôle peuvent être intégrés aux chaînes transmises. Ils peuvent être saisis depuis la station de programmation PC3000 à l'aide d'une structure spéciale $ 'dollar' qui est décrite dans l'annexe.

Vidage de la mémoire tampon d'émission

Il existe deux méthodes pour vider la mémoire tampon d'émission. Lorsqu'on utilise le câblage ST, **Tx_Value** peut être positionné sur la chaîne vide, c'est-à-dire '', puis **Tx_Trig** peut être passé de Off à On, comme s'il envoyait la chaîne. Le module Raw_Comms traite cela comme un cas spécial et met fin à la transmission en cours.

Deuxième méthode : en cas de contrôle par programme séquentiel, le passage de **Tx_State** de En attente à Ok provoque l'arrêt de la transaction en cours, le vidage de la mémoire tampon d'émission et la remise à zéro (Ok) de **Tx_Error**.

Réception des chaînes

La réception des chaînes est contrôlée de la même manière que l'envoi des chaînes. La réception des caractères par le bloc Raw_Comms est placée en mémoire tampon de manière à permettre le renvoi d'écho, la suppression automatique et le traitement ligne par ligne et non caractère par caractère, par le programme utilisateur. Initialement, pour des raisons de simplicité, on part du

principe que le renvoi d'écho et la suppression ne sont pas nécessaires. Pour la réception, il est toujours nécessaire de prendre en compte les paramètres **Rx_Term** et **Rx_Max_Len**. Ces deux paramètres agissent sur le point auquel une chaîne reçue est transférée de la mémoire tampon de réception, au sein du bloc fonction, dans **Rx_Value**. Si un caractère **Rx_Term** a été défini (c'est-à-dire si la chaîne n'est pas vide), la saisie (caractère d'arrêt compris) apparaîtra dans **Rx_Value** dès la réception de ce caractère. Par exemple, s'il a fallu traiter l'entrée qui se terminait par <CR>, **Rx_Term** serait positionné sur '$R' et une valeur reçue dans **Rx_Value** pourrait être This is some inputSR'.

> **N.B. :** dans les chaînes ST, SR représente le caractère Retour chariot, ASCII 13.

Le paramètre **Rx_Max_Len** sert à spécifier le nombre maximal de caractères qui doivent être reçus avant qu'ils soient transférés dans **Rx_Value**. Si **Rx_Max_Len** était positionné sur 1, les caractères seraient saisis un par un. S'ils sont utilisés ensemble, **Rx_Term** et **Rx_Max_Len** provoquent la production de **Rx_Value** lorsqu'une des deux conditions est remplie.

Si les caractères reçus étaient :

```
'This_is_the_first_sentence._This_is_a_longer_second_sentence.'
```

et si **Rx_Term** et **Rx_Max_Len** étaient respectivement (.) et 30, les chaînes reçues dans **Rx_Value** seraient :

```
'This_is_the_first_sentence.'
'_This_is_a_longer_second_sente'
'nce.'
```

> **N.B. :** si le message reçu contient des caractères de contrôle non imprimables, ils sont indiqués sur la station de programmation PC3000 avec la structure $ 'dollar' qui est décrite dans les paragraphes ultérieurs.

Vidage de la mémoire tampon de réception

La mémoire tampon de réception peut être vidée à tout moment par positionnement de **Rx_State** sur Vidage ou par passage du paramètre **Rx_Flsh_Trig** de Off à On. Cette opération ne supprime pas les caractères déjà dans **Rx_Value** mais supprime les caractères qui ont été reçus et sont placés dans la file d'attente avant d'être copiés dans **Rx_Value.**

Echo et suppression

La mémoire tampon de réception dans le bloc fonction Raw_Comms permet d'utiliser deux fonctions supplémentaires : écho et suppression.

Echo

Si le paramètre Echo est sur On, l'émetteur renvoie un écho des caractères au fur et à mesure de leur réception. Ces caractères d'écho n'interfèrent avec aucune émission en cours mais seront mis en mémoire tampon jusqu'à ce que l'émetteur soit libre. Au cours du renvoi d'écho des caractères, le paramètre **Tx_State** est sur En attente tout comme si une chaîne avait été écrite par le programme utilisateur. Pour cette raison, il est nécessaire de vérifier que **Tx_State** est sur Ok ou Erreur avant d'essayer d'écrire des chaînes lorsque l'écho est sur On.

Suppression

Pour que la suppression de caractères dans la mémoire tampon avant qu'ils atteignent **Rx_Value** soit possible, il faut configurer deux paramètres supplémentaires : **Rx_Del_Key** et **Rx_Del_Echo**. Si la suppression est demandée, le paramètre **Rx_Del_Key** doit contenir le caractère à utiliser pour la suppression ; ce caractère est très souvent `$7F' ou `$08'. Si la fonction de suppression n'est pas demandée, il faut laisser **Rx_Del_Key** comme chaîne vide. Etant donné qu'il est souvent nécessaire de renvoyer un écho d'un certain nombre de caractères pour appliquer une suppression sur un terminal, les caractères qui ont fait l'objet d'un écho pour le caractère de suppression proviennent du paramètre **Rx_Del_Echo**. Par exemple, sur un terminal VT100, la touche de suppression restitue le code 7Fh et, pour supprimer un caractère de l'affichage, il faut envoyer '<BS><espace><BS>'.

Afin de configurer la suppression pour un VT100, il est par conséquent nécessaire de positionner **Rx_Del_Key** sur `$7F' et **Rx_Del_Echo** sur `$08 $08'. Si Echo n'est pas sur On, il n'est pas nécessaire de configurer le paramètre **Rx_Del_Echo** mais, si la suppression dans la mémoire tampon de réception reste nécessaire, il faut définir **Rx_Del_Key** .

Exemple

L'exemple présenté montre la manière dont on peut utiliser un terminal VT100 comme panneau de contrôle opérateur simple pour un process. Les détails du pilotage de panneau seront fournis mais le contrôle du process sera exclu pour des raisons de simplicité.

Pour cet exemple, nous supposerons que l'application est un process batch simple qui nécessite la saisie du numéro de batch et d'un point de consigne avant que le process puisse tourner. Le panneau est nécessaire pour permettre la saisie de ces valeurs et, une fois la saisie effectuée, pour afficher l'heure de démarrage du batch, la date et le point de consigne.

```
Please enter batch number and setpoint to start next run.


Batch number = 123456

Setpoint = 12.5

Batch number 123456, Setpoint 12.5 - Run started at 12:15:30 on 29-10-
90.

PHASE 1 .........Completed.

PHASE 2 .........Completed.

PHASE 3 .........Completed.

Batch number 123456 completed at 12:20:47 on 29-10-90

Please enter batch number and setpoint to start next run.

Batch number =
```

Figure 3-23 Exemple d'affichage de sortie

Pendant que le process est en marche, il doit indiquer lorsque chaque phase du process est terminée et, lorsque l'exécution est terminée, il doit indiquer l'heure et la date d'achèvement. Il demande ensuite un autre numéro de batch et un autre point de consigne pour l'exécution suivante. La figure 3-23 montre un exemple d'affichage de sortie après une exécution.

Configuration du bloc Raw_Comms

On suppose que le terminal VT100 a été configuré pour 9600 Bauds, 7 bits de données, une parité paire, 1 bit d'arrêt et aucun écho local. On suppose également que la touche DEL doit être utilisée pour la suppression et qu'elle produit le code ASCII 7Fh. Aucun contrôle de débit ne sera utilisé dans un sens et dans l'autre car on suppose que le terminal et le PC3000 ont une mémoire tampon de réception suffisamment importante. Le tableau 3-34 montre les valeurs des paramètres du bloc fonction qui sont nécessaires pour cet exemple.

Paramètre	Valeur	Commentaires
Port	'OB'	Port B sur le LCM.
Tx_Baud	_9600	Débit d'émission : 9600 Bauds.
Tx_CTS_Ctl	Off	Le contrôle de débit CTS n'est pas nécessaire.
Rx_Baud	_9600	Débit de réception : 9600 Bauds.
Rx_RTS_Ctl	Off	Le contrôle de débit RTS n'est pas nécessaire.
Data_Bits	_7	7 bits de données (réception et émission).
Parity	Even	Parité paire (réception et émission).
Stop_Bits	_1	1 bit d'arrêt (réception et émission).
Tx_Value	''	†
Tx_Break	Off	Il n'est pas nécessaire d'envoyer BREAK dans cette application.
Tx_State	Ok	†
Tx_Trig	Off	Tx_Trig n'est pas nécessaire car l'émission est contrôlée par le programme séquentiel avec Tx_State .
Rx_State	Ok	†
Rx_Trig	Off	Rx_Trig n'est pas nécessaire dans ce cas car la réception est contrôlée par le programme séquentiel avec Rx_State .
Flsh_Rx_Trig	Off	Flsh_Rx_Trig n'est pas nécessaire car cette application ne nécessite pas le vidage de la mémoire tampon de réception.
Rx_Term	'$R'	Le retour chariot met fin à la saisie.
Rx_Max_Len	128	Nombre maximal de caractères qui peuvent être saisis : 128.
Rx_Del_Key	'$7F'	La touche suppression du VT100 restitue le code ASCII 7Fh.
Rx_Del_Echo	'$08 $08'	Pour supprimer un caractère sur le VT100, il est nécessaire d'envoyer '<BS><espace><BS>'.
Rx_Echo	On	Le renvoi d'un écho par le PC3000 est nécessaire.

† Il s'agit de valeurs initiales ; le paramètre sera contrôlé par le programme séquentiel.

Tableau 3-34 Exemple de configuration des paramètres

Contrôle séquentiel de la machine

On suppose que cet exemple d'application est entièrement contrôlé par le programme séquentiel. Les SFC (grafcets) pour cette application sont présentés sur la figure 3-9.

N.B. : les trois phases du process sont rigoureusement identiques dans l'exemple et leurs SFC/ST sont indiqués une seule fois sous la forme Phase_n. Il est évident que les trois phases du process ne seraient pas identiques dans la pratique mais, dans l'exemple, la partie contrôle de process de l'application n'est pas représentée pour des raisons de clarté.

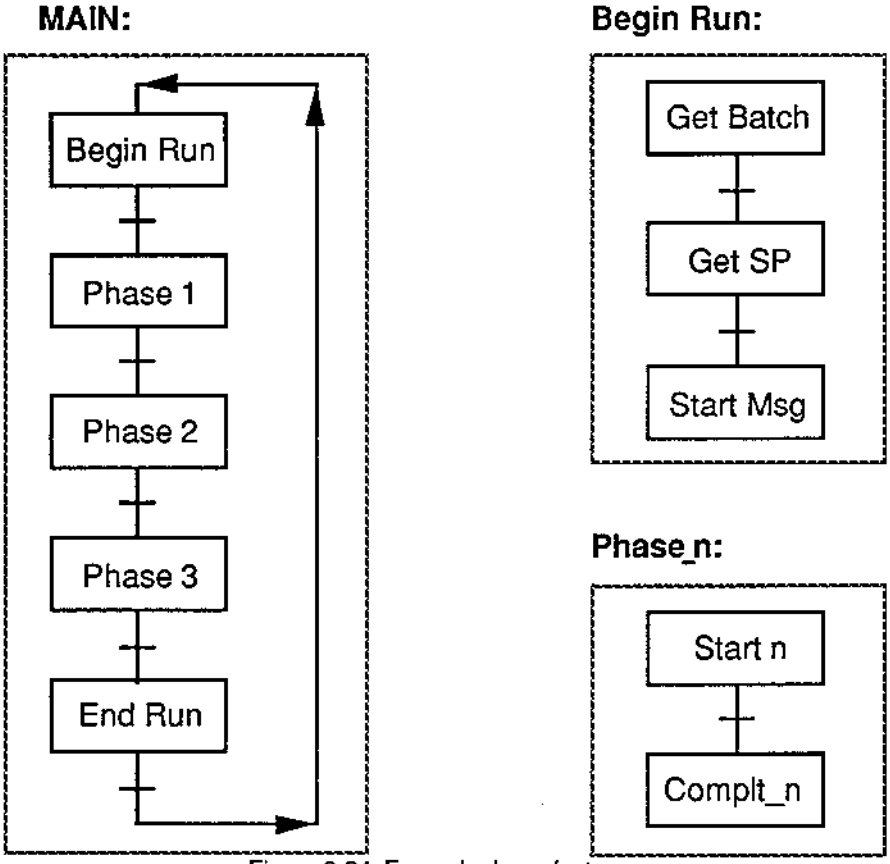


Figure 3-24 Exemple de grafcets

BeginRun: Get Batch, la première étape dans BeginRun , sort un message/message-guide en affectant une chaîne à **Tx_Value** et en positionnant **Tx_State** sur **Write** .

> **N.B. :** l'utilisation de 'RL' (retour chariot, nouvelle ligne) pour passer à la ligne suivante permet l'affichage de deux lignes par l'envoi d'une chaîne.

Une fois que le message-guide a été envoyé, **Rx_State** est positionné sur **Read** qui déclenche la saisie du numéro de batch. Le passage de GetBatch à Get_SP ne se fait pas avant que la saisie soit terminée. La saisie des données est arrêtée soit par l'opérateur qui tape un retour chariot soit par la saisie de 128 caractères. La limite de 128 caractères ne devrait normalement pas être utilisée mais est nécessaire car il s'agit du nombre maximal de caractères qui peut être contenu dans la chaîne **Rx_Value**. Un test permettant de savoir si **Tx_State** est **Ok** est également inclus dans la transition entre GetBatch et Get_SP, afin de garantir que le message-guide de numéro de batch a été affiché avant que la phase Get_SP affiche le message-guide de point de consigne. Une fois que la chaîne de numéro de batch a été saisie, Get_SP élimine le retour chariot de la fin de **Rx_Value** et le place dans une variable utilisateur.

> **N.B. :** pour des raisons de simplicité de l'exemple, aucune vérification n'a été effectuée sur la chaîne, bien qu'il puisse être nécessaire de valider le numéro de batch, d'une manière ou d'une autre, dans une application réelle. Il faut aussi noter que le numéro de batch est conservé dans une structure de chaîne. Cela permet l'utilisation d'un numéro de batch alpha-numérique.

La phase Get_SP, une fois qu'elle a traité le numéro de batch, affiche un message demandant le point de consigne et lance la lecture comme phase GetBatch. Une fois que le point de consigne a été lu, il est décodé (à l'aide de la fonction **STRING_TO_REAL**) et mémorisé dans une variable utilisateur par la phase StartMsg. Le reste de la phase StartMsg compose le message '... Calcul démarré ...' dans **Tx_Value** à l'aide d'un certain nombre d'enchaînements puis l'envoie en positionnant **Tx_State** sur **Write**. La fonction CONCAT/ sert à regrouper les chaînes.

> **N.B. :** aucune validation du point de consigne n'est indiquée dans cet exemple pour des raisons de clarté, bien qu'elle puisse être nécessaire dans une application réelle.

Phase_n : les trois phases du process sont mises en oeuvre sous la forme de macros composées de deux phases et sont toutes identiques ; elles ne diffèrent que par le nom. La phase Start_n sort le message '**PHASE n .**' de la même manière que précédemment. Tout séquencement du process serait inclus dans cette phase. Ensuite, la phase Cmpltd_n sort le message 'Terminé.' pour montrer que la phase est achevée.

End_Run : la phase End_Run est très comparable à la deuxième moitié de la phase StartMsg step, sauf qu'elle compose et démarre l'émission du message

'Batch ... terminé ..'. Une fois que le message a été envoyé, le flux de contrôle revient sur BeginRun qui demande l'exécution du batch suivant.

ST pour l'exemple de macro d'application MAIN

```
INITIAL_STEP  MAIN : (* MACRO *)

TRANSITION
      FROM BeginRun (* MACRO *)
      TO   Phase_1 (* MACRO *)
:=  VT100.Tx_State = 0(*Ok*)
END_TRANSITION

TRANSITION
      FROM Phase_1 (* MACRO *)
      TO   Phase_2 (* MACRO *)
:=  VT100.Tx_State = 0(*Ok*)
END_TRANSITION

TRANSITION
      FROM Phase_2 (* MACRO *)
      TO   Phase_3 (* MACRO *)
:=  VT100.Tx_State = 0(*Ok*)
END_TRANSITION

TRANSITION
      FROM Phase_3 (* MACRO *)
      TO   End_Run
:=  VT100.Tx_State = 0(*Ok*)
END_TRANSITION

STEP  End_Run :
      VT100_Tx_Value          := CONCAT( IN1 :=  'Batch number ' , IN2
:=                                      BtchName.Val ) ;
      VT100_Tx_Value          :=  CONCAT( IN1 :=  VT100.Tx_Value , IN2
:=  '                                     completed at ' ) ;
      VT100_Tx_Value          :=  CONCAT( IN1 :=  VT100.Tx_Value , IN2
:=                                      TIME_OF_DAY_TO_STR( IN :=
                                              RT_Clock.Time_Of_Day ) )
;
      VT100_Tx_Value          :=  CONCAT( IN1 :=  VT10.Tx_Value , IN2 :=
'                                       on ' ) ; VT100_Tx_Value   :=
                                      CONCAT( IN1 :=  VT100.Tx_Value ,
IN2 :=                                        DATE_TO_EURO_STRING(
IN:=                                                  RT_Clock.Date )
) ;
```

```
      VT100_Tx_Value          :=  CONCAT( IN1 :=  VT100.Tx_Value , IN2
:=                                      '.RLRL' ) ;
      VT100_Tx_State          := 3(*Write*) ;
      END_STEP
TRANSITION
      FROM End_Run
      TO   BeginRun (* MACRO *)
   :=  VT100.Tx_State = 0(*Ok*)
   END_TRANSITION

END_STEP (* MAIN *)
```

ST pour l'exemple de macro d'application BeginRun

```
STEP  BeginRun : (* MACRO *)
   STEP  GetBatch :
      VT100_Tx_Value    :=
               Please enter batch number and setpoint to start
               next run.$R$LBatch number = ' ;
      VT100_Tx_State    :=  3(*Write*) ;
      VT100_Rx_State    :=  3(*Read*) ;
   END_STEP

   TRANSITION
      FROM GetBatch
      TO   Get_SP
   :=  ( VT100.Rx_State = 0(*Ok*) ) AND ( VT100.Tx_State = 0(*Ok*) )
   END_TRANSITION

   STEP  Get_SP :
      BtchName_Val      := LEFT( IN := VT100.Rx_Value , L :=
                              LEN( IN :=  VT100.Rx_Value )- 1 ) ;
      VT100_Tx_Value    := 'LSetpoint = ' ;
      VT100_Tx_State    := 3(*Write*) ;
      VT100_Rx_State    := 3(*Read*) ;
END_STEP

TRANSITION
      FROM Get_SP
      TO   StartMsg
   :=  ( VT100.Rx_State = 0(*Ok*) ) AND (
   VT100.Tx_State = 0(*Ok*) )
END_TRANSITION

STEP  StartMsg :
```

```
        Setpoint_Val      := STRING_TO_REAL( IN := VT100.Rx_Value ) ;
        VT100_Tx_Value    := CONCAT( IN1 := 'LBatch number ' , IN2 :=
                               BtchName.Val ) ;
        VT100_Tx_Value    := CONCAT( IN1:= VT100.Tx_Value , IN2 := ',
                               Setpoint ' );
        VT100_Tx_Value    := CONCAT( IN1 := VT100.Tx_Value , IN2:=
                               REAL_TO_STRING( IN := Setpoint.Val, DPS:=
                               1 ) ) ;
        VT100_Tx_Value    := CONCAT( IN1 := VT100.Tx_Value , IN2 := '
                               - Run started at ' ) ;
        VT100_Tx_Value    := CONCAT( IN1 := VT100.Tx_Value , IN2 :=
                               TIME_OF_DAY_TO_STR( IN:=
                             RT_Clock.Time_Of_Day ) ) ;
        VT100_Tx_Value    := CONCAT( IN1 := VT100.Tx_Value ,  IN2 := '
                               on ' ) ;
        VT100_Tx_Value    := CONCAT( IN1 := VT100.Tx_Value , IN2 :=
                               DATE_TO_EURO_STRING( IN :=
                               RT_Clock.Date ) ) ;
        VT100_Tx_Value    := CONCAT( IN1 := VT100.Tx_Value ,  IN2 :=
                               '.RL' ) ;
        VT100_Tx_State    := 3(*Write*) ;
    END_STEP
END_STEP (* BeginRun *)
```

ST pour l'exemple de macros d'application Phase_ n

```
STEP  Phase_n : (* MACRO *)

    STEP  Start_n :
        VT100_Tx_Value  := 'Phase n.  '  ;
        VT100_Tx_State  := 3(*Write*) ;
    END_STEP

    TRANSITION
        FROM Start_n
        TO   Cmpltd_n
    :=  VT100.Tx_State = 0(*Ok*)
    END_TRANSITION

    STEP  Cmpltd_n :
        VT100_Tx_Value   := 'Completed.RL' ;
        VT100_Tx_State   := 3(*Write*) ;
    END_STEP
END_STEP (* Phase_n *)
```

Affichage des caractères de contrôle dans les chaînes

Sur la station de programmation PC3000, les caractères de contrôle non imprimables peuvent être insérés dans une chaîne utilisant la structure $ 'dollar'. La même structure est utilisée pour afficher le contenu des chaînes. Le signe $ sert à indiquer que le(s) caractère(s) qui suit(suivent) définit(définissent) un caractère de contrôle. Les structures utilisées sont les suivantes :

$nn	- caractère avec un code ASCII 'nn' en
	hexadécimal, par exemple 0A
$$	- signe dollar
$'	- apostrophe
$L	- changement de ligne (ASCII 0A en hexadécimal)
$N	- nouvelle ligne (converti en une paire RL)
$P	- changement de page (ASCII 0C en hexadécimal)
$R	- retour chariot (ASCII OD en hexadécimal)
$T	- tabulation (ASCII 09 en hexadécimal)

Tableau 3-35 Caractères de contrôle pour les chaînes

Signalisation des erreurs

Les codes d'erreur qui apparaissent dans **Rx_Error_No** et **Tx_Error_No** sont indiqués dans le tableau 3-36. Certains codes d'erreur peuvent apparaître lors de la première exécution du bloc, indiquant un défaut d'initialisation du port. Ces erreurs sont consignées à la fois dans **Rx_Error_No** et **Tx_Error_No** et sont repérées par un † dans le tableau. Le code d'erreur 0 représente OK et indique que le bloc fonctionne normalement. Le tableau montre dans le code d'erreur 0 (OK) les situations dans lesquelles **Tx/Rx_Error_No** s reviendra à zéro à la suite d'une erreur.

> **N.B. :** il est possible de forcer à nouveau les codes d'erreur à zéro en écrivant Ok dans le paramètre Rx/Tx_State qui convient.

Rx_Error_No	Tx_Error_No	Description de l'erreur
0	0	OK •Le port a été initialisé correctement. •Une réception a été arrêtée par passage de Rx_State de En attente à Ok . •Une erreur de réception a été supprimée par passage de Rx_State d'Erreur à Ok . •Une transmission a été arrêtée par passage de Tx_State de En attente à Ok . •Une erreur d'émission a été supprimée par passage de Rx_State d'Erreur à Ok . •Une émission s'est achevée correctement. •Une chaîne a été reçue correctement. •Un vidage de mémoire tampon de réception s'est achevé correctement.
1	1	ABSENCE D'ADRESSE DE PORT † •La chaîne Port a moins de deux caractères.
2	2	BITS DE DONNEES DEMANDES PAS DISPONIBLES † •Le port spécifié ne peut pas prendre en charge le nombre de bits de données sélectionné.
3	3	CONFIGURATION DE PARAMETRE DEMANDEE PAS DISPONIBLE † •La parité sélectionnée n'est pas prise en charge par ce port.
4	4	BIT D'ARRET DEMANDES PAS DISPONIBLES † •Le nombre de bits d'arrêt demandé n'est pas pris en charge par ce port.
5	5	DEBIT RX PAS DISPONIBLE † •Le débit de réception choisi n'est pas disponible sur ce port.
6	6	DEBIT TX PAS DISPONIBLE † •Le débit d'émission choisi n'est pas disponible sur ce port.
7	7	RTS PAS DISPONIBLE † •Le contrôle de débit RTS n'est pas disponible sur ce port.
8	8	CTS PAS DISPONIBLE † •Le contrôle de débit CTS n'est pas disponible sur ce port.

Tableau 3-36 Codes d'erreur Raw_Comms

Rx_Error_No	Tx_Error_No	Description de l'erreur
11	11	EMPLACEMENT INTERDIT † •Le numéro d'emplacement spécifié dans le paramètre Port ne contient aucun port série.
12	12	PORT INTERDIT † •L'emplacement spécifié n'a aucun port possédant le numéro spécifié dans le paramètre Port.
14	14	DEBITS RX & TX INCOMPATIBLES † •La combinaison des débits d'émission et de réception choisis n'est pas possible sur ce port.
17	17	PORT UTILISE † •Un autre bloc de module a été déjà annexé à ce port.
-	95	ERREUR TX FORCEE •Le paramètre Tx_State a été positionné sur Erreur pour forcer une erreur.
96	-	ERREUR RX FORCEE •Le paramètre Rx_State a été positionné sur Erreur pour forcer une erreur.
97	-	ECHO PERDU •Tx_Break était sur On lorsqu'il a reçu un caractère qui devait faire l'objet d'un écho, ce qui a provoqué sa perte.. •La capacité de la mémoire tampon d'écho a été dépassée, ce qui a provoqué une perte des caractères d'écho.
98	-	CARACTERES RX PERDUS •La capacité de la mémoire tampon de réception a été dépassée, ce qui a provoqué la perte de certains caractères. •Rx_State a été passé d'En attente à Lecture, provoquant le démarrage d'une nouvelle lecture après le vidage de la mémoire tampon de réception.
-	99	CARACTERES TX PERDUS •Tx_Break a été activé pendant l'émission des caractères, ce qui a provoqué la perte de certains caractères. •Il y a eu une tentative d'émission de caractères pendant que Tx_Break était activé, ce qui a provoqué un arrêt de l'émission.
100	-	ERREUR DE DEBORDEMENT RX •Une erreur de débordement a été détectée sur un caractère reçu.
101	-	ERREUR DE PARITE RX •Une erreur de parité a été détectée sur un caractère reçu.

Tableau 3-36 Codes d'erreur Raw_Comms (suite)

Rx_Error_No	Tx_Error_No	Description de l'erreur
102	-	ERREUR DE PARITE ET DE DEBORDEMENT RX •Une erreur de parité et de débordement a été détectée au cours de la réception.
103	-	ERREUR D'ENCADREMENT RX •Une erreur d'encadrement a été détectée sur un caractère reçu.
104	-	ERREUR D'ENCADREMENT ET DE DEBORDEMENT RX •Une erreur d'encadrement et de débordement a été détectée au cours de la réception.
105	-	ERREUR D'ENCADREMENT ET DE PARITE RX •Une erreur d'encadrement et de parité a été détectée au cours de la réception.
106	-	ERREUR D'ENCADREMENT, DE DEBORDEMENT ET DE PARITE RX •Une erreur d'encadrement, de parité et de débordement a été détectée au cours de la réception.

Tableau 3-36 Codes d'erreur Raw_Comms (suite)

Attributs de paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	STRING	'0A'	Oper	Config		
Tx_Baud	ENUM	_600	Config	Config	Valeurs énumérées	_75(0) _300(1) _600(2) _1200(3) _2400(4) _4800(5) _9600(6) _19200(7) _38400(8) _57600(9) _115200(10)
Tx_CTS_Ctl	BOOL	ON	Config	Config	Détection	OFF(0) ON(1)
Rx_Baud	ENUM	_600	Config	Config	Valeurs énumérées	_75(0) _300(1) _600(2) _1200(3) _1200(3) _2400(4) _4800(5) _9600(6) _19200(7) _38400(8) _57600(9) _115200(10)
Rx_RTS_Ctl	BOOL	ON	Config	Config	Détection	OFF(0) ON(1)
Data_Bits	ENUM	_7	Config	Config	Valeurs énumérées	_5(0) _6(1) _7(2) _8(3)
Parité	ENUM	SPACE	Config	Config	Valeurs énumérées	PAIR(0) IMPAIR(1) ESPACE(2) MARQUE(3) NEANT(4)
Stop_Bits	ENUM	_2	Config	Config	Valeurs énumérées	_1(0) _1_5(1) _2(2)

Tableau 3-37 Attributs des paramètres Raw_Comms (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Tx_Value	STRING	''	Oper	Oper		
Tx_Break	BOOL	ON	Oper	Oper	Détection	OFF(0) ON(1)
Rx_Trig	BOOL	YES	Oper	Oper	Détection	NO(0) YES(1)
Fish_Rx_Trig	BOOL	YES	Oper	Oper	Détection	NO(0) YES(1)
Rx_Term	STRING	''	Oper	Oper		
Tx_State	ENUM	ERROR	Oper	Oper	Valeurs énumérées	OK(0) EN ATTENTE(1) ERREUR(2) ECRITURE(3)
Rx_Max_Len	SINT	1	Oper	Oper	Lim. haute Lim. basse	128 1
Rx_Del_Key	STRING	''	Oper	Oper		
Rx_Del_Echo	STRING	''	Oper	Oper		
Rx_Echo	BOOL	ON	Oper	Oper		
Status	BOOL	GO	Oper	Block	Valeurs énumérées	NOGO(0) Go(1)
Rx_Error_no	SINT	0	Oper	Block	Lim. haute Lim. basse	255 0
Tx_Error_no	SINT	0	Oper	Block	Lim. haute Lim. basse	255 0
Tx_State_Trk	BOOL	YES	Super	Block	Détection	NO(0) YES(1)
Rx_State_Trk	BOOL	YES	Super	Block	Détection	NO(0) YES(1)
Rx_Value	STRING	''	Oper	Block		
Rx_Break	BOOL	YES	Oper	Block	Détection	NO(0) YES(1)

Tableau 3-37 Attributs des paramètres Raw_Comms (suite)

BLOC FONCTION SIEMENS_M_S

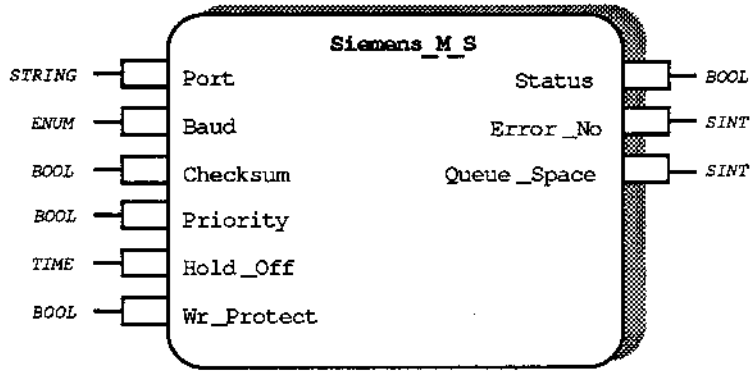


Figure 3-25 Diagramme du bloc fonction

Description fonctionnelle

Le bloc fonction Siemens_M_S prend en charge les communications série sur un port de communications série désigné à l'aide du protocoleSiemens 3964(R). Ce bloc fonction applique la procédure 3964(R) et l'interpréteur RK 512 comme cela est indiqué dans la spécification du protocole Siemens, avec des modifications pour garantir la conformité avec la méthodologie des blocs fonctions PC3000.

Deux fonctions de l'interpréteur RK 512 ne sont pas prises en charge par la mise en oeuvre du PC3000 :

- les télégrammes de suivi ne sont pas pris en charge.
- le seul 'type de commande' pris en charge est le bloc de données (D).

Avant de lire cette description, vous avez intérêt à acquérir des connaissances générales sur le système de communications PC3000 en lisant la présentation des communications PC3000.

Ce bloc fonction est nécessaire lors de la conception ou de la programmation du PC3000 pour qu'il communique avec un seul PLC Siemens utilisant une liaison série pour une connexion point à point. Les détails de ce manuel seront utiles pour le développement d'applications qui utilisent le bloc fonction de module de communications Siemens. Du fait que le module Siemens peut fonctionner à la fois en mode maître et en mode esclave, ce document comporte aussi des information sur l'utilisation des variables esclaves et déportées.

Il faut consulter les documents suivants si l'on souhaite connaître en détail le protocole Siemens 3964(R) :

- Siemens Aktiengesellschaft. Logiciel de programmation Siemens COM525 pour les processeurs de communications CP524 et CP525 (S5-DOS), guide utilisateur, volume 1 (chapitre 7, section 5).

•Siemens Aktiengesellschaft. Logiciel de programmation Siemens COM525 pour les processeurs de communications CP524 et CP525 (S5-DOS), guide utilisateur, volume 1 (chapitre 7, section 4).

•Siemens Aktiengesellschaft. Logiciel de programmation Siemens COM525 pour les processeurs de communications CP524 et CP525 (S5-DOS), guide utilisateur, volume 1 (chapitre 7, section 4.1).

•Siemens Aktiengesellschaft. Logiciel de programmation Siemens COM525 pour les processeurs de communications CP524 et CP525 (S5-DOS), guide utilisateur, volume 1 (chapitre 7, section 8.2).

Terminologie spéciale

Pour respecter la terminologie Siemens, nous avons adopté le terme suivant :

Partenaire : autre terme pour désigner le périphérique déporté qui est en communication avec le PC3000 ; il s'agit généralement d'un PLC Siemens.

Ce bloc fonction traite les détails des communications avec un PLC Siemens propres au protocole ; il est pris en charge par des blocs fonctions génériques de variables déportées et de variables esclaves. Les blocs de variables déportées et de variables esclaves sont liés au module au moyen d'une adresse propre au protocole. Les blocs de variables déportées correspondent au côté maître du module et déclenchent les lectures et les écritures dans les périphériques déportés. Du côté esclave, les blocs de variables esclaves définissent les valeurs qui peuvent être lues ou écrites par un périphérique déporté utilisant le type de protocole défini dans le paramètre Adresse ('SI' pour Siemens dans le cas présent).

N.B. : ce module prend en charge les blocs fonctions Chaîne déportée/esclave et Entier déporté/esclave.

Attributs du bloc fonction

Type : 8 70

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port Status Queue_Space

Capacité mémoire nécessaire : 2580 octets

Description des paramètres

Paramètres de configuration du module

Le bloc Siemens_M_S possède un certain nombre d'entrées qui configurent différents aspects du module et doivent être fixées avant le lancement du programme. La modification de ces paramètres au cours de l'exécution du programme utilisateur n'a aucun effet sur le module, sauf dans des circonstances particulières : cf. 'Présentation des communications PC3000', section

'Modification temporaire des paramètres de configuration'. Les seules entrées dans le bloc fonction Siemens_M_S qui peuvent être modifiées lorsqu'il est en marche sont **Hold_Off** et **Wr_Protect**.

Port

Le paramètre **Port** est l'adresse à deux caractères du port sur laquelle doit être exécutée la procédure 3964(R). Le premier caractère est un chiffre compris entre 0 et 5 représentant l'emplacement dans le rack et le deuxième caractère est une lettre représentant le port dans cet emplacement. Par exemple, '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' pourrait désigner son emplacement supérieur.

Baud

Le paramètre **Baud** offre un choix de 11 débits différents compris entre 75 Bauds et 115,2 kBauds (cf. tableau 4-2), avec une valeur par défaut de 9600 Bauds.

> **N.B.** : tous les ports ne peuvent pas prendre en charge la totalité des débits. Si un débit n'est pas pris en charge, une erreur sera indiquée lors de la première exécution du bloc fonction (cf. la description d'**Error_No**).

Valeur énumérée	Débit
0	75
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	115200

Tableau 3-38 Débits Siemens_M_S

Checksum

Lorsque le paramètre **Checksum** est sur **Yes**, la procédure 3964R qui comporte la vérification par total de contrôle est utilisée. Lorsque **Checksum** est sur **No**, la procédure 3964, qui n'effectue pas la vérification par total de contrôle, est utilisée. Les deux procédures sont identiques à tous les autres égards.

Priorité

Le paramètre **Priorité** doit être positionné de telle manière qu'une extrémité de la liaison soit **Haute** et l'autre **Basse**. Il servira à déterminer l'extrémité de la liaison qui doit céder en cas de conflit sur la liaison.

Hold_Off

Le paramètre **Hold_Off** est la durée pendant laquelle le module attendra lorsqu'une écriture est protégée avant de renvoyer un télégramme d'erreur au partenaire. Pour avoir une explication sur la manière de bloquer les travaux et sur les valeurs suggérées pour ce paramètre, cf. page 3-134.

Paramètres d'état du module

L'état du module est indiqué par trois paramètres de sortie dans le bloc fonction Siemens_M_S.

Etat

Le paramètre **Etat** est une indication booléenne de l'état de la liaison contrôlée par le module. En l'absence de problème sur la liaison, ce paramètre est sur **Go** mais, lorsqu'une erreur se produit, il passe sur **NoGo**. Si **Etat** est sur **NoGo**, le paramètre **Error_No** indique la cause du problème.

Error_No

Le paramètre **Error_No** indique la cause des éventuelles erreurs sur la liaison. Si l'**Etat** de la liaison est sur **Go**, **Error_No** sera 0 (OK). Pour avoir des détails complets sur le traitement des erreurs et les codes d'erreur, consulter la section Signalisation des erreurs page 3 et les codes d'erreur page 3.

Queue_Space

Le paramètre **Queue_Space** indique l'espace restant dans la file d'attente pour les opérations sur les paramètres déportés. S'il atteint zéro, cela implique que la largeur de bande de la liaison est insuffisante pour faire face au nombre de demandes de variables déportées effectuées et les données seront perdues. Si ce cas se produit, il faut réduire les vitesses d'interrogation du paramètre.

Protection en écriture du module

Wr_Protect

Le paramètre de saisie **Wr_Protect** sur le bloc de module Siemens_M_S est une protection en écriture globale qui peut servir à désactiver l'écriture dans toutes les adresses par ce port de module.

Fonctionnement avec les variables déportées

Les demandes maîtres effectuées par le PC3000 sont contrôlées par un ou plusieurs blocs de variables déportées. Si seul le fonctionnement esclave est nécessaire, les blocs de variables déportées ne sont pas indispensables. Le module Siemens_M_S prend en charge les paramètres Remote_Str, Remote_Int et Remote_SW.

Adressage

Il est nécessaire de définir une **Adresse** propre au protocole dans le bloc de variable déportée qui est l'adresse utilisée pour accéder au partenaire. La figure 4-2 montre un exemple d'adresse. Le port est défini, comme dans le paramètre **Port** du bloc fonction Siemens_M_S, sous la forme d'un numéro d'emplacement dans un rack suivi d'une lettre pour le port dans cet emplacement. La partie de l'adresse propre au protocole commence par le numéro d'unité centrale qui doit être normalement compris entre 1 et 7. Elle est suivie de la lettre de type de commande qui doit être D (bloc de données). Le numéro de bloc de données et le numéro de mot de données viennent ensuite, séparés par un point. La longueur du bloc à chercher ou à envoyer peut être indiquée de trois manières.

- La première option consiste à terminer l'adresse à cet endroit, en laissant le type de plage et l'adresse de longueur ou de fin. Cela implique une longueur de bloc implicite d'un mot, par exemple 0A1D7.5
- La deuxième option consiste à indiquer la longueur des mots ; dans ce cas, la longueur est précédée d'un signe plus, par exemple 0A2D20.16+8
- La dernière possibilité consiste à préciser une plage. Dans ce cas, le numéro de mot de données est suivi d'un tiret et du numéro de bloc de données final, par exemple 0A5D78.1-50

Dans la pratique, on utilise une des deux dernières méthodes pour indiquer une adresse et les autres méthodes sont uniquement données pour des raisons de facilité d'utilisation. Par exemple, '0A1D78.10+1', '0A1D78.10-10' et '0A1D78.10' représentent la même adresse. Il faut noter que la plage est inclusive.

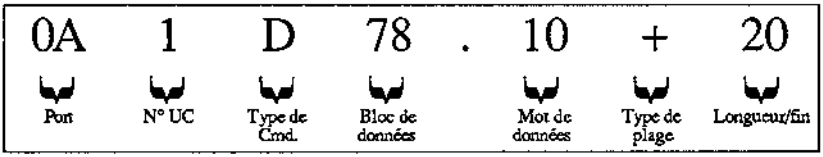


Figure 3-26 Exemple d'adresse de variable déportée

Utilisation de Remote_Str

Se reporter à la 'Présentation des communications PC3000' pour avoir des informations supplémentaires sur les blocs fonctions de variables déportées.

L'adresse indiquée dans le paramètre **Adresse** du bloc Remote_Str doit avoir une longueur de bloc inférieure à la moitié du paramètre **New_Value** lors de l'écriture, sinon une erreur est signalée, indiquant un nombre insuffisant de données à envoyer. Les modes de données sont représentés par des paires de caractères dans la chaîne, l'octet le plus significatif étant en première position.

Utilisation de Remote_Int

L'adressage d'entiers déportés est identique à l'adressage des chaînes déportées, sauf pour un aspect : dans le cas des entiers déportés, l'adresse doit indiquer une longueur de bloc d'un ou deux mots seulement. Si l'adresse indique une longueur de bloc d'une unité, elle peut servir à lire ou écrire un entier d'un mot. Pour les écritures d'un entier d'un mot, les 16 bits les moins significatifs de la valeur seront écrits. Si l'adresse a une longueur de deux mots, l'entier complet sera écrit sous la forme de deux mots, le mot le plus significatif étant en tête.

Utilisation de Remote_S_W

Les mots d'état déporté sont une forme spéciale d'entier qui est décomposé en bits dont il est constitué. Les adresses des mots d'état doivent avoir une longueur d'un mot seulement. Avec le module Siemens, le mot d'état (SW) est traité de la même manière qu'un entier et peut donc avoir une longueur de deux mots. Si l'on utilise une adresse de deux mots de long, le module n'émettra pas d'objection mais le mot le plus significatif sera écrit comme étant 0 et ne sera pas pris en compte lors de la lecture.

Fonctionnement avec la variable esclave

La moitié esclave du module Siemens sert à simuler un PLC Siemens dans le PC3000. L'espace d'adressage de ce pseudo PLC est défini par un certain nombre de blocs fonctions de variables esclaves. Le module Siemens prend **UNIQUEMENT** en charge Slave_Str et Slave_Int.

Adressage

L'adresse de la variable esclave utilise la même structure que l'adresse de la variable déportée, sauf qu'elle a un mnémonique de module à deux lettres qui, dans le cas présent, est 'SI' pour Siemens, au lieu de l'adresse de port (cf. figure 3-27)

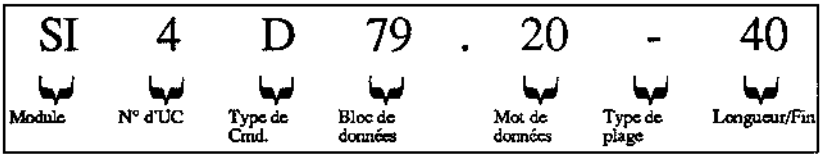


Figure 3-27 Exemple d'adresse de variable esclave

Utilisation des variables esclaves.

Consulter la 'Présentation des communications PC3000' et le Manuel des blocs fonctions PC3000 pour avoir des informations sur l'utilisation des blocs fonctions de variables esclaves.

L'esclave répond à n'importe quelle lecture ou écriture du partenaire (périphérique déporté) si elles se situent dans la plage spécifiée par le paramètre **Adresse**. Si un certain nombre de blocs fonctions de variables esclaves ont des adresses consécutives, par exemple 1D79.10-19, 1D79.20+10, 1D79.30..., le partenaire peut écrire dans plusieurs variables esclaves avec une seule écriture de bloc (mots 10-30 par exemple). Pour les écritures de paramètres multiples, si un paramètre d'un bloc est protégé en écriture, l'ensemble de l'écriture sera bloqué. Il faut noter que le module Siemens PC3000 ne prend pas en charge les drapeaux de coordination. L'adresse est spécifiée en termes de type de module et non d'adresse de port car une variable esclave peut répondre à des demandes provenant de n'importe quels ports possédant le même module.

> **N.B. :** comme les entiers déportés, les entiers esclaves doivent avoir une adresse d'un ou deux mots de longueur pour représenter les entiers 16 ou 32 bits. Dans ce cas aussi, les entiers 32 bits sont représentés avec le mot le plus significatif en première position.

Protection en écriture.

Une variable esclave peut être protégée de deux manières contre l'écriture par le protocole de communications Siemens. Dans le bloc du module, le paramètre **Wr_Protect** peut servir à assurer une protection globale contre toutes les écritures par le port spécifié dans le paramètre **Port** de ce bloc. Si, par exemple, un périphérique de surveillance devait être relié au PC3000 à l'aide de Siemens 3964(R), le paramètre **Wr_Protect** serait positionné sur **Yes** pour protéger les variables esclaves PC3000 contre les écritures.

La deuxième forme de protection s'effectue variable esclave par variable esclave. Chaque variable esclave possède un paramètre **Mode** qui peut être **Rd_Wr**, **Rd_Only** ou **Wr_Once**. Dans le mode **Rd_Wr**, il n'y a aucune restriction à la lecture ou à l'écriture du paramètre par un périphérique déporté. En mode **Rd_Only**, les lectures sont permises sans restriction mais les écritures

provenant d'un périphérique déporté renvoient un télégramme de réponse indiquant une erreur, informant le périphérique déporté que l'écriture a été rejetée.

> **N.B. :** il est possible de bloquer cette réponse indiquant une erreur pendant un court moment pour permettre le traitement des valeurs reçues et la suppression de la protection en écriture.

Cette situation entraînera la consignation de l'erreur DB/DX DISABLED BY CF AT PARTNER dans le périphérique déporté. Le mode **Wr_Once** est utilisé dans les cas où il est nécessaire de traiter chaque valeur telle qu'elle est reçue dans le bloc de la variable esclave. Le mode **Wr_Once** fonctionne initialement de la même manière que le mode **Rd_Wr**, permettant la lecture et l'écriture par le périphérique déporté. La différence survient une fois que la valeur a été écrite. Lorsqu'une écriture se produit alors que l'on est en mode **Wr_Once**, elle est autorisée mais, ensuite, le **Mode** passe immédiatement sur **Rd_Only**, empêchant l'écrasement de la valeur qui vient d'être reçue. Une fois que la valeur a été traitée par le programme utilisateur, le **Mode** peut revenir à **Wr_Once**, prêt pour la réception de la valeur suivante. Cette opération s'effectue depuis le programme séquentiel, par affectation de **Wr_Once** au paramètre **Mode**, ou dans le câblage en faisant passer le paramètre **Trig_Wr1** de **Off** à **On**, ce qui provoque alors le passage de **Mode** sur **Wr_Once**.

Blocage

Si la protection en écriture d'un paramètre n'est pas activée (c'est-à-dire si le **Mode** de la variable esclave est **Rd_Wr** ou **Wr_Once** et si le bloc de module **Wr_Protect** est sur **Off**), une écriture dans ce paramètre fait immédiatement l'objet d'un envoi d'accusé de réception à l'émetteur, comme le montre la figure 3-28, et la valeur est écrite dans le paramètre.

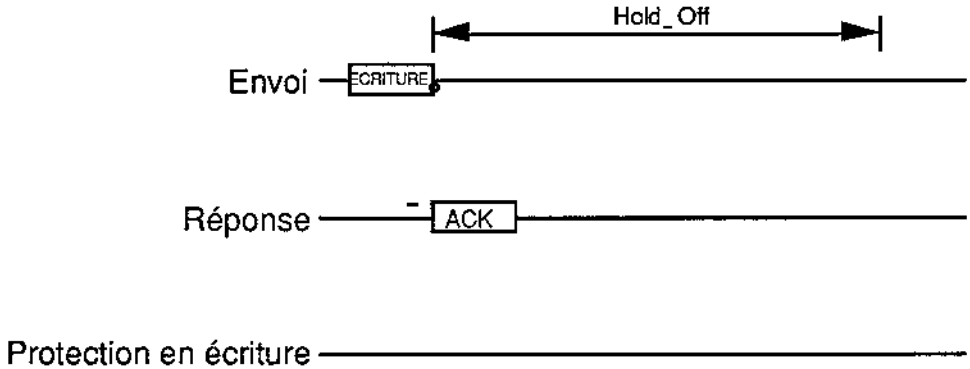


Figure 3-28 Ecriture avec la protection en écriture désactivée

Afin d'empêcher l'écrasement de la valeur de la variable esclave avant que le programme utilisateur ait pu la traiter, on peut faire retarder l'écriture et son accusé de réception par le module Siemens à l'aide des fonctions de protection en écriture décrites précédemment. Si la protection en écriture est activée lors de la réception d'une écriture, l'écriture et son accusé de réception seront retardés jusqu'à ce que la protection soit désactivée, comme le montre la figure 3-29.

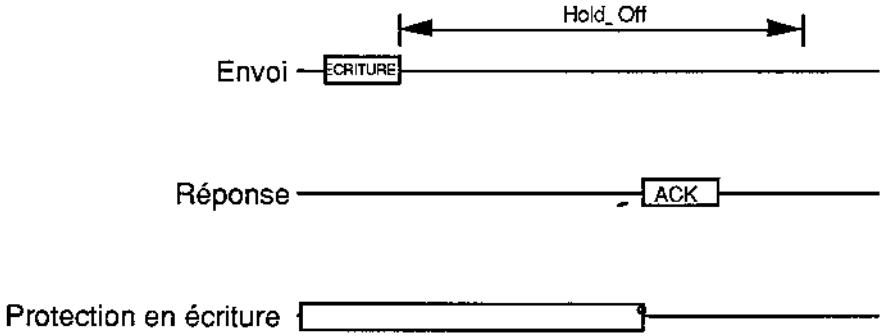


Figure 3-29 Temporisation de l'écriture lorsque la protection en écriture est activée

Si la protection en écriture n'est pas désactivée avant la fin de la durée de **Hold_Off**, un télégramme d'erreur sera renvoyé et l'écriture sera arrêtée. Cf. figure 3-30.

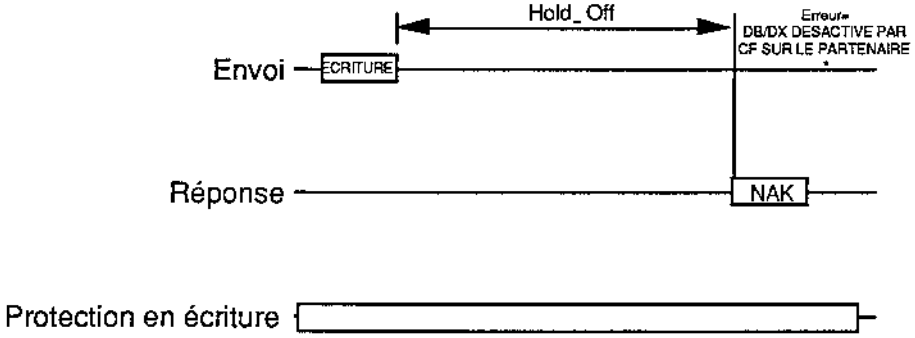


Figure 3-30 Echec de l'écriture provoquée par le fait que la protection en écriture est activée

Il faut positionner le paramètre **Hold_Off** dans le bloc fonction de module sur la durée maximale nécessaire pour traiter les données reçues. Cela garantit que, dans le cas où la protection en écriture reste activée, l'émetteur est informé le plus tôt possible que le paramètre est protégé. S'il n'est pas nécessaire de bloquer les écritures pour un application donnée, il est possible de positionner **Hold_Off** sur zéro ; dans ce cas, les écritures dans les paramètres protégés seront immédiatement signalées à l'émetteur comme une erreur. La figure 3-31 montre cette situation.

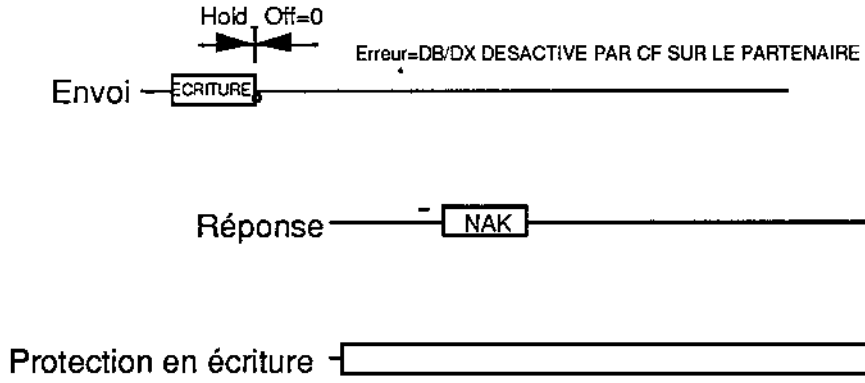


Figure 3-31 Ecriture avec Hold_Off égal à 0 et la protection en écriture activée

Il est important de noter que la temporisation de l'émetteur impose une limite supérieure à **Hold_Off**. La moitié de la procédure 3964(R) qui effectue l'envoi fixe une 'Durée de surveillance' après laquelle le télégramme est considéré comme perdu. Cette durée de surveillance varie en fonction du débit, elle est indiquée dans le tableau 3-41.

Débit	Durée de surveillance	Blocage maximal recommandé
75	40s	38s 400ms
300	10s	9s 500ms
600	7s	6s 700ms
1200	5s	4s 800ms
2400	5s	4s 800ms
4800	5s	4s 800ms
9600	5s	4s 800ms
19200	5s	4s 900ms
38400	5s	4s 900ms
57600	5s	4s 900ms
115200	5s	4s 900ms

Tableau 3-39 Durées de surveillance

La figure 3-32 montre ce qui se passe lorsqu'on donne à **Hold_Off** une valeur supérieure à la durée de surveillance.

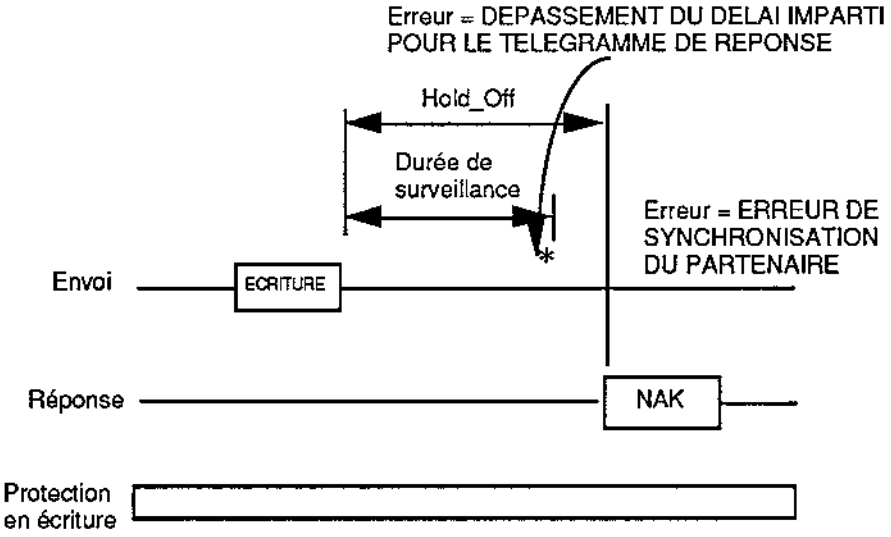


Figure 3-32 Ecriture avec une valeur de Hold_Off supérieure à la durée de surveillance

N.B. 1 : deux erreurs seront consignées par l'émetteur dans cette situation si la protection en écriture reste activée. La première erreur indiquera que la transaction d'émission a dépassé le délai imparti. La deuxième erreur sera due au fait que, lors de l'arrivée du télégramme de réponse, aucun télégramme d'envoi ne pourra lui être associé (parce que l'émission sera considérée comme ayant échoué). Il est conseillé de positionner **Hold_Off** sur le minimum et toujours à une valeur inférieure au maximum indiqué dans le tableau 3-41. Cela permet de garder une marge de manoeuvre pour le temps d'attente et de laisser au télégramme de réponse le temps de parvenir à l'émetteur .

N.B. 2 : les figures 3-42 à 3-45 ne montrent pas le temps d'attente. Par exemple, sur la figure 3-42, le télégramme d'accusé de réception ne sera pas renvoyé immédiatement après la réception du télégramme d'envoi, il y aura une légère temporisation de traitement.

Exemple

Cette section montre un exemple d'application. Nous ne donnons pas la totalité de la programmation séquentielle de l'application pour des raisons de clarté mais les blocs fonctions de communications feront l'objet d'une discussion détaillée.

Dans cet exemple d'application, un PLC Siemens sert à contrôler une machine. Ce contrôleur de machine accepte les commandes écrites dans un mot de données (DW 70.1 par exemple) sur le PLC et met à jour un mot d'état (DW 71.9 par exemple) qui peut être lu par le PC3000. Il envoie également des codes d'alarme non demandés à un mot de données (DW 50.0 par exemple) dans le PC3000. Lors de la première exécution, le contrôleur lit aussi un nom de batch à 6 caractères sur le PC3000 (DW 51.0-2 par exemple). Le fonctionnement détaillé de la machine n'a pas d'importance pour l'exemple. Les tableaux 3-40, 3-41 et 3-42 montrent les codes de commande de la machine, les bits d'état et les codes d'alarme pris pour cet exemple.

Commande	Code
marche	1
blocage	2
continuation	3
arrêt	4
réinitialisation	5

Tableau 3-40 Exemple de codes de commande pour une application sur une machine

Etat	Numéro de bit
marche	0
blocage	1
manuel/auto	2
alarme active	3
avertissement batterie	4
tolérance dépassée	5
échec m/c	6
arrêt d'urgence	7
mise en marche !	8

Tableau 3-41 Exemple de bits d'état pour une application sur une machine

Type d'alarme	Code
tolérance dépassée	1
échec m/c	2
arrêt d'urgence	3
reprise manuelle	4
mise en marche !	5

Tableau 3-42 Exemple de codes d'alarme pour une application sur une machine

Le port B sur le LCM est considéré comme relié au PLC Siemens, qui sera considéré comme ayant été programmé pour attendre la procédure 3964R avec un débit de 9600 Bauds et une priorité élevée. La priorité du Siemens serait positionnée sur "haute" pour que les messages d'alarme provenant de la machine aient la priorité sur les commandes émanant du PC3000. Le bloc fonction serait configuré comme le montre le tableau 3-43 pour cette application.

Paramètre	Valeur	Commentaires
Port	'0B'	'0' désigne l'emplacement 0 dans le rack (LCM) et 'B' désigne le port B.
Baud	_9600 (6)	
Checksum	Yes (1)	Ce paramètre sert à effectuer un choix entre les procédures 3964 et 3964R. 3964R, qui est utilisée dans cet exemple, est la procédure 3964 avec l'ajout d'un caractère de contrôle de bloc (total de contrôle).
Priority	Low (0)	Une extrémité de la liaison doit être sur priorité Basse et l'autre sur priorité Haute. L'extrémité à priorité basse de la liaison passera en deuxième position si une situation de conflit survient.
Hold_Off	1s	Lorsqu'un code d'alarme est reçu dans le programme utilisateur, il faut qu'il soit traité correctement dans la seconde suivante. S'il n'est pas traité dans ce délai et si une autre alarme se produit, elle sera signalée au PLC comme étant protégée en écriture.
Wr_Protect	No (0)	Dans cette application, il n'est jamais nécessaire de protéger en écriture l'ensemble des accès.

Tableau 3-43 Exemple de configurations du bloc fonction Siemens_M_S

Cet exemple d'application nécessite quatre blocs fonctions de paramètres de communications : deux variables déportées et deux variables esclaves. Les blocs de variables déportées sont un Remote_Int, pour envoyer des commandes, et un Remote_SW, pour récupérer les informations relatives à l'état de la machine. Les variables esclaves sont une Slave_Int, pour recevoir les codes d'alarme, et une Slave_Str, pour le nom de batch à lire. Il faut noter la différence entre les blocs de variables esclaves et les blocs de variables déportées. Les variables déportées sont associées à la moitié maître du module et aux données *récupérées* ou *envoyées*. Les blocs de variables esclaves sont associés à la moitié esclave du module et peuvent être *lus* ou *écrits* par le partenaire, c'est-à-dire un PLC Siemens.

Pour que l'envoi de commandes au PLC soit possible, il faut créer un bloc fonction de variable déportée appelé MC_Cmd dans cet exemple. Le tableau 3-44 montre la configuration de ce bloc.

Paramètre	Valeur du démarrage à froid	Commentaires
Address	`0B1D70.1'	Cette adresse indique que le paramètre doit être envoyé depuis le port B sur le LCM à l'UC numéro 1, DW 70.1.
Mode	Demand	Le mode Demande est choisi lorsque les commandes doivent uniquement être envoyées lorsque le programme séquentiel PC3000 l'exige.
Trig_Read	Off (0)	Ce paramètre n'est pas utilisé pour cette application car il n'est pas indispensable à la lecture de cette adresse. Ce paramètre peut conserver sa valeur par défaut Off.
Trig_Write	Off (0)	Ce paramètre n'est pas utilisé pour cette application et peut conserver sa valeur par défaut Off. Dans cette application, le programme séquentiel contrôlera l'écriture des commandes, ce qui signifie que le paramètre State est une manière plus pratique de déclencher l'écriture des commandes.
Refresh	10s	Etant donné que le mode Demande est utilisé pour cette application, ce paramètre n'est pas utilisé et peut conserver sa valeur par défaut de 10 sec.
New_Value	-	Valeur à envoyer au PLC pour représenter une commande machine. N.B. : seuls les 16 bits les moins significatifs seront envoyés au PLC Siemens ou seront lus dans ce PLC car un seul bloc est spécifié.
State	Ok (0)	Le programme séquentiel effectue une écriture dans le paramètre. State est écrit par le programme séquentiel pour déclencher l'envoi de la commande.

Tableau 3-44 Valeurs de démarrage à froid du bloc fonction MC_Cmd Remote_Int

Pour tester en continu l'état d'une machine depuis le PLC Siemens, une déclaration de bloc fonction Remote_SW, appelée MC_Stat, est configurée dans le PC3000 comme dans le tableau 3-45. On suppose que seul l'état doit être mis à jour à la fréquence d'une fois par seconde sur le PC3000.

Paramètre	Valeur de démarrage à froid	Commentaires
Address	'0B1D71.9'	Cette adresse indique la lecture de l'UC 1, DW 71.9 depuis le périphérique relié au port B sur le LCM. Il s'agit de l'adresse du mot d'état dans le PLC Siemens.
Mode	R_Cont (1)	Le mode R_Cont est choisi pour lire les informations d'état provenant du PLC à intervalles réguliers.
Trig_Read	Off (0)	Ce paramètre n'est pas utilisé pour cette application car le mode R_Cont sert à déclencher la lecture. Le paramètre peut rester sur sa valeur par défaut Off.
Trig_Write	Off (0)	Ce paramètre n'est pas utilisé pour cette application et peut conserver sa valeur par défaut Off.
Refresh	1s	Une fréquence d'interrogation de 1 fois par seconde est considérée comme suffisante pour cette application.
New_Value	Off	Du fait que l'état de la machine ne sera pas écrit, New_Value s ne sera pas utilisé et peut conserver sa valeur par défaut.
State	Ok (0)	Il ne faut pas initialiser le paramètre State lorsque le mode de lecture continu est utilisé pour contrôler l'interrogation. State peut rester sur la valeur par défaut et sera contrôlé par le bloc fonction pour lancer les lectures.

Tableau 3-45 Valeurs de démarrage à froid du bloc fonction MC_Stat Remote_SW

Les alarmes provenant du PLC Siemens seront reçues par la moitié esclave du module. Pour contrôler ce processus, un bloc fonction Slave_Int, appelé MC_Alarm, est produit. Les valeurs de démarrage à froid sont présentées dans le tableau 3-46.

Paramètre	Valeur de démarrage à froid	Commentaires
Address	'SI1D50.0'	Cette adresse indique que le bloc simule une adresse Siemens UC 1, DW 50.0.
Trig_Wr1	Off (0)	Le paramètre Trig_Wr1 est prévu pour des raisons pratiques, lorsque le contrôle de la protection en écriture du paramètre est assuré par câblage. Dans le cas de cette application, le traitement des alarmes sera réalisé par le programme séquentiel, par conséquent ce paramètre conservera sa valeur par défaut Off.
Mode	Wr_Once (2)	Mode sera initialement Wr_Once. Il permet une écriture mais, après cette écriture, il passe à Rd_Only. Se reporter au texte pour avoir une explication détaillée du fonctionnement.
Changed	No (0)	Le drapeau Changed est initialement mis sur la valeur par défaut No, ce qui permet de voir le moment où une alarme est reçue par l'esclave.

Tableau 3-46 Valeurs de démarrage à froid du bloc fonction MC_Alarm Slave_Int

Le nom du batch sera lu dans un bloc de chaîne esclave du PC3000 appelé Batch_Nm. Les valeurs de démarrage à froid de ce bloc sont présentées dans le tableau 3-47.

Paramètre	Valeur de démarrage à froid	Commentaires
Address	'SI1D51.0+3'	Cette adresse indique que le bloc simule les adresses Siemens UC 1, DW 51.0-2. N.B. : trois mots peuvent contenir six caractères ASCII .
Trig_Wr1	Off (0)	Le paramètre Trig_Wr1 est conçu pour des raisons de facilité, lorsque la protection en écriture du paramètre est assurée par câblage. Dans le cas de cette application, le numéro de batch sera traité par le programme séquentiel, ce paramètre sera donc laissé sur sa valeur par défaut Off.
Mode	Rd_Only (1)	Le nom de batch est seulement lu par la machine, il n'est pas écrit et Mode doit par conséquent être positionné sur Rd_Only pour garantir une sécurité maximale.
Changed	No (0)	Le drapeau Changed ne présente aucun intérêt car le paramètre est uniquement lu ; le drapeau peut donc rester sur sa valeur par défaut No.
Value	'	Le programme séquentiel donnera à Value le nom de batch avant que la machine soit positionnée sur 'marche'.

Tableau 3-47 Valeurs de démarrage à froid du bloc fonction Batch_Nm Slave_Str

Le programme séquentiel remplira le paramètre **Value** de Batch_Nm avec la chaîne qui convient avant l'envoi de la commande de mise en marche de la machine.

Contrôle des communications

Les blocs fonctions de communications doivent être contrôlés dans une certaine mesure par la programmation séquentielle, ce qui est illustré dans cet exemple alors que le détail de l'application du programme séquentiel n'est pas présenté.

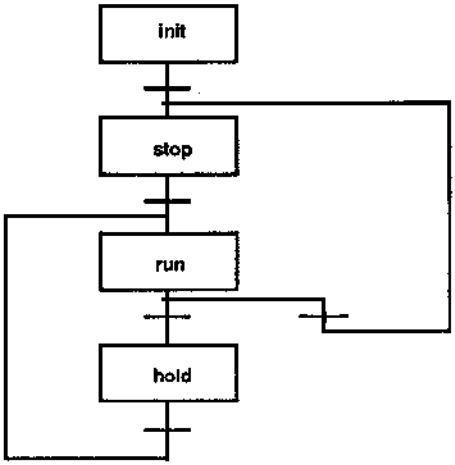


Figure 3-33 Grafect de l'exemple d'application

La figure 3-46 montre le grafcet (SFC) du séquencement du contrôle principal avec le texte structuré (ST) :

```
STEP  init :
    MC_Cmd.New_Value := 5 ; (*reset*)
    MC_Cmd.State := 3 (*Write*) ;
END_STEP

TRANSITION
    FROM init
    TO   stop
:= MC_Cmd.State = 0 (*Ok*)
END_TRANSITION

STEP  stop :
    MC_Cmd.New_Value := 4 ; (* stop *)
    MC_Cmd.State := 3 (*Write*) ;
END_STEP

TRANSITION
    FROM stop
    TO   run
:= ( MC_Cmd.State = 0 (*Ok*) ) AND
    ( run_proc.Val = 1 (*On*) ) ;
    (* run -> On to start process *)
```

```
END_TRANSITION

STEP  run :
    Batch_Nm.Value := Next_Batch.Val;
    MC_Cmd.New_Value := 1 ; (* run *)
    MC_Cmd.State := 3 (*Write*) ;
END_STEP

TRANSITION
    FROM run
    TO hold
:= ( MC_Cmd.State = 0 (*Ok*) ) AND
       ( hold.Val = 1 (*On*) ) ;
END_TRANSITION

TRANSITION
FROM run
TO   stop
:= ( MC_Cmd.State = 0 (*Ok*) ) AND
      ( run_proc.Val = 0 (*Off*) );
END_TRANSITION

STEP hold :
    MC_Cmd.New_Value := 2; (* hold *)
    MC_Cmd.State := 3 (*Write*) ;
END_STEP

TRANSITION
    FROM hold
    TO run
:= ( MC_Cmd.State = 0 (*Ok*) ) AND
       ( hold.Val = 0 (*Off*) ) ;
END_TRANSITION
```

Les quatre noms d'état correspondent aux commandes envoyées à la machine par le texte structuré dans ces pas. Le pas d'initialisation envoie une commande de remise à zéro à la machine pour l'initialiser. Il commence à positionner **New_Value** sur le code de commande 'réinitialisation' puis **State** est positionné sur **Write**. La transition de sortie de ce pas attend ensuite que **State** revienne sur **Ok**, indiquant que la valeur a été écrite avec succès. Le pas d'arrêt est identique, à la différence près qu'il reste inchangé tant que la commande d'arrêt n'a pas été envoyée et que la variable utilisateur booléenne run_proc n'est pas sur **On**. Tous les autres états et transitions fonctionnent de la même manière que la commande d'arrêt ou run_proc. Il faut noter l'utilisation d'une autre variable utilisateur booléenne pour contrôler le blocage du process d'une manière identique à la variable utilisateur run_proc. Le traitement des erreurs n'a pas été

indiqué pour des raisons de clarté. Toutefois, il est indispensable que les transitions de sortie après lancement d'une écriture déportée vérifient **Error** et **Ok** dans **State**. L'exemple indiqué suppose que la liaison est sans problème et se verrouillerait si une erreur se produisait. Dans une application réelle, il serait indispensable de prendre en compte le traitement des erreurs de manière détaillée.

Traitement des alarmes

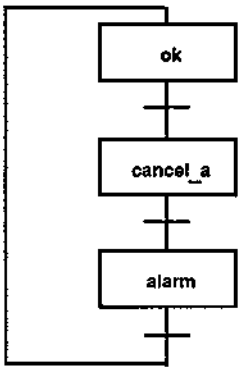


Figure 3-34 Grafcet de traitement des erreurs pour l'exemple d'application

La figure 3-34 représente le grafcet (SFC) pour le traitement des alarmes machine avec le texte structuré (ST) :

```
STEP  ok :
END_STEP
TRANSITION
     FROM ok
     TO   alarm
:= MC_Alarm.Value <> 0 ;
END_TRANSITION

STEP  alarm :
     run.Val := 0 (*Off*) ;(* stop the machine *)
     hold.Val := 0 (*Off*) ;
END_STEP

TRANSITION
     FROM alarm
     TO   cancel_a
:= Al_Clr.Val = 1 (*On*) ;
```

```
          (* setting Al_Clr.Val On (1) clears the alarm
and acknowledges it *)
END_TRANSITION

STEP  cancel_a :
     Al_Clr.Val := 0 (*Off*) ;
     MC_Alarm.Value = 0 ;
         (* clear the MC_Alarm slave variable to acknowledge the
            alarm *)
MC_Alarm.Mode = 2 (* Wr_Once *);
         (* Re-enable writes ready for next alarm *)
END_STEP

TRANSITION
     FROM alarm
     TO   ok
:= 1
END_TRANSITION
```

Etant donné que le code d'alarme est reçu par l'esclave, il n'est pas nécessaire de lancer des transactions avec le partenaire, ce qui simplifie considérablement le codage. Le pas ok ne contient aucun ST. Il possède un état de sortie simple qui fait passer la séquence au pas alarm si Alarm.Val est différent de zéro.

> **N.B.** : le bloc fonction de variable esclave passe du mode **Wr_Once** au mode **Rd_Only** pour assurer une protection contre la perte de certaines alarmes lorsque cela se produit.

Dans le pas alarm , les valeurs booléennes 'marche' et 'blocage' sont effacées pour arrêter le process. L'alarme est supprimée lorsque la valeur booléenne **Al_Clr.Val** est positionnée sur **On**. Le passage d'**Al_Clr.Val** sur **On** amène le programme séquentiel sur le pas cancel_a qui efface **Al_Clr.Val**, ce qui prépare pour la suppression de l'alarme suivante. Ce pas positionne aussi **MC_Alarm.Val** sur 0 pour indiquer au PLC Siemens que l'alarme a été notée/traitée. **MC_Alarm.Val** sera interrogé par le PLC Siemens PLC pour détecter l'accusé de réception ou la suppression de l'alarme. Enfin, **MC_Alarm.Mode** est repositionné sur **Wr_Once** afin de permettre à nouveau l'écriture du code d'alarme par le PLC.

Signalisation d'état

Les bits d'état interrogés peuvent être utilisés de nombreuses manières, en fonction de l'application, mais pour cet exemple, certains seront affichés à l'aide du bloc fonction de messages. Le câblage ci-après affichera l'état marche/blocage et l'état de l'alarme sur la ligne de messages du logiciel de la station de programmation PC3000. Les autres bits d'état peuvent être affichés d'une manière identique si besoin est.

```
Messages.P_Message := CONCAT( IN1 := SEL_STRING( G :=
                      MC_Status.Value_1  IN0 := 'HOLDING ',IN1 :=
                      'RUNNING ' ) , IN2 := SEL_STRING
                   ( G := MC_Status.Value_2 , IN0 := 'IN AUTO',IN1 :=
                      'IN MANUAL' ) );
Messages.S_Message := SEL_STRING( G := MC_Status.Value_3 ,IN0 :=
                      '*** ALARM
ACTIVE             ***', IN1 := 'Ok' );
```

Signalisation des erreurs

La majorité des codes d'erreur ont été créés pour correspondre aux erreurs SYSTAT utilisées par Siemens, afin qu'ils puissent être indiqués dans le manuel COM 525. Il existe une erreur qui figure dans le manuel COM 525 et qui n'a pas de code d'erreur correspondant, compte tenu des restrictions de mise en oeuvre : l'erreur BREAK (255 (FFh)) qui a dû être déplacée à 127 (7Fh) pour le PC3000. Il existe aussi certaines erreurs supplémentaires, propres à la mise en oeuvre du PC3000, qui ont des codes compris entre 128 et 164 et 255.

Les erreurs peuvent être signalées à trois endroits différents par le bloc fonction. Les erreurs sont signalées dans le paramètre **Error_No** de Siemens_M_S, dans les blocs fonctions de variables esclaves et dans les blocs fonctions de variables déportées, en fonction du type d'erreur impliqué. Si, par exemple, au cours de l'envoi de Remote_Str, le partenaire n'envoie aucune réponse, l'erreur 226 (DEPASSEMENT DU TEMPS IMPARTI A LA LIAISON) apparaît dans les paramètres **Error_No** des blocs fonctions Siemens_M_S et Remote_Str. En règle générale :

- si une erreur peut être attribuée directement à une variable déportée donnée, elle est signalée à ce paramètre.
- si une erreur est détectée et implique un problème de liaison, elle est signalée au bloc du module.
- seules les erreurs dans l'adresse d'une variable esclave sont signalées au bloc de paramètres esclaves.

Le tableau 3-48 montre l'emplacement où sont signalés différents types d'erreurs.

Type d'erreur détectée et moment où elle est détectée	Bloc(s) signalé(s) à		
	Module	Déporté	Esclave
Erreur détectée dans la liaison, lorsqu'elle est inactive, qui ne peut être attribuée à aucun paramètre. Ex. : ERREUR DE SYNCHRO DUE AU PARTENAIRE	•		
Lors du traitement de la lecture/écriture d'une variable.[1] Ex. : ERREUR DE STRUCTURE DANS LE TELEGRAMME DE REPONSE.	•		
Erreur d'initialisation de Siemens_M_S lors de la première utilisation de ce bloc fonction. Ex. : ERREUR DE PORT : PORT INTERDIT.	•		
BREAK reçue à n'importe quel moment.	•	•[2]	
Lors du traitement d'une lecture/écriture de variable déportée, une erreur a été détectée dans la liaison au cours de l'envoi de la demande. Ex. : ERREUR AU COURS DE L'ETABLISSEMENT DE LA LIAISON.	•	•	
Lors du traitement d'une lecture/écriture de variable déportée, une erreur a été 'REPTEL'. Ex. : ERREUR D'ACCES DB/DX SUR LE PARTENAIRE.		•	
Une erreur a été détectée dans le paramètre Adresse d'une variable déportée lorsque le bloc de variables déportées a lancé une lecture/écriture. Ex. : NUMERO D'UC HORS PLAGE.		•	
Lors du traitement d'une lecture/écriture de variable déportée, aucun module n'a été trouvé sur le port spécifié dans la variable déportée Adresse. Ex. : ERR DE PORT : AUCUN PARAM DEPORTE SERV.		•	
Une erreur dans le paramètre Adresse d'une variable esclave a été détectée lors de la première initialisation du paramètre esclave.[3] Ex. : ADRESSE HORS PLAGE.			•
Lors de la première initialisation de la variable esclave, un dépassement de capacité de la variable esclave système ou de tableaux de types de modules s'est produit. Ex. : ERREUR DE PORT : PARAM. ESCLAVES TROP NOMBREUX.			•

Tableau 3-48 Emplacements de signalisation des erreurs

[1]Le télégramme déformé ne peut pas être attribué à la demande en cours de la variable déportée parce qu'il aurait pu s'agir d'une demande arrivante pour slave_variable.

[2]Lorsqu'une INTERRUPTION est détectée, les lectures/écritures de variables déportées en cours sont arrêtées avec une erreur. Si d'autres lectures/écritures sont entamées avant la suppression de l'INTERRUPTION, elles sont immédiatement arrêtées avec une erreur.

[3]Les variables esclaves sont déclarées au module lors de leur première exécution. Le module code et vérifie ensuite leur adresse lors de sa première exécution, lorsque tous les blocs ont été exécutés une fois. Cela signifie que le code d'erreur VARIABLE ESCLAVE PAS INITIALISEE (255) peut être visible dans **Error_No** de la variable esclave quelque temps après le lancement du programme utilisateur. Si cette erreur persiste, cela signifie qu'il n'existe aucun module portant le nom déclaré qui puisse traiter la variable esclave.

Codes d'erreur

Les codes d'erreur énumérés dans le tableau 3-49 correspondent aux codes SYSTAT figurant dans le chapitre 7 du manuel CP525, sauf pour les erreurs 128 à 164 et 255 qui sont propres au PC3000. L'exception est l'erreur INTERRUPTION qui est 255 dans le SYSTAT Siemens et 127 sur le PC3000. Il faut noter que toutes les erreurs ne sont pas visibles dans le bloc de module. Les erreurs peuvent aussi être indiquées par le module dans les blocs de variables esclaves et déportées (cf. les renvois à la fin des codes d'erreur).

Error_No	Description de l'erreur
0	OK •Aucune erreur n'a été détectée.[1] [3] •La dernière opération s'est terminée sans erreur.[2] [3]
33	ERREUR DANS LE TYPE DE DONNEES[1] [2] •Le type de données de la variable esclave/déportée a été positionné sur une valeur autre que 'D' (bloc de données). Pour l'instant, seul le type 'D' est pris en charge.
34	ADRESSE HORS PLAGE[1] [2] •Le numéro de bloc de données spécifié dans l'Adresse de la variable esclave/déportée est supérieur à 255. Le numéro du bloc de données doit être compris entre 0 et 255. •Le numéro de mot de données spécifié dans l'Adresse de la variable esclave/déportée est supérieur à 255. Le numéro de mot de données doit être compris entre 0 et 255. •Il y a une erreur dans la structure du bloc/mot de données, dans le paramètre Adresse de la variable esclave/déportée. •La longueur spécifiée dans l'Adresse de la variable esclave/déportée est supérieure à 65535. La longueur du bloc doit être comprise entre 0 et 65535. •L'adresse de fin de bloc spécifiée dans l'Adresse de la variable esclave/déportée est supérieure à 65535. L'adresse de fin de bloc doit être comprise entre 0 et 65535. •L'adresse de fin de bloc spécifiée dans l'Adresse de la variable esclave/déportée est inférieure à l'adresse de début de bloc. •Le numéro de mot de données dans l'Adresse est suivi d'un caractère interdit. •La taille du bloc spécifiée est trop importante pour le type de bloc esclave/déporté utilisé. •Il y a des caractères en surnombre dans l'Adresse.
36	NUMERO D'UC HORS PLAGE[1] •Le numéro d'UC spécifié pour la variable esclave/déportée n'est pas autorisé. Il doit être compris entre 0 et 7.

Tableau 3-49 Codes d'erreur Siemens_M_S

Error_No	Description de l'erreur
38	LONGUEUR TROP IMPORTANTE •La longueur de bloc spécifiée dans l'Adresse de la variable déportée est trop importante pour la longueur de la chaîne New_Value .
41	ERREUR DE SYNCHRON. DUE AU PARTENAIRE [3] •Un télégramme de réponse est arrivé alors qu'aucune demande n'avait été effectuée ou il n'a pas été terminé.
42	ERREUR DANS LA STRUCTURE DU TELEGRAMME DE REPONSE[3] •Le premier octet est différent de 0. N.B. : le module ne prend pas en charge les télégrammes de suivi donc $FF est interdit.
43	REPONSE A UNE RECUPERATION TROP LONGUE[3] •La réponse à une récupération contient un trop grand nombre de données.
44	REPONSE A UNE RECUPERATION TROP COURTE[3] •La réponse à une récupération contient un nombre de données trop faible.
45	LA REPONSE A UN ENVOI CONTIENT DES DONNEES[3] •Une réponse suivant un envoi contenait des données.
47	DEPASSEMENT DU DELAI IMPARTI POUR LE TELEGRAMME DE REPONSE[2 3] •Aucun télégramme de réponse n'est parvenu du partenaire dans le temps de contrôle après l'envoi d'un télégramme de RECUPERATION/ENVOI.
48	DB/DX DESACTIVE PAR CF (OU WR_PROTECT) DANS LE PARTENAIRE[2] •Cette erreur sera signalée en cas d'écriture dans une autre variable esclave PC3000 Siemens_M_S protégée en écriture. •Les drapeaux de coordination ne sont actuellement pas pris en charge par le module, cette erreur ne devrait donc jamais se produire dans les cas de connexion avec les PLC Siemens.
49	ERREUR DE MATERIEL DANS LE PARTENAIRE[2] Ex. : avec un partenaire CP525 : •Le partenaire signale la détection d'un type de source/destination interdit. •Le partenaire signale une erreur mémoire dans le PC partenaire. •Le partenaire signale une erreur dans le message de colloque entre CP/CPU dans le partenaire. S'applique uniquement aux PC SIEMENS. •Le partenaire signale que le PC est sur ARRET.

Tableau 3-49 Codes d'erreur Siemens_M_S (suite)

Error_No	Description de l'erreur
50	ERREUR D'ACCES MEMOIRE SUR LE PARTENAIRE[2] Ex. : avec CP525 comme partenaire : •Le partenaire signale une zone erronée pour le mot de code d'état. •Le partenaire signale que la zone de données n'existe pas. •Le partenaire signale que la zone de données est trop petite (sauf DB/DX).
52	ERREUR DE COMMANDE SUR LE PARTENAIRE[2] Ex. : Avec CP525 comme partenaire: •Le partenaire signale une première lettre de commande erronée dans l'en-tête de télégramme.
53	ERREUR DE TYPE DE COMMANDE SUR LE PARTENAIRE[2] Ex. : Avec CP525 comme partenaire: •Le partenaire signale une deuxième lettre de commande erronée dans l'en-tête de télégramme.
54	LE PARTENAIRE DETECTE UNE ERREUR DE LONGUEUR DE TELEGRAMME[2] Ex. : Avec CP525 comme partenaire: •Le partenaire signale que la longueur codée dans l'en-tête ne correspond pas à la longueur réelle du télégramme qui a été lue.
55	LE PARTENAIRE DETECTE UNE ERREUR DE SYNCHRO.[2] Ex. : Avec CP525 comme partenaire: •Le partenaire signale que l'ordre des télégramme est erroné.
56	ABSENCE DE DEMARRAGE A FROID SUR LE PARTENAIRE[2] Ex. : Avec CP525 comme partenaire: •Le partenaire signale qu'aucun 'SYNCHRON' HDB n'a tourné depuis la mise sous tension. •Le partenaire signale que le sélecteur de mode est positionné sur STOP/PGR.
57	LE PARTENAIRE SIGNALE QUE LA COMMANDE SYS EST INTERDITE[2] Ex. : Avec CP525 comme partenaire: •Il s'agit d'une réaction incorrecte du partenaire. Le CP525 ne sort jamais de commande système!
58	NUMERO D'ERREUR INCONNU PROVENANT DU PARTENAIRE[2] •Un numéro d'erreur inconnu a été reçu dans le télégramme de réponse provenant du partenaire.

Tableau 3-49 Codes d'erreur Siemens_M_S (suite)

Error_No	Description de l'erreur
64	ERREUR DANS LE PREMIER OCTET DE COMMANDE[3] •Le premier octet de commande est différent de 0. N.B. : le module ne prend pas en charge les télégrammes de suivi, $FF est donc interdit.
127	INTERRUPTION[2 3] •Une interruption (espace continu) a été détectée sur la ligne série.
128	EVENEMENT INTERDIT[3] •Erreur interne dans le bloc qui ne doit jamais apparaître.
129	ETAT INTERDIT[3] •Erreur interne dans le bloc qui ne doit jamais apparaître.
130	CHEVAUCHEMENT D'ADRESSES ESCLAVES[1] •Deux adresses esclaves se chevauchent.
145	ERREUR DE PORT : ABSENCE D'ADRESSE[4] •Le paramètre Port du bloc fonction contient moins de deux caractères.
149	ERREUR DE PORT : DEBIT RX PAS DISPONIBLE[4] •Le débit de réception demandé n'est pas disponible sur ce port série.
150	ERREUR DE PORT : DEBIT TX PAS DISPONIBLE[4] •Le débit d'émission demandé n'est pas disponible sur ce port série.
155	ERREUR DE PORT : EMPLACEMENT INTERDIT[4] •Le numéro d'emplacement sélectionné n'est pas autorisé. Le numéro d'emplacement est le premier caractère du paramètre Port.
156	ERREUR DE PORT : PORT INTERDIT[4] •Le numéro de port sélectionné n'est pas autorisé. Le numéro de port est le deuxième caractère du paramètre Port.
160	ERREUR DE PORT : ABSENCE DE PRISE EN CHARGE DE PARAMETRE DEPORTE[2] •Aucun module n'est affecté au port figurant dans le paramètre Adresse du bloc fonction de variable déportée.

Tableau 3-49 Codes d'erreur Siemens_M_S (suite)

Error_No	Description de l'erreur
161	ERREUR DE PORT : PORT UTILISE[4] •Le port sélectionné est déjà utilisé pour un autre module.
162	ERREUR DE PORT : TROP GRAND NOMBRE DE PARAMETRES ESCLAVES[1] •Un trop grand nombre de variables esclaves ont été déclarées au système.
163	ERREUR DE PORT : TROP GRAND NOMBRE DE TYPES DE MODULES[1] •Un trop grand nombre de types de modules esclaves ont été déclarés au système.
164	ECRITURE INTERDITE DU TEMPS HOLD OFF[4] •Le temps Hold_Off sélectionné est trop long pour ce débit.
225	ERREUR AU COURS DE L'ETABLISSEMENT DE LA LIAISON[2 3] •Après l'envoi d'un <STX>, un caractère autre que <DLE> ou <STX> a été reçu.
226	DEPASSEMENT DU DELAI IMPARTI POUR L'ETABLISSEMENT DE LA LIAISON[2 3] •Après l'envoi d'un <STX>, le partenaire n'a pas répondu dans le délai imparti pour la réponse.
227	ARRET PROVOQUE PAR LE PARTENAIRE[2 3] •Un caractère <NAK> ou un autre caractère a été reçu au cours de l'envoi, ce qui a mis fin à cet envoi.
228	ERREURS A LA FIN DE LA LIAISON[2 3] •Après l'envoi d'un télégramme, celui-ci a été rejeté par le partenaire à l'aide dun caractère <NAK> ou d'un autre caractère différent de <DLE>.
229	DEPASSEMENT DU DELAI IMPARTI A LA FIN D'UNE LIAISON[2 3] •Après envoi d'une fin de liaison <DLE><ETX>**BCC**, aucun accusé de réception n'est parvenu en provenance du partenaire dans le délai imparti.
241	ERREUR DE FIN DE LIAISON[3] •Un caractère autre que <NAK> ou <STX> a été reçu après la fin de la liaison, c'est-à-dire pendant que la ligne était inactive.
242	ERREUR LOGIQUE AU COURS DE LA RECEPTION[3] •Un caractère interdit a été reçu après un caractère <DLE>. Seuls <DLE> ou <ETX> sont autorisés à la suite de <DLE>

Tableau 3-49 Codes d'erreur Siemens_M_S (suite)

Error_No	Description de l'erreur
244	ERREUR BCC[3] •Le caractère de contrôle de bloc (BCC) est en conflit avec la valeur calculée en interne. S'applique uniquement lorsque Checksuming est activé.
246	AUCUNE MEMOIRE TAMPON RX LIBRE[3] • La mémoire tampon de réception n'a pas été libérée dans le délai imparti. Cela ne doit jamais se produire et doit être signalé comme un défaut s'il se produit. •Un caractère a été reçu après un caractère <STX> mais avant qu'un caractère <DLE> ait accusé réception de l'établissement de la liaison. •La capacité de la mémoire tampon de réception a été dépassée alors qu'elle constituait un message reçu.
254	ERREUR DE TRANSMISSION[2] [3] •Une erreur a été détectée au cours de la réception d'un caractère. Exemple : des erreurs d'encadrement, de parité ou de débordement ont été détectées.
255	VARIABLE ESCLAVE PAS INITIALISEE[1] •La variable esclave n'a pas encore été initialisée. •Il n'existe dans le programme utilisateur aucun bloc de module qui porte le nom indiqué dans l'adresse de la variable esclave.

Tableau 3-49 Codes d'erreur Siemens_M_S

[1]Cette erreur peut être signalée dans un bloc de variable esclave lors de la première exécution du programme utilisateur.

[2]Cette erreur peut être signalée dans un bloc de variable déportée lors de la première exécution d'une lecture ou d'une écriture.

[3]Cette erreur peut être signalée dans le bloc fonction Siemens_M_S à n'importe quel moment.

[4]Cette erreur peut être signalée dans le bloc fonction Siemens_M_S lors de sa première exécution.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	**STRING**	'0A'	Oper	Config		
Baud	**ENUM**	_9600	Oper	Config	Valeurs énumérées	_75(0) _300(1)* _600(2)* _1200(3)* _2400(4)* _4800(5)* _9600(6)* _19200(7)* _38400(8) _57600(9) _115200(10) * Débits pris en charge par Siemens
Checksum	**BOOL**	Nb	Oper	Super	Détection	Non(0) Oui(1)
Priority	**BOOL**	Low	Oper	Super	Détection	Bas(0) Haut(1)
Hold_Off	**TIME**	0s	Oper	Config	Lim. haute Lim. basse	40s 0s
Wr_protect	**BOOL**	Nb	Oper	Super	Détection	Non(0) Oui(1)
Status	**BOOL**	NOGO	Oper	Bloc	Détection	NOGO(0) Go(1)
Error_No	**SINT**	0	Oper	Bloc	Lim. haute Lim. basse	255 0
Queue_Space	**SINT**	0	Oper	Bloc	Lim. haute Lim. basse	255 0

Tableau 3-50 Attributs des paramètres Siemens_M_S

BLOC FONCTION JBus_M

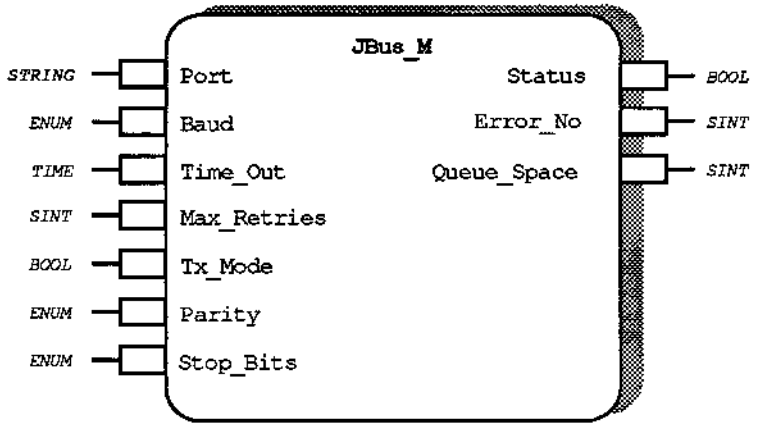


Figure 3-35 Diagramme du bloc fonction

Description fonctionnelle

Le bloc fonction JBus_M prend en charge les communications série sur un port série de communications avec le protocole JBus en mode maître. Le bloc fonction prend aussi en charge le protocole Modbus qui est identique à JBus, à la différence près que les emplacements esclaves sont référencés avec un décalage d'une unité. Il n'est normalement pas nécessaire de connaître le protocole JBus de manière détaillée pour utiliser ce bloc fonction. Toutefois, vous pouvez vous reporter au Guide de base du protocole Gould Modbus de Gould Inc. Programmable Control Division pour avoir des détails, si besoin est. La partie consacrée au bloc fonction JBus_S comprend une présentation du protocole JBus. Reportez-vous à la 'Base du protocole JBus', page 3-180. Avant de lire cette description, vous avez intérêt à acquérir des connaissances générales sur le système de communications PC3000 en lisant la présentation des communications PC3000.

Ce bloc fonction sera nécessaire pour concevoir ou programmer le PC3000 afin qu'il utilise une interface JBus ou Modbus qui fonctionne en mode maître, c'est-à-dire que le PC3000 est relié à un ou plusieurs périphérique(s) esclave(s) JBus par l'intermédiaire d'une liaison série.

Le bloc fonction Jbus_M fournit un module maître JBus et traite les détails des communications série propres au protocole. Il est pris en charge par des blocs fonctions génériques de variables déportées liés au module par une adresse propre au protocole. Les blocs de variables déportées utilisent le bloc fonction du module maître pour lancer les lectures et les écritures dans les périphériques déportés. Pour avoir des détails supplémentaires sur l'utilisation des blocs

fonctions de variables déportées, consulter la 'Présentation des communications PC3000'. Des informations supplémentaires relatives au protocole Jbus sont fournies dans le chapitre 3, dans la description du bloc fonction JBus_S.

Attributs du bloc fonction

Type : 8 75

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port Status Queue_Space

Besoins de capacité mémoire : 1992 octets

Durée d'exécution : ..... 23µ Secs

Description des paramètres

Paramètres de configuration du module

Le bloc JBus_M possède un certain nombre de paramètres de saisie de configuration qui définissent divers aspects du module et doivent être configurés avant le lancement du programme utilisateur. La modification de ces paramètres au cours de l'exécution du programme utilisateur n'a aucun effet sur le module, sauf dans des circonstances particulières : cf. 'Présentation des communications PC3000', partie 'Modification temporaire des paramètres de configuration'.

Port

Le paramètre **Port** est l'adresse à deux caractères du port sur lequel tourne le protocole JBus. Le premier caractère est un chiffre compris entre 0 et 5, représentant l'emplacement dans le rack, et le deuxième caractère est une lettre qui représente le port dans cet emplacement ; par exemple, '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' pourrait désigner son port supérieur.

Baud

Le paramètre **Baud** offre un choix entre 11 débits différents, compris entre 75 Bauds et 115,2 kBauds (indiqués dans le tableau 3-51), avec un débit par défaut de 9600 Bauds.

> **N.B. :** tous les ports ne peuvent pas prendre en charge la totalité des débits. Si un débit n'est pas pris en charge, cela est indiqué par une erreur lors de la première exécution du bloc fonction (cf. la description d'**Error_No**).

Valeur énumérée	Débit
0	75
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	115200

Tableau 3-51 Débits JBus_M

Parité

Le paramètre **Parité** définit la parité utilisée pour la transmission des caractères. Il peut prendre les valeurs **Néant, Paire, Impaire, Espace** ou **Marque**.

Stop_Bits

Le paramètre **Stop_Bits** définit le nombre de bits d'arrêt utilisés pour la transmission des caractères. Les valeurs autorisées sont **1, 1,5** et **2.**

Time_Out

Le paramètre **Time_Out** spécifie la durée pendant laquelle JBus_Master doit attendre un message de réponse à une demande émise. A la fin de cette durée, le module estime qu'il y a une erreur de transmission et la demande peut être réémise ou une erreur peut être renvoyée au bloc de paramètres déportés qui a émis la demande.

Max_Retries

Le paramètre **Max_Retries** spécifie le nombre de nouvelles tentatives de transmission d'une demande en cas de détection d'une erreur, comme un dépassement du délai imparti ou une erreur de total de contrôle de message. Si aucune réponse valable n'est reçue après ce nombre de nouvelles tentatives, une erreur est renvoyée à la variable déportée qui a émis la demande.

Paramètres d'état du module

L'état du module est indiqué par trois paramètres de sortie dans le bloc fonction JBus_M.

Etat

Le paramètre **Etat** est une indication booléenne de l'état de la liaison contrôlée par le module. En l'absence de problèmes sur cette liaison, le paramètre indique **Go** mais, lorsqu'une erreur se produit, il passe sur **NoGo**. Si l'**Etat** est **NoGo**, le paramètre **Error_No** indique la cause du problème.

Error_No

Le paramètre **Error_No** indique la cause des erreurs éventuelles survenant sur la liaison. Si l'**Etat** de la liaison est **Go**, **Error_No** sera 0 (OK). Pour avoir des détails complets sur le traitement des erreurs et les codes d'erreur, se reporter au tableau 3-52, page 3-166.

Queue_Space

Le paramètre **Queue_Space** indique l'espace restant dans la file d'attente pour les transactions de variable déportée. S'il devient égal à zéro, cela implique que la largeur de bande de la liaison est insuffisante pour faire face au nombre de demandes de variables déportées effectuées et les données seront perdues. Si cette situation survient, il faut diminuer les fréquences d'interrogation des paramètres.

Fonctionnement avec les variables déportées

Les demandes du maître effectuées par le PC3000 sont contrôlées par une ou plusieurs variables déportées. Le module JBus_M prend en charge les paramètres Remote_Bool, Remote_Real, Remote_Int, Remote_Str et Remote_SW.

Adressage

Il est nécessaire de configurer une **Adresse** propre au protocole dans le bloc de variable déportée qui est l'adresse servant à accéder aux périphériques déportés. La figure 3-21 montre un exemple de structure d'adresse. Le port est défini, comme dans le paramètre **Port** du bloc fonction, comme un numéro d'emplacement dans le rack suivi d'une lettre pour le port situé dans cet emplacement.

La partie de l'adresse propre au protocole commence par une identité esclave qui spécifie l'adresse du périphérique esclave qui doit répondre à une demande. Il s'agit d'un nombre hexadécimal à deux chiffres compris entre 0 et 247. Une identité esclave égale à zéro correspond au mode multidiffusion où tous les esclaves sont adressés. Cette identité est suivie d'une adresse à quatre chiffres qui identifie une adresse initiale dans le périphérique esclave sélectionné. Le

caractères suivant (T majuscule ou minuscule) est facultatif et précise si l'emplacement adressé est accessible uniquement comme saisie. La spécification d'une adresse en saisie seule est importante car elle peut sélectionner un espace d'adresse différent dans le périphérique esclave. Les derniers caractères sont des caractères de structure de données qui spécifient, pour chaque type de variable déportée, la manière dont doivent être interprétées les données.

Figure 3-36 Exemple d'adresse de variable déportée

Paramètre booléen déporté

La chaîne d'adresse du bloc possède les caractères de structure suivants.

B - Mode d'adressage bit (1 bit).

R - Mode d'adressage registre (1 mot).

Exemples

1000 - Valeur par défaut du mode d'adressage bit.

1000B - Mode d'adressage bit.

1000R - Mode d'adressage registre.

Les valeurs booléennes sont codées de la manière suivante :

Mode bit : une valeur booléenne nulle est convertie en un résultat en bits égal à zéro. Une valeur boolénenne de 1 est convertie en un résultat en bits égal à 1.

Mode mot : une valeur booléenne nulle est convertie en un résultat en mots égal à zéro. Une valeur booléenne de 1 est convertie en un résultat en mots différent de zéro.

Paramètre réel déporté

La chaîne d'adresse du bloc possède les caractères de structure suivants :

En : adressage de registre, longueur = 1, mode exposant. n facultatif = -9..9.

Haut, bas : adressage registre, longueur = 1, mode limites, bas < haut.

Exemples

3000 - Valeur par défaut du mode exposant, index = 0.

3000E - Mode exposant, index = 0.

3000E-4 - Mode exposant, index = -4.

3001E+2 - Mode exposant, index = 2.

3001L1.5,3.5 - Mode limites, bas = 1,5, haut = 3,5.

3002L-4.0,+4.0 - Mode limites, bas = -4,0, haut = 4,0.

Les valeurs réelles sont codées de la manière suivante :

Mode exposant - Mot - valeur entière de réel * 10^{index}

Mode limites - Mot - valeur entière de $\frac{Real-Low}{High-Low}$ * FFFFh

Paramètre entier déporté

La chaîne d'adresse du bloc possède les caractères de structure suivants :

Bn - Mode d'adressage bit. n facultatif = 1..32.

Rn - Mode d'adressage registre. n facultatif = 1 ou 2.

Exemples

2000 - Valeur par défaut du mode d'adressage registre, longueur 1 mot.

2000B - Mode d'adressage bit, longueur 1 bit.

2000R - Mode d'adressage registre, longueur 1 mot.

2001B12 - Mode d'adressage bit, longueur 12 bits.

2001R2 - Mode d'adressage registre, longueur 2 mots.

Les valeurs entières sont codées de la manière suivante :

Mode bit - Un octet est formé par extraction du nombre nécessaire de bits dans l'entier double. L'octet résultant est complété par des bits zéro si besoin est.

Mode mot - Aucune conversion n'est nécessaire. Si l'entier double est défini comme ayant une longueur d'un mot, c'est le mot le moins significatif qui est utilisé.

Paramètre de chaîne déportée

La chaîne d'adresse du bloc possède les caractères de structure suivants.

Bn - Mode d'adressage bit. n facultatif ≤ longueur maximale de la chaîne en bits.

Rn - Mode d'adressage registre. n facultatif ≤ longueur maximale de la chaîne en mots.

Exemples

4000 - Valeur par défaut du mode d'adressage registre, longueur 1 mot, 2 caractères.

4000B - Mode d'adressage bit, longueur 1 bit.

4000R - Mode d'adressage registre, longueur 1 mot, 2 caractères.

4001B12 - Mode d'adressage bit, longueur 12 bits.

4001R6 Mode d'adressage registre, longueur 6 mots, 12 caractères.

Les valeurs de chaîne sont codées de la manière suivante :

Mode bit - Un octet est formé par extraction du nombre demandé de bits dans la chaîne. L'octet résultant est complété par des bits zéro si besoin est.

Mode mot - Les mots sont codés dans une structure de type entier, l'octet le plus significatif étant en première position et l'octet le moins significatif en deuxième position.

Par conséquent, mot = octet_0$_{hi}$ octet_1 $_{lo}$.

Les lectures et écritures de blocs sont uniquement possibles à l'aide des paramètres de chaînes déportées où les données seront mémorisées sous la forme de mots 16 bits ou de bits condensés en octets. Il ne convient donc que pour la mémorisation de caractères ou d'entiers, sauf si la conversion en un autre type de données est effectuée auparavant.

Signalisation des erreurs

Les erreurs suivantes sont signalées par le bloc fonction JBus_M

Error_No	Description de l'erreur
145	ERREUR DE PORT : ABSENCE D'ADRESSE •Le paramètre Port du bloc fonction contient moins de deux caractères.
149	ERREUR DE PORT : DEBIT RX PAS DISPONIBLE •Le débit de réception demandé n'est pas disponible sur ce port série.
150	ERREUR DE PORT : DEBIT TX PAS DISPONIBLE •Le débit d'émission demandé n'est pas disponible sur ce port série.
155	ERREUR DE PORT : EMPLACEMENT INTERDIT •Le numéro d'emplacement sélectionné n'est pas autorisé. Le numéro d'emplacement est le premier caractère du paramètre Port.
156	ERREUR DE PORT : PORT INTERDIT •Le numéro de port sélectionné n'est pas autorisé. Le numéro de port est le deuxième caractère du paramètre Port.
160	ERREUR DE PORT : ABSENCE DE PRISE EN CHARGE DE PARAM DEPORTE •Aucun module maître correct n'est affecté au port indiqué dans le paramètre Adresse du bloc fonction de paramètre déporté.
161	ERREUR DE PORT : PORT UTILISE Le port sélectionné est déjà utilisé pour un autre module.

Table 3-52 Codes d'erreur JBus_M

Codes d'erreur de variables déportées

Les erreurs suivantes sont signalées par les blocs fonctions de variables déportées fonctionnant avec un port série de communications affecté à JBus_M.

Error_No	Description de l'erreur
128	CHAINE D'ADRESSE TROP COURTE •La chaîne d'adresse contenue dans le paramètre Adresse est trop courte pour être valable.
129	ADRESSE HORS PLAGE •L'adresse spécifiée dans le paramètre Adresse ne se situe pas dans la plage valable comprise entre 0000 et 9999.
130	CARACTERE NON VALABLE DANS LA CHAINE D'ADRESSE •Un caractère non valable a été détecté dans le champ de structure de données du paramètre Adresse.
131	VALEUR NUMERIQUE DE LA STRUCTURE HORS PLAGE •Une valeur de taille de paramètre spécifiée dans le paramètre Adresse est hors plage pour ce type de données.
135	IDENTITE ESCLAVE PAS VALABLE •Une identité esclave qui a été spécifiée ne se situait pas dans la plage valable comprise entre 0 et 247.
136	DIFFUSION INTERDITE •Une lecture a été tentée à l'aide du mode diffusion alors que seules les écritures sont autorisées.
137	ECRITURE DANS LE PARAMETRE D'ENTREE IMPOSSIBLE •Une écriture a été tentée dans un emplacement esclave spécifié comme étant en saisie seule dans le paramètre Adresse.

Tableau 3-53 Codes d'erreur de variables déportées JBus_M

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	STRING	'0A'	Oper	Config		
Baud	ENUM	_9600	Oper	Config	Valeurs énumérées	_75(0) _300(1) _600(2) _1200(3) _2400(4) _4800(5) _9600(6) _19200(7) _38400(8) _57600(9) _115200(10)
Time_Out	TIME	1s	Oper	Super	Lim. haute Lim. basse	24jours 100ms
Max_Retries	SINT	2	Oper	Super	Lim. haute Lim. basse	1000 0
Tx_Mode	BOOL	RTU	Oper	Config	Détection	RTU(0) Ascii(1)
Parity	ENUM	NONE	Config	Config	Valeurs énumérées	EVEN(0) ODD(1) SPACE(2) MARK(3) NONE(4)
Stop_Bits	ENUM	_1	Config	Config	Valeurs énumérées	_1(0) _1_5(1) _2(2)
Status	BOOL	NoGo	Oper	Bloc	Détection	NOGO(0) Go(1)
Error_No	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0
Queue_Space	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0

Tableau 3-54 Attributs des paramètres

BLOC FONCTION JBUS_S

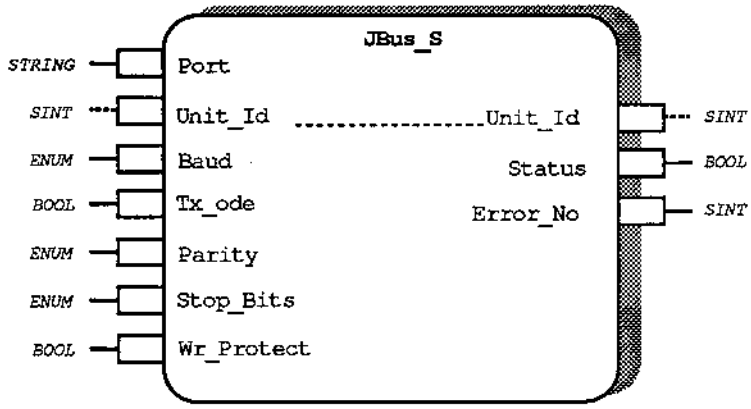


Figure 3-37 Diagramme de communications JBus_S

Description fonctionnelle

Ce document détaille le module esclave du protocole JBus pour le PC3000 et le bloc fonction associé.

> **N.B. :** le protocole Modbus est également pris en charge par ce module ; Modbus est identique à JBus, à la différence près que seuls les emplacements esclaves sont référencés avec un décalage d'une unité.

Vous avez intérêt à acquérir des connaissances générales sur le système de communications PC3000 qui figurent dans la 'Présentation des communications PC3000' avant de lire ce document.

Ce bloc fonction sera nécessaire pour concevoir ou programmer le PC3000 pour qu'il utilise une interface JBus ou Modbus fonctionnant en mode esclave, c'est-à-dire que le PC3000 est relié à un périphérique maître JBus par l'intermédiaire d'une liaison série.

Le bloc fonction JBus_S offre un module pour le protocole JBus permettant à un port série PC3000 de fonctionner comme esclave JBus. Le bloc de module traite les détails des communications propres au protocole et est pris en charge par les blocs fonctions génériques de variables esclaves. Les blocs de variables esclaves sont liés au module par une adresse propre au protocole et définissent les valeurs qui peuvent être lues ou écrites par un périphérique déporté utilisant le type de protocole spécifié dans le paramètre Adresse, 'JB' pour JBus dans le cas présent. Les valeurs des paramètres peuvent être placées soit dans un espace d'adresse de registre soit dans un espace d'adresse de bit distincte si besoin est.

Pour avoir des détails supplémentaires sur le bloc de variable esclave, se reporter à la 'Présentation des communications PC3000'.

Attributs du bloc fonction

Type : 8 80

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port Status Wr_Protect Unit_Id

Besoins de capacité mémoire : 1706 octets

Durée d'exécution : ..... 17 µ Secs

Description des paramètres

Paramètres de configuration du module

Le bloc JBus_S possède plusieurs paramètres de saisie de configuration qui définissent divers aspects du module et qui doivent être configurés avant le lancement du programme utilisateur. La modification de ces paramètres au cours de l'exécution du programme utilisateur n'aura aucun effet sur le module, sauf dans des circonstances particulières : cf. 'Présentation des communications PC3000', partie 'Modification temporaire des paramètres de configuration'. Le seul paramètre de saisie qui peut avoir un effet sur le bloc au cours de son exécution est **Wr_Protect** .

Port

Le paramètre **Port** est l'adresse à deux caractères du port sur lequel doit tourner le protocole JBus. Le premier caractère est un chiffre compris entre 0 et 5, représentant l'emplacement dans le rack et le deuxième caractère est une lettre représentant le port dans cet emplacement. Par exemple, '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' pourrait désigner son port supérieur.

Baud

Le paramètre **Baud** offre un choix de 11 débits différents, compris entre 75 Bauds et 115,2 kBauds (comme l'indique le tableau 3-55), avec une valeur par défaut de 9600 Bauds.

> **N.B. :** tous les ports ne peuvent pas prendre en charge la totalité des débits. Si un débit n'est pas pris en charge, une erreur l'indiquera lors de la première exécution du bloc fonction (cf. la description d'**Error_No**).

Valeur énumérée	Débit
0	75
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	38400
9	57600
10	115200

Tableau 3-55 Débits de Bus_S

Parité

Le paramètre **Parité** définit la parité utilisée pour la transmission de caractères. Il peut prendre les valeurs **Paire, Impaire, Espace** ou **Marque.**

Stop_Bits

Le paramètre **Stop_Bits** définit le nombre de bits d'arrêt utilisés pour la transmission des caractères. Les valeurs autorisées sont 1, 1,5 et 2.

Unit_Id

Le paramètre **Unit_Id** spécifie l'adresse JBus à laquelle répondra le PC3000. La plage est comprise entre 0 et 247.

Si **Unit_Id** est égal à zéro, un identificateur sélectionné par le matériel est utilisé. Il est configuré à l'aide de liaisons/d'un commutateur rotatif sur le LCM ou d'un commutateur rotatif sur un ICM, selon le module sur lequel est situé le port.

Liaisons LCM	Commutateur ICM	Unit_Id
0000	0	1
0001	1	16
0010	2	31
0011	3	46
0100	4	61
0101	5	76
0110	6	91
0111	7	106
1000	8	121
1001	9	136
1010		151
1011		166
1100		181
1101		196
1110		211
1111		1111

Tableau 3-56 Unit_Id JBus sélectionnée par le matériel

Tx_Mode

Le paramètre **Tx_Mode** spécifie le type de codage utilisé pour envoyer et recevoir les messages JBus et peut prendre les valeurs `RTU` ou `Ascii`. La différence entre les deux modes réside dans le fait que les messages à structure Ascii utilisent des caractères imprimables et ont une taille environ double de celle des messages à structure `RTU` qui utilisent un codage binaire. Le contenu du message est le même dans les deux modes.

Wr_Protect

Lorsqu'il est réglé, le paramètre **Wr_Protect** inhibe toutes les écritures JBus. Si une opération d'écriture est tentée alors que la protection en écriture est activée, un message de réponse erronée est émis par le module.

Paramètres d'état du module

L'état du module est indiqué par deux paramètres de sortie dans le bloc fonction JBus_S.

Status

Le paramètre **Etat** est une indication booléenne de l'état de la liaison contrôlée par le module. En l'absence de problème sur la liaison, ce paramètre indique Go mais, lorsqu'une erreur se produit, il passe sur NoGo. Si l'**état** est sur NoGo, le paramètre **Error_No** indique la cause du problème.

Error_No

Le paramètre **Error_No** indique la cause des éventuelles erreurs survenues sur la liaison. Si l'état de la liaison est Go, Error_No sera 0 (OK). Pour avoir des détails complets sur les codes d'erreur, se reporter à la partie Signalisation des erreurs.

Les emplacements de blocs fonctions de variables esclaves sont créés dans le PC3000 pour définir les paramètres qui peuvent être lus et écrits par un périphérique déporté JBus. Pour l'instant, la plupart des types de données élémentaires et quelques types de données vectorielles sont pris en charge.

Tous les types de blocs de variables esclaves ont les paramètres suivants :

Protocole **Adresse** et l'adresse permettant d'accéder au paramètre (cf. la suite pour avoir des détails).

Valeur : paramètre d'entrée/sortie qui contient la valeur du paramètre.

Mode : paramètre d'entrée/sortie permettant de sélectionner les modes lecture seule, lecture et écriture ou écriture unique.

Trig_Wr1 sur un flanc ascendant fait passer le mode sur l'écriture unique.

Le paramètre booléen **Etat** indique que le bloc est opérationnel.

La sortie **Error_No** indique la raison pour laquelle le bloc n'est pas opérationnel.

Les données de fonctionnement des variables esclaves sont écrites et lues dans le PC3000 par l'intermédiaire des variables esclaves.

Adresses

Le paramètre Adresse est composé d'un identificateur de protocole à deux caractères suivi d'informations d'emplacement et de structure propres au protocole. Exemple :

Chaîne d'adresse	Identificateur de protocole	Emplacement	Structure
'JB1234R1'	JB	1234	R1

Il est possible d'accéder à une variable esclave à l'aide de n'importe quel port de communications configuré pour prendre en charge le protocole sélectionné par la partie 'identificateur de protocole' de l'adresse ; dans le cas présent, on utilise 'JB' pour sélectionner le bloc fonction de module Jbus_S.

La partie 'emplacement' de l'adresse reçoit une valeur décimale comprise entre 0000 et 3999 (4000 à 9999 sont réservés pour un accès futur aux paramètres système). Etant donné que JBus possède deux espaces d'adresses, Bit et Registre, l'emplacement doit seulement être unique dans chacun de ces deux espaces. Selon le type et la structure de variable sélectionnés, une variable peut s'étendre sur plusieurs emplacements. Par exemple, un bloc fonction de chaîne esclave (Slave_Str) avec un emplacement de 0000 et une structure de B64 occuperait les emplacements 0000-0063. Il faut noter que les adresses qui se chevauchent feront passer une ou plusieurs variables esclaves sur un état d'erreur.

Il est possible d'accéder à des variables consécutives à l'aide d'une lecture ou d'une écriture de bloc. Les tentatives de lecture ou d'écriture dans des 'emplacements vides' entre des variables provoqueront le renvoi d'une erreur sur JBus. Une lecture ou une écriture de bloc peut aussi accéder à une partie de variable si cette dernière a une longueur supérieure à un emplacement.

Toutes les fois qu'une variable esclave est placée dans un mode 'écriture unique', l'écriture suivante dans cette variable à l'aide de JBus la fera passer en lecture seule. Cela protège la valeur jusqu'à ce que le programme utilisateur PC3000 puisse en accuser réception en ramenant le mode sur 'écriture unique'. Le document 'Présentation des communications PC3000' donne une description plus détaillée à ce sujet.

La valeur d'une variable esclave est codée dans un mot 16 bits pour l'adressage registre et dans un octet condensé pour l'adressage bit.

Paramètre booléen esclave

La chaîne d'adresse du bloc possède les caractères de structure suivants.

B - Mode d'adressage bit (1 bit).

R - Mode d'adressage registre (1 mot).

Exemples

1000 - Valeur par défaut du mode d'adressage bit.

1000B - Mode d'adressage bit.

1000R - Mode d'adressage registre.

Les valeurs booléennes sont codées de la manière suivante :

Mode bit - Une valeur booléenne nulle est convertie en un résultat bit égal à zéro. Une valeur booléenne de 1 est convertie en un résultat bit égal à 1.

Mode mot - Une valeur booléenne nulle est convertie en un résultat mot égal à zéro. Une valeur booléenne de 1 est convertie en un résultat mot différent de zéro.

Paramètre réel esclave

Cette chaîne d'adresse du bloc possède les caractères de structure suivants :

E<n> - Adressage registre, longueur = 1, mode exposant. n facultatif = -9..9.

L<bas>,<haut> - Adressage registre, longueur = 1, mode limites, bas < haut.

Exemples

3000 - Valeur par défaut du mode exposant, index = 0.

3000E - Mode exposant, index = 0.

3000E-4 - Mode exposant, index = -4. 3001E+2 - Mode exposant, index = 2.

3001L1.5,3.5 - Mode limites, bas = 1,5, haut = 3,5.

3002L-4.0,+4.0 - Mode limites, bas = -4,0, haut = 4,0.

Les valeurs réelles sont codées de la manière suivante :

Mode exposant - Mot = valeur entière de réel * 10^{index}

Mode limites - Mot = valeur entière de $\frac{Real-Low}{High-Low}$ * FFFFh

Paramètre entier esclave

La chaîne d'adresse du bloc possède les caractères de structure suivants :

Bn - Mode d'adressage bit. n facultatif = 1..32.

Rn - Mode d'adressage registre. n facultatif = 1 ou 2.

Exemples

2000 - Valeur par défaut du mode d'adressage registre, longueur 1 mot.

2000B - Mode d'adressage bit, longueur 1 bit.

2000R - Mode d'adressage registre, longueur 1 mot.

2001B12 - Mode d'adressage bit, longueur 12 bits.

2001R2 - Mode d'adressage registre, longueur 2 mots.

Les valeurs entières sont codées de la manière suivante :

Mode bit - Un octet est formé par extraction du nombre nécessaire de bits dans l'entier double. L'octet résultant est complété par des bits nuls si besoin est.

Mode mot - Aucune conversion n'est nécessaire. Si l'entier double est défini comme ayant une longueur d'un mot, c'est le mot le moins significatif qui est utilisé.

Paramètre de chaîne esclave

La chaîne d'adresse du bloc possède les caractères de structure suivants :

Bn - Mode d'adressage bit. n facultatif ≤ longueur maximale de la chaîne exprimée en bits.

Rn - Mode d'adressage registre. n facultatif ≤ longueur maximale de la chaîne exprimée en mots.

Exemples

4000 - Valeur par défaut du mode d'adressage registre, longueur 1 mot, 2 caractères.

4000B - Mode d'adressage bit, longueur 1 bit.

4000R - Mode d'adressage registre, longueur 1 mot, 2 caractères.

4001B12 - Mode d'adressage bit, longueur 12 bits.

4001R6 - Mode d'adressage registre, longueur 6 mots, 12 caractères.

Les valeurs de la chaîne sont codées de la manière suivante :

Mode bit - Un octet est formé par extraction du nombre nécessaire de bits dans la chaîne. L'octet résultant est complété par des bits nuls si besoin est.

Mode mot - Les mots sont codés dans une structure de type entier, l'octet le plus significatif étant placé en première position et l'octet le moins significatif en deuxième position. On obtient ainsi : mot = octet_0$_{hi}$ octet_1$_{lo}$.

Paramètre esclave vectoriel

Les types vectoriels pris en charge actuellement sont : booléen, entier et réel, chacun contenant huit valeurs. Ils équivalent à huit paramètres simples du même type situés à des adresses consécutives et ayant tous la même spécification de structure.

Exemples d'adresses

Si l'on crée les variables esclaves suivantes :

Paramètre	Type de bloc	Adresse
A	Slave_Bool	JB0000
B	Slave_Str	JB0001B7
C	Slave_Int	JB0008B16
D	Slave_Real	JB0000
E	Slave_Int	JB0001
F	Slave_Str	JB0002R4

Le mappage d'adresse tel qu'il serait vu par JBus :

Domaine	Emplacement	Paramètre
Bit	0000	A
Bit	0001-0007	B
Bit	0008-0023	C
Register	0000	D
Register	0001	E
Register	0002-0005	F

Paramètres à structure Xycom

La mise en oeuvre du protocole JBus par le PC3000 prend en charge 3 types de paramètres supplémentaires non standard. Cela est utilisé de manière spécifique pour structurer les types de données pour les communications avec un terminal Xycom mais peut être utilisé dans n'importe quelle application pour laquelle une conformité stricte avec le protocole JBus n'est pas nécessaire. La principale caractéristique est l'utilisation d'un mot JBus supplémentaire pour transmettre les informations des paramètres.

Entier double (DINT)

Un paramètre xycom DINT est spécifié à l'aide du caractère X pour le champ propre au protocole dans l'adresse du paramètre, par exemple JB1000X.

La valeur du paramètre est transmise de la manière suivante à l'aide de deux registres JBus :

registre LS = (abs(i) MOD 10000) * 2 + sign_bit

registre MS = abs(i) DIV 10000

où sign_bit est 1 pour un i négatif, 0 dans les autres cas.

Cette structure permet l'utilisation d'une plage comprise entre -99999999 et +99999999.

Virgule flottante (REEL)

Le champ adresse d'un paramètre xycom REEL est spécifié comme pour un paramètre xycom DINT, par exemple JB2000X.

La valeur du paramètre est transmise de la manière suivante à l'aide de deux registres JBus :

registre LS = (abs(R_m) mod 1000) * 16 + R_e * 2 + sign_bit

registre MS = abs(R_m) div 1000

où sign_bit est 1 pour une valeur négative, 0 dans les autres cas, R_m est une valeur entière et R_e est une puissance de 10.

Cette structure permet l'utilisation d'une plage comprise entre -9999999 et 9999999 avec sept chiffres significatifs.

Durée (TIME)

Le champ adresse pour un paramètre xycom TIME possède deux caractères de structure spéciaux, par exemple JB3000.

La valeur du paramètre est transmise de la manière suivante à l'aide de deux registres JBus :

registre LS = T_{sec} * 1000 + $T_{millisec}$

$$\text{registre MS} = T_{day} * 10000 + T_{hour} * 100 + T_{min}$$

Cela permet l'utilisation d'une plage comprise entre 0ms et 6d_23h_59m_59s_999ms.

Informations sur les durées d'exécution

Le tableau ci-dessous donne des informations sur les durées d'exécution esclaves pour plusieurs types différents d'opérations avec des paramètres. Dans les applications PC3000 actuelles, l'esclave JBus_S est normalement associé à la tâche à priorité élevée 10ms. La durée d'exécution totale pour tous les blocs fonctions 10ms ne doit pas être supérieure à 10ms et le tableau ci-après peut donc servir à estimer la taille maximale de bloc utilisable.

Opération	Taille du bloc	Durée d'exécution (ms)	Paramètre temps (µs)
Lecture de mot BOOL	3	1,2	60
Ecriture de mot BOOL	3	1,6	43
Lecture de mot DINT	1	0,8	65
Ecriture de mot DINT	1	0,8	48
Lecture de mot STRING	20	3,1	69
Lecture de mot STRING	40	5,3	69
Ecriture de mot STRING	20	3,7	70
Ecriture de mot STRING	40	6,0	71
Lecture de mode REAL E	1	1,3	326
Lecture de mode REAL E	8	4,5	326
Ecriture de mode REAL E	1	0,8	187
Ecriture de mode REAL E	8	3,7	181
Lecture de mode REAL L	1	2,1	385
Lecture de mode REAL L	8	5,1	385
Ecriture de mode REAL L	1	0,9	254
Ecriture de mode REAL L	8	4,4	249
Lecture de mode REAL X	1	2,2	600
Lecture de mode REAL X	6	7,8	646
Ecriture de mode REAL X	1	2,7	212
Ecriture de mode REAL X	6	5,0	206
Lecture de bit STRING	128	1,5	390
Ecriture de bit STRING	128	1,9	361

Tableau 3-57 Informations de temps JBus_S

Chaque transaction JBus est composée d'un message de demande et d'un message de réponse. La taille de chaque message est liée à la taille du bloc et au mode de transmission (RTU ou ASCII.) Le tableau 3-39 indique la taille des messages produits pour chaque type d'opération de paramètre pour le mode RTU. Les messages en mode ASCII sont liés aux messages en mode RTU par la formule simple suivante : Ascii = RTU * 2 + 1.

Fonction	Espace d'adresse	Taille de bloc	Message de demande	Message de réponse
1/2	Bit	n	8	5 + 2n
3/4	Mot	n	8	5 + 2n
5	Bit	1	8	8
6	Mot	1	8	8
15	Bit	n	$10+\frac{(n-1)}{8}$	8
16	Mot	n	9 + 2n	8

Tableau 3-58 Taille des messages JBus_S

La taille de message et le débit de transmission ont un effet direct sur le délai d'exécution pour chaque transaction. Avec :

```
T_rx = Request_Msg_Size * Bits_Per_Character/Baud_Rate
et
T_tx = Response_Msg_Size * Bits_Per_Character/Baud_Rate
alors
T_total = T_rx + Slave_Variable_Execution_Time + T_tx
```

Base du protocole JBus

Une définition du protocole Modbus est fournie dans le 'Guide de base Gould du protocole Modbus' de Gould Inc. Programmable Control Division. Modbus est pratiquement identique à Jbus, à cette différence près que les emplacements de périphériques esclaves sont référencés avec un décalage d'une unité.

Le protocole JBus est prévu pour un maître et un maximum de 247 esclaves sur un seul réseau. Seul le maître lance une transaction. Les transactions sont soit de type question/réponse (un seul esclave est adressé) soit de type diffusion/absence de réponse (tous les esclaves sont adressés). Une transaction est composée d'une trame à question unique et réponse unique ou d'une trame à diffusion unique.

Certaines caractéristiques du protocole JBus comme la structure de la trame, le traitement des erreurs ou les fonctions effectuées sont fixes. D'autres, comme le débit, la parité, le nombre de bits d'arrêt et les modes de transmission (ASCII ou RTU) sont sélectionnables par l'utilisateur. Toutes les caractéristiques sont

sélectionnées lors de la mise en route et ne peuvent être modifiées pendant que le système tourne.

Les messages émis par le maître ont la forme d'une adresse esclave, d'un code fonction, de données et d'un code d'erreur. En l'absence de messages d'erreur, l'esclave adressé effectue l'action définie par le code fonction. L'esclave envoie ensuite un message de réponse composé de l'adresse esclave, de l'action effectuée, de la date acquise et d'un code de contrôle d'erreur. Aucune réponse n'est envoyée si le message est une diffusion indiquée par une adresse égale à 0.

En règle générale, le maître peut envoyer un autre message à n'importe quel esclave dès qu'il reçoit une réponse valable ou après une période donnée, sélectionnée par l'utilisateur, s'il ne reçoit pas de réponse. Tous les messages peuvent être envoyés sous la forme de demandes provoquant une réponse de la part de l'esclave. Toutefois, seuls les messages qui ne nécessitent pas de réponse, comme une fonction d'écriture, peuvent être envoyés sous la forme de messages de diffusion.

Modes de transmission

Deux modes de transmission sont utilisables dans un système JBus : les modes ASCII et RTU décrits ci-après. Il est possible d'utiliser un seul mode à la fois dans un système.

Caractéristique	ASCII	RTU
Système de codage	Hexadécimal (0-9,A-F)	8 bits binaire
Bits de départ	1	1
Bits de données	7	8
Bits de parité (facultatifs)	1	1
Bits d'arrêt	1 ou 2	1 ou 2
Contrôle d'erreur	Contrôle de redondance longitudinale	Contrôle de redondance cyclique

Tableau 3-59 Modes de transmission JBus

Structure de message

Le protocole définit deux types de structure de message : ASCII et RTU. L'interprétation des champs dans les deux types de messages est rigoureusement identique. La différence principale réside dans le type de contrôle d'erreur effectué et dans le fait que le nombre de caractères utilisé en ASCII est à peu près double du nombre de caractères utilisé dans le mode RTU. A la place d'un caractère unique à 8 bits, la paire équivalente de caractères ASCII 7 bits (0-9,A-F) est envoyée.

Encadrement ASCII

L'encadrement en mode de transmission ASCII est effectué à l'aide du caractère deux points pour indiquer le commencement d'un encadrement et un retour chariot (CR) saut de ligne (LF) pour en indiquer la fin.

Début de l'encradrement	Addressse	Fonction	Données	Contrôle d'erreur	Fin de l'encadrement
:	2 caractères	2 caractères	N x 4 caractères	2 caractères	CR LF
Deux points	16 bits	16 bits	N x 16 bits	16 bits	

Tableau 3-60 Encadrement de message JBus

Encadrement d'unité terminale déportée (RTU)

La fin d'un message en mode RTU est définie sous la forme d'un silence de trois caractères au débit de service. Cette durée peut ne pas être exacte dans la mise en oeuvre actuelle. Le début du message suivant est censé s'être produit lorsque cette interruption est détectée.

Début de l'encadrement	Adresse	Fonction	Données	Contrôle d'erreur	Fin de l'encadrement
T1 T2 T3	8 bits	8 bits	8 bits	16 bits	T1 T2 T3

Tableau 3-61 Structure du message JBus

Champ Adresse

Le champ adresse suit immédiatement le début de l'encradrement et est composé de 8 bits (RTU) ou de 2 caractères (ASCII). Ce champ indique l'adresse de l'esclave, définie par l'utilisateur, qui doit recevoir le message envoyé par le maître. Chaque esclave doit recevoir une adresse unique. Une adresse égale à 0 est utilisée dans un message de diffusion. Tous les esclaves répondront à un message de diffusion mais aucune réponse ne sera envoyée.

Champ de fonction

Le champ Code de fonction indique à l'esclave adressé l'action qu'il doit effectuer. La liste des fonctions prises en charge est la suivante :

Code fonction	Action
1 ou 2	Lecture de N bits
3 ou 4	Lecture de N mots
5	Ecriture d'1 bit
6	Ecriture d'1 mot
15	Ecriture de N bits
16	Ecriture de N mots

Tableau 3-62 Codes fonction JBus

où N = 1..256.

Champ de données

Ce champ de données contient des informations dont l'esclave a besoin pour effectuer la fonction indiquée ou contient des données collectées par l'esclave en réponse à une question.

Champ de contrôle d'erreur

Ce champ permet aux périphériques maîtres et esclaves de rechercher des erreurs de transmission dans un message. Le champ de contrôle d'erreur utilise un contrôle de redondance longitudinale (LRC) en mode ASCII et un contrôle CRC 16 en mode RTU.

Réponses d'exception

Les codes de réponse d'exception pris en charge sont énumérés ci-dessous. Lorsqu'un esclave détecte une de ces erreurs, il envoie un message de réponse au maître ; ce message est composé des champs adresse de l'esclave, code fonction, code d'erreur et contrôle d'erreur. Pour indiquer que la réponse est une signalisation d'erreur, le bit 7 du code fonction est positionné sur 1.

Code	Nom	Signification
01	FONCTION INTERDITE	Le message reçu n'est pas une action autorisée pour l'esclave adressé
02	ADRESSE DE DONNEES INTERDITE	L'adresse indiquée dans le champ de données n'est pas valable
04	DEFAILLANCE DANS LE PERIPHERIQUE ASSOCIE	Une erreur provoquant une interruption s'est produite (tentative d'écriture dans une adresse protégée en écriture par exemple).

Tableau 3-63 Codes de réponse d'exception JBus

Exemple de réponse d'exception :

Adresse esclave	Fonction	Code d'exception	Contrôle d'erreur
0A	81	02	73

Tableau 3-64 Codes de réponse d'exception JBus

N.B. : La réponse d'exception 03 - VALEUR DE DONNEES INTERDITE n'est pas mise en oeuvre car le programme utilisateur devrait s'assurer que les valeurs ne peuvent pas être supérieures aux plages définies.

Détails des fonctions

La présente section a pour objet de définir la structure générale des fonctions spécifiques disponibles dans le protocole JBus. La structure du message de demande émanant du maître est représentée, suivie du message de réponse envoyé par l'esclave. La structure est représentée en RTU, qui peut être converti facilement en ASCII. Les mots sont transmis avec l'octet le plus significatif en première position. Les fonctions bit permettent un transfert maximal de 1024 bits et les fonctions mot un maximum de 125 mots dans une seule opération.

Lecture de N bits (Code fonction 01 ou 02)

Demande :

Octet 1	Adresse (1..247).
Octet 2	Code fonction (1 ou 2).
Octets 3 & 4	Adresse du premier bit à lire.
Octets 5 & 6	Nombre de bits à lire.
Octets 7 & 8	Total de contrôle CRC-16.

Réponse :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (1 ou 2).
Octet 3	Nombre d'octets lus N.
Octets 4 à 4+N-1	Octets contenant les bits lus.
Octets 4+N & 4+N+1	Total de contrôle CRC-16

Les bits lus sont condensés en octets commençant au bit 0 pour la premiée adresse. Le dernier octet est complété par des bits 0 si besoin est.

Exemple : bits lus entre 0020 et 0056 à partir du numéro d'esclave 17.

Demande : 11 01 00 14 00 25 CRC-16

Réponse : 11 01 05 CD 6B B2 0E 1B CRC-16

Lecture de N mots (Code fonction 03 ou 04)

Demande :

Octet 1	Adresse esclave (1..247)
Octet 2	Code fonction (3 ou 4).
Octets 3 & 4	Adresse du premier mot à lire.
Octets 5 & 6	Nombre de mots à lire.
Octets 7 & 8	Total de contrôle CRC-16.

Réponse :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (3 ou 4).
Octet 3	Nombre d'octets lus N.
Octets 4 à 4+N-1	Mots mémorisés octet le plus significatif, octet le moins significatif
Octets 4+N & 4+N+1	Total de contrôle CRC-16.

Exemple : bits lus de 0107 à 0110, à partir du numéro d'esclave 17.

Demande : 11 03 00 6B 00 03 CRC-16

Réponse : 11 03 06 02 2B 00 00 00 64 CRC-16

Ecriture de 1 bit (code fonction 05)

Demande :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (5).
Octets 3 & 4	Adresse du bit à écrire.
Octet 5	Valeur bit. FF pour fixer, 00 pour supprimer.
Octet 6	Egal à 0.
Octets 7 & 8	Total de contrôle CRC-16.

Réponse :

Identique à la demande.

Exemple : activer le bit à l'adresse 0173 dans l'esclave numéro 17.

Demande : 11 05 00 AD FF 00 CRC-16

Réponse : 11 05 00 AD FF 00 CRC-16

Ecriture de 1 mot (code fonction 06)

Demande :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (6).
Octets 3 & 4	Adresse du mot à écrire.
Octets 5 & 6	Valeur du mot.
Octets 7 & 8	Total de contrôle CRC-16.

Réponse :

Identique à la demande.

Exemple : écrire 128 dans le mot à l'adresse 0135 dans l'esclave numéro 17.

Demande : 11 06 00 87 00 80 CRC-16

Réponse : 11 06 00 87 00 80 CRC-16

Ecriture de N bits (code fonction 15)

Demande :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (15).
Octets 3 & 4	Adresse du premier bit à écrire.
Octets 5 & 6	Nombre de bits à écrire n.
Octet 7	Nombre d'octets à écrire N = int(n/8)+1.
Octets 8 à 8+N-1	Octets contenant les bits à écrire.
Octet 8+N & 8+N+1	Total de contrôle CRC-16.

Réponse :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (15).
Octets 3 & 4	Adresse du premier bit écrit.
Octets 5 & 6	Nombre de bits écrits.
Octets 7 & 8	Total de contrôle CRC-16.

Exemple : écriture de 10 bits commençant à l'adresse 0020 dans l'esclave numéro 17.

Demande : 11 0F 00 14 00 0A 02 CD 00 CRC-16

Réponse : 11 0F 00 14 00 0A CRC-16

Ecriture de N mots (code fonction 16)

Demande :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (16).
Octet 3 & 4	Adresse du premier mot à écrire.
Octet 5 & 6	Nombre de mots à écrire n.
Octet 7	Nombre d'octets à écrire N = n * 2.
Octets 8 à 8+N-1	Mots mémorisés octet le plus significatif, octet le moins significatif
Octet 8+N & 8+N+1	Total de contrôle CRC-16.

Réponse :

Octet 1	Adresse esclave (1..247).
Octet 2	Code fonction (16).
Octet 3 & 4	Adresse du premier mot écrit.
Octet 5 & 6	Nombre de mots écrits.
Octet 7 & 8	Total de contrôle CRC-16

Exemple : écriture de 2 mots commençant à l'adresse 0135 dans l'esclave numéro 17.

Demande : 11 10 00 87 00 02 04 00 0A 01 02 CRC-16

Réponse : 11 10 00 87 00 02 CRC-16

Signalisation des erreurs

En cas d'erreur spécifique du bloc fonction JBus_S Bloc ou du protocole JBus, l'erreur est signalée par le paramètre **Error_No** du bloc fonction. Les erreurs portant sur une variable spécifique sont signalées par le bloc de variable esclave associé.

Error_No	Description de l'erreur
145	ERREUR DE PORT : ABSENCE D'ADRESSE •Le paramètre Port du bloc fonction contient moins de deux caractères.
149	ERREUR DE PORT : DEBIT RX PAS DISPONIBLE •Le débit de réception demandé n'est pas disponible sur ce port série.
150	ERREUR DE PORT : DEBIT TX PAS DISPONIBLE •Le débit d'émission demandé n'est pas disponible sur ce port série.
155	ERREUR DE PORT : EMPLACEMENT INTERDIT •Le numéro d'emplacement sélectionné n'est pas autorisé. Le numéro d'emplacement est le premier caractère du paramètre Port.
156	ERREUR DE PORT : PORT INTERDIT •Le numéro de port sélectionné n'est pas autorisé. Le numéro de port (deuxième caractère du paramètre Port) doit être valable pour le module, par exemple 'A' à 'C' pour un port LCM ou 'A' à 'D' pour un port ICM.
161	ERREUR DE PORT : PORT UTILISE •Le Port sélectionné est déjà utilisé pour un autre module.
162	ERREUR DE PORT : TROP GRAND NOMBRE DE PARAMETRES ESCLAVES •Un trop grand nombre de paramètres esclaves ont été déclarés dans le système.
163	ERREUR DE PORT : TROP GRAND NOMBRE DE TYPES DE MODULES •Un trop grand nombre de types de modules esclaves ont été déclarés dans le système.

Tableau 3-65 Codes d'erreur JBus_S

Codes d'erreur de variables esclaves

Error_No	Description de l'erreur
128	CHAINE D'ADRESSE TROP COURTE •La chaîne d'adresse contenue dans le paramètre Adresse est trop courte pour être valable.
129	ADRESSE HORS PLAGE •L'adresse spécifiée dans le paramètre Adresse ne se situe pas dans la plage valable comprise entre 0000 et 3999.
130	CARACTERE PAS VALABLE DANS LA CHAINE D'ADRESSE •Un caractère pas valable a été détecté dans le champ de structure de données du paramètre Adresse.
131	VALEUR NUMERIQUE DE STRUCTURE HORS PLAGE •Une valeur de taille de paramètre spécifiée dans le paramètre Adresse est située en dehors de la plage de ce type de données.
132	TYPE DE PARAMETRE PAS VALABLE •Un type de bloc de variable esclave qui a été créé ne peut pas être traité par la configuration actuelle du module.
134	CHEVAUCHEMENT D'ADRESSES •La plage d'adresse spécifiée dans le paramètre Adresse chevauche partiellement ou entièrement la plage d'adresse d'un autre paramètre esclave.
135	TYPES MULTI-ELEMENTS MELANGES •Un bloc de variable esclave possède des éléments matriciels de types différents. La configuration actuelle du module ne peut pas prendre en charge les types mélangés.

Tableau 3-66 Codes d'erreur de la variable esclave J Bus_S

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	STRING	'0A'	Oper	Config		
Unit_Id	SINT	0	Oper	Oper	Lim. haute Lim. basse	247 0
Baud	ENUM	_9600	Oper	Config	Valeurs énumérées	_75(0) _300(1) _600(2) _1200(3) _2400(4) _4800(5) _9600(6) _19200(7) _38400(8) _57600(9) _115200(10)
Tx_Mode	BOOL	RTU	Oper	Config	Détection	RTU(0) Ascii(1)
Parity	ENUM	NONE	Config	Config	Valeurs énumérées	PAIR(0) IMPAIR(1) ESPACE(2) MARQUE(3) NEANT(4)
Stop_Bits	ENUM	_1	Config	Config	Valeurs énumérées	_1(0) _1_5(1)
Wr_Protect	BOOL	No	Oper	Config	Détection	No(0) Yes(1)
Status	BOOL	NoGo	Oper	Bloc	Détection	NOGO(0) Go(1)
Error_No	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0

Figure 3-67 Attributs des paramètres JBus_S

BLOC FONCTION TOSHIBA_M

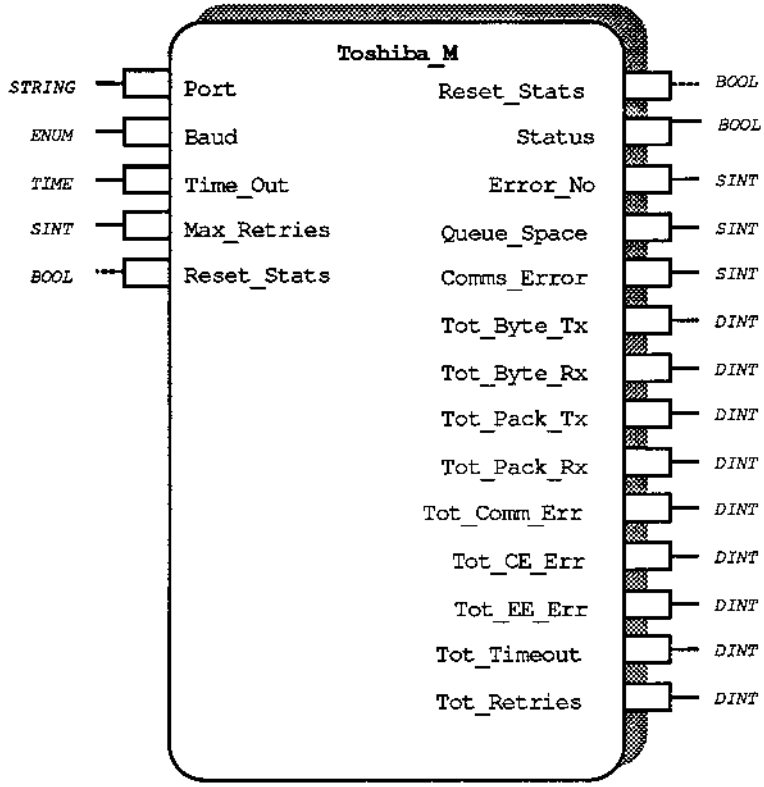


Figure 3-38 Diagramme du bloc fonction

Description fonctionnelle

Le bloc fonction Toshiba_M prend en charge les communications série sur un port série de communications désigné en utilisant le protocole PLC Toshiba. Cette description du bloc fonction comprend les informations relatives aux commandes et au registre Toshiba qui permettent de se documenter rapidement sur le sujet. Avant de lire cette description, vous avez intérêt à acquérir des connaissances générales sur le système de commmunications PC3000 en lisant la présentation des communications PC3000.

Ce bloc fonction sera nécessaire pour concevoir ou programmer le PC3000 de telle manière qu'il puisse utiliser une interface PLC Toshiba fonctionnant en mode maître, c'est-à-dire que le PC3000 est relié par une liaison série à un PLC Toshiba configuré comme esclave.

Nous vous conseillons de consulter les documents suivants si vous avez besoin de comprendre en détail le protocole PLC Toshiba :

•EX250500 Toshiba Corporation. *Module de liaison informatique sur automate programmable Toshiba. Manuel utilisateur CMP-6236 EX250/500 (Référence : UM-EX25U-E104(Mar. '87(2)))*

•EX2000 Toshiba Corporation. *Manuel utilisateur de liaison informatique sur automate programmable Toshiba EX2000. (Référence : UM-EX2K***-E004(Mar. '88(1)))*

Le bloc fonction Toshiba_M fournit un module pour le protocole de communication destiné aux périphériques Toshiba EX250, EX500 et EX2000 ; le bloc fonction permet au PC3000 d'agir en qualité de maître de la liaison série.

Ce bloc fonction traite des détails des communications propres au protocole et est pris en charge par des blocs fonctions génériques de variables déportées. Les blocs de variables déportées sont liés au module par une adresse propre au protocole et peuvent nécessiter l'interrogation ou la mise à jour des paramètres d'un appareil en communication par l'intermédiaire d'un port de communications spécifié.

Pour avoir des détails supplémentaires sur les blocs fonctions de variables déportées, cf. la 'Présentation des communications PC3000'.

Seule la version ASCII du protocole est prise en charge. Le module fonctionne toujours en **Parité paire.**

Attributs du bloc fonction

Type: 8 88

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port status Queue_Space Comms_Error

Besoin de capacité mémoire : . 2026 octets

Description des paramètres

Paramètres de configuration du module

Le bloc **Toshiba_M** possède deux paramètres de saisie de configuration, Port et Baud, qui configurent le module et doivent être définis avant l'exécution du programme utilisateur. La modification de ces paramètres au cours de l'exécution du programme utilisateur n'aura aucun effet sur le module, sauf dans des circonstances particulières : cf. 'Présentation des communications PC3000', section 'Modification temporaire des paramètres de configuration'. Les autres paramètres de saisie peuvent être modifiés au cours de l'exécution du programme utilisateur.

Port

Le paramètre **Port** est l'adresse à deux caractères du port sur lequel tourne le protocole Toshiba. Le premier caractère est un chiffre compris entre 0 et 5, qui représente l'emplacement dans le rack et le deuxième caractère est une lettre qui représente le port dans cet emplacement ; par exemple, '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' pourrait désigner son port supérieur.

Baud

Le paramètre **Baud** offre un choix de 6 débits différents compris entre 300 et 9600 Bauds (indiqués dans le tableau 4-20), avec une valeur par défaut de 9600 Bauds.

N.B. : tous les ports ne peuvent pas prendre en charge la totalité des débits. Si un débit ne peut pas être pris en charge, cela sera indiqué par une erreur lors de la première exécution du bloc fonction (cf. la description d'**Error_No**).

Valeur de l'énumération	Débit
0	300
1	600
2	1200
3	2400
4	4800
5	9600

Tableau 3-68 Débits Toshiba_M

Time_Out

Le paramètre **Time_Out** spécifie la durée pendant laquelle le maître Toshiba attendra un message-réponse à une demande. Une fois cette durée écoulée, le module considère qu'il y a eu une erreur de transmission et que la demande peut

être retransmise ou qu'une erreur peut être renvoyée au bloc de variable déportée qui a émis la demande.

Time_Out prend par défaut la valeur 1 seconde.

Max_Retries

Le paramètre **Max_Retries** spécifie le nombre de réémissions d'une demande si une erreur de transmission comme un dépassement du délai imparti ou une erreur de total de contrôle de message est détectée. Si aucune réponse valable n'est reçue après ce nombre de tentatives, une erreur est renvoyée au bloc de variable déportée qui a émis la demande.

Max_Retries prend par défaut la valeur 2, une demande sera donc envoyée trois fois avant qu'une erreur soit signalée et la demande arrêtée.

Paramètres d'état du module

L'état du module est indiqué par trois paramètres de sortie dans le bloc fonction **Toshiba_M**.

Etat

Le paramètre **Etat** est une indication booléenne de l'état de la liaison contrôlée par le module. En l'absence de problèmes sur la liaison, ce paramètre indique **Go** mais, lorsqu'une erreur se produit, il passe sur **NOGO**. Si l'**Etat** est sur **NOGO**, le paramètre **Error_No** indique la cause du problème.

Error_No

Le paramètre **Error_No** indique la cause des erreurs éventuelles sur la liaison. Si l'**Etat** est sur **Go**, **Error_No** sera sur 0 (OK). Pour avoir des détails complets sur les codes d'erreur, cf. la partie Signalisation des erreurs, page 3-

Queue_Space

Le paramètre **Queue_Space** indique l'espace restant dans la file d'attente pour les opérations avec les paramètres déportés. S'il atteint zéro, cela implique que la largeur de bande de la liaison est insuffisante pour faire face au nombre de demandes de variables déportées effectuées et les données seront perdues. Si cette situation survient, il faut diminuer les fréquences d'interrogation des paramètres.

Comms_Error

Le paramètre **Comms_Error** indique l'état de la dernière transaction exécutée par le module. La détection d'une erreur apparaît ici. Il faut noter que, si un grand nombre de transactions se produisent, toute erreur détectée et signalée ne restera dans ce paramètre que jusqu'à l'achèvement de la transaction suivante.

Paramètres statistiques du module

Les données statistiques du module sont indiquées par neuf paramètres de sortie dans le bloc fonction **Toshiba_M**.

Reset_Stats

Le paramètre **Reset_Stats** sert à réinitialiser toutes les sorties d'informations statistiques, ce qui s'effectue par positionnement de la valeur booléenne sur **Oui**. Ce paramètre passe automatiquement sur **Non** une fois que les sorties statistiques ont été réinitialisées.

Tot_Byte_Tx

Le paramètre **Tot_Byte_Tx** indique le nombre total d'octets qui ont été transmis par le module depuis le début de l'exécution de celui-ci ou depuis la dernière réinitialisation des sorties statistiques.

Tot_Byte_Rx

Le paramètre **Tot_Byte_Rx** indique le nombre total d'octets qui ont été reçus par le module depuis le début de l'exécution de celui-ci ou depuis la dernière réinitialisation des sorties statistiques.

Tot_Packet_Tx

Le paramètre **Tot_Packet_Tx** indique le nombre total de paquets qui ont été transmis par le module depuis le début de l'exécution de celui-ci ou depuis la dernière réinitialisation des sorties statistiques.

Tot_Packet_Rx

Le paramètre **Tot_Packet_Rx** indique le nombre total de paquets qui ont été reçus par le module depuis le début de l'exécution de celui-ci ou depuis la dernière réinitialisation des sorties statistiques.

Tot_Comm_Err

Le paramètre **Tot_Comm_Err** indique le nombre total d'erreurs de communications qui ont été détectées par le module depuis le début de l'exécution de celui-ci ou depuis la dernière réinitialisation des sorties statistiques. Il comprend à la fois les valeurs **Tot_CE_Err** et **Tot_EE_Err**, ainsi que le nombre total de dépassements du délai imparti, d'erreurs de total de contrôle et les autres erreurs éventuelles qui ont été détectées au cours du processus d'émission/réception.

Tot_CE_Err

Le paramètre **Tot_CE_Err** indique le nombre total d'**erreurs de liaisons informatiques** qui ont été reçues du périphérique Toshiba EX et qui ont été

détectées par le module depuis le début de l'exécution ou depuis la dernière réinitialisation des sorties statistiques.

Tot_EE_Err

Le paramètre **Tot_EE_Err** indique le nombre total de **Fausses erreurs** qui ont été reçues du périphérique Toshiba EX et qui ont été détectées par le module depuis le début de l'exécution ou depuis la dernière réinitialisation des sorties statistiques.

Tot_Timeouts

Le paramètre **Tot_Timeouts** indique le nombre total de dépassements du délai imparti qui ont été détectés par le module depuis le début de l'exécution ou depuis la dernière réinitialisation des sorties statistiques. Un dépassement du délai imparti se produit lorsque la durée entre l'émission d'une demande et la réception de la réponse en provenance du périphérique Toshiba est supérieure à la limite fixée par le paramètre d'entrée **Time_Out**.

Tot_Retries

Le paramètre **Tot_Retries** indique le nombre total de nouvelles tentatives qui se sont produites depuis le début de l'exécution ou depuis la dernière réinitialisation des sorties statistiques. Une nouvelle tentative se produit en cas de détection d'un dépassement du délai imparti ou d'une erreur dans l'opération de communications.

Opérations avec les variables déportées

Les demandes émanant du PC3000 sont contrôlées par un ou plusieurs blocs de variables déportées.

Le module **Toshiba_M** prend actuellement en charge les blocs de types **Remote_Int**, **Remote_Str** et **Remote_SW**.

Adressage

Il est nécessaire de configurer une **Adresse** propre au protocole dans le bloc de variable déportée qui est l'adresse servant à accéder aux périphériques déportés. L'adresse est composée de la manière suivante.

Le port est défini, comme dans le paramètre **Port**, sous la forme d'un numéro d'emplacement dans le rack suivi d'une lettre correspondant au port dans cet emplacement.

La partie de l'adresse propre au protocole commence par un numéro de station précisant le périphérique Toshiba qui doit répondre à la demande. Ce numéro est composé d'un ou deux caractères, selon le périphérique Toshiba auquel est envoyée la demande. Pour un périphérique Toshiba EX250 ou EX500, l'adresse est un caractère compris entre '0' et '7', du fait que huit périphériques seulement peuvent être reliés ensemble sur une liaison série provenant de l'ordinateur hôte.

Pour un périphérique Toshiba EX2000, le numéro de station est composé de deux caractères compris entre '01' et '32' si l'EX2000 est sur une liaison série RS485 ou entre '01' et '08' s'il est sur une liaison série RS422 ; là aussi, ces plages sont limitées par le nombre de périphériques qui peuvent être placés sur une liaison série provenant de l'ordinateur hôte.

Le numéro de station est suivi de la commande qui doit être exécutée sur le périphérique Toshiba. La commande est toujours un mnémonique à deux caractères décrit dans la documentation Toshiba pour les périphériques EX250, EX500 ou EX2000.

La fin de l'adresse dépend de la commande spécifiée. Dans le cas des commandes de lecture ou d'écriture de périphérique ou de registre, l'adresse doit se terminer par le type de périphérique ou de registre, le périphérique ou le registre initial et le nombre de périphériques ou de registres à lire ou écrire. La structure de ces informations est présentée dans les exemples ci-dessous et figure également dans les documents Toshiba. Pour toutes les autres commandes, aucune information supplémentaire n'est nécessaire.

Les commandes de lecture et d'écriture de périphérique ou de registre, **DR** et **DW**, peuvent être facultativement spécifiées sous la forme **D?** ; le mnémonique de lecture ou d'écriture qui convient sera remplacé en interne, selon que l'opération déclenchée est une lecture ou une écriture.

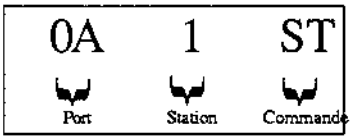


Figure 3-39 Exemple 1 d'adresse de variable déportée

N.B. 1 : l'exemple de la figure 3-39 montre une commande **'Lecture de l'état'** émise par un périphérique EX250/500.

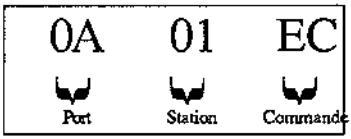


Figure 3-40 Exemple 2 d'adresse de variable déportée

N.B. 2 : l'exemple de la figure 3-40 montre une commande **'Contrôle EX'** émise en direction d'un périphérique EX2000.

Figure 3-41 Exemple 3 d'adresse de variable déportée

N.B. 3 : l'exemple de la figure 3-41 montre une commande **'Lecture de périphérique/registre'** émise en direction d'un périphérique EX2000 pour qu'il lise le registre d'horloge 12.

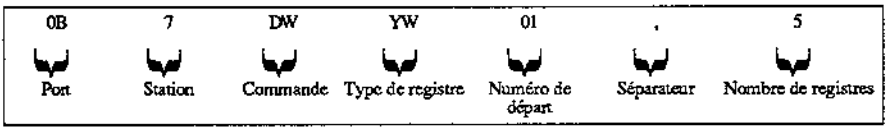


Figure 3-42 Exemple 4 d'adresse de variable déportée

N.B. 4 : l'exemple de la figure 3-42 montre une commande **'Ecriture de périphérique/registre'** émise en direction d'un périphérique EX250/500 pour qu'il écrive dans cinq registres de sortie externes commençant au registre 1.

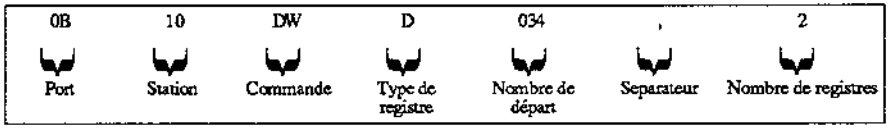


Figure 3-43 Exemple 5 d'adresse de variable déportée

N.B. 5 : l'exemple de la figure 3-43 montre une commande **'Ecriture de périphérique/registre'** émise en direction d'un périphérique EX2000 pour qu'il écrive dans deux registres de données commençant au registre 34.

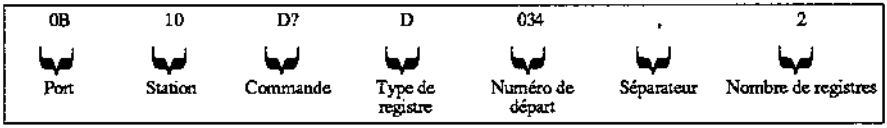


Figure 3-44 Exemple 6 d'adresse de variable déportée

N.B. 6 : l'exemple de la figure 3-44 montre une commande **'Lecture ou écriture de périphérique ou de registre'** (selon le déclenchement) émise en direction d'un périphérique EX2000 pour qu'il écrive deux registres de données commençant au registre 34.

Figure 3-45 Exemple 7 d'adresse de variable déportée

N.B. 7 : l'exemple de la figure 3-45 montre une commande **'Ecriture de périphérique/registre'** émise en direction d'un périphérique EX250/500 pour qu'il écrive dans 4 périphériques de sortie externes commençant au périphérique 1B (Module 1, canal B). Cela provoque ainsi l'écriture dans les périphériques Y1B, Y1C, Y1D et Y1E (c'est-à-dire Module 1, canaux B, C, D et E).

Le paramètre Adresse d'un bloc fonction de variable déportée peut être modifié à tout moment, ce qui permet de réutiliser le bloc pour communiquer avec d'autres appareils et d'autres paramètres. Il est déconseillé de le modifier lorsque le paramètre Etat des blocs fonctions est sur 'En attente', c'est-à-dire qu'une demande est en cours.

Structures des données

Cette section décrit les différents codes de données sélectionnables pour chaque type de données élémentaire.

Entier

Le bloc fonction Remote_Int donne accès à un emplacement 'Registre' unique ou, dans le cas des périphériques d'entrée, de sortie et de relais, à un maximum de 16 emplacements de 'Périphériques'. Dans le cas d'autres 'périphériques', il donne accès à un seul emplacement. Le bloc fonction Remote_SW comprime/décomprime la valeur de seize paramètre numériques.

Chaîne

Le bloc fonction Remote_Str donne accès à un ou plusieurs emplacements de 'périphériques/registres' ou à des informations 'système ou diagnostic' sur le périphérique esclave. Lorsque ce bloc est utilisé avec les commandes 'Périphérique/registre' ou avec des commandes qui accèdent à des valeurs entières uniques, les données sont saisies sous forme hexadécimale, chaque caractère étant précédé du symbole '$'. Lorsqu'il est utilisé avec d'autres

commandes, les données sont saisies sous forme de texte ASCII simple (utilisé uniquement pour la commande 'TS').

Erreurs

S'il y a une erreur dans la chaîne d'adresse ou s'il y a une erreur de communications, le paramètre Etat du bloc de paramètres à distance est sur NOGO et le paramètre Error_No indiquera la cause.

N.B. : le contenu de la chaîne d'adresse est uniquement validé lors du lancement d'une lecture ou d'une écriture déportée.

Base des commandes Toshiba

Les commandes énumérées ci-dessous comprennent les commandes EX250/500 et EX2000. Lorsque ces commandes s'appliquent en particulier à un des périphériques, elles sont clairement identifiées.

Commande	Description
ER	Lecture d'état d'erreur EX •Lit l'état d'erreur du périphérique EX.
TS	Test •Renvoie le texte transmis comme s'il avait été reçu sur le périphérique EX.
ST	Lecture d'état EX •Lit l'état du périphérique EX (MARCHE/ARRET/ERREUR).
DR	Lecture de périphérique/registre •Lit le contenu du périphérique ou du registre.
DW	Ecriture de périphérique/registre •Ecrit le contenu du périphérique ou du registre.
BR†§	Lecture des blocs de programmes (EX250/500 uniquement) •Lit les programmes bloc par bloc.
BR¥§§	Lecture des blocs d'informations de programmes (EX2000 uniquement) •Lit les informations de programmes bloc par bloc.

Tableau 3-69 Commandes Toshiba EX250/500 et EX2000

† Périphériques EX250 et EX500 uniquement.

¥ Périphérique EX2000 uniquement.

§ Utilisation de cette commande par EX250/500.

§§ Utilisation de cette commande par EX2000.

Commande	Description
RB¥	Lecture des blocs de programmes •Lit les programmes bloc par bloc.
BW†§	Ecriture des blocs de programmes •Ecrit les programmes bloc par bloc.
BW¥§§	Ecriture des blocs d'informations de programmes. •Ecrit les informations de programmes bloc par bloc.
WB¥	Ecriture des blocs de programmes •Ecrit les programmes bloc par bloc.
EC	Contrôle EX •Contrôle du mode de fonctionnement du périphérique EX.
IR	Lecture du haut de la zone de sauvegarde •Lit le registre de départ pour le haut de la zone de sauvegarde.

Tableau 3-70 Commandes Toshiba EX250/500 et EX2000

† Utilisation de cette commande par EX250/500.

†† Utilisation de cette commande par EX2000.

§ Utilisation de cette commande par EX250/500.

§§ Utilisation de cette commande par EX2000.

Commande	Description
TR	Lecture de tableau de diagnostic •Lit les informations d'erreur définies par l'utilisateur.
SR†	Lecture des paramètres système •Lit les informations système EX250/500.
S1¥	Lecture du paramètre système 1 •Lit les informations système 1 du périphérique EX2000.
S2¥	Lecture du paramètre système 2 •Lit les informations 2 du périphérique EX2000.
RT¥	Lecture du calendrier ou de l'horloge •Lit les informations du calendrier ou de l'horloge.
WT¥	Ecriture du calendrier ou de l'horloge •Ecrit les informations du calendrier ou de l'horloge.

Tableau 3-71 Commandes Toshiba EX250/500 et EX2000

† Périphériques EX250 et EX500 uniquement.

¥ Périphérique EX2000 uniquement.

Références des périphériques/registres Toshiba

Registre	EX250	EX250 (extension RAM)	EX500	EX2000 (RAM 16Ko)	EX2000 (RAM 32 Ko)
Registres d'entrée XW	XW00 - XW31	XW00 - XW31	XW00 - XW63	XW0000 - XW0499	XW0000 - XW0499
Registres de sortie YW	YW00 - YW31	YW00 - YW31	YW00 - YW63	YW0000 - YW0499	YW0000 - YW0499
Registres d'entrée RW	RW00 - RW63	RW00 - RW63	RW00 - RW63	RW0000 - RW0999	RW0000 - RW0999
Registres d'entrée ZW	ZW00 - ZW31	ZW00 - ZW31	ZW00 - ZW31	ZW0000 - ZW1999	ZW0000 - ZW1999
Registres de données D	D000 - D511 or 0000 - 0511	D000 - D999 ou 0000 - 1535†	D000 - D999 ou 0000 - 1535†	D00000 - D08191	D00000 - D16383
Registres d'horloge T	T000 - T127	T000 - T127	T000 - T127	T00000 - T00499	T00000 - T00499
Registres de compteur C	C000 - C095	C000 - C095	C000 - C095	C00000 - C00499	C00000 - C00499
Périphériques d'entrée externes X	X000 - X15F	X000 - X15F	X000 - X31F	X00000 - X0499F	X00000 - X0499F
Périphériques de sortie externes Y	Y000 - Y15F	Y000 - Y15F	Y000 - Y31F	Y00000 - Y0499F	Y00000 - Y0499F
Périphériques de relais auxiliaires R	R000 - R63F (Bobine spéciale : R600 - R63F)	R000 - R63F (Special coil: R600 - R63F)	R000 - R63F (Bobine spéciale : R600 - R63F)	R00000 - R0999F (Bobine spéciale : R09000 - R0999F)	R00000 - R0999F (Bobine spéciale : R09000 - R0999F)
Registres d'horloge Z	Z000 - Z31F	Z000 - Z31F	Z000 - Z31F	Z00000 - Z0999F	Z00000 - Z0999F
Registres d'horloge T ¥				T.0000 - T.0499	T.0000 - T.0499
Registres d'horloge C. ¥				C.0000 - C.0499	C.0000 - C.0499

Tableau 3-72 Noms et plages de périphériques/registres Toshiba EX250/500 et EX2000

† Périphériques EX250 et EX500 uniquement.

¥ Périphérique EX2000 uniquement.

Signalisation des erreurs

Codes d'erreur de module

Ces erreurs apparaissent sous la forme d'**Error_No** sur le bloc **Toshiba_M**.

Error_No	Description de l'erreur
1	ERREUR DE PORT : ABSENCE D'ADRESSE •Le paramètre Port du bloc fonction contient moins de deux caractères.
5/6	ERREUR DE PORT : DEBIT PAS DISPONIBLE •Le débit demandé n'est pas disponible sur ce port série.
11	ERREUR DE PORT : EMPLACEMENT INTERDIT •Le numéro d'emplacement sélectionné n'est pas autorisé. Le numéro d'emplacement est le premier caractère du paramètre Port et doit être compris entre '0' et '5'.
12	ERREUR DE PORT : PORT INTERDIT •Le numéro de port sélectionné n'est pas autorisé. Le numéro de port est le deuxième caractère du paramètre Port et doit être compris entre 'A' et 'C' pour un LCM ou entre 'A' et 'D' pour un ICM.
17	ERREUR DE PORT : PORT UTILISE •Le Port sélectionné est déjà utilisé pour un autre module.

Tableau 3-73 Codes d'erreur Toshiba_M

Codes d'erreur de variables déportées

Ces erreurs apparaissent sous la forme d'**Error_No** sur le bloc de la variable déportée et **Comms_Error** sur le bloc **Toshiba_M**.

Error_No	Description de l'erreur
11	ERREUR D'ADRESSE : EMPLACEMENT INTERDIT •Le premier caractère du paramètre Adresse n'est pas compris dans la plage valable comprise entre '0' et '5'.
12	ERREUR D'ADRESSE : PORT INTERDIT •Le deuxième caractère du paramètre Adresse n'est pas situé dans la plage valable comprise entre 'A' et 'C' pour un port LCM ou entre 'A' et 'D' pour un port ICM.
16	ERREUR D'ADRESSE : ABSENCE DE PRISE EN CHARGE DE PARAMETRE DEPORTE •Aucun module maître correct n'est affecté au port indiqué dans le paramètre Adresse.
50	ERREUR DE COMMUNICATIONS : SURCHARGE RX •Une erreur de surcharge a été détectée sur un caractère reçu.
51	ERREUR DE COMMUNICATIONS : PARITE RX •Une erreur de parité a été détectée sur un caractère reçu.
52	ERREUR DE COMMUNICATIONS : PARITE & SURCHARGE RX •Une erreur de parité et de surcharge a été détectée au cours de la réception.
53	ERREUR DE COMMUNICATIONS : ENCADREMENT RX •Une erreur d'encadrement a été détectée sur un caractère reçu.
54	ERREUR DE COMMUNICATIONS : ENCADREMENT ET SURCHARGE RX •Une erreur d'encadrement et de surcharge a été détectée au cours de la réception.
55	ERREUR DE COMMUNICATIONS : ENCADREMENT ET PARITE RX •Une erreur d'encadrement et de parité a été détectée au cours de la réception.
56	ERREUR DE COMMUNICATIONS : ENCADREMENT, SURCHARGE ET PARITE RX •Une erreur d'encadrement, de parité et de surcharge a été détectée au cours de la réception.
57-64	ETAT D'INTERRUPTION MODIFIE •Un état d'interruption a été détecté ou supprimé. •La ligne peut avoir été coupée. •Le débit du périphérique déporté peut être trop faible.

Tableau 3-74 Codes d'erreur de variables déportées

Error_No	Description de l'erreur
80	ADRESSE DE LA STATION PAS VALABLE •L'adresse de la station spécifiée dans la chaîne d'adresse de variable déportée n'est pas valable.
81	NOMBRE DE PERIPHERIQUES PAS VALABLE •Le nombre de périphériques ou de registres spécifié n'est pas valable.
82	NOMBRE DE PERIPHERIQUES SPECIFIE TROP IMPORTANT •Le nombre de périphériques spécifié est supérieur à la limite maximale fixée à 32.
83	DISCORDANCE ENTRE LE NOMBRE DE PERIPHERIQUES ET LES DONNEES •Il y a un nombre insuffisant de données dans la valeur du paramètre de la chaîne déportée pour le nombre de périphériques ou de registres spécifié.
84	ERREUR DE TOTAL DE CONTROLE DE COMMUNICATIONS •Une erreur de total de contrôle a été détectée dans les données de réponse reçues provenant du périphérique esclave.
85	ERREUR DE DEPASSEMENT DU TEMPS IMPARTI DAUX COMMUNICATIONS •Une erreur de dépassement du temps imparti s'est produite après le nombre de nouvelles tentatives nécessaire, sans réponse du périphérique esclave dans la période impartie.
101	ERREUR DE COMMANDE •La commande correcte manque.
102	ERREUR DE STRUCTURE •Détection d'une discordance de la structure de transmission.
103	ERREUR DE TOTAL DE CONTROLE •La demande reçue par le périphérique esclave présentait une erreur de total de contrôle.
104†	ERREUR DE CODE DE FIN •Le code de fin de paquet ')' et le retour chariot n'ont pas été reçus par le périphérique esclave.
105†	LONGUEUR DE TEXTE EXCESSIVE •Le texte envoyé dépasse 255 octets.
106†	ERREUR DE PROTECTION •Commande d'écriture reçue par l'esclave alors que 'Protection en écriture' était toujours activée.

Tableau 3-74 Codes d'erreur de variables déportées (suite)

† Périphériques EX250 et EX500 uniquement.

Error_No	Description de l'erreur
107†	ERREUR DE CONVERSION •Les données provenant de l'EX250/500 ne peuvent pas être converties.
108†	TIME-OUT 1 •Un blanc d'une seconde ou plus s'est produit au cours de la réception des données sur le périphérique esclave.
109†	TIME-OUT 2 •La demande envoyée au périphérique esclave EX250/500 n'a pas été acceptée dans les 30 secondes.
110†	ERREUR DE PARITE •Erreur de parité détectée sur le périphérique esclave.
111†	ERREUR DE SURCHARGE •Erreur de surcharge détectée sur le périphérique esclave.
112†	ERREUR D'ENCADREMENT •Erreur d'encadrement détectée sur le périphérique esclave.
131†	ABSENCE D'INSTRUCTION FIN •Absence d'instruction FIN dans le programme esclave.
132†	INSTRUCTION DE PAIRE INTERDITE •Erreur dans le programme esclave lors du contrôle initial de MARCHE.
133†	ECHEC DU PROGRAMME •Programme détruit ou erreur de total de contrôle de programme dans l'esclave.
134†	MEMOIRE SATUREE •Impossibilité d'effectuer une commande d'écriture de programme, il n'y a plus de mémoire disponible.
135†	NUMERO DE PAGE/CIRCUIT INTERDIT •La page ou le circuit demandé(e) n'existe pas.
136	DISCORDANCE DE MODE •Réception d'une commande de mode pas valable.
137†	ERREUR PROM •Tentative d'écriture de programme après installation de PROM dans l'esclave.
138†	ERREUR D'OPERANDE •L'opérande du programme de l'esclave ne correspond pas à l'affectation du périphérique d'E/S disponible.

Tableau 3-74 Codes d'erreur de variable déportée (suite)

† Périphériques EX250 et EX500 uniquement.

Error_No	Description de l'erreur
139	NUMERO/TAILLE DE REGISTRE ERRONE(E) •Le registre spécifié n'est pas défini ou son numéro ne se situe pas dans la plage fixée.
140†	DISCORDANCE D'E/S •L'E/S programmée ne coïncidait pas avec l'E/S esclave disponible.
141†	ABSENCE DE SYNCRHO D'E/S •Aucune réponse n'a été reçue d'un module d'E/S spécifié.
142†	TRANSMISSION ERRONEE •Erreur de traitement du texte reçu ou commande non identifiée.
143	DISCORDANCE DE TYPE •Le type de mémoire et d'E/S stocké dans le périphérique de sauvegarde ne coïncide pas avec celui du périphérique esclave.
144†	PAGE PLEINE •La page du programme contient plus de 255 instructions.
146¥	PROTECTION DE LA MEMOIRE •La mémoire du périphérique esclave est protégée.
147¥	TRANSMISSION OCCUPEE •Commande en cours d'exécution depuis un programmateur universel local.
203¥	ERREUR DE BUS D'E/S •Une erreur de bus d'E/S a été détectée lorsque la commande MARCHE a été envoyée à l'esclave.
234¥	ERREUR : ABSENCE DE SYNCHRO E/S •Absence de réponse du périphérique d'E/S lorsque la commande MARCHE a été émise.
235¥	ERREUR DE DISCORDANCE D'E/S •Le module d'E/S installé ne correspond pas au tableau d'affectation E/S.

Tableau 3-74 Codes d'erreur de variables déportées (suite)

† Périphériques EX250 et EX500 uniquement.

¥ Périphérique EX2000 uniquement.

Error_No	Description de l'erreur
237¥	ERREUR DE CHEVAUCHEMENT D'E/S •La duplication d'un numéro de registre d'E/S affecté est détectée lors de l'émission de la commande MARCHE.
238¥	ERREUR DE NUMERO D'E/S •Le numéro de registre d'E/S affecté dépasse les limites fixées.
239¥	ERREUR D'ECRITURE PROM •Une erreur est détectée dans l'écriture PROM.
241¥	ERREUR : ABSENCE DE FIN •Aucune instruction FIN ne figure dans le programme principal, le sous-programme et le programme d'interruption.
242¥	ERREUR D'INSTRUCTION DE PAIRE •Une erreur est détectée dans une instruction de paire pour MCS/MCR, JCS,JCR lors de l'émission d'une commande MARCHE.
243¥	ERREUR D'OPERANDE •Une erreur est détectée dans une instruction d'opérande lors de l'émission d'une commande MARCHE.
244¥	PROGRAMME PAS VALABLE •Une erreur est détectée dans le tableau de gestion des programmes lors de l'émission d'une commande MARCHE.
245¥	ERREUR DE BRANCHEMENT •Une erreur est détectée dans l'utilisation d'une instruction BRANCHEMENT lors de l'émission d'une commande MARCHE.
246¥	ERREUR : ABSENCE D'ETIQUETTE •Aucune ETIQUETTE n'est enregistrée pour l'instruction BRANCHEMENT.
247¥	ERREUR : ABSENCE DE SOUS-PROGRAMME •Le sous-programme n'est pas enregistré.
248¥	ERREUR : ABSENCE DE RETOUR •L'instruction RETOUR dans un sous-programme n'est pas enregistrée.

Tableau 3-74 Codes d'erreur de variables déportées (suite)

¥ Périphérique EX2000 uniquement.

Error_No	Description de l'erreur
249¥	EMBOITEMENT DE SOUS-PROGRAMME •Une erreur d'emboîtement d'un sous-programme est détectée lors de l'émission d'une commande MARCHE.
250¥	ERREUR DE NUMERO DE PAS •Une erreur de numéro de pas du grafcet Toshiba est détectée lors de l'émission d'une commande MARCHE.
251¥	PAS DU CONNECTEUR •Une erreur de connexion séquentielle est détectée dans la page du grafcet Toshiba lors de l'émission d'une commande MARCHE.
252¥	ERREUR DE SOUS-PROGRAMME DE GRAFCET TOSHIBA •Aucune instruction APPEL n'est enregistrée dans le sous-programme de grafcet Toshiba.
253¥	INSTRUCTION UTILISATEUR INTERDITE •Une commande interdite est détectée dans un programme utilisateur.

Tableau 3-74 Codes d'erreur de variables déportées

¥ Périphérique EX2000 uniquement.

Codes d'erreur des communications standard

Les codes suivants s'appliquent aux autres modules de protocoles de communications. Des codes d'erreur supplémentaires sont utilisés pour tous les modules lorsque des codes d'erreur particuliers sont nécessaires.

Code	Etat d'erreur	Module esclave	Variable esclave	Module maître†	Variable déportée
0	OK	•	•	•	•
1	CHAINE D'ADRESSE TROP COURTE •Adresse du port trop courte •Adresse déportée trop courte •Adresse esclave trop courte	•	•	•	•
2	BITS DE DONNEES INTERDITS	•	•	•	•
3	PARITE INTERDITE	•	•	•	•
4	BITS D'ARRET INTERDITS	•	•	•	•
5	DEBIT RX PAS DISPONIBLE	•	•	•	•
6	DEBIT TX PAS DISPONIBLE	•	•	•	•
7	RTS PAS DISPONIBLE	•	•	•	•
8	CTS PAS DISPONIBLE	•	•	•	•
11	EMPLACEMENT INTERDIT •Le premier caractère du paramètre Adresse ne se situe pas dans la plage valable comprise entre '0' et '5'	•	•	•	•
12	PORT INTERDIT •Le deuxième caratère du paramètre Adresse ne se situe pas dans la plage valable pour ce module, par exemple 'A' à 'C' pour un port LCM ou 'A' à 'D' pour un port ICM.	•	•	•	•
14	RX TX INCOMPATIBLES	•	•	•	•
16	VARIABLES DEPORTEES PAS PRISES EN CHARGE •Le module de communications ne prend pas en charge les variables déportées.				•
17	PORT UTILISE	•	•	•	•
18	TROP GRAND NOMBRE DE VARIABLES ESCLAVES		•		
19	TROP GRAND NOMBRE DE TYPES DE MODULES		•		

Tableau 3-75 Codes d'erreur de communications standard

N.B. : †Ces erreurs s'appliquent aussi au bloc fonction du module de communications brutes.

Ces codes d'erreur peuvent être décalés vers un ensemble de valeurs supérieures pour certains modules pour qu'ils n'entrent pas en conflit avec les codes d'erreur standard associés au protocole considéré.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	STRING	'OA'	Oper	Config		
Baud	ENUM	_9600	Oper	Config	Valeurs énumérées	_300(0) _600(1) _200(2) _400(3) _4800(4) _9600(5)
Time_Out	TIME	1s	Oper	Super	Lim. haute Lim. basse	24 jours 1sec
Max_Retries	SINT	2	Oper	Super	Lim. haute Lim. basse	1000. 0
Enable_82X	BOOL	Nb	Config	Config	Détection	Non(0) Oui(1)
Reset_Stats	BOOL	Nb	Oper	Oper	Détection	Non(0) Oui(1)
Status	BOOL	NoGo	Oper	Bloc	Détection	NOGO(0) Go(1)
Error_Nb	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0
Queue_Space	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0
Comms_Error	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0

Tableau 3-76 Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Tot_Byte Tx	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_Byte_Rx	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_Pack_Tx	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_pack_Rx	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_Comms_Err	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_CE_Err	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_EE_Err	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999999999 0
Tot_Timeouts	DINT	0	Oper	Block	Lim. haute Lim. basse	999999999 0
Tot_Retries	DINT	0	Oper	Block	Lim. haute Lim. basse	999999999 0

Tableau 3-76 Attributs des paramètres (suite)

BLOC FONCTION EURO_PANEL

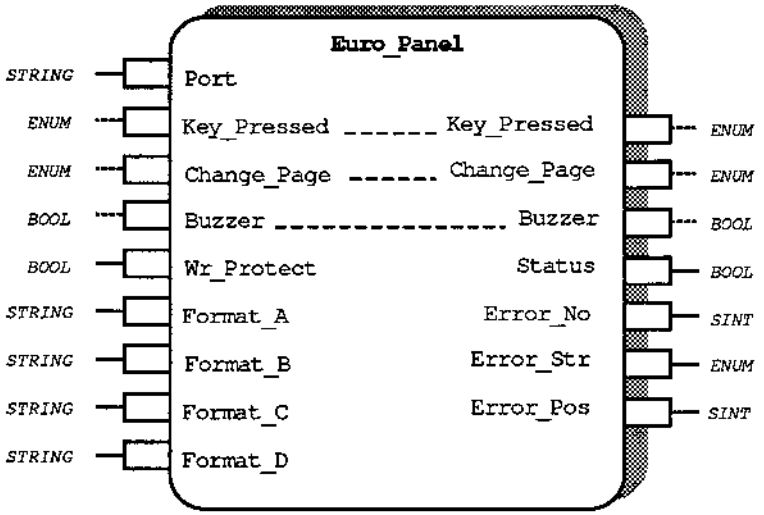


Figure: 3-46 Diagramme du bloc fonction Euro_Panel

Description fonctionnelle

Le bloc fonction est conçu pour simplifier la production d'affichages sur l'afficheur Eurotherm 2 lignes x 40 caractères. Ce bloc fonction est nécessaire pour programmer le PC3000 afin qu'il fonctionne avec l'afficheur Euro_Panel et il offre des fonctions permettant de contrôler l'affichage des messages et de prendre en charge l'interaction avec l'utilisateur et la saisie de données à partir de l'afficheur.

Avant de lire cette description, vous avez intérêt à acquérir des connaissances générales au sujet du système de communications PC3000 en lisant la Présentation des communications PC3000.

Nous avons utilisé la terminologie suivante pour cette description :

OIFL Langage structuré d'interface opérateur servant à définir la structure des messages affichés et des valeurs pour l'afficheur Eurotherm.

Page Terme utilisé dans ce document pour désigner une structure particulière pour l'afficheur complet 2 lignes x 40 caractères.

L'afficheur Eurotherm est un afficheur 2 lignes x 40 caractères, un pavé numérique, des touches fléchées haut/bas et de défilement qui peut communiquer avec le PC3000 par l'intermédiaire de n'importe quel port de communications RS422. Un seul afficheur peut être relié à un port RS422 unique.

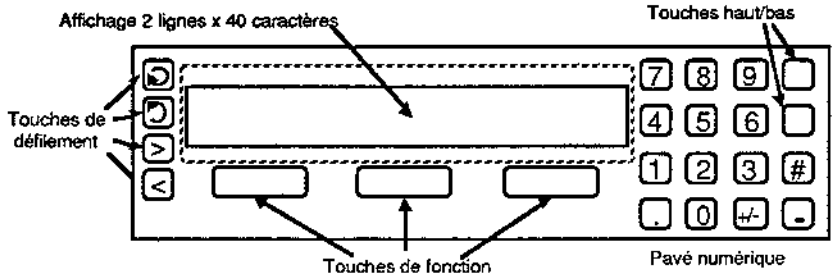


Figure: 3-47 Disposition de l'afficheur

Le bloc fonction de module de communications Euro_Panel offre une prise en charge des programmes PC3000 et gère toutes les fonctions élémentaires comme la saisie des données, le déplacement du curseur, les mises à jour d'écrans, etc. et configure la liaison série pour l'utilisation du débit, des bits d'arrêt, etc. qui conviennent. Les paramètres affichés sont les variables esclaves sur le PC3000, le bloc fonction Euro_Panel fonctionnant en tant que maître de la liaison série. Une variable esclave peut être associée à un ou plusieurs blocs fonctions de communications Euro_Panel à l'aide du sélecteur de protocole dans l'adresse de variable esclave.

Les messages de l'afficheur peuvent être mis en forme par configuration de quatre paramètres de chaîne dans le bloc fonction Euro_Panel. Les structures sont décrites à l'aide du langage structuré d'interface opérateur (OIFL). OIFL peut faire référence aux blocs fonctions de variables esclaves par un nom fourni comme partie de l'adresse de la variable esclave, par exemple OIFL peut servir à placer la valeur d'une variable esclave à un endroit donné dans la fenêtre de l'afficheur. Pour avoir des informations sur les blocs fonctions de variables esclaves, se reporter à la 'Présentation des communications PC3000'.

Programmation

Lors de la conception d'un système pour l'utilisation de l'afficheur Eurotherm, les paramètres internes de blocs fonctions qui doivent être affichés ou mis à jour par l'afficheur doivent être 'câblés' avec un ensemble de blocs fonctions de variables esclaves. Par exemple, pour afficher la sortie d'une boucle de contrôle PID, il faut créer un bloc fonction Slave_Real qui est 'câblé' au paramètre de sortie PID. Le bloc fonction Euro_Panel se rapporte à ces variables esclaves par l'intermédiaire des noms donnés aux adresses esclaves associées.

Sur la figure 3-13, trois variables esclaves sont associées à deux blocs fonctions de module de communications affectés aux ports '0C' et '1B'. Le bloc fonction affecté au port '0C' possède une chaîne de structure qui fait référence à la variable Slave_Real 'reél1', ce qui provoque l'affichage du message 'Sortie = 100,0' sur l'afficheur, 100,0 étant la valeur actuelle de la variable Slave_Real. Si Slave_Real est câblé à la sortie d'un bloc fonction PID, la valeur affichée sera automatiquement mise à jour par le bloc fonction Euro_Panel à chaque exécution.

Une deuxième variable esclave entière identifiée dans le bloc Euro_Panel par l'adresse 'EPval_I' fournit la valeur pour un numéro de batch affiché sur le panneau relié au port B de l'ICM.

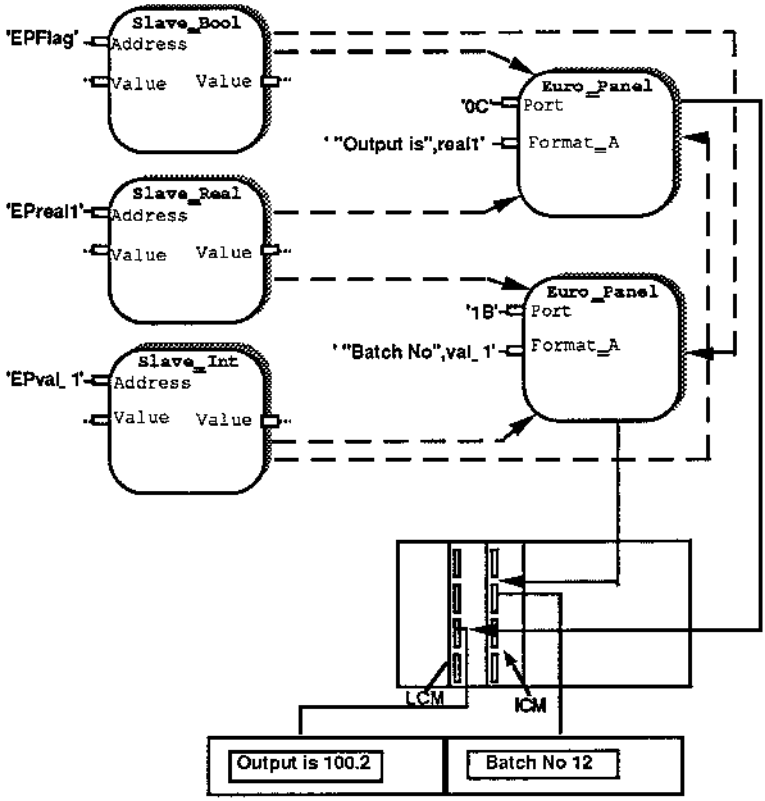


Figure 3-48 Exemple d'utilisation d'Euro_Panel

Variables esclaves

Les deux premiers caractères de l'adresse de variable esclave définissent le protocole auquel est associée la variable esclave. Dans ce cas, les deux premiers caractères doivent être 'EP' pour sélectionner Euro_Panel ; le reste de l'adresse doit servir à affecter un nom à la variable esclave qui peut être référencée dans le langage OIFL. Cette partie est appelée nom du paramètre OIFL et peut être constituée par n'importe quel ensemble de caractères alphanumériques, soulignement '_' inclus, jusqu'à une longueur de 12 caractères.

```
Exemples d'adresses de variables esclaves :
        'EPpv'
        'EPset_point'
        'EPs34_6_tt'
```

Les variables esclaves peuvent aussi avoir un paramètre de protection en écriture qui peut servir à activer et à désactiver l'autorisation d'écriture sur une variable donnée comme sur l'état d'un autre paramètre dans le programme utilisateur, ce qui peut être par exemple utilisé pour inhiber les changements sur certains paramètres au cours de certaines phases d'un process.

Bloc fonction Euro_Panel

Chaque bloc fonction Euro_Panel est associé à un port de communications unique et donc à un afficheur Eurotherm unique. Il effectue la conversion dans différents formats d'affichage, selon un jeu de chaînes de formats. Il traite le protocole nécessaire pour piloter l'afficheur et assure la prise en charge des touches pour la saisie des données. Il n'offre aucune fonction permettant de modifier le débit ou les caractéristiques de liaison série identiques car ces caractéristiques ne peuvent pas être modifiées dans l'afficheur.

Utilisation avec les grafcets

Dans une application simple qui nécessite l'affichage d'un jeu fixe de valeurs sur l'afficheur, il suffit de configurer une fois les chaînes de structure Euro_Panel, par exemple sous la forme de valeurs de démarrage à froid. Toutefois, avec les applications complexes, il peut être nécessaire de modifier les valeurs affichées à l'aide des listes et menus de défilement. On peut obtenir ce résultat en modifiant les chaînes de structure dans un grafcet (SFC). Quatre chaînes de structure sont fournies, de telle sorte que certaines parties d'une zone d'affichage peuvent être modifiées séparément. Par exemple, il est possible que la ligne inférieure fasse partie de la liste d'interrogation alors que la ligne supérieure affiche toujours les mêmes informations. Dans ce cas, le grafcet doit uniquement modifier la chaîne de structure pour la ligne inférieure.

N.B. : les quatre chaînes de structure sont toujours traitées par l'Euro_Panel comme si elle formaient une chaîne unique. Un programme utilisateur peut modifier n'importe laquelle de ces chaînes à tout moment.

Se reporter à la partie relative aux 'Exemples de programmes utilisateur' pour avoir des informations supplémentaires.

Attributs du bloc fonction

Type: 8 90

Classe : ..... COMMS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Port Status Error_No

Besoins de capacité mémoire : 2356 octets

Durée d'exécution : ..... 1150 µ Secs

Description des paramètres

Port

Il s'agit d'une chaîne de deux caractères définissant le port auquel est attaché le bloc fonction de l'afficheur. Le premier caractère est un chiffre compris entre 0 et 5, représentant l'emplacement dans le rack et le deuxième caractère est une lettre représentant le port dans cet emplacement. Par exemple, '0C' serait le port C sur le LCM et, s'il y avait un ICM dans l'emplacement 3, '3A' désignerait son port supérieur. La modification de ce paramètre au cours de l'exécution du programme utilisateur n'a aucun effet sur le module, sauf dans des circonstances particulières. Cf. la 'Présentation des communications PC3000', section ' Modification temporaire des paramètres de configuration'.

Key_Pressed

Il s'agit d'une entrée/sortie énumérée qui donne la dernière touche qui a été enfoncée sur l'afficheur, par exemple 1 (* 1 *), 19 (* Entrée *), 21 (* Haut *). Le tableau ci-dessous montre une liste complète des valeurs de ce paramètre.

Nom de la touche	Valeur
Zéro	0
Un	1
Deux	2
Trois	3
Quatre	4
Cinq	5
Six	6
Sept	7
Huit	8
Neuf	9
Sens horaire	10
Sens anti-horaire	11
Droite	12
Gauche	13
F1	14
F2	15
F3	16
Point	17
Changement de signe	18
Entrée	19
Bas	20
Haut	21
Aucune touche	22

Tableau 3-77 Codage des touches Euro_Panel

Afin d'être sûr qu'un nouvel appui sur une touche sera détecté, il faut positionner ce paramètre sur 22 (*AUCUNE TOUCHE*) avant de tester une touche particulière.

Change_Page

Il s'agit d'une entrée/sortie qui sert à déclencher le bloc fonction Euro_Panel pour relire toutes les chaînes de structure A,B,C et D, c'est-à-dire pour analyser les instructions de structure OIFL afion de créer un nouvel affichage sur l'afficheur. L'affichage reste vide tant que le bloc fonction analyse toutes les chaînes de structure. S'il y a une erreur dans une des chaînes de structure, **Change_Page** passe sur Erreur et l'affichage restera vide ; s'il n'y a aucune

erreur, **Change_Page** passera sur 0(OK) et le nouvel affichage sera visible. Ce paramètre peut prendre les valeurs suivantes :

Ok	0	Etat normal
En attente	1	Nouvelle page en cours de traitement
Erreur	2	Erreur détectée au cours du changement de page
Nxt_Pge	3	Demande de la page suivante

Tableau 3-78 Etats de changement de page Euro_Panel

Seule la valeur **Nxt_Pge** doit être configurée par le programme utilisateur pour demander l'analyse des chaînes de structure pour créer une nouvelle page, les autres valeurs sont configurées lors de l'exécution du bloc fonction Euro_Panel.

Buzzer

Buzzer est une entrée/sortie configurée par le programme utilisateur et remise à zéro par le bloc fonction ; par conséquent, le fait de positionner buzzer sur 1(BEEP) provoquera toujours l'émission d'un signal sonore par l'afficheur.

Wr_Protect (protection en écriture)

Il s'agit d'une entrée qui désactive toute écriture dans les paramètres figurant sur l'affichage. Elle offre une fonction de protection en écriture globale sur un panneau et peut servir à inhiber l'accès en écriture pour des raisons de sécurité.

Format _A-D

Il y a quatre chaînes OIFL dont chacun peut avoir une longueur maximale de 255 caractères. Le bloc fonction lit toujours les quatre chaînes commençant par A, toutes les fois qu'il est déclenché par le paramètre **Change_Page**. Chaque paramètre doit contenir des définitions de champ OIFL complètes ou des chaînes vides.

> **N.B. :** le fait de modifier ces paramètres sans modifier ensuite le paramètre **Change_Page** n'aura aucun effet sur le panel visible.

Etat

Etat est une sortie qui peut prendre les valeurs Go ou NOGO, indiquant l'état du bloc fonction.

Error_No

Les codes d'erreur sont scindés en deux groupes : erreurs de port (cf. tableau 3-51) et erreurs d'analyse (cf. tableau 3-50). Les erreurs de port se produisent lorsque le programme utilisateur est remis à zéro et lorsque le module se réinitialise, les erreurs d'analyse se produisent lorsqu'une nouvelle page est analysée, c'est-à-dire que le paramètre **Change_Page** est positionné sur 1.

Error_Str

Il s'agit d'un type énuméré indiquant la chaîne de structure qui contenait la première erreur détectée par l'analyseur. Par exemple, la valeur 2 (* Fmt_B *) indique que l'erreur a été détectée dans la chaîne de structure B.

Error_Pos

Cet entier donne une indication approximative de la position de l'erreur dans la chaîne de structure.

Saisie des données

Protection en écriture

Il existe trois formes de protection en écriture dans le bloc fonction Euro_Panel :

Protection en écriture globale : il s'agit d'une entrée dans le bloc fonction. Lorsque le bloc est en mode protection en écriture, aucune écriture ne peut être effectuée dans les paramètres affichés sur la page.

Autorisation d'écriture OIFL : les différents paramètres sont déclarés dans la chaîne de structure comme pouvant ou non être écrits.

Autorisation d'écriture de variable esclave : l'autorisation en écriture dans la variable esclave peut servir à activer et à désactiver les écritures, en fonction des autres variables.

Modes

Trois modes sont associés à la saisie des données (cf. figure 3-15).

Mode Affichage

Dans ce mode, tous les paramètres sont des réels et sont affichés en fonction de leur structure dans OIFL.

Mode saisie de données

Lors de la première entrée en mode saisie de données, un curseur apparaît sous le premier paramètre de la page qui peut recevoir des écritures. S'il n'y a aucun paramètre qui peut recevoir des écritures, il est impossible d'entrer en mode saisie de données et l'appui sur '#' provoque l'émission d'un signal sonore par l'afficheur. En mode saisie de données, les touches gauche et droite peuvent servir à déplacer le curseur sur le

prochain paramètre qui peut recevoir des écritures. La valeur au-dessus du curseur clignote, ce qui indique qu'il s'agit de la valeur réelle de la variable esclave.

Mode saisie de valeurs

L'appui sur Entrée ou le commencement de la frappe d'une valeur fait entrer l'afficheur en mode saisie de valeur. La valeur au-dessus du curseur N'EST PAS la valeur réelle de la variable esclave. Pour indiquer cela, l'afficheur arrête de clignoter. Une fois que la valeur a été saisie, l'appui sur Entrée confirme la valeur et amène en mode saisie de données, l'appui sur '#' met fin à la saisie de valeurs et amène en mode saisie de données avec la valeur du paramètre d'origine inchangée.

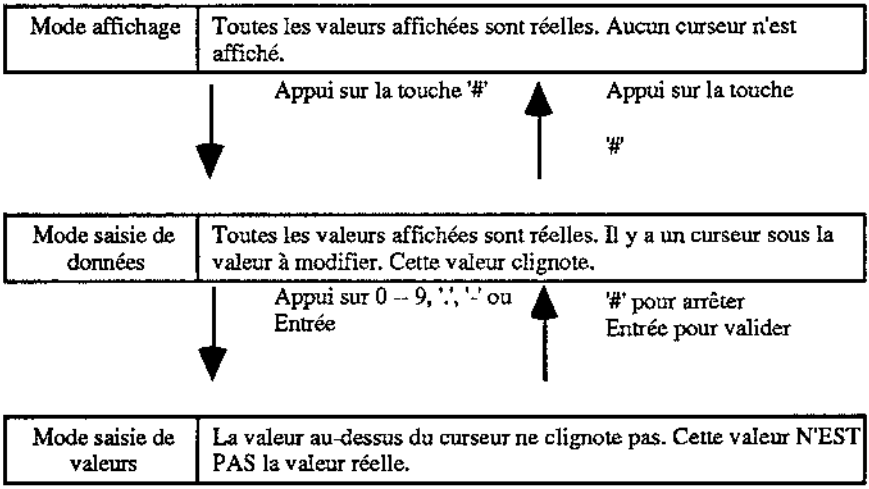


Figure 3-49 Modes de saisie de données sur le bloc fonction Euro_Panel

Echo des touches dans les différents modes

En mode affichage, la seule touche qui ne fait pas l'objet d'un écho pour le paramètre **Key_Pressed** est la touche '#'. En modes saisie de données ou saisie de valeurs, aucune touche ne fait l'objet d'un écho pour le paramètre **Key_Pressed.** L'appui sur une touche non valable (par exemple une touche de fonction) provoque l'émission d'un signal sonore par l'afficheur.

Langage structuré d'interface opérateur (OIFL).

Présentation

OIFL offre une méthode facile permettant de produire une page unique sur un affichage. Cette page est composée de champs séparés par des virgules, chaque champ définissant une des quatre possibilités suivantes :

- •des chaînes constantes
- •des variables esclaves avec des informations de structuration
- •des commandes de positionnement
- •des macros

L'accès aux variables esclaves s'effectue par l'intermédiaire de l'adresse de variable esclave. Les définitions de champs OIFL sont contenues dans des chaînes affectées aux paramètres de structure **Format_A, Format_B, Format_C, Format_D.** L'ajout d'une virgule ',' est facultatif à la fin de la dernière définition de champ OIFL pour une chaîne de structure lorsque les définitions OIFL continuent sur la chaîne de structure suivante. Tous les paramètres de structure doivent contenir des définitions OIFL valables ou des chaînes nulles, c'est-à-dire ' ---two blits not'.

Exemples

Cette section est composée d'exemples d'OIFL et des affichages obtenus. Les affichages sont ceux qui apparaissent sur l'afficheur Eurotherm 2 lignes x 40 colonnes. Les caractères gras servent à indiquer les caractères qui clignotent sur l'affichage. Exemple :

'Des caractères sur un affichage.'
,@0:1,'1234567890123456789012345678901234567890'

Des caractères sur un affichage. 1234567890123456789012345678901234567890

Attributs de champs

Tout champ imprimé à partir d'un panel d'opérateur possède un certain nombre d'attributs qui lui sont associés. Ils servent à prendre en charge les champs clignotants et soulignés, les différentes justifications, les largeurs de champs, etc. Les attributs de champs sont normalement situés après deux points ou une accolade droite.

Largeur de champ

La largeur de champ est un nombre indiquant la largeur maximale qu'un paramètre peut occuper. La justification est effectuée dans le champ défini de cette manière. Si un nombre est trop grand pour être affiché dans la largeur de champ qui lui est attribuée, le champ tout entier est rempli de points

d'interrogation. Si une chaîne énumérée est trop grande pour sa largeur de champ, elle est tronquée de manière à tenir dans le champ et il en est de même pour les variables esclaves de chaînes.

Justification

Pour utiliser la justification gauche, droite ou centrée, il faut utiliser respectivement les caractères L, R et C. La justification s'effectue dans la largeur de champ du paramètre.

Clignotement et soulignement

F et U s'appliquent respectivement aux champs clignotement et soulignement.

Autorisation d'écriture

Pour saisir un paramètre par l'intermédiaire de l'interface opérateur, il faut que le paramètre ait été autorisé en écriture, ce qui permet d'afficher le même paramètre sur différents panels, l'autorisation d'écriture s'appliquant à un seul d'entre eux. L'autorisation d'écriture est indiquée par un W dans OIFL.

Ecriture seule

Sert à utiliser des mots de passe lorsqu'un mot de passe est saisi mais pas affiché. Un '*' indique un paramètre en écriture seule. L'autorisation d'écriture est implicite.

Exemples

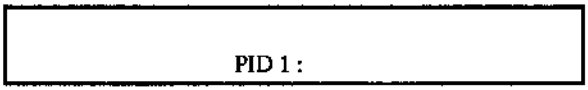
'PID 1 :':LUF

'PID 1 :':10R

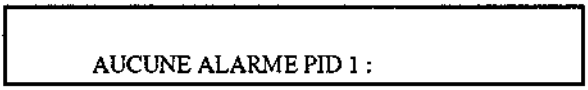
Positionnement

Le signe @ sert à indiquer l'emplacement à atteindre ; avec un paramètre, le champ suivant commence sur l'emplacement de ce caractère sur la ligne en cours. Avec deux paramètres séparés par un ':', le deuxième paramètre définit la ligne. Exemples :

@10:1,'PID 1 :'



@10:1,'PID 1 :',@0,'AUCUNE ALARME'



Chaînes constantes

Les termes constants de l'affichage sont délimités par des guillemets. Pour utiliser le caractère guillemets proprement dit, taper ".

La syntaxe est la suivante :

'display_string'[:[field_width][LRCUF]]

Exemple :

'abc"def'

Euro_Panel prend en charge un jeu de caractères qui inclut les symboles des signaux sonores d'alarme : oméga, pi, flèches, etc. Pour faire afficher ces symboles sur le panel, il faut faire précéder le code de caractère hexadécimal par le caractère spécial $. Un tableau de codes de caractères est fourni à la fin de cette section.

A titre d'exemple, supposons que l'on place du texte au-dessus de la première touche de fonction.

La chaîne de structure

```
'@S:1, 'ACK $CA'; F
```

donnerait un ACK clignotant avec une flèche vers le bas ( ) au-dessus de la touche de fonction gauche.

Réels

Trois structures de base sont prises en charge pour les variables réelles : un nombre fixe de décimales (valeur par défaut), une forme standard ('S') et les unités techniques ('E') (forme standard mais uniquement avec des multiples de trois pour l'exposant). Le nombre de décimales pour chaque structure est défini à l'aide de '.N' placé juste après la largeur de champ.

Si la valeur à afficher ne tient pas dans la largeur de champ, le champ est rempli de points d'interrogation. Si des décimales nulles sont spécifiées, aucune virgule décimale n'est affichée.

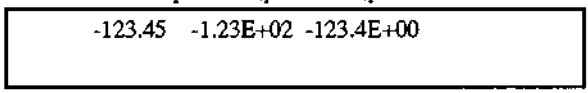
Il est possible d'ajouter des limites aux variables inscriptibles à l'aide du signe '<=', comme le montre l'exemple ci-dessous.

La syntaxe est la suivante :

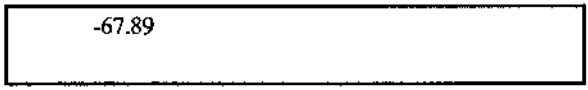
[Min<=]Param_Name[<=Max]:field_width.decimal_places{ESLRCUFW*]

Exemples :-

pv1:11.2,pv1:11.2S,pv1:11.2E



-100.0<=pv2<=100.0:10.2LW



Valeurs entières et booléennes

Les valeurs entières et booléennes peuvent être représentées de deux manières : sous la forme d'un nombre ou sous la forme d'un type énuméré. Lorsqu'on affiche une valeur entière sous la forme d'un nombre, il faut spécifier une largeur de champ ; pour l'affichage énuméré, la largeur de champ par défaut est la longueur du nom de type le plus grand et, pour une valeur booléenne affichée sous la forme d'un nombre, la largeur de champ par défaut est 1.

La syntaxe pour l'affichage sous la forme d'un nombre est la suivante :

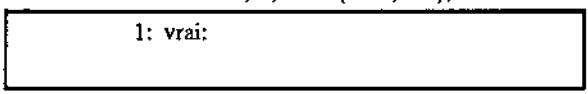
[Min<=]Param_Name[<=Max]:field_width[LRCUFW*]

La syntaxe pour un type énuméré est la suivante :

[Min<=]Param_Name[<=Max] {name0,name1,...}[[field_width][LRCUFW]]

Exemples :

bool1:5,' :',bool1 {faux,vrai},':'



int1 {zéro,un,deux,trois},':',int1{OK,Erreur,En attente,Ecriture}

Variables à chaînes

La syntaxe est la suivante :

Param_Name:field_width[LRCUF]

Exemples :-

@27:1,String2:11F

L'exemple suppose que 'String2' est l'adresse du paramètre OIFL d'une variable esclave qui contient la chaîne '***ALARM***'. Elle est affichée sur la deuxième ligne de l'afficheur, commençant à l'emplacement 27. Le 'F' indique que le texte va clignoter.

Variables temporelles

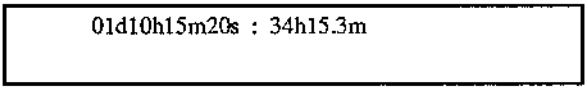
La chaîne de structure pour une variable temporelle définit les champs de temps affichés. Il s'agit de D(jours), H(heures), M(minutes), S(secondes) et m(millisecondes). Seuls les premier et dernier champs à afficher sont définis dans la chaîne. En outre, il est possible de définir une partie fractionnelle en spécifiant un certain nombre de décimales.

La syntaxe est la suivante :

[Min<=]Param_Name[<=Max]:[DHMSm][.dp][:field_width[LRCUFW]]

Exemples :-

Time1:DS,' : ',Time1:HM.1



Variables heure du jour

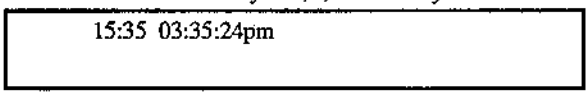
Pour une heure du jour, la résolution peut être indiquée en minutes (M) ou en secondes (S). Par défaut, la sortie sera en mode 24 heures mais l'ajout d'un A au spécificateur de présentation fait passer la sortie en mode 12 heures (am/pm).

La syntaxe est la suivante :

[Min<=]Param_Name[<=Max]:[MS][A][:field_width[LRCUFW]]

Exemples :-

TimeOfDay1:M,' ',TimeOfDay1:SA



Variables date

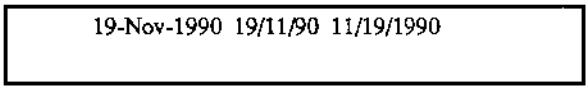
Trois présentations sont définies pour la date : standard, européenne et américaine. Par défaut, l'année est affichée avec deux caractères mais peut inclure le siècle par ajout de X.

La syntaxe est la suivante :

[Min<=]Param_Name[<=Max]:[SEA][X][:field_width[LRCUFW]]

Exemples :-

Date1:SX,' ',Date1:E,' ',Date1:AX



Prise en charge de macros

Les macros sont prévues pour réduire le texte OIFL nécessaire lorsque des parties de texte de présentation identique sont associées à un certain nombre de variables esclaves. Les définitions des macros sont normalement indiquées à côté du début de l'ensemble de chaînes de structure et peuvent être ensuite appelées à n'importe quel endroit du texte OIFL qui suit.

Deux formes de définitions de macros sont prises en charge. L'une définit une macro avec des paramètres remplaçables mais ne produit aucune sortie immédiate sur l'afficheur. Dans le texte OIFL qui suit, la sortie est produite lorsque la macro est appelée avec des noms de paramètres particuliers.

L'autre type définit la macro et produit une sortie en même temps car les noms de paramètres sont fournis avec la définition de la macro.

Avec les deux types, il est uniquement possible de remplacer les noms de paramètres OIFL par des variables esclaves dans la macro. Il n'est pas possible de remplacer des données de structure. L'appel de macro doit aussi contenir le bon nombre de paramètres pour correspondre à la définition. Les paramètres sont remplacés dans le même ordre que celui de la définition.

Définition de macro ne produisant aucune sortie d'affichage

Les paramètres remplaçables sont définis à l'aide du symbole'#'. La syntaxe est la suivante :

MacroNumber(#:format1,#:format2 ...)

Par exemple :

```
Macro definition :
    1('PV :',#:10.2,'SP :',#:10.2)
Macro call :
    1(pv1,sp1)
```

Macro produisant une sortie d'affichage

Dans cette forme de définition, un nom de paramètre donné est utilisé dans la définition pour que la macro soit utilisable au point où elle est définie. Le nom de paramètre fourni dans la définition est remplacé par des noms donnés lors de l'appel de la macro.

La syntaxe est la suivante :

MacroNumber(ParamName1:fmt1,ParamName2:fmt2 ...)

Par exemple:

```
Macro definition :
    2(AlarmStat1:{OK,Alarm,Shutdown}:10L)
Macro call :
    2(AlarmStat2)
```

Esclaves multi-éléments

Le PC3000 prend en charge un certain nombre de variables esclaves qui contiennent des ensembles de valeurs comme Slave_Real_8 Slave_Bool_8 etc. L'accès à ces variables s'effectue de la manière suivante :

ParamName[Element]:Format

Par exemple, avec un ensemble de variables esclaves à virgule flottante (REEL) portant le nom 'réel' dans l'adresse esclave, le troisième paramètre doit être affiché de la manière suivante :

real[3]:10.2SW

Exemples

Les exemples ci-après montrent la manière dont on peut structurer un programme utilisateur pour utiliser efficacement le bloc fonction du panel.

Le grafcet constitue une représentation naturelle de la hiérarchie des pages de panel. Un seul pas peut servir à affecter les chaînes de structure et la demande de traitement de la nouvelle page par le bloc fonction. Le pas peut aussi faire passer **panel. Key_Pressed** sur 22 (*NO_KEY*) pour être prêt pour le prochain appui sur une touche. Les transitions qui suivent le pas sont toutes **panel. Key_Pressed** = <valeur>, où valeur représente le code attendu pour l'appui sur une touche. Dans l'exemple de la figure 15, les transitions en-dessous du pas MENU seraient **panel. Key_Pressed** =14(F1), 15(F2) ou 16(F3). Les transitions en-dessous de chacune des macros sont toujours vraies, par conséquent, lors de la sortie d'un niveau, le panel revient au menu.

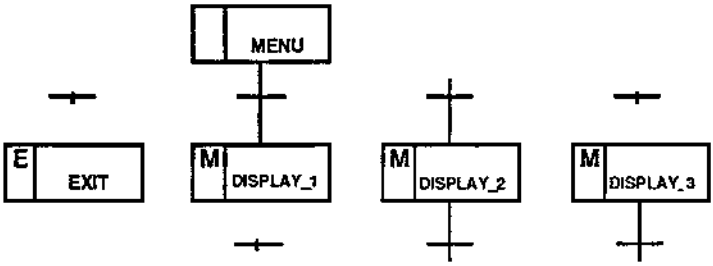


Figure 3-50 Menu à trois niveaux inférieurs

N.B. : Le pas Fin n'est pas obligatoire. Il est inclus ici pour faciliter la compréhension.

Pages défilantes

La figure 3-51 donne un exemple de jeu de page défilantes avec bouclage. Les transitions en-dessous de chaque page définissent respectivement la touche sortie, la touche flèche vers le bas et la touche flèche vers le haut.

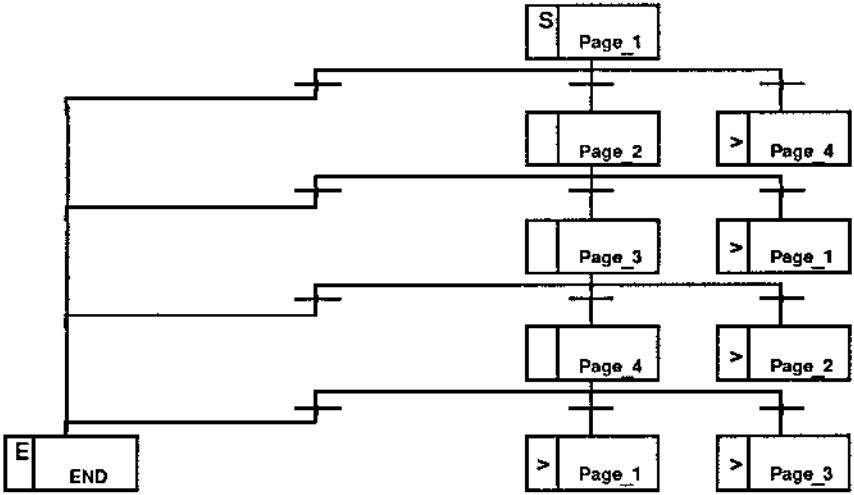


Figure 3-51 Pages défilantes sur un seul niveau

Exemple de ST contenu dans ces pas :

```
(* SINGLE SHOT *)
INITIAL_STEP  PAGE_1 :
                              (* Reset the key pressed value to detect a
          change *)
                              (* in the following transition *)
                              panel.Key_Pressed       := 22(*No_Key*) ;
                              (* Set up the new format for the page *)
                              panel.Format_A          :=
                                 '' Loop 1 Overview : Time Left
          ',time1:MS' ;       panel.Format_B :='@0:1,' PV:',pv1:7.2,'
          SP:',sp1:7.2,
                                 'OP:',op1:7.2' ;
                              (* Update the panel output *)
                              panel_Change_Page       := 3(*Nxt_Pge*) ;
END_STEP
TRANSITION
       FROM PAGE_1
       TO   END
:=     panel.Key_Pressed = 14(*F1*) OR
       panel.Key_Pressed = 15(*F2*) OR
            panel.Key_Pressed = 16(*F3*) ;
                                          (* Any function key *)
END_TRANSITION
TRANSITION
       FROM PAGE_1
       TO   PAGE_2
:=     panel.Key_Pressed = 10(*Clkwise*);
END_TRANSITION
TRANSITION
       FROM PAGE_1
       TO   PAGE_4
:=     panel.Key_Pressed = 11(*Aclkwise*);
END_TRANSITION
```

Gros systèmes

Lorsqu'il y a beaucoup de paramètres à afficher sur le panel, il est possible de définir un petit ensemble de variables esclaves générales qui peuvent être réutilisées à chaque changement de l'affichage. Au lieu de relier la variable esclave à un paramètre, un pas continu est utilisé de la manière suivante :

```
(* SINGLE SHOT *)
INITIAL_STEP  L1_DISP :
       panel_Key_Pressed       := 22(*No_Key*) ;
       panel_Format_A          :=
```

```
        '' Loop 1 Overview : Time Left ',time1:MS' ;   panel_Format_B
:=
        '@0:1,' PV:',real1:7.2,'  SP:',real2:7.2,'
OP:',real3:7.2' ;
      (* Update the panel output *)
      panel_Change_Page       := 3(*Nxt_Pge*) ;
END_STEP
TRANSITION
      FROM L1_DISP
      TO   L1_WIRE
:= 1; (* NULL transition - default TRUE *) END_TRANSITION
(* CONTINUOUS *)
STEP  L1_WIRE :
      (* Connect the variables esclaves to the correct
      paramètres *)
      real1_Value          := pid1.Process_Val ;
      real2_Value          := pid1.Setpoint ;
      real3_Value          := pid1.Ch1_Output -
      pid1.Ch2_Output ;
END_STEP
TRANSITION
      FROM L1_WIRE
      TO   LP_END
:=  panel.Key_Pressed = 14(*F1*) OR
            panel.Key_Pressed = 15(*F2*) OR
               panel.Key_Pressed = 16(*F3*) ;
          (* Any function key pressed *)
END_TRANSITION
```

Signalisation des erreurs

Les codes d'erreur relatifs à la fois au module Euro_Panel et à la variable esclave se trouvent dans l'annexe C, à la fin de ce chapitre. D'autres codes d'erreur, appelés erreurs d'analyseur, peuvent être produits par l'analyseur 01FL, lors de la demande d'une nouvelle page. Ces codes sont énumérés ci-après.

Erreurs de l'analyseur	
Numéro de l'erreur	**Nom de l'erreur**
100	Aucune virgule après l'utilisation ou la définition d'une macro
101	Absence de deux points ou d'une virgule
102	Cet emplacement de coordonnées n'existe pas
103	Pas inférieur au minimum suivant
104	Aucun nom de paramètre spécifié
105	Aucune accolade de fermeture ne suit la liste d'énumération
106	Codes point, deux points ou virgule manquants
107	Largeur de champ nulle interdite
108	Largeur de champ obligatoire manquante
109	Virgule manquante
110	Aucune parenthèse de fermeture après l'utilisation d'une macro
111	Aucune virgule après la définition d'une macro
112	Aucune virgule après l'utilisation d'une macro
113	Chaîne terminée sans parenthèse de fermeture à la fin de la définition de la macro
114	Numéro prévu
115	Chaîne terminée avant la parenthèse de fermeture
116	Nom de paramètre manquant
117	Paramètre pas trouvé
118	Minimum réel pas autorisé avec le paramètre dint
119	Maximum inférieur au minimum
120	Minimum interdit pour ce type de paramètre

Tableau 3-79 Erreurs de l'analyseur Euro_Panel

Erreurs de k'analyseur	
Numéro de l'erreur	**Nom de l'erreur**
121	Trop grand nombre de noms de détection pour une valeur booléenne
122	Valeur pas alphanumérique dans la liste d'énumération
123	Chaîne terminée avant la fin de la liste d'énumération
124	Champ de la liste d'énumération vide
125	Trop grand nombre de décimales demandé
126	Décimales interdites avec ce type
127	Plusieurs justifications spécifiées
128	Les chaînes constantes ne sont pas inscriptibles
129	Notation scientifique spécifiée pour un paramètre non réél
130	Impossibilité d'avoir à la fois la notation scientifique et la notation technique
131	Notation technique spécifiée pour un paramètre non réel
132	Plus d'espace pour les attributs
133	Largeur de champ de paramètres de chaîne manquante
134	Largeur de champ de paramètre réel manquante
135	Largeur de champ de paramètre dint manquante
136	Chaîne terminée avant la fin de l'utilisation de la macro
137	Argument de macro pas terminé par une virgule ou une parenthèse de fermeture
138	Pas de place pour la chaîne const sur l'affichage
139	Signe égal manquant après un signe inférieur à

Tableau 3-79 Erreurs de l'analyseur Euro_Panel (suite)

Codes standard d'erreurs de communications

Les codes suivants s'appliquent à l'ensemble des modules. Des codes d'erreur supplémentaires seront utilisés pour tous les modules lorsque des codes d'erreur propres au protocole seront nécessaires.

Code	Etat d'erreur	Module esclave	Variable esclave	Module maître†	Variable déportée
0	OK	*	*	*	*
1	CHAINE D'ADRESSE TROP COURTE •Adresse du port trop courte •Adresse déportée trop courte •Adresse esclave trop courte	*	*	*	*
2	BITS DE DONNEES INTERDITS	*	*	*	*
3	PARITE INTERDITE	*	*	*	*
4	BITS D'ARRET INTERDITS	*	*	*	*
5	DEBIT RX PAS DISPONIBLE	*	*	*	*
6	DEBIT TX PAS DISPONIBLE	*	*	*	*
7	RTS PAS DISPONIBLE	*	*	*	*
8	CTS PAS DISPONIBLE	*	*	*	*
11	EMPLACEMENT INTERDIT •Le premier caractère du paramètre Adresse ne se situe pas dans la plage valable comprise entre '0' et '5'	*	*	*	*
12	PORT INTERDIT •Le deuxième caractère du paramètre Adresse ne se situe pas dans la plage valable pour le module , par exemple 'A' à 'C' pour un port LCM ou 'A' à 'D' pour un port ICM.	*	*	*	*

Tableau 3-80 Codes standard d'erreurs de communications

Code	Etat d'erreur	Module esclave	Variable esclave	Module maître†	Variable déportée
14	RX TX INCOMPATIBLES	•	•	•	•
16	VARIABLES DEPORTEES PAS PRISES EN CHARGE •Le module de communications ne prend pas en charge les variables déportées				•
17	PORT UTILISE	•	•	•	•
18	TROP GRAND NOMBRE DE VARIABLES ESCLAVES		•		
19	TROP GRAND NOMBRE DE TYPES DE MODULES		•		

Tableau 3-80 Codes standard d'erreurs de communications (suite)

N.B. : †Ces erreurs s'appliquent aussi au bloc fonction de module de communications brutes.

Ces codes d'erreur peuvent être décalés vers un ensemble de valeurs supérieures pour certains modules afin de ne pas entrer en conflit avec des codes d'erreur standard associés au protocole considéré.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Port	STRING	'0A'	Oper	Config		Oui(1)
Key_Pressed	ENUM	No_Key	Oper	Oper	Valeurs énumérées	Zéro(0) Un(1) Deux(2) Trois(3) Quatre(4) Cinq(5) Six(6) Sept(7) Huit(8) Neuf(9) Sens horaire(10) Sens anti-horaire(11) Droite(12) Gauche(13) F1(14) F2(15) F3(16) Chg(17) Sgn(18) Entrée(19) Bas(20) Haut(21) Aucune touche(22)
Change_Page	ENUM	Ok	Oper	Oper	Valeurs énumérées	Ok(0) En attente(1) Erreur(2) Page suivante(3)
Buzzer	BOOL	Off	Oper	Config	Détection	Arrêt(0) Signal sonore(1)
Wr_Protect	BOOL	No	Oper	Super	Détection	Non(0) Oui(1)
Format_A	STRING	'	Oper	Config		
Format_B	STRING	'	Oper	Config		
Format_C	STRING	'	Oper	Config		
Format_D	STRING	'	Oper	Config		
Status	BOOL	NoGo	Oper	Bloc	Détection	NoGo(0) Go(1)
Error_No	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0
Error_str	ENUM	No_Err	Oper	Bloc	Valeurs énumérées	No_Err(0) Fmt_A(1) Fmt_B(2) Fmt_C(3) Fmt_D(4)
Error_Pos	SINT	0	Oper	Bloc	Lim. haute Lim. basse	255 0

Tableau 3-81 Attributs des paramètres Euro_Panel

Déc	Hex	Caractère	Déc	Hex	Caractère
32	20	Espace	66	42	B
33	21	!	67	43	C
34	22	"	68	44	D
35	23	#	69	45	E
36	24	$	70	46	F
37	25	%	71	47	G
38	26	&	72	48	H
39	27	'	73	49	I
40	28	(	74	4A	J
41	29	)	75	4B	K
42	2A	*	76	4C	L
43	2B	+	77	4D	M
44	2C	,	78	4E	N
45	2D	-	79	4F	O
46	2E	.	80	50	P
47	2F	/	81	51	Q
48	30	0	82	52	R
49	31	1	83	53	S
50	32	2	84	54	T
51	33	3	85	55	U
52	34	4	86	56	V
53	35	5	87	57	W
54	36	6	88	58	X
55	37	7	89	59	Y
56	38	8	90	5A	Z
57	39	9	91	5B	[
58	3A	:	92	5C	\
59	3B	;	93	5D	]
60	3C	<	94	5E	^
61	3D	=	95	5F	_
62	3E	>	96	60	`
63	3F	?	97	61	a
64	40	@	98	62	b
65	41	A	99	63	c

Tableau 3-82 Codes de caractères pour Euro_Panel

Déc	Hex	Caractère	Déc	Hex	Caractère
100	64	d	135	87	ø
101	65	e	136	88	Ù
102	66	f	137	89	à
103	67	g	138	8A	ä
104	68	h	139	8B	â
105	69	i	140	8C	à
106	6A	j	141	8D	æ
107	6B	k	142	8E	ë
108	6C	l	143	8F	ê
109	6D	m	144	90	é
110	6E	n	145	91	è
111	6F	o	146	92	ï
112	70	p	147	93	î
113	71	q	148	94	ì
114	72	r	149	95	ñ
115	73	s	150	96	ô
116	74	t	151	97	Φ
117	75	u	152	98	ö
118	76	v	153	99	ò
119	77	w	154	9A	œ
120	78	x	155	9B	û
121	79	y	156	9C	ù
122	7A	z	157	9D	ü
123	7B	{	158	9E	§
124	7C	|	159	9F	⊆
125	7D	}	160	A0	Ç
126	7E	~	161	A1	¿
127	7F	DEL	162	A2	¥
128	80	Å	163	A3	∂
129	81	Ä	164	A4	€
130	82	Æ	165	A5	⊖
131	83	ß	166	A6	∅
132	84	É	167	A7	π
133	85	Ñ	168	A8	µ
134	86	Ö	169	A9	Ω

Tableau 3-82 Codes de caractères pour Euro_Panel (suite)

Déc	Hex	Caractère	Déc	Hex	Caractère
170	AA	∫	201	C9	→
171	AB	Σ	202	CA	↓
172	AC	≠	203	CB	↑
173	AD	≡	204	CC	«
174	AE	≤	205	CD	»
175	AF	≥	206	CE	
176	B0	0 super	207	CF	
177	B1	1 super	208	D0	0 sub
178	B2	2 super	209	D1	1 sub
179	B3	3 super	210	D2	2 sub
180	B4	4 super	211	D3	3sub
181	B5	9 super	212	D4	4 sub
182	B6	6 super	213	D5	5 sub
183	B7	7 super	214	D6	6 sub
184	B8	8 super	215	D7	7 sub
185	B9	9 super	216	D8	8 sub
186	BA	√	217	D9	9 sub
187	BB	±	218	DA	C inv
188	BC	▪	219	DB	H inv
189	BD	x	220	DC	inv
190	BE	÷	221	DD	P inv
191	BF	°	222	DE	S inv
192	C0	←	223	DF	A inv
193	C1	⊢	224	E0	E inv
194	C2	¦	225	E1	¤
195	C3	£	226	E2	¢
196	C4	⟨	227	E3	DEL
197	C5	⟩	228	E4	
198	C6	V	229		
199	C7	🔔	230		
200	C8	←	231		

Tableau 3-82 Codes de caractères pour Euro_Panel (suite)

ANNEXE A GLOSSAIRE DE TERMES

ASCII American Standard Code for Information Interchange : ensemble de caractères utilisé dans le système PC3000 (cf. annexe D pour voir la totalité des caractères ASCII).

AWG American Wire Gauge

Baud Nombre de changements d'état des signaux de ligne par seconde pour l'envoi d'informations série.

Caractère(s) de contrôle de bloc (BCC) Un ou plusieurs caractères qui contiennent une valeur créée à l'aide d'un algorithme comme un total ou une parité longitudinale appliquée aux octets d'un message de communications . Le BCC est généralement ajouté à la fin d'un message de transmission. Il peut être ensuite utilisé par le périphérique de réception pour effectuer un contrôle croisé de l'intégrité du message en relançant l'algorithme appliqué aux mêmes octets dans le message.

Identité de canal Dans certains protocoles, il est possible d'accéder à un sous-ensemble de paramètres à l'aide d'une identité de canal. Avec EI Bisync, une identité de canal (CHID) ayant la forme d'un caractère imprimable unique est parfois nécessaire.

Caractères de contrôle Certains caractères dans le jeu de caractères ASCII qui servent à encadrer les messages ou à signaler les informations de contrôle entre les périphériques. Les caractères de contrôle utilisés dépendent du protocole de communications utilisé. Exemples : <escape> 1BH.

EI Bisync Protocole série adopté par le groupe de sociétés Eurotherm International, utilisé avec un grand nombre d'appareils. Pour avoir une définition détaillée de cette norme, se reporter au 'Manuel Eurotherm International Bisync EI HP022047C' et aux 'Extensions du protocole de communications EI Bisync EI HP024113'.

ESP Système de supervision Eurotherm qui tourne sur PC.

Duplex intégral Lorsqu'il fonctionne dans ce mode, un périphérique peut émettre et recevoir simultanément des données. Cela implique normalement l'existence de supports séparés pour l'émission et pour la réception des données..

GID Identité de groupe, octet utilisé dans le protocole EI Bisync pour accéder à un groupe de périphériques esclaves qui, utilisé avec l'identité d'unité (UID), forme une adresse de communications de périphérique esclave.

Semi-duplex Ce mode de fonctionnement implique que, à tout moment, un périphérique peut soit émettre soit recevoir des données ; l'émission et la réception simultanées sont impossibles.

ICM	Module de communications intelligent : module de communications général PC3000 qui offre 4 ports spéciaux configurables par l'utilisateur.
LCM	Module de contrôle local : module de traitement principal d'un système PC3000 qui offre 3 ports série configurables par l'utilisateur.
Maître	Un périphérique maître peut émettre une demande de lecture ou d'écriture d'informations sur un périphérique déporté et tester des informations sur un certain nombre de périphériques déportés (cf. esclave).
Mnémonique	Abréviation texte qui peut être utilisée dans un protocole pour accéder à un élément, une fonction ou un paramètre donné(e) dans un périphérique. Avec le protocole EI Bisync, un mnémonique peut être n'importe quelle paire de lettres et de chiffres, par exemple 'SP', 'PV', P1, '23'.
Multi-points	Réseau de communications série dans lequel de nombreux périphériques esclaves sont reliés à un périphérique maître unique.
Parité	Bit de contrôle ajouté à un octet pour la transmission série afin d'améliorer la détection des erreurs. Une parité impaire implique qu'il y a toujours un nombre impair de bits qui sont transmis pour chaque octet ; une parité paire implique qu'il y a toujours un nombre pair de bits qui sont transmis.
Egal à égal	Ce type de protocole permet à n'importe quel périphérique relié à un réseau de communications d'envoyer des informations et d'en demander à n'importe quel autre périphérique du réseau.
Interrogation	Méthode de collecte d'informations provenant d'une série de périphériques déportés, reliés par l'intermédiaire d'une liaison série, consistant à demander successivement des informations à chacun d'entre eux. Cela implique que le périphérique qui interroge soit le 'maître' de la liaison.
Point à point	Liaison série utilisée uniquement pour relier deux périphériques
Port	Point de liaison physique sur le PC3000, auquel peuvent être reliés des périphériques déportés à l'aide d'une liaison série.
PLC	Automate programmable
Protocole	Définit les règles par lesquelles deux périphériques qui sont en communication échangent des informations. Cela inclut les méthodes servant à "emballer" et à coder les informations envoyées sur une liaison série.
PS	Station de programmation PC3000

Périphérique déporté Terme générique désignant tout périphérique communiquant avec le PC3000.

RS232, RS422, RS485 Désignent trois normes différentes utilisées pour la signalisation électrique d'informations sur une liaison série de communications.

Communications série Système de communications où une suite de caractères est envoyée en série, c'est-à-dire que les données sont décomposées en une série de bits envoyés l'un après l'autre sur une liaison série.

Liaison série Terme générique désignant le support physique servant à relier deux périphériques qui sont en communication ; peut être un fil simple, une paire de fils torsadés, des câbles à fibre optique, etc.

SFC Grafcet

Esclave Périphérique configuré de telle manière qu'il réponde aux demandes de lecture ou d'écriture d'informations émanant d'un périphérique 'maître' mais qui ne peut jamais émettre de demandes de communications.

Bits d'arrêt Temporisation mesurée en fonction de la durée de transmission d'un bit, insérée après la transmission d'un caractère. Servent à garantir qu'un temps suffisant s'écoule entre la transmission de caractères consécutifs et, par conséquent, délimitent chaque caractère.

Dépassement du délai imparti Technique utilisée dans la plupart des modules à protocole pour détecter un état d'erreur par la mesure de la durée d'un événement, comme l'arrivée d'un message. Si cette durée est supérieure à une valeur fixée, on dit que l'événement a dépassé le délai qui lui était imparti.

Transaction Terme générique désignant un échange d'informations entre des périphériques et qui peut impliquer une opération de lecture ou d'écriture.

UID Identité d'unité, qui fait partie de l'adresse esclave EI Bisync utilisée avec la GID.

ANNEXE B STRUCTURE DU FICHIER .CEL

Le fichier .CEL peut être créé lorsqu'un programme utilisateur est formé sur la station de programmation et peut servir à obtenir une liste de paramètres adressables dans un programme utilisateur. A chaque reconstitution d'un programme utilisateur, il est nécessaire d'obtenir un nouveau fichier .CEL dans le cas où les adresses de paramètres de blocs fonctions ont changé.

Segmentation d'un fichier .CEL.

Programme Essai 21-Nov-1993-14-48-30

N°	Nom de la déclaration	ID	Nom du paramètre	ID	Type
1	PcsSTATE	001	Ok_Down_Time	1	6
2	PcsSTATE	001	Seq_Mode	2	4
3	PcsSTATE	001	Alarm	3	4
4	PcsSTATE	001	Alarm_Count	4	2
5	PcsSTATE	001	HW_Fault	7	2
6	PcsSTATE	001	Module_Fault	8	2
7	PcsSTATE	001	HW_Condition	9	2
8	PcsSTATE	001	Barrery_Cond	10	3
9	PcsSTATE	001	Sys_Alarms	14	2
10	PcsSTATE	001	StartUp_Flag	15	4
11	PcsSTATE	001	Last_Down_Tm	16	6
12	PcsSTATE	001	Last_StartUp	17	3
13	PcsSTATE	001	Start_Strat	18	3
14	PcsSTATE	001	Reg_IO_Fail	19	4
15	PcsSTATE	001	HW_Links	20	2
16	Task_1	002	Single	1	4
17	Task_1	002	Intervalle	2	6
18	Task_1	002	Priorité	3	2

19	Task_1	002	Wdog_Time	4	6
20	Task_1	002	Overrun_Cnt	5	2
21	Task_1	002	Exec_Time	6	6
22	Task_1	002	Act_Interval	7	6
etc.....					
127	MAIN	014	Fini	3	4
128	hhh	015	Horloge	1	4
129	hhh	015	Process_Val	2	2
130	hhh	015	Réinitialisation	3	4
131	hhh	015	Q_Output	4	4
132	hhh	015	Count_Val	5	2

Nombre total de portes : 132

56775 (ID unique servant à assortir le logiciel utilisateur pour l'utilisation en ligne)

Le fichier contient les informations suivantes :

En-tête donnant le nom du programme, ('essai' dans l'exemple) et la date de création,

Liste de paramètres (ou portes) qui sont accessibles à un périphérique déporté,

Résumé à la fin du fichier, qui donne le décompte des paramètres et une identité unique. L'ID unique peut être utilisée par certains outils pour effectuer un contrôle croisé entre la liste et le programme utilisateur contenu dans un fichier .cfg. Elle est différente pour chaque création d'un programme utilisateur.

Les informations suivantes sont données pour chaque paramètre :

Numéro de saisie de paramètre classé séquentiellement,

Nom et numéro de déclaration des blocs fonctions,

Nom et numéro de paramètre, les paramètres d'une déclaration donnée de bloc fonction sont classé séquentiellement.

Type de données de paramètre, codé de la manière suivante :

Code	Type de données IERC1131	Commentaire
1	REAL	Nombre à virgule flottante, par exemple 1,123
2	INT	Entier 32 bits, par exemple 2341
3	ENUM	Valeur énumérée, codée sous la forme d'un entier 32 bits sans signe
4	BOOL	Booléen
5	IOADDRESS	Adresse d'E/S PC3000, codée sous la forme d'un entier 32 bits sans signe
6	TIME	Durée exprimée en ms, codée sous la forme d'un entier 32 bits sans signe
7	DATE	Date exprimée en secondes depuis le 1er jan 1970 00:00:00, codée sous la forme d'un entier 32 bits sans signe
8	TIMEOFDAY	Heure du jour exprimée en secondes depuis 00:00:00, codée sous la forme d'un entier 32 bits sans signe
9	DATEANDTIME	Date et heure exprimées en secondes depuis le 1er jan 1970 00:00:00, codées sous la forme d'un entier 32 bits sans signe
10	STRING	Chaîne de caractères qui peut être à la fois imprimable et non imprimable

Pour accéder à un paramètre donné, se reporter au module esclave de communications concerné afin d'avoir l'algorithme utilisé pour convertir les numéros de déclarations de blocs fonctions et les numéros de paramètres en adresses de protocole.

Par exemple, consulter la section 'Paramètre de bloc fonction' dans la description du bloc fonction 'Module esclave EI Bisync' pour avoir l'algorithme permettant de convertir les numéros de déclaration et de paramètre dans les champs d'adresse UID, Canal et Mnémonique EI Bisync.

ANNEXE C CODES STANDARD D'ERREURS DE COMMUNICATIONS

Les codes suivants s'appliquent à tous les modules. Des codes d'erreur supplémentaires seront utilisés pour tous les modules lorsque des codes d'erreur propres au protocole seront nécessaires.

Code	Etat d'erreur	Module esclave	Variable esclave	Module maître †	Variable déportée
0	OK	•	•	•	•
1	CHAINE D'ADRESSE TROP COURTE	•	•	•	•
	Adresse de port trop courte				
	Adresse déportée trop courte				
	Adresse esclave trop courte				
2	BITS DE DONNEES INTERDITS	•	•	•	•
3	PARITE INTERDITE	•	•	•	•
4	BITS D'ARRET INTERDITS	•	•	•	•
5	DEBIT RX PAS DISPONIBLE	•	•	•	•
6	DEBIT TX PAS DISPONIBLE	•	•	•	•
7	RTS PAS DISPONIBLE	•	•	•	•
8	CTS PAS DISPONIBLE	•	•	•	•
11	EMPLACEMENT INTERDIT Le premier caractère du paramètre Adresse ne se situe pas dans la plage valable comprise entre '0' et '5'.	•	•	•	•
12	PORT INTERDIT Le deuxième caractère du paramètre Adresse ne se situe pas dans la plage valable pour le module, par exemple 'A' à 'C' pour un port LCM ou 'A' à 'D' pour un port ICM.	•	•	•	•

Code	Etat d'erreur	Module esclave*	Variable esclave	Module maître †	Variable déportée
14	RX TX INCOMPATIBLES	*	*	*	*
16	VARIABLES DEPORTEES PAS PRISES EN CHARGE Le module de communications ne prend pas en charge les variables déportées.	*	*	*	*
17	PORT UTILISE	*	*	*	*
18	TROP GRAND NOMBRE DE VARIABLES ESCLAVES		*		
19	TROP GRAND NOMBRE DE TYPES DE MODULES		*		

N.B.: †Ces erreurs s'appliquent aussi au bloc fonction de module Raw_Comms.

Ces codes d'erreur peuvent être décalés vers un ensemble de valeurs supérieur pour certains modules, afin de ne pas entrer en conflit avec des codes d'erreur standard associés au protocole considéré. Par exemple, erreur 1, CHAINE D'ADRESSE TROP COURTE est décalée à 145 pour le module JBus_Slave.

ANNEXE D TABLEAU DE CODES ASCII

Le tableau de codes ASCII est donné à titre de référence.

Tableau de codes ASCII - caractères hexadécimaux

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 |	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 2	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E .	3F
40 @	41 A	42 B	43 C	44 D	45 E	46 F	7 47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E -	5F _
60 '	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E -	7F DEL

Chapitre 4

SYSTEME DE FICHIERS

Edition 2

Présentation

La classe de blocs fonctions Système de fichiers contient un ensemble de blocs fonctions permettant de définir et de commander le système de fichiers PC3000. Le système de fichiers est utilisé par les blocs fonctions Programmeur pour le stockage des données et peut également servir pour les applications simples de consignation des données. Les outils de programmation DOS contiennent des fonctions de téléchargement des fichiers depuis et vers le PC3000.

BLOC FONCTION FORMAT D'ENREGISTREMENT FICHIER

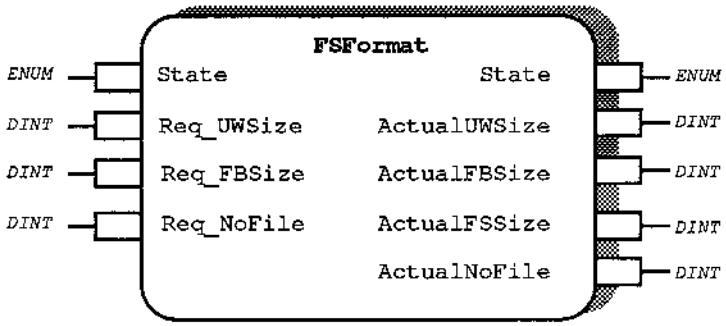


Figure 4-1 Bloc fonction format d'enregistrement fichier

Description fonctionnelle

L'enregistrement fichier PC3000 est créé dans l'espace mémoire RAM de réserve dans l'unité centrale (LCM.). L'espace mémoire dans l'unité centrale (LCM) est également nécessaire pour les programmes utilisateur, la source utilisateur téléchargée et les blocs fonctions téléchargeables. Par conséquent, l'espace précis qui peut être affecté à l'enregistrement fichier dépend de la mémoire nécessaire à ces autres utilisations et de la quantité de RAM installée sur l'unité centrale (LCM). Se reporter à la documentation technique de l'unité centrale (LCM) pour voir les configurations mémoire valables.

Le bloc fonction enregistrement fichier permet d'établir l'enregistrement fichier dans la RAM de réserve. Une fois que cet enregistrement fichier a été créé, il reste intact, avec l'ensemble des fichiers présents, jusqu'à ce qu'il soit explicitement reformaté par une déclaration du bloc fonction format d'enregistrement fichier.

Il est possible de déterminer l'espace mémoire nécessaire au programme utilisateur à partir des informations d'état en ligne fournies par le logiciel de programmation PC3000 une fois qu'un programme a été téléchargé. Il faut prévoir un espace suffisant pour le plus grand programme utilisateur qui sera téléchargé sur le PC3000 ainsi que les éventuels blocs fonctions téléchargeables associés qui sont nécessaires. L'unité centrale (LCM) passe à l'état de repos lorsqu'on essaie d'effectuer un téléchargement avec un espace insuffisant pour le programme utilisateur et les blocs fonctions téléchargeables.

Ne pas oublier que l'enregistrement fichier est uniquement supprimé de la mémoire si l'alimentation électrique de l'unité centrale (LCM) est coupée et la batterie débranchée. Il est possible de modifier sa taille (bien que cela signifie que son contenu sera supprimé) à l'aide du bloc fonction Format de l'enregistrement fichier.

En cas de tentative de création d'enregistrement fichier supérieur au maximum permis par d'autres utilisations de la mémoire pendant qu'un programme est en cours d'exécution, l'enregistrement fichier de la taille maximale possible est créé avec le programme en cours et les blocs fonctions téléchargeables restent intacts. Toutefois, cela signifie qu'un programme plus gros ou nécessitant davantage d'espace de bloc fonction téléchargeable ne pourrait pas être téléchargé avant le reformatage de l'enregistrement fichier.

Attributs du bloc fonction

Type : D80

Classe : .....FILESYSTEM

Tâche par défaut : .....Task_2

Liste résumée : .....State, ActualUWSize, ActualFBSize, ...ActualFSSize.

Mémoire nécessaire : .....36 octets

Description des paramètres

State (S)

Lorsqu'il est piloté comme entrée, ce paramètre permet le formatage de l'enregistrement fichier. Il affiche également le résultat de l'opération de formatage.

Ok :	état de repos de cette entrée/sortie : aucune action de formatage n'a été encore commencée ou le dernier formatage a réussi.
Ecriture :	sert à lancer une opération de formatage. Le paramètre State revient automatiquement à Ok ou Erreur, selon que l'opération a ou non réussi.
Erreur :	la dernière action de formatage a échoué.

Req_UWSize (RUW)

Taille de RAM disponible que l'utilisateur souhaite réserver pour les programmes utilisateur. Si elle est inférieure à la taille du programme utilisateur actuel, c'est la taille de ce dernier qui est utilisée.

Req_FBSize (RFB)

Taille de RAM disponible que l'utilisateur souhaite réserver pour les bibliothèques de blocs fonctions téléchargeables. Si elle est inférieure à ActualFBSize, c'est la taille de la bibliothèque de blocs fonctions téléchargeables actuelle qui est utilisée.

Req_NoFile (RNF)

Permet à l'utilisateur de limiter le nombre de fichiers qui peuvent être contenus à un moment quelconque dans l'enregistrement fichier.

ActualUWSize (AUW)

Quantité de RAM disponible actuellement réservée pour les programmes utilisateur. Ne représente pas la taille réelle du programme utilisateur actuel.

ActualFBSize (AFB)

Quantité de RAM disponible actuellement réservée pour les bibliothèques de blocs fonctions téléchargeables. Ne représente pas la taille réelle de la bibliothèque de blocs fonctions actuellement téléchargée.

ActualFSSize (AFS)

Quantité de RAM disponible actuellement réservée pour l'enregistrement fichier PC3000.

ActualNoFile (ANF)

Nombre effectif de fichiers qui peuvent être contenus à un moment quelconque dans l'enregistrement fichier.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en écriture	Accès en lecture	Informations propres au type	
State	ENUM	Ok (0)	Oper	Oper	Cf. liste des paramètres	
Req_UWSize	DINT	0	Oper	Oper	Limite haute Limite basse	2147483647 0
Req_FBSize	DINT	0	Oper	Oper	Limite haute Limite basse	2147483647 0
Req_NoFile	DINT	0	Oper	Oper	Limite haute Limite basse	255 20
ActualUWSize	DINT	Memory in LCM, RAM1, RAM2	Oper	Block	Limite haute Limite basse	Dépend de la RAM de l'unité centrale 0
ActualFBSize	DINT	Memory in LCM, RAM3,	Oper	Block	Limite haute Limite basse	Dépend de la RAM de l'unité centrale 0
ActualFSSize	DINT	0	Oper	Block	Limite haute Limite basse	Dépend de la RAM de l'unité centrale 0
ActualNoFile	DINT	0	Oper	Oper	Limite haute Limite basse	255 20

Table 4-2 Bloc fonction Format de l'enregistrement fichier

BLOC FONCTION MANIPULATION DES FICHIERS DE L'ENREGISTREMENT FICHIER

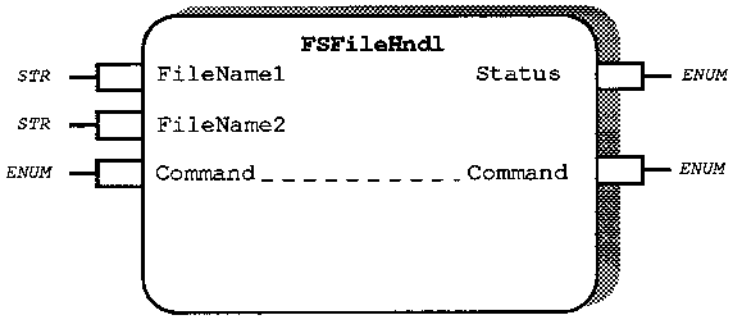


Figure 4-2 Bloc fonction Manipulation des fichiers de l'enregistrement fichier

Description fonctionnelle

Ce bloc permet la réalisation d'opérations simples de manipulation des fichiers dans l'enregistrement fichier préformatéeoù sont présents des fichiers. Pour effectuer les opérations qui nécessitent de connaître les noms de fichiers, il peut être nécessaire d'utiliser au préalable le bloc fonction FSDirectry.

Attributs du bloc fonction

Type : D82
Classe : .....FILESYSTEM
Tâche par défaut : .....Task_2
Liste résumée : .....FileName1, FileName2, Command, Status.
Mémoire nécessaire : .....42 octets

Description des paramètres

FileName1 (FN1)

Nom du premier fichier à utiliser en liaison avec Command. La lettre de la mémoire (R: pour la RAM ou E: pour la ROM) doit être insérée devant le nom du fichier. Les noms de mémoires et de fichiers dépendent de la casse des caractères.

FileName2 (FN2)

Nom du deuxième fichier à utiliser en liaison avec Command si besoin est. Les noms de mémoires et de fichiers dépendent de la casse des caractères.

Command (CMD)

Paramètre de commande de ce bloc fonction.

Ok :	état au repos de cette entrée/sortie : aucune action de manipulation de fichier n'a été encore commencée ou la dernière action demandée a été achevée ou arrêtée.
Copie :	copie du fichier dont le nom est donné par FileName1 vers le fichier dont le nom est donné par FileName2. Si FileName2 n'existe pas, il est créé. S'il existe, son contenu est écrasé. Command revient à Ok lorsque la copie a réussi ou lorsque son échec est certain.
Suppression :	suppression du fichier dont le nom est donné par FileName1. Le fichier dont le nom est donné par FileName2 n'est pas touché. Command revient à Ok lorsque la suppression a réussi ou lorsque son échec est certain.
CloseAl :	l'ensemble des fichiers dans l'enregistrement fichier sont examinés pour voir les fichiers qui sont censés être ouverts. Tous les fichiers ouverts mais auquel aucun bloc fonction n'a actuellement accès est fermé.

Status (ST)

Cette sortie indique l'état de l'opération en cours ou l'état d'achèvement de l'opération précédente si aucune opération n'est en cours.

Ok :	la dernière opération s'est achevée correctement.
Copie :	une opération de copie est en cours.
Erreur :	une erreur indéfinie s'est produite dans la dernière opération.
OpenErr :	une erreur s'est produite lors de la tentative d'ouverture d'un fichier.
ClsErr :	une erreur s'est produite lors de la tentative de fermeture d'un fichier.
CopyErr :	une erreur s'est produite lors de la tentative de copie d'un fichier.
ReadErr :	une erreur s'est produite lors de la tentative de lecture d'un fichier.
WrtErr :	une erreur s'est produite lors de la tentative d'écriture d'un fichier.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type
FileName1	**STR**	"	Oper	Oper	12 caractères au maximum
FileName2	**STR**	"	Oper	Oper	12 caractères au maximum
Command	**ENUM**	Ok(0)	Oper	Oper	Cf. liste des paramètres
Status	**ENUM**	Ok(0)	Oper	Block	Cf. liste des paramètres

Tableau 4-2 Bloc fonction Manipulation de fichier d'enregistrement fichier

BLOC FONCTION ESPACE LIBRE D'ENREGISTREMENT FICHIER

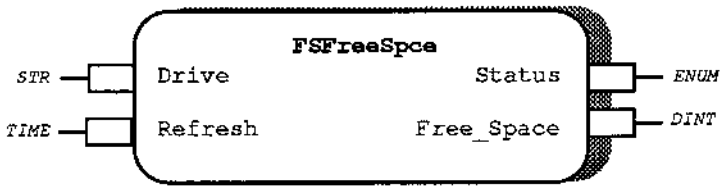


Figure 4-3 Bloc fonction Espace libre d'enregistrement fichier

Description fonctionnelle

Ce bloc fonction permet de contrôler l'espace libre actuel dans l'enregistrement fichier du PC3000. Il est possible de faire varier la fréquence de rafraîchissement afin de limiter le temps d'immobilisation du système par ce bloc fonction sur le système.

Il est possible d'examiner différents types d'enregistrement fichier du PC3000. Les seuls types actuellement valables sont 'R' (enregistrement fichier RAM) et 'E' (enregistrement fichier ROM).

Attributs du bloc fonction

Type : D84
Classe : ..... FILESYSTEM
Tâche par défaut : ..... Task_2
Liste résumée : ..... Drive, Refresh, Status, Free_Space.
Mémoire nécessaire : ..... 20 octets

Description des paramètres

Drive (DRV)

Type de mémoire à examiner. Valeurs valables : 'R' (système de fichiers RAM) et 'E' (système de fichiers ROM).

Refresh (R)

Fréquence à laquelle l'espace mémoire est examiné pour mettre à jour la sortie Free_Space.

Status (ST)

Indique l'état du lecteur sélectionné.

OK :	le lecteur a été formaté et est fonctionnel.
NoSuprt :	le type de lecteur sélectionné n'est pas reconnu ou n'est pas valable pour le système de fichiers actuel.
NoFile :	de la mémoire est affectée à l'enregistrement fichier mais elle a perdu la structure de son répertoire. Un reformatage sera nécessaire.
NoFrmat :	le lecteur n'est pas formaté. Cela n'indique pas nécessairement que le lecteur pourrait être formaté ou est valable pour le système actuel.

Free_Space (FRE)

Espace libre sur le lecteur sélectionné. Il s'agit de l'espace disponible pour les fichiers nouveaux.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Drive	STR	'R'	Oper	Oper	Seuls 'R ' et 'E' sont actuellement valables	
Refresh	TIME	1s	Oper	Oper	Limite haute	23d 23h 59m 59s 999ms
					Limite basse	0ms
Status	ENUM	Ok(0)	Oper	Bloc	Cf. liste des paramètres	
Free_Space	DINT	Configuration du format	Oper	Bloc	La plage dépend de la taille de la mémoire de l'unité centrale	

Table 4-3 Bloc fonction Espace libre d'enregistrement fichier

BLOC FONCTION ACCES À L'ENREGISTREMENT FICHIER

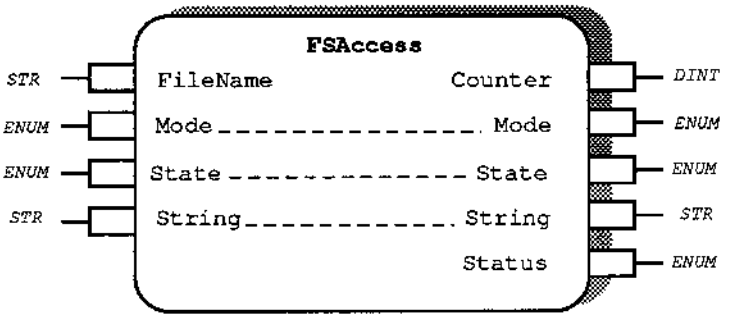


Figure 4-4 Bloc fonction Accès à l'enregistrement fichier

Description fonctionnelle

Ce bloc permet la création de fichiers nouveaux et la lecture ou l'écriture ligne par ligne ou caractère par caractère du contenu des fichiers existants.

Attributs du bloc fonction

Type : D86
Classe : .....FILESYSTEM
Tâche par défaut : .....Task_2
Liste résumée : .....FileName, Mode, State, String.
Mémoire nécessaire : .....252 octets

Description des paramètres

FileName (FNM)

Nom du fichier qui sera lu ou écrit. La lettre de la mémoire (R: pour RAM ou E: pour ROM) doit être insérée devant le nom du fichier. Les noms de lecteurs et de fichiers dépendent de la casse des caractères.

Mode (M)

Détermine l'opération qui sera effectuée à l'aide des entrées et sorties State et String.

Ok — état de repos du bloc fonction. Le paramètre State n'a aucun effet lorsque mode est sur Ok.

Read — le paramètre State peut servir à lire dans le fichier ligne par ligne ou caractère par caractère. Les informations lues apparaissent sur le paramètre d'entrée/sortie String. Une fois que la fin du fichier est atteinte, Mode revient automatiquement à Ok : aucune action du programme utilisateur n'est nécessaire pour cela.

Write — le paramètre State peut servir à écrire dans le fichier ligne par ligne ou caractère par caractère. Les informations à écrire sont saisies à l'aide du paramètre d'entrée/sortie String et commencent à la première ligne du fichier : toutes les informations du fichier situées au début de l'opération Write seront écrasées.

L'écriture ne peut commencer que lorsqu'aucun autre bloc fonction n'accède au fichier en question, aussi bien pour la lecture que pour l'écriture. Une manière utile de vérifier consiste à utiliser le paramètre FSDirectry.FileRdOpen qui doit être égal à zéro avant que l'écriture puisse commencer.

Append — le paramètre State peut servir à ajouter des éléments au fichier ligne par ligne ou caractère par caractère. Les informations à ajouter sont saisies à l'aide du paramètre d'entrée/sortie String et seront placées après les informations déjà contenues dans le fichier.

Il faut noter que le premier caractère ou la première ligne qui doit commencer un fichier doit être inséré(e) à l'aide de Mode=Write. Il est ensuite possible d'ajouter des caractères et des lignes.

L'ajout ne peut commencer que lorsqu'aucun autre bloc fonction n'accède au fichier en question, aussi bien pour la lecture que pour l'écriture. Une manière utile de vérifier consiste à utiliser le paramètre FSDirectry.FileRdOpen qui doit être égal à zéro avant que l'ajout puisse commencer.

State (S)

Permet de lire ou d'écrire ligne par ligne ou caractère par caractère les informations du fichier dont le nom est donné par FileName.

Ok	état au repos du bloc fonction. State revient automatiquement à OK. Aucune action du programme utilisateur n'est nécessaire.
NextLn	si le paramètre Mode est positionné sur Read, cela provoque la lecture de la ligne suivante du fichier et l'affichage du résultat sur l'entrée/sortie String. Le caractère de fin de ligne (<CR><LF>) n'est pas visible. Si le paramètre Mode est positionné sur Write, la chaîne affichée sur l'entrée/sortie String est écrite comme ligne suivante du fichier et se termine par 'RL' i.e. <CR><LF>.
NextChr	si le paramètre Mode est positionné sur Read, cela provoque la lecture du caractère suivant du fichier et l'affichage du résultat sur l'entrée/sortie String. Le caractère de fin de ligne (<CR><LF>) n'est pas visible. Si le paramètre Mode est positionné sur Write, cela provoque l'écriture de la chaîne affichée sur l'entrée/sortie String comme caractère suivant du fichier. Il est possible de terminer une ligne en écrivant $R puis $L comme les deux derniers caractères écrits ou ajoutés.

String (STR)

Valeur de la chaîne à écrire dans le fichier lorsqu'on est en mode Write ou Append Mode ou valeur de la chaîne lue dans le fichier lorsqu'on est en mode Read.

Counter (CNT)

Numéro de ligne actuel dans le fichier lorsqu'on utilise le mode NextLn.

Status (ST)

Indique si la dernière opération demandée à l'aide du paramètre State dans le fichier dont le nom est donné par FileName a ou non réussi.

Ok	la dernière opération a réussi.
Opn_Err	il y a eu une erreur lors de l'ouverture du fichier.
Cls_Err	il y a eu une erreur lors de la fermeture du fichier.
EOF	la dernière opération a provoqué le passage à la fin du fichier.
LnToLng	la chaîne fournie par l'intermédiaire de String comme entrée à écrire dans le fichier était trop longue. Il faut noter que, lorsqu'on écrit avec State=NextChr, String devrait être un seul caractère.
WrtErr	une erreur s'est produite lors de la tentative d'écriture dans le fichier.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
FileName	**STR**	"	Oper	Oper	12 caractères au maximum	
Mode	**ENUM**	Ok(0)	Oper	Oper	Cf. liste des paramètres	
State	**ENUM**	Ok(0)	Oper	Oper	Cf. liste des paramètres	
String	**STR**	"	Oper	Oper	80 caractères au maximum	
Counter	**DINT**	0	Oper	Bloc	Limite haute Limite basse	2147483647 0
Status	**ENUM**	Ok(0)	Oper	Bloc	Cf. liste des paramètres	

Table 4-4 Attributs des paramètres Accès à l'enregistrement fichier

BLOC FONCTION ACCES A L'ENREGISTREMENT FICHIER

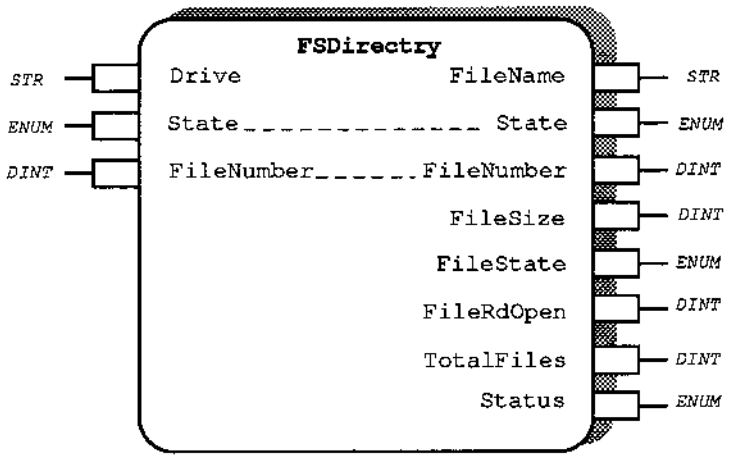


Figure 4-5 Bloc fonction Accès à l'enregistrement fichier

Description fonctionnelle

Ce bloc fonction donne la possibilité de déterminer le nom et la taille des fichiers actuellement dans l'enregistrement fichier. L'espace libre dans l'enregistrement fichier est également indiqué.

Les informations des sorties ne sont pas mises automatiquement à jour, elles doivent être mises à jour par le programme utilisateur à l'aide de l'entrée State.

Attributs du bloc fonction

Type : D88
Classe : .....FILESYSTEM
Tâche par défaut : .....Task_2
Liste résumée : .....Drive, State, FileNumber, FileName
Mémoire nécessaire : .....54 octets

Description des paramètres

Drive (DRV)

Spécifie le type de mémoire dont il faut examiner les informations de l'enregistrement fichier.

Les types de mémoires possibles sont RAM (désigné par 'R') et EPROM (désigné par 'E').

State (S)

Ce paramètre permet au programme utilisateur de commander l'examen du répertoire du lecteur choisi.

Ok — état au repos du bloc.

Top — provoque l'examen du haut du répertoire. **FileNumber** est positionné sur zéro et State revient immédiatement à Ok. Aucune action n'est nécessaire de la part du programme utilisateur pour parvenir à ce résultat.

S'il y a des fichiers dans l'enregistrement fichier, cette entrée supérieure porte le **FileName** <FREE-SPACE> et FileSize est égal à l'espace mémoire disponible dans l'enregistrement fichier, à l'exclusion de l'espace déjà utilisé par les fichiers. La valeur est égale à celle donnée par FSFreeSpce.**Free_Space** mais pas à celle de FSFormat.**ActualFSSize.**

S'il n'y a aucun fichier dans l'enregistrement fichier, cette entrée supérieure porte le **FileName** 'FREE'. **FileSize** est égal à l'espace mémoire disponible dans l'enregistrement fichier moins l'espace nécessaire à la mise à jour de l'enregistrement fichier. La valeur est égale à celle donnée par FSFreeSpce.**Free_Space** mais pas à celle de FSFormat.ActualFSSize, même si l'enregistrement fichier est vide.

ReadNxt — les informations relatives à l'entrée suivante de la liste du répertoire du fichier dont FileNumber est indiqué sont présentées sur les sorties. State revient immédiatement à Ok : aucune action n'est nécessaire de la part du programme utilisateur pour parvenir à ce résultat. **FileNumber** est automatiquement incrémenté.

Si la fin du répertoire a été atteinte (indiquée par Status), **FileNumber** continue à être incrémenté mais aucune des autres sorties ne change.

ReadAgn les informations relatives à l'entrée de la liste du répertoire dont FileNumber est indiqué sont présentées sur les sorties. State revient immédiatement à Ok : aucune action n'est nécessaire de la part du programme utilisateur pour parvenir à ce résultat. Cela permet la mise à jour des sorties avec l'état actuel d'un fichier donné.

FileNumber (FNB)

Le bloc fonction met automatiquement à jour ce paramètre d'entrée/sortie si le paramètre State est utilisé pour se déplacer dans le répertoire à l'aide de ReadNxt.

Il est toutefois possible de positionner **FileNumber** sur une valeur donnée et de faire afficher sur les sorties des détails relatifs au fichier choisi en positionnant State sur **ReadAgn**.

FileName (FNM)

Nom du dernier fichier mis à jour par les actions des entrées State et **FileNumber**. Quatre noms de fichiers intégrés sont utilisés :

" Aucun enregistrement fichier n'est configuré pour ce lecteur.

'FREE' Aucun fichier dans l'enregistrement fichier.

'<FREE-SPACE>' Le haut du répertoire est examiné. **FileSize** montre l'espace disponible pour les fichiers sur le lecteur indiqué et FileState affiche Unused.

'<END-OF_DIR>' **FileNumber** est le nombre de fichiers plus un et Status affiche DirEnd.

A part ces noms, les noms de fichiers apparaissent pour les fichiers qui ont été chargés par l'utilisateur ou le programme utilisateur.

FileSize (FSZ)

Taille, exprimée en octets, du fichier indiqué par **FileName.**

FileState (S)

Indique l'état actuel du fichier indiqué par **FileName.**

Unused	Cet espace n'est actuellement utilisé par aucun fichier (normalement visible uniquement en haut et à la fin du répertoire).
Closed	Le fichier est fermé, aucun autre bloc fonction n'a accès au fichier.
WrtOpen	Un bloc fonction écrit dans le fichier ou il y a un accès au fichier par une liaison de communications.
RdOpen	Un ou plusieurs bloc(s) fonction(s) lit(lisent) dans le fichier. Le nombre de blocs fonctions qui lisent actuellement dans le fichier est affiché par **FileRdOpen.**
RdWtOpn	Un ou plusieurs bloc(s) fonction(s) lit(lisent) dans le fichier et un bloc fonction écrit dans le fichier. Le nombre de blocs fonctions qui lisent actuellement dans le fichier est affiché par **FileRdOpen.**
NoSpce	Une erreur s'est produite dans le système de fichiers du fait que l'enregistrement fichier est saturé.
NoSpCls	
NoSpWrt	[Erreurs système de niveau. élémentaire. S'adresser à Eurotherm
Automation]	
NoSpRd	
NoSpRdWt	

FileRdOpen (FRO)

Nombre de blocs fonctions actuellement lisant actuellement dans le fichier dont le nom est donné par FileName.

TotalFiles (TFL)

Nombre total de fichiers dans le répertoire la dernière fois que ce bloc fonction a lu le répertoire en commençant par le haut.

Status (ST)

Indique l'état d'achèvement de la dernière opération demandée par State.

Ok	La dernière opération s'est terminée correctement.
DirEnd	La dernière opération a provoqué une lecture jusqu'à la fin du répertoire.
Error	Une erreur a eu lieu lors de la dernière opération. Les sorties ne donnent pas des informations fiables.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Drive	STR	'R'	Oper	Oper	Seuls 'R' et 'E' sont actuellement valables	
State	ENUM	Ok(0)	Oper	Oper	Cf. liste des paramètres	
FileNumber	DINT	0	Oper	Oper	Limite haute Limite basse	2147483647 0
FileName	STR	"	Oper	Block	12 caractères au maximum	
FileSize	DINT	"	Oper	Block	Limite haute Limite basse	2147483647 0
FileState	ENUM	Unused(0)	Oper	Block	Cf. liste des paramètres	
FileRdOpen	DINT	0	Oper	Block	Limite haute Limite basse	30 0
TotalFiles	DINT	0	Oper	Block	Limite haute Limite basse	2147483647 0
Status	ENUM	Ok(0)	Oper	Block	Cf. liste des paramètres	

Table 4-5 Attributs des paramètres Accès au répertoire de l'enregistrement fichier

Chapitre 5

MODULES

Edition 1

Présentation

Ce chapitre est réservé aux blocs fonctions de modules qui offrent des paramètres de configuration et des informations pour le diagnostic applicables en général à un module matériel et non propres à un canal donné.

Des informations complémentaires au sujet du module de saisie d'impulstions (PIM2) et du module d'impulsions profilées (PPM) seront fournies courant 1994.

Chapitre 6

ENTREES

Edition 1

Sommaire (suite)

Présentation

Ce chapitre décrit la classe de blocs fonctions ENTREES.

Les blocs fonctions Entrées sont automatiquement créés comme élément de la définition matérielle PC3000, c'est-à-dire l'opération consistant à déclarer le type de module d'E/S situé dans chaque emplacement dans le rack. Chaque bloc fonction Entrée est 'relié' à un canal physique d' E/S. Une fois définis, ces blocs peuvent être manipulés de la même manière que les blocs fonctions des autres classes.

BLOC FONCTION DIGITAL_IN

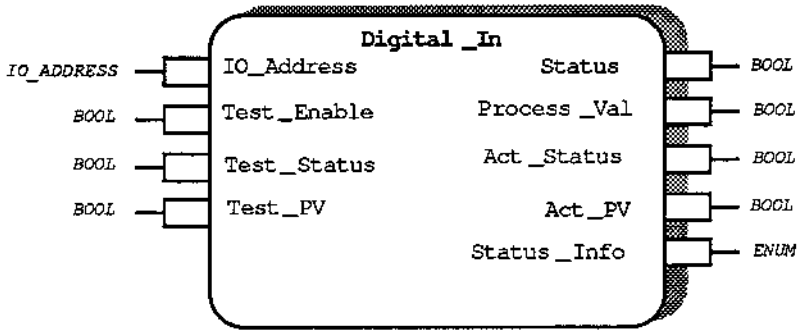


Figure 6-1 Bloc fonction Digital_In

Description fonctionnelle

Le bloc fonction Digital_In assure l'interface avec le module matériel d'entrée numérique. Il dispose d' un paramètre de sortie booléen qui définit le sens d'une entrée logique physique associée.

Des outils de test sont fournis pour permettre le remplacement de l'état des entrées physiques par des valeurs de test.

Attributs du bloc fonction

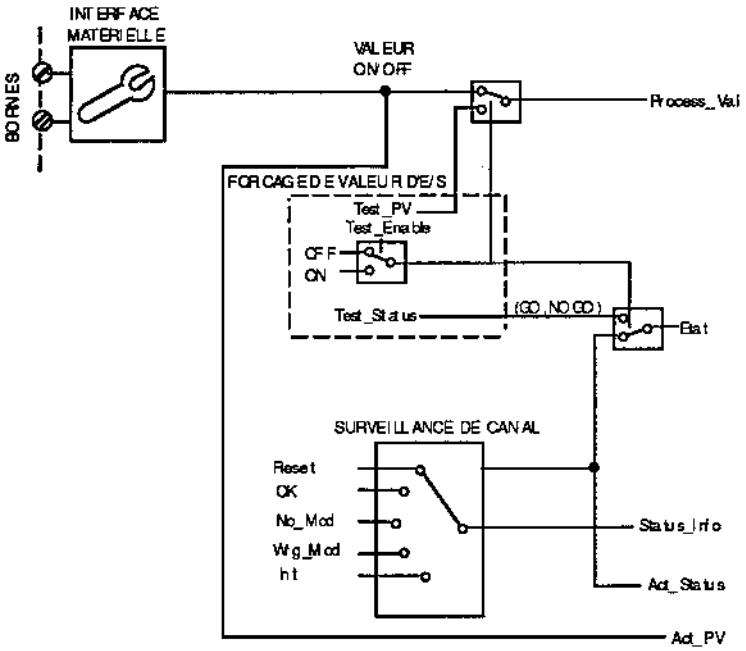
Type : 10 10

Classe : ..... INPUTS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Process_Val, Act_PV, Status, Status_Info

Besoins de capacité mémoire : 12 octets



Description des paramètres

IO_Address (IOA)

IO_Address associe la déclaration de bloc fonction aux entrées physiques du module matériel auxquelles elle se rapporte. Sa valeur est définie automatiquement lorsque la déclaration du bloc fonction est effectuée Sa valeur a la forme X:YY:ZZ, où X représente le numéro du rack dans lequel réside le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro du canal dans le module. Par exemple, 1:02:03 signifie que la déclaration du bloc fonction se rapporte au troisième canal d'un module situé dans le deuxième emplacement du premier rack du système PC3000.

Test_Enable (TEN)

Test_Enable permet à l'utilisateur de commuter la valeur lue par Process_Val entre le canal d'entrée matériel et l'entrée Test_PV dans le bloc fonction. Si Test_Enable est positionné sur Off (0), Process_Val lira sa valeur dans l'entrée matérielle et Status montrera l'état du matériel. Si Test_Enable est sur On (1), Process_Val lira sa valeur dans Test_PV et Status sera piloté par l'entrée Test_Status.

Test_Status (TST)

La valeur de Test_Status est reportée dans Status lorsque Test_Enable est sur On (1). Lorsque Test_Enable est sur Off (0), Test_Status n'est pas utilisé.

Test_PV (TPV)

Test_PV est reporté dans la sortie Process_Val lorsque Test_Enable est sur On (1). Lorsque Test_Enable est sur Off (0), Test_PV n'est pas utilisé.

Status

Lorsque Test_Status est sur Off (0), le paramètre Status montre l'état du canal matériel adressé par le bloc fonction. Si Test_Status est sur On (1), Status lira directement sa valeur dans Test_Status. Status peut prendre les valeurs NOGO (0) ou Go (1).

Process_Val (PV)

En fonctionnement normal, lorsque Test_Enable est sur Off (0), Process_Val montre l'entrée dans le canal d'entrée logique adressé par le bloc fonction. Si Test_Enable est sur On (1), Process_Val lit directement sa valeur dans Test_PV. Il s'agit de la valeur utilisée par la stratégie de contrôle.

Act_Status (AST)

Act_Status montre l'état du canal matériel adressé par le bloc fonction. Act_Status montre toujours l'état du matériel alors qu'Status peut être commuté entre l'entrée matérielle et Test_Status en mode test. Ce paramètre doit uniquement être utilisé à des fins de diagnostic.

Act_PV (APV)

Act_PV montre l'entrée dans le canal d'entrée logique adressé par le bloc fonction. Act_PV montre toujours l'entrée matérielle alors que Process_Val peut être commutée entre l'entrée matérielle et Test_PV en mode test. Ce paramètre doit uniquement être utilisé à des fins de diagnostic.

Status_Info (STI)

Status_Info est un paramètre de diagnostic qui sert à montrer la valeur de Status. Il peut avoir 5 valeurs possibles :

Reset (0) :	le programme utilisateur ne tourne pas.
Ok (1) :	le canal fonctionne normalement.

No_Mod (2) : il n'y a aucun module dans l'emplacement matériel eadressé par le bloc fonction.

Wrg_Mod (3) : un type de module incorrect a été placé dans l'emplacement adressé par le bloc fonction.

Init (4) : le module ou le canal est en cours d'initialisation. Cet état est passager et peut être observé lors de la mise en route.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Act_PV	**BOOL**	Off (0)	Config	Bloc	Détection	Off (0) On (1)
Act_Status	**BOOL**	NOGO (0)	Config	Bloc	Détection	NOGO(0) Go (1)
IO_Address	**IO_ADDRESS**		Config	Config		
Process_Val	**BOOL**	Off (0)	Oper	Bloc	Détection	Off (0) On(1)
Status	**BOOL**	NOGO (0)	Oper	Bloc	Détection	NOGO(0) Go(1)
Status_Info	**ENUM**	Reset (0)	Oper	Bloc	Détection	Cf. Description des paramètres
Test_Enable	**BOOL**	Off (0)	Config	Config	Détection	Off(0) On(1)
Test_PV	**BOOL**	Off (0)	Config	Config	Détection	Off(0) On(1)
Test_Status	**BOOL**	NOGO (0)	Config	Config	Détection	NOGO(0) Go(1)

Tableau 6-1 Attributs des paramètres

BLOC FONCTION DEBOUNCE_IN

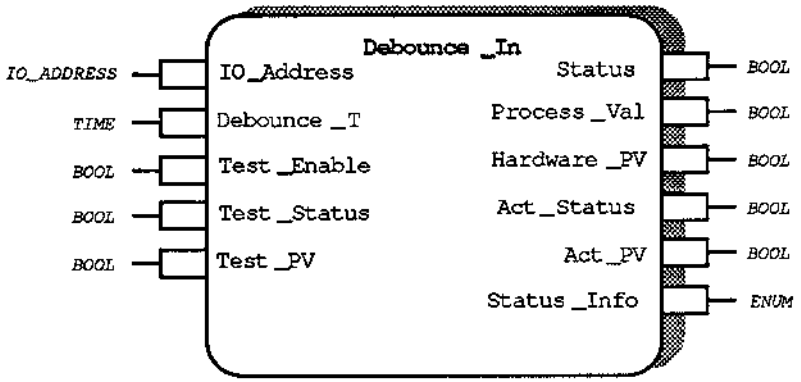


Figure 6-2 Bloc fonction Debounce_In

Description fonctionnelle

Le bloc fonction Debounce_In assure l'interface avec le module matériel d'entrée logique. Une fonction antirebond est incluse pour compenser le rebond de contact. Lorsque l'antirebond est utilisé, Process_Val change sur le premier front détecté et ne variera pas pendant toute la durée de l'antirebond.

Attributs du bloc fonction

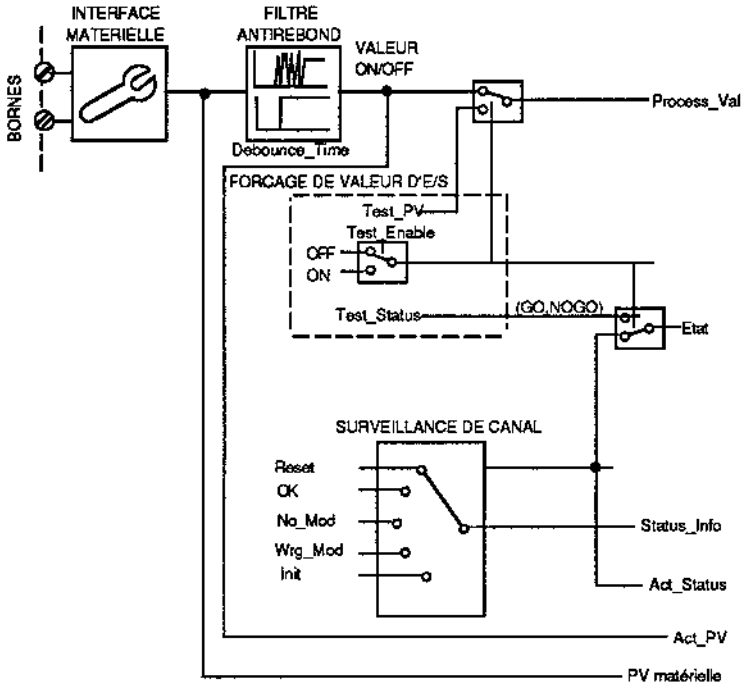
Type : 10 14

Classe : ..... INPUTS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Process_Val, Act_PV, Status, Status_Info

Besoins de capacité mémoire : 22 octets



Description des paramètres

IO_Address (IOA)

IO_Address associe la déclaration du bloc fonction aux entrées physiques sur le module matériel auxquelles elle se rapporte. Sa valeur est définie automatiquement lorsque la déclaration du bloc fonction est effectuée. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro de l'emplacement du rack dans lequel se trouve le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro du canal dans le module. Par exemple, 1:02:03 signifie que la déclaration du bloc fonction se rapporte au troisième canal d'un module qui se trouve dans le deuxième emplacement du premier rack du système PC3000.

Debounce_T (DT)

Debounce_T définit la durée pendant laquelle Process_Val est conservée à la suite de la détection d'un front sur le canal d'entrée.

Test_Enable (TEN)

Test_Enable permet à l'utilisateur de commuter la valeur lue par Process_Val entre le canal d'entrée matérielle et l'entrée Test_PV dans le bloc fonction. Si Test_Enable est sur Off (0), Process_Val et Hardware_PV liront leur valeur dans l'entrée matérielle et Status montrera l'état du matériel.

Si Test_Enable est sur On (1), Process_Val et Hardware_PV liront leur valeur dans Test_PV et Status sera piloté par l'entrée Test_Status.

Test_Status (TST)

La valeur de Test_Status est reportée dans Status lorsque Test_Enable est sur On (1). Lorsque Test_Enable est sur Off (0), Test_Status n'est pas utilisé.

Test_PV (TPV)

Test_PV est reporté dans les sorties Process_Val et Hardware_PV lorsque Test_Enable est sur On (1). Lorsque Test_Enable est sur Off (0), Test_PV n'est pas utilisé.

Status

Lorsque Test_Status est sur Off (0), le paramètre Status montre l'état du canal matériel en cours d'adressage par le bloc fonction. Si Test_Status est sur On (1), Status lira directement sa valeur dans Test_Status. Status peut prendre les valeurs NOGO (0) ou Go (1).

Process_Val (PV)

Au cours du fonctionnement normal, lorsque Test_Enable est sur Off (0), Process_Val montre l'entrée dans le canal d'entrée logique en cours d'adressage par le bloc fonction. Si Test_Enable est sur On (1), Process_Val lit directement sa valeur dans Test_PV.

Dans les deux modes, la fonction d'antirebond agit pour garantir que Process_Val répond uniquement au premier front de l'entrée qui change. Il s'agit de la valeur utilisée par la stratégie de contrôle.

Hardware_PV (HPV)

Hardware_PV montre la valeur de process avant son passage par le filtre antirebond.

Act_Status (AST)

Act_Status montre l'état du canal matériel adressé par le bloc fonction. Act_Status montre toujours l'état du matériel alors qu'Status peut être commuté entre l'entrée matérielle et Test_Status en mode test. Ce paramètre doit être uniquement utilisé à des fins de test.

Act_PV (APV)

Act_PV montre l'entrée dans le canal d'entrée logique adressé par le bloc fonction, la fonction d'antirebond étant incluse pour garantir qu'Act_PV répond uniquement au premier front de l'entrée qui change.

Act_PV montre toujours l'entrée matérielle alors que Process_Val peut être commutée entre l'entrée matérielle et Test_PV en mode test. Ce paramètre doit être uniquement utilisé à des fins de diagnostic.

Status_Info (STI)

Status_Info est un paramètre de diagnostic qui sert à indiquer la valeur de Status. Il peut avoir 5 valeurs :

Réinitialisation (0) : le programme utilisateur ne tourne pas.

Ok (1) : le canal fonctionne normalement.

No_Mod (2) : il n'y a aucun module dans l'emplacement matériel adressé par le bloc fonction.

Wrg_Mod (3) : un type de module incorrect a été placé dans l'emplacement adressé par le bloc fonction

Init (4) : le module ou le canal est en cours d'initialisation. Il s'agit d'un état temporaire qui peut être observé lors de la mise en route

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Act_PV	BOOL	Off (0)	Config	Bloc	Détection	Off(0) On(1)
Act_Status	BOOL	NOGO (0)	Config	Bloc	Détection	NOGO(0) Go(1)
Debounce_T	TIME	0	Oper	Oper	Lim. haute Lim. basse	1d_3h_46m_0
Hardware_PV	BOOL	Off (0)	Oper	Bloc	Détection	Off(0) On(1)
IO_Address	IO_ADDRESS		Config	Config		
Process_Val	BOOL	Off (0)	Oper	Bloc	Détection	Off(0) On(1)
Status	BOOL	NOGO (0)	Oper	Bloc	Détection	NOGO(0) Go(1)
Status_Info	ENUM	Reset (0)	Oper	Bloc	Détection	Cf. Description des paramètres
Test_Enable	BOOL	Off (0)	Config	Config	Détection	Off(0) On(1)
Test_PV	BOOL	Off (0)	Config	Config	Détection	Off(0) On(1)
Test_Status	BOOL	NOGO (0)	Config	Config	Détection	NOGO(0) Go(1)

Tableau 6-2 Attributs des paramètres Debounce_In

BLOC FONCTION ANALOG_IN

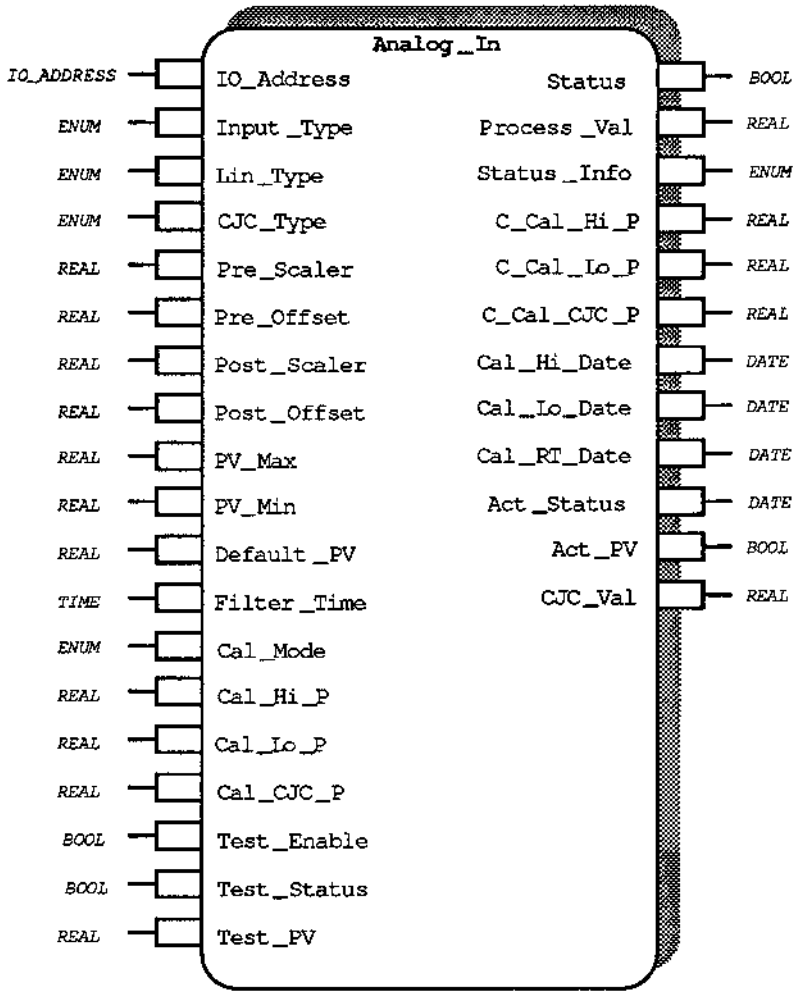


Figure 6-3 Diagramme du bloc fonction Analog_In

Description fonctionnelle

Le bloc fonction Analog_In assure l'interface avec le module matériel d'entrée analogique. La linéarisation des entrées et le filtrage passe bas sont offerts en même temps que la mise à l'échelle et le décalage de la valeur mesurée. Il existe aussi des fonctions de test qui permettent de remplacer la valeur physique mesurée et l'état par des valeurs de test.

Le bloc fonction analogique est utilisé avec les modules matériels à 2 et 4 canaux.

Attributs du bloc fonction

Type : 16 32

Classe : ..... INPUTS

Tâche : ..... Task_2

Liste récapitulative : ..... Process_Val, Act_PV, Status, Status_Info

Besoins de capacité mémoire : 126 octets

Durée d'exécution : ..... 175 µ Secs

Description des paramètres

Les paramètres qui sont repérés par un '*' **doivent** être configurés avant le lancement du programme. Tous les autres paramètres sont facultatifs.

IO_Address (IOA)

IO_Address associe la déclaration du bloc fonction aux entrées physiques sur le module matériel auxquelles elle se rapporte. Sa valeur est définie automatiquement lorsque la déclaration de bloc fonction est effectuée. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack où réside le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro du canal dans le module.

Par exemple, 1:02:03 signifie que la déclaration du bloc fonction se rapporte au troisième canal d'un module situé dans le deuxième emplacement du premier rack du système PC3000.

Input_Type (IT) *

La plage d'entrées d'un canal d'entrée analogique se configure à l'aide d'un paramètre (Input_Type). La plage d'entrées est considérée comme étant la plage de service normale maximale de l'entrée. Dans la pratique, il y a environ 20 à 25 % de marge en plus de la plage maximale, après quoi le canal d'entrée déclarera un état de dépassement matériel de plage.

Ce paramètre n'est pas automatiquement lié au type de thermocouple ou à la gamme maximale et minimale du capteur ; ces valeurs doivent être configurées par sélection indépendante du type de linéarisation (Lin_Type), de la température ou de l'entrée maximale attendue (PV_Max) et de la valeur minimale d'entrée (PV_Min).

Configuration de plage à deux canaux

Le module à deux canaux possède une entrée universelle, ce qui permet de mélanger différents types d'entrées sur la même carte.

Les configurations sont les suivantes :

'Ancienne' gamme	Gamme	Gamme matérielle
mV50	Range_1	entrées 0 à 50mV linéaires ou tc
mV100	Range_2	entrées 0 à 100mV linéaires ou tc
mV50RT	Range_3	Sonde à résistance deux fils 0 à 500 Ohms
mV100RT	Range_4	Sonde à résistance trois fils 0 à 500 Ohms
V0_5	Range_5	0 à 5V linéaire
V0_10	Range_6	0 à 10V linéaire

Configuration de plage à quatre canaux

Les modules à 4 canaux existent en 4 versions différentes. La configuration dépend du type de module :

Version d'entrée millivolt quatre canaux :

PC3000/AI/VERSION2/MV4

Gamme	Gamme matérielle
Range_1	-10 à 10mV
Range_2	-10 à 20mV
Range_3	-10 à 50mV
Range_4	-10 à 100mV
Range_5	s.o.
Range_6	s.o.

Version d'entrée milliampère quatre canaux : PC3000/AI/VERSION2/MA4

On suppose que le module est équipé de résistances 5 Ohms.

Gamme	Gamme matérielle
Range_1	0 à 2mA (inutilisé)
Range_2	0 à 4mA (inutilisé)
Range_3	0 à 10mA
Range_4	0 à 20mA
Range_5	s.o.
Range_6	s.o.

Version d'entrée tension quatre canaux : PC3000/AI/VERSION2/V4

Noter la gamme d'entrée bipolaire

Gamme	Gamme matérielle
Range_1	-1 à 1V
Range_2	-2 à 2V
Range_3	-5 à 5V
Range_4	-10 à 10V
Range_5	s.o.
Range_6	s.o.

Version d'entrée sonde à résistance quatre canaux : PC3000/AI/VERSION2/RT4

Gamme	Gamme matérielle
Range_1	0 à 50 Ohms
Range_2	0 à 100 Ohms
Range_3	0 à 250 Ohms
Range_4	0 à 500 Ohms
Range_5	s.o.
Range_6	s.o.

Lin_Type (LT) *

Lin_Type définit le type de linéarisation d'entrée utilisée par le bloc fonction. Le paramètre est un type énuméré qui peut prendre les valeurs indiquées dans le tableau suivant :

Valeur	Mnémonique	Type de linéarisation	Gamme
0	Linéaire	Néant	
1	J	Thermocouple Fe / Const	-210 à 1200 °C
2	L	Thermocouple Fe / Const (DIN)	-21 à 600 °C
3	K	Thermocouple NiCr / NiAl	-265 à 1372 °C
4	T	Thermocouple Cu / Const	-270 à 400 °C
5	R	Thermocouple Pt13%Rh / Pt	-50 à 1767 °C
6	S	Thermocouple Pt10%Rh / Pt	-50 à 1767 °C
7	B	Thermocouple Pt30%Rh / Pt6%Rh	0 à 1820 °C
8	E	Thermocouple NiCr / Const	-270 à 1000 °C
9	Pt_100	Sonde Pt 100Ω resistance	-200 à 800 °C
10	G1	Thermocouple W / W26%Re	0 à 2300 °C
11	WRe5_26	Thermocouple W5%Re / W26%Re	0 à 2300 °C
12	PR10_40	Thermocouple Pt10%Rh / Pt40%Rh	200 à 1800 °C
13	C	Thermocouple W5%Re / W26%Re	0 à 2300 °C
14	PR20_40	Thermocouple Pt20%Rh / Pt40%Rh	0 à 2000 °C
15	Plat_II	Thermocouple Platinel II	0 à 1200 °C

Tableau 6-3 Linéarisation d'entrée

Valeur	Mnémonique	Type de linéarisation	Gamme
16	G2	Thermocouple W / W26%Re	0 à 2200 °C
17	NC_NM	Thermocouple Ni / Ni18%Mo	0 à 1100 °C
18	D	Thermocouple W3%Re / W25%Re	0 à 2400 °C
19	WRe26_5	Thermocouple WRe5% / WRe26%	0 à 2000 °C
20	Nic_Nis	Thermocouple Nicrosil / Nisil	0 à 1300 °C
21	Q004	Pyromètre Q004	700 à 1600 °C
22	Q003	Pyromètre Q003	600 à 1500 °C
23	R_026	Pyromètre R026	0 à 500 °C
24	IVDI	Pyromètre IVDI	1000 à 2500 °C
25	DT1	Pyromètre DT1	750 à 2500 °C
26	DT1_10	Pyromètre DT1 / 10	1000 à 3000 °C
27	R_023	Pyromètre R023	700 à 1700 °C
28	FP_GP10	Pyromètre FP / GP 10	450 à 900 °C
29	FP_GP11	Pyromètre FP / GP 11	600 à 1300 °C
30	FP_GP12	Pyromètre FP / GP	750 à 1850 °C
31	FP_GP20	Pyromètre FP / GP 20	300 à 750 °C
32	FP_GP21	Pyromètre FP / GP 21	500 à 1100 C°
33	SQRT	Fonction racine carrée	

Tableau 6-3 Linéarisation d'entrée

CJC_Type (CT)

CJC_Type définit le type de compensation de soudure froide utilisée par le bloc fonction. Il est nécessaire pour compenser l'effet de Seebeck qui se produit à la liaison entre le câble du thermocouple et l'appareil. La compensation de soudure froide est uniquement appelée par le bloc fonction lorsqu'une linéarisation de thermocouple a été sélectionnée sur le paramètre Lin_Type. CJC_Type peut être positionné sur une option parmi trois :

Interne (0) :

Doit être utilisé lorsque le thermocouple est relié directement au module d'entrée analogique PC3000.

Ext_0C (1) :

Doit être utilisé lorsque le thermocouple est relié à une soudure froide externe fixée à 0°C. L'algorithme Ext_0C (1) effectue une compensation pour une température de soudure froide de 0°C.

Ext_50C (2) :

Doit être utilisé lorsque le thermocouple est relié à une soudure froide externe fixée à 50°C. L'algorithme Ext_50C (1) effectue une compensation pour une température de soudure froide de 50°C.

Sélection de la plage de valeurs de process *

Configurée par les paramètres PV_Max et PV_Min. PV_Max représente la valeur maximale que peuvent prendre les canaux Process_Value pour que le status du module repasse de Go (1) à NOGO (0). PV_Min est la valeur minimale de Process_Value.

Les valeurs de PV_Max et PV_Min doivent être configurées en prenant en compte les valeurs de changement d'échelle et de décalage pré et post linéarisation qui ont pu être configurées. A titre d'exemple, une entrée 0 à 5V devant fournir une sortie linéaire située dans la plage 0 à 100 % sera configurée de la manière suivante :

Paramètre	Valeur
Input_Type	Range_5
Pre_Scaler	20
Pre_Offset	0
PV_Max	110% (marge 10 %)
PV_Min	-10% (marge 10 %)

Les outils de changement d'échelle et les décalages font l'objet d'une explication dans une section ultérieure.

N.B. :

PV_Max et PV_Min doivent être configurés explicitement. Ils ne sont pas liés au tableau de linéarisation sélectionné à partir de Lin_Type.

PV_Max et PV_Min doivent être configurés dans les limites du tableau (cf. la spécification technique) ou dans la plage maximale de fonctionnement.

Si les limites définies sont dépassées, le paramètre Status_Info en indique la cause : soit Over_R(8) si l'entrée est supérieure à la limite maximale définie par PV_Max ou à la limite d'entrée matérielle soit Under_R(9) si l'entrée est inférieure à la limite minimale définie par PV_Min ou à la limite d'entrée matérielle.

PV_Max (PMX) *

PV_Max sert à définir la limite supérieure de Process_Val. Cette limite est appelée une fois que le changement d'échelle et le décalage pré et post entrée ont été effectués.

Si Process_Val est supérieure à PV_Max, Status sera positionné sur NOGO (0), Staus_Info sera positionné sur Over_R (8) et Process_Val sera positionné sur Default_PV.

PV_Min (PMN) *

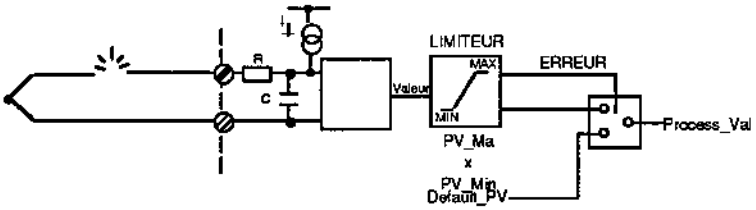
PV_Min sert à définir la limite inférieure de Process_Val. Cette limite est appelée une fois que le changement d'échelle et le décalage pré et post entrée ont été effectués.

Si Process_Val est inférieur à PV_Min, Status sera positionné sur NOGO (0), Staus_Info sera positionné sur Under_R (9) et Process_Val sera positionné sur Default_PV.

Rupture capteur *

Il est possible d'utiliser le paramètre Default_PV en association avec PV_Max et PV_Min pour fixer une valeur 'sûre' à Process_Value lorsque les limites PV définies sont franchies.

A titre indicatif, il est courant de positionner Default_PV sur la valeur de PV_Max-1, ce qui garantit que, en cas d'interruption du capteur, PV sera forcé à une valeur supérieure et la puissance de sortie par l'intermédiaire du bloc fonction PID sera désactivée.



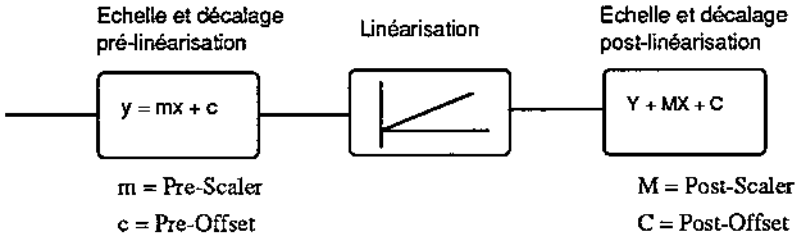
Default_PV (DPV) *

Default_PV sert à piloter Process_Val lorsque le status est NOGO (0).

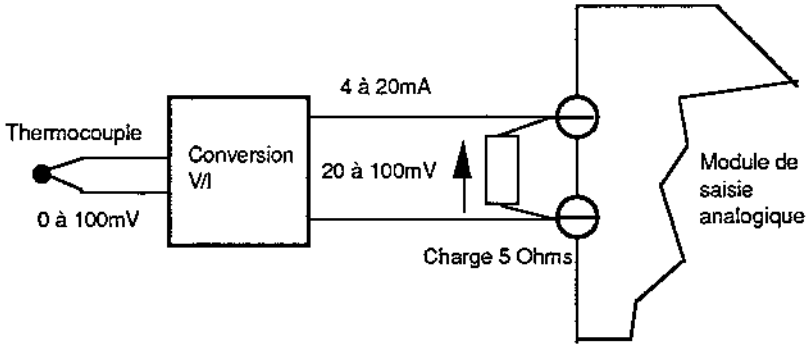
Outil de changement d'échelle et décalage pré-linéarisation

Deux paramètres, Pre_Scaler et Pre_Offset, offrent un moyen de changer l'échelle du signal d'entrée avant d'effectuer la linéarisation.

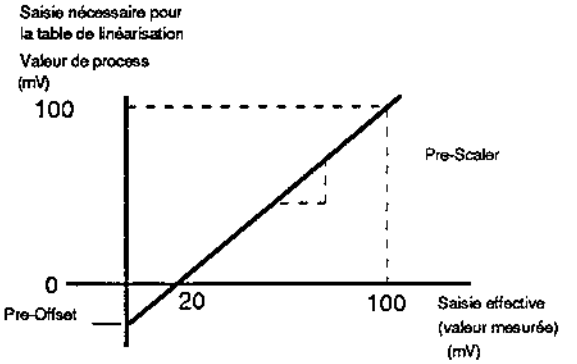
L'outil de changement d'échelle et le décalage pré-linéarisation agissent de la manière suivante :



A titre d'exemple, prenons un signal d'entrée 4 à 20mA provenant d'un conditionneur de signaux de thermocouple déporté. Ce thermocouple produit un signal d'entrée 0 à 100mV qui est ensuite fourni au PC3000 sous la forme 4 à 20mA.



L'entrée PC3000 nécessite qu'une résistance 5 Ohms convertisse cette entrée en 20 à 100mV. Le tableau de linéarisation convertit les millivolts en degrés Celsius. L'outil de changement d'échelle et le décalage doivent donc être configurés de la manière suivante :

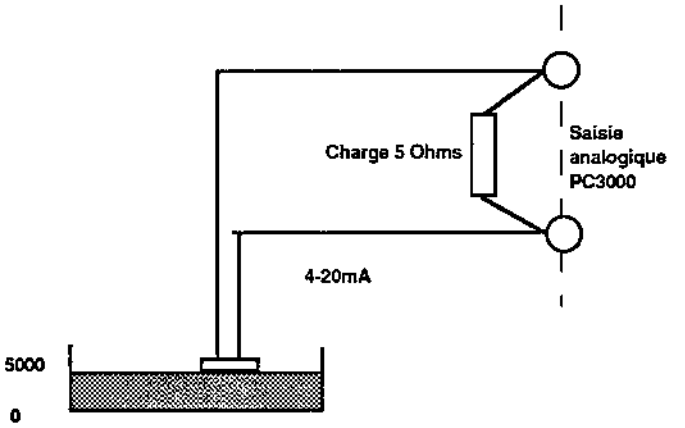


pre-scale = 100/(100-20) = 1,25

pre-offset = 100-(1.25*100 = -25

(à partir de y = mx + c)

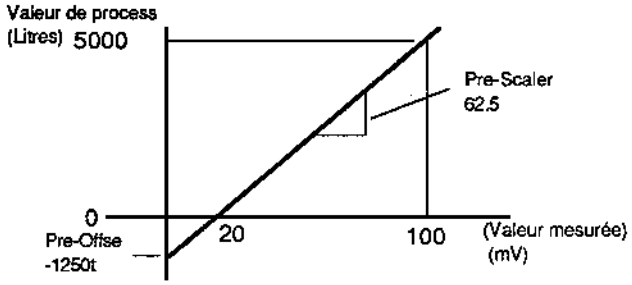
Autre exemple :



La cuve se remplit à 500 litres

L'émetteur délivre 4 à 20mA

Le signal d'entrée envoyé au PC3000 est compris entre 20 et 100mV en raison de la charge 5 Ohms



Pre-Scaler = 5000/(100-20) = 62.5

Pre-Offset = 5000/ - (62.5 * 100) = -1250

Règle:

Pre-Scaler = (MaxPV - MinPV) / (MaxMV - MinMV)

Pre_Offset = (MaxPV) - (Pre_Scaler * MaxMV)

où : MaxPV est la valeur de process maximale, c'est-à-dire 5000 litres

MinPV est la valeur de process minimale, c'est-à-dire 0 litre

MaxMV est la valeur maximale mesurée

(sur l'entrée PC3000) c'est-à-dire 100mV

MinMV est la valeur minimale mesurée

(sur l'entrée PC3000) c'est-à-dire 20mV

Pre_Scaler (PRS)

Pre_Scaler est un outil de changement d'échelle qui est appliqué au signal d'entrée avant l'appel de l'algorithme de linéarisation.

Pre_Offset (PRO)

Pre_Offset est un décalage qui est appliqué au signal d'entrée avant l'appel de l'algorithme de linéarisation.

Outil de changement d'échelle et décalage post -linéarisation

Deux paramètres, Post_Scaler et Post_Offset, offrent un moyen de changer l'échelle du signal d'entrée une fois que la linéarisation a été effectuée.

L'outil de changement d'échelle et le décalage post-linéarisation agissent de la manière indiquée sur le diagramme de la page 6-23

A titre d'exemple, prenons un signal de thermocouple qui produit une valeur de sortie de 873,5°C. L'outil de changement d'échelle et le décalage post-linéarisation peuvent servir à fournir une valeur en °F. L'outil de changement d'échelle et le décalage post-linéarisation doivent par conséquent être configurés de la manière suivante :

post-scaler = 9/5

post-offset = 32

à partir de $y = mx + c$

Post_Scaler (POS)

Post_Scaler est un outil de changement d'échelle qui est appliqué au signal d'entrée après l'appel de l'algorithme de linéarisation.

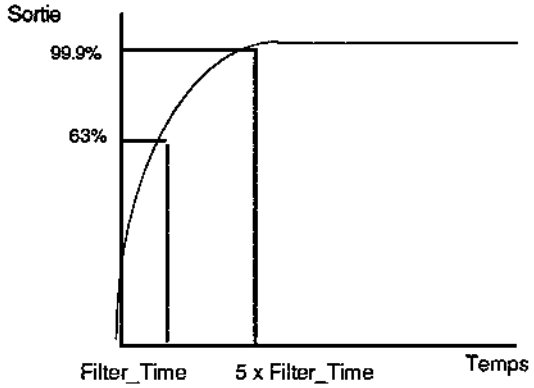
Post_Offset (POO)

Post_Offset est un décalage qui est appliqué au signal d'entrée après l'appel de l'algorithme de linéarisation.

Filter_Time (FT)

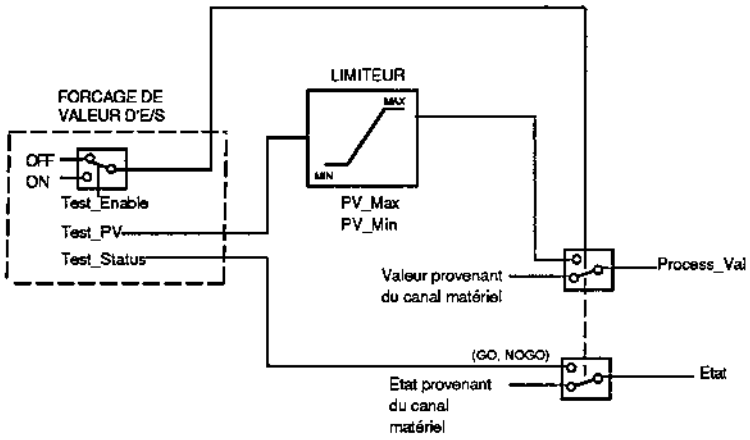
Le module d'entrée comprend un filtre passe bas programmable. La constante de temps de ce filtre peut être fixée à l'aide de ce paramètre. Par défaut, le temps du filtre est positionné sur zéro, ce qui désactive l'effet du filtre.

Le filtre peut servir à diminuer l'effet du bruit aléatoire sur la valeur mesurée. On peut obtenir une augmentation de la résolution effective au détriment du temps de réponse de l'entrée en faisant la moyenne de l'entrée sur une longue période de temps.



Forçage de la valeur d'E/S

Ces paramètres offrent la possibilité de découpler le programme de la valeur réelle de l'E/S. Cela permet de 'forcer' les valeurs sur lesquelles agit le programme PC3000 indépendamment de le status réel. Cela permet de tester facilement les mécanismes de défaillance ou de résoudre les problèmes de mise en service comme 'limiteur de course pas encore installé'.



Test_Enable (TEN)

Doit être positionné sur On (1) pour forcer Process_Value. Une fois positionnées, les valeurs de Status et de Process_Value sont configurées par les paramètres suivants.

Test_Status (TST)

Lorsque le canal est placé en mode test, ce paramètre peut servir à contrôler directement le paramètre Status du canal.

> **N.B. :** la valeur du paramètre Status_Info indique toujours OK dans ce mode de fonctionnement.

Test_PV (TPV)

Lorsque le canal est placé en mode test, ce paramètre peut servir à contrôler directement le paramètre Process_Value du canal.

Status (ST)

Lorsque Test_Status est sur Off (0), le paramètre Status montre le status du canal matériel auquel se rapporte le bloc fonction. Si Test_Status est sur On (1), Status lira directement sa valeur dans Test_Status.

L'état peut être NOGO (0) ou Go (1)

Process_Val (PV)

En service normal, lorsque Test_Enable est sur Off (0), Process_Val montre l'entrée dans le canal d'entrée analogique adressé par le bloc fonction, une fois que la linéarisation, le changement d'échelle et le décalage ont été effectués. Si Test_Enable est sur On (1), Process_Val lit directement sa valeur dans Test_PV.

Il s'agit de la valeur utilisée par la stratégie de contrôle.

Status_Info (STI)

Status_Info est un paramètre de diagnostic qui sert à montrer la valeur de Status. Il peut avoir neuf valeurs :

Reset (0) :	le programme utilisateur ne tourne pas.
Ok (1) :	le canal fonctionne normalement.
No_Mod (2) :	il n'y a aucun module dans l'emplacement matériel adressé par le bloc fonction.
Wrg_Mod (3) :	un type de module incorrect a été installé dans l'emplacement adressé par le bloc fonction
Init (4) :	le module ou le canal est en cours d'initialisation
_ (5) :	absence de fonction Status_Info
No_Cal (6) :	la gamme d'entrée qui a été sélectionnée dans Input_Type n'a pas été étalonnée

Calib (7) : le module est actuellement en mode étalonnage

Over_R (8) : Process_Val est supérieure à PV_Max ou la gamme d'entrée matérielle a été dépassée.

Under_R (9) : Process_Val est inférieure à PV_Min ou la gamme d'entrée matérielle a été dépassée.

Act_PV (APV) et Act_Status (AST)

Il s'agit des sorties du bloc fonction. Act_PV indique la valeur restituée par le canal. En fonctionnement normal, la valeur Act_PV coïncide avec celle de Process_Value. Act_PV inclut l'effet de la linéarisation et de la configuration de tous les outils de changement d'échelle et des décalages. De même, Act_Status coïncide normalement avec l'état indiqué par Status. Toutefois, les situations suivantes peuvent se produire :

(1) La valeur produite par le bloc fonction est située en dehors des limites définies par PV_Max et PV_Min.

PV_Max représente la valeur maximale que peuvent prendre les canaux Process_Value sans que l'e Status des modules repasse de Go(1) à NOGO(0). PV_Min est la valeur minimale de Process_Value.

A titre d'exemple, les valeurs de paramètres suivants ont été configurées :

Input_Type ..... plage 1 (50mV)

Lin_Type ..... J

PV_Max ..... 500°C

PV_Min ..... -10°C

Tous les autres paramètres prennent les valeurs par défaut. Si le signal d'entrée millivolt provoque le dépassement de 500°C par le bloc, les paramètres prennent les valeurs suivantes :

Act_PV 634.3

Act_Status ..... Go

Process_Value ..... Default_PV

Status ..... NOGO

Si le thermocouple avait eu une défaillance, Act_Status serait aussi positionné sur NOGO(0) et ACT_PV serait bloqué sur la dernière valeur avant que le dépassement de gamme se produise. Le paramètre Status_Info enregistre la cause de l'état NOGO(0).

(2) Utilisation du mode Test

Si Process_Value est forcée à l'aide du mode test, les données produites par le canal sont toujours indiquées à l'aide d'Act_PV et d'Act_Status. Cf. la section relative au 'Forçage des E/S' pour avoir des détails supplémentaires.

3) En cours d'étalonnage

Si le canal est placé en mode étalonnage par positionnement de son mode sur I/P_Hi, les données produites par le canal sont indiquées à l'aide d'Act_PV et Act_Status. Cf. la section relative à l''étalonnage' pour avoir des détails supplémentaires.

Au cours de l'étalonnage, les paramètres prennent les valeurs suivantes :

Act_PV ..... Entrée d'étalonnage

Act_Status ..... NOGO

Process_Value ..... Default_PV

Status ..... NOGO

Status_Info ..... Cal (ibration)

Act_PV indique maintenant la valeur 'brute' produite par le canal matériel, étant donné que l'effet de la linéarisation et les outils de changement d'échelle ou les décalages qui peuvent avoir été définis sont désactivés au cours de l'étalonnage matériel.

Paramètres d'étalonnage

Les paramètres énumérés ci-dessous sont utilisés pour l'étanlonnage du module matériel.

Cal_Mode	C_Cal_CJC_P
Cal_Hi_P	Cal_Hi_Date
Cal_Lo_P	Cal_Lo_Date
Cal_CJC_P	Cal_CJC_Date
C_Cal_Hi_P	Cal_RT_Date
C_Cal_Lo_P	CJC_Val

Cf. le manuel 'Installation du PC3000' HA022231 pour avoir des détails complets sur la procédure d'étalonnage.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Act_PV	REAL	0.0	Config	Bloc	Lim. haute Lim. basse	PV_Max PV_Min
Act_Status	BOOL	NOGO (0)	Config	Bloc	Détection	NOGO (0) Go (1)
C_Cal_CJC_P	REAL	0	Config	Bloc	Lim. haute Lim. basse	1000°C -273°C
C_Cal_Hi_P	REAL	0.0	Config	Bloc	Lim. haute Lim. basse	100 000 -10 000
C_Cal_Lo_P	REAL	0.0	Config	Bloc	Lim. haute Lim. basse	100 000 -10 000
Cal_CJC_Date	DATE	00:00:00 1 jan 1970	Config	Bloc	Lim. haute Lim. basse	19 jan 2038 00:00:00 1 jan 1970 00:00:00 N.B. 1
Cal_CJC_P	REAL	0.0	Config	Config	Lim. haute Lim. basse	1000 °C -1000 °C
Cal_Hi_Date	DATE	00:00:00 1 jan 1970	Config	Bloc	Lim. haute Lim. basse	19 jan 2038 00:00:00 1 jan 1970 00:00:00 N.B. 1
Cal_Hi_P	REAL	0.0	Config	Config	Lim. haute Lim. basse	100,000 Cal_Lo_P
Cal_Lo_Date	DATE	00:00:00 1 jan 1970	Config	Bloc	Lim. haute Lim. basse	19 jan 2038 00:00:00 1 jan 1970 00:00:00 N.B. 1
Cal_Lo_P	REAL	0.0	Config	Config	Lim. haute Lim. basse	Cal_Hi_P -10 000
Cal_Mode	ENUM	Run (0)	Config	Config	Détection	Run (0) Save (1) I/P_Hi (2) I/P_Lo (3) CJC (4)

Tableau 6-4 Attributs des paramètres Analog_In (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Cal_RT_Date	**DATE**	1 jan 1970 00:00:00	Config	Bloc	Lim. haute Lim. basse	19 jan 2038 00:00:00 1 jan 1970 00:00:00 N.B. 1
CJC_Type	**ENUM**	Intern (0)	Config	Config	Détection	Cf. Description des paramètres
CJC_Val	**REAL**	0.0	Config	Bloc	Lim. haute Lim. basse	10 000°C - 273°C
Default_PV	**REAL**	0.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Filter_Time	**TIME**	0	Super	Super	Lim. haute Lim. basse	1d_3h_46m_40s_000ms 0
Input_Type	**ENUM**	Range_1 (0)	Config	Config	Détection	Cf. Description des paramètres
IO_Address	**IO_Address**		Super	Super		
Lin_Type	**ENUM**	None (0)	Config	Config	Détection	Cf. Description des paramètres
Post_Offset	**REAL**	0.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Post_Scaler	**REAL**	1.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Pre_Offset	**REAL**	0.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Pre_Scaler	**REAL**	1.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Process_Val	**REAL**	0.0	Oper	Bloc	Lim. haute Lim. basse	PV_Max PV_Min
PV_Max	**REAL**	0.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
PV_Min	**REAL**	0.0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Status	**BOOL**	NOGO (0)	Oper	Bloc	Détection	NOGO (0) Go (1)

Tableau 6-4 Attributs des paramètres Analog_In (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Status_Info	ENUM	Reset (0)	Oper	Bloc	Détection	Cf. Description des paramètres
Test_Enable	BOOL	Off (0)	Config	Config	Détection	Off (0) On (1)
Test_PV	REAL	0.0	Config	Config	Lim. haute Lim. basse	PV_Max PV_Min
Test_Status	BOOL	NOGO (0)	Config	Config	Détection	NOGO (0) Go (1)

Tableau 6-4 Attributs des paramètres Analog_In (suite)

N.B. 1 : les dates d'étalonnage ne sont pas prises en compe pour l'instant.

BLOC FONCTION xFAST_AN_I (ne convient pas pour les nouvelles versions)

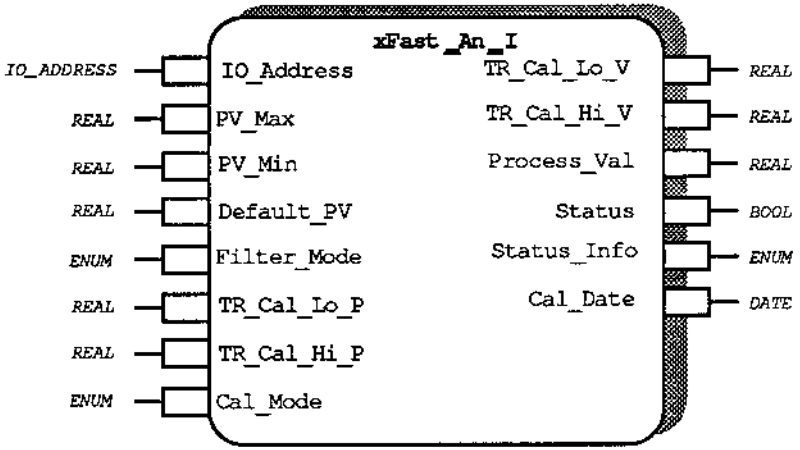


Figure 6-4 Diagramme du bloc fonction xFast_An_I

Description fonctionnelle

Le bloc fonction xFast_An_I offre l'interface avec n'importe quel matériel qui peut prendre en charge un canal d'entrée analogique rapide, par exemple, le module d'E/S analogique rapide (Fast_An_8).

Il est adapté aux situations dans lesquelles une entrée analogique doit être traitée à des fréquences de balayage inférieures à 100 ms.

Il faut noter que le bloc x_I_Fast_An_I doit être pris en considération afin de diminuer la marge arithmétique à virgule flottante pour les fréquences de balayage inférieures à 50 ms.

Attributs du bloc fonction

Type : 10 30

Classe : ..... INPUTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Process_Val, Status, Status_Info

Besoins de capacité mémoire : 86 octets

Description des paramètres

Paramètres d'étalonnage

Les paramètres énumérés ci-après servent à l'étalonnage du module matériel.

TR_Cal_Lo_P
TR_Cal_Hi_P
Cal_Mode
TR_Cal_Lo_V
TR_Cal_Hi_V
Cal_Date

IO_Address (IOA)

IO_Address associe la déclaration de bloc fonction aux entrées physiques sur le module matériel auxquelles elle se rapporte. Sa valeur est définie automatiquement lorsque la déclaration de bloc fonction est effectuée. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack dans lequel réside le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro du canal dans le module. Par exemple, 1:02:03 signifie que la déclaration du bloc fonction se rapporte au troisième canal d'un module situé dans le deuxième emplacement du premier rack du système PC3000.

PV_Max (PMX)

PV_Max sert à définir la limite supérieure de Process_Val. Si l'entrée provenant du matériel est supérieure à cette valeur, Status sera positionné sur NOGO (0), Status_Info sur Over_R (6) et Process_Val sera positionnée sur la valeur de Default_PV.

PV_Min (PMN)

PV_Min est l'entrée inférieure de la valeur d'entrée. Si l'entrée provenant du matériel est inférieure à PV_Min, Status sera positionné sur NOGO (0), Status_Info sur Under_R (7) et Process_Val sera positionnée sur la valeur de Default_PV.

Default_PV (DPV)

Default_PV sert à piloter Process_Val lorsque Status est sur NOGO (0).

Filter_Mode (FM)

Filter_Mode définit la fréquence d'angle du filtre passe-bas bipole qui est employé par le canal matériel. Les options disponibles comprennent 16 Hz (0), 32 Hz (1), 80 Hz (2) et 160 Hz (3).

Process_Val (PV)

Process_Val montre l'entrée de la valeur analogique dans le canal matériel adressé par le bloc fonction.

Status (ST)

Status montre l'état du canal matériel adressé par le bloc fonction.

Status_Info (STI)

Status_Info est un paramètre de diagnostic qui explique la valeur de Status. Il peut avoir huit valeurs :

Reset (0) :	le programme utilisateur ne tourne pas
Ok (1) :	le Status est sur Go (1)
No_Mod (2) :	il n'y a aucun module dans l'emplacement matériel adressé par le bloc fonction.
Wrg_Mod (3) :	un type de module incorrect a été placé dans l'emplacement adressé par le bloc fonction
Calib (4) :	le module est actuellement en mode étalonnage.
Init (5) :	le module ou le canal est actuellement en cours d'initialisation.
Over_R (6) :	l'entrée de la valeur analogique dans le canal matériel est supérieure à PV_Max.
Under_R (7) :	l'entrée de la valeur analogique dans le canal matériel est inférieure à PV_Min.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Propres au type	
Cal_Date	**DATE**		Config	Config		
Cal_Mode	**ENUM**	Run (0)	Config	Config	Détection	Run (0) Save (1) Hcal_Hi (2) Hcal_Lo (3) Tcal_Hi (4) Tcal_Lo (5) Dflt_TR (6)
Default_PV	**REAL**	0	Config	Config	Lim. haute Lim. basse	110 -110
Filter_Mode	**ENUM**	16Hz (0)	Oper	Oper	Valeurs énumérées	16Hz (0) 32 Hz(1) 80Hz (2) 160Hz (3)
IO_Address	**IO_ADDRESS**		Config	Config		
Process_Val	**REAL**	0 %	Oper	Oper	Lim. haute Lim. basse	Fixé par PV_Max Fixé par PV_Min
PV_Max	**REAL**	110	Config	Config	Lim. haute Lim. basse	110 -110
PV_Min	**REAL**	0	Config	Config	Lim. haute Lim. basse	110 -110
Status	**BOOL**	Go (1)	Oper	Oper	Détection	NOGO (0) Go (1)
Status_Info	**ENUM**	Ok (1)	Oper	Oper	Détection	Cf. Description des paramètres
TR_Cal_Hi_P	**REAL**	100 %	Config	Config	Lim. haute Lim. basse	110 TR_Cal_Lo_P
TR_Cal_Hi_V	**REAL**	100 %	Config	Config	Lim. haute Lim. basse	110 -110
TR_Cal_Lo_P	**REAL**	0 %	Config	Config	Lim. haute Lim. basse	TR_Cal_Hi_P -110
TR_Cal_Lo_V	**REAL**	0 %	Config	Config	Lim. haute Lim. basse	110 -110

Tableau 6-5 Attributs des paramètres xFast_An_I

BLOC FONCTION XI_FAST_AN_I (ne convient pas pour les nouvelles versions)

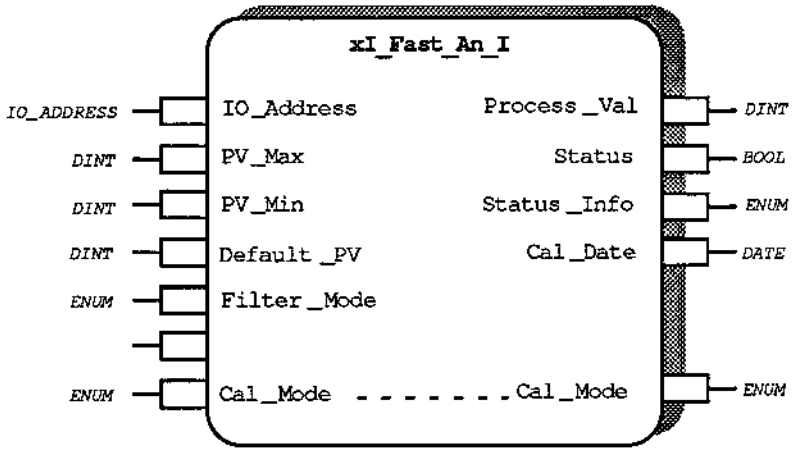


Figure 6-5 Diagramme du bloc fonction xI_Fast_An_I

Description fonctionnelle

Le bloc fonction xI_Fast_An_I offre l'interface avec n'importe quel matériel qui peut prendre en charge un canal d'entrée analogique rapide, par exemple le module d'E/S analogique rapide (FAST_AN_8). Le bloc fonction xI_Fast_An_I est identique au bloc fonction Fast_An_In, à la différence près qu'il s'agit d'un entier pur et qu'il ne possède aucune fonction d'étalonnage de transducteur. Il représente une marge de performances plus faible que Fast_An In et Analogue_In et doit être utilisé lorsqu'une entrée analogique rapide est nécessaire, c'est-à-dire des durées de balayage inférieures à 100ms. La sortie entière est comprise entre -2000 et +2000 pour représenter une plage d'entrée de -10 Volts à +10 Volts.

Attributs du bloc fonction

Type : 10 38

Classe : ..... INPUTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Process_Val, Status, Status_Info

Besoins de capacité mémoire : 86 octets

Description des paramètres

Paramètres d'étalonnage

Les paramètres énumérés ci-dessous servent à l'étalonnage du module matériel.

Cal_Mode

Cal_Date

IO_Address (IOA)

IO_Address associe la déclaration du bloc fonction aux entrées physiques sur le module auxquelles elle se rapporte. Sa valeur est définie automatiquement lorsque la déclaration du bloc fonction est effectuée. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack dans lequel réside le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro du canal dans le module. Par exemple, 1:02:03 signifie que la déclaration du bloc fonction se rapporte au troisième canal d'un module situé dans le deuxième emplacement du premier rack du système PC3000.

PV_Max (PMX)

PV_Max sert à définir la limite supérieure de Process_Val. Si l'entrée provenant du matériel est supérieure à cette valeur, Status sera positionné sur NOGO (0), Status_Info sur Over_R (6) et Process_Val sera positionné sur la valeur de Default_PV.

PV_Min (PMN)

PV_Min est l'entrée inférieure de la valeur d'entrée. Si l'entrée provenant du matériel est inférieure à PV_Min, Status sera positionné sur NOGO (0), Status_Info sur Under_R (7) et Process_Val sera positionné sur la valeur de Default_PV.

Default_PV (DPV)

Default_PV sert à piloter Process_Val lorsque Status est sur NOGO (0).

Filter_Mode (FM)

Filter_Mode définit la fréquence d'angle du filtre passe-bas bipole qui est employé par le canal matériel. Les options disponibles comprennent 16 Hz (0), 32 Hz (1), 80 Hz (2) et 160 Hz (3).

Process_Val (PV)

Process_Val montre l'entrée de la valeur analogique dans le canal matériel adressé par le bloc fonction.

Status (ST)

Status montre l'état du canal matériel adressé par le bloc fonction.

Status_Info (STI)

Status_Info est un paramètre de diagnostic qui montre la valeur de Status. Il peut avoir huit valeurs :

Reset (0) :	le programme utilisateur ne tourne pas
Ok (1) :	le Status est sur Go (1)
No_Mod (2) :	il n'y a aucun module dans l'emplacement matériel adressé par le bloc fonction.
Wrg_Mod (3) :	un type de module incorrect a été placé dans l'emplacement adressé par le bloc fonction
Calib (4) :	le module est actuellement en mode étalonnage.
Init (5) :	le module ou le canal est en cours d'initialisation.
Over_R (6) :	la valeur analogique entrée dans le canal matériel est supérieure à PV_Max.
Under_R (7) :	la valeur analogique entrée dans le canal matériel est inférieure à PV_Min.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Propres au type	
Cal_Date	DATE		Config	Config		
Cal_Mode	ENUM	Run (0)	Config	Config	Détection	Run (0) Save (1) Hcal_Hi (2) Hcal_Lo (3) Tcal_Hi (4) Tcal_Lo (5) Dflt_TR (6)
Default_PV	DINT	0	Config	Config	Lim. haute Lim. basse	2048 -2048
Filter_Mode	ENUM	16Hz (0)	Oper	Oper	Valeurs énumérées	16Hz (0) 32 Hz(1) 80Hz (2) 160Hz (3)
IO_Address	IO_ADDRESS		Config	Config		
Process_Val	DINT	0 %	Oper	Oper	Lim. haute Lim. basse	PV_Max PV_Min
PV_Max	DINT	110	Config	Config	Lim. haute Lim. basse	2048 -2048
PV_Min	DINT	0	Config	Config	Lim. haute Lim. basse	2048 -2048
Status	BOOL	Go (1)	Oper	Oper	Détection	NOGO (0) Go (1)
Status_Info	ENUM	Ok (1)	Oper	Oper	Détection	Cf. Description des paramètres

Tableau 6-6 Attributs des paramètres xl_Fast_I

BLOC FONCTION PI_SMPL_CTR

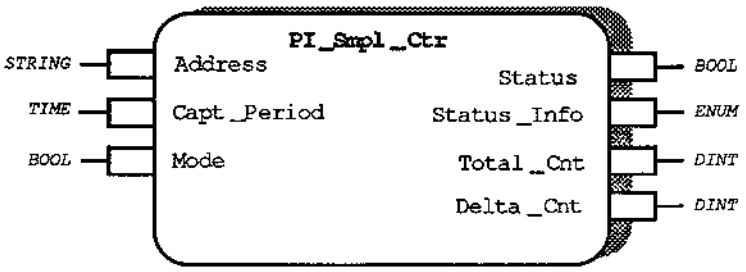


Figure 6-6 Bloc fonction PI_Smpl_Ctr

Description fonctionnelle

Le bloc fonction PI_Smpl_Ctr assure l'interface avec le module d'entrée d'impulsions (PIM) pour fournir un compteur d'entrées d'impulsions. Il s'agit d'un mécanisme de lecture d'un décompte à intervalles échantillonnés avec précision, les intervalles étant calculés par le PIM. Le bloc fonction sort le décompte total et la différence entre la dernière valeur saisie et la précédente valeur saisie.

Le bloc fonction PI_Smpl_Cntr reçoit les informations sur les impulsions par l'intermédiaire du bloc fonction PIM qui a été affecté au module matériel. PI_Smpl_Cntr fonctionne uniquement s'il a été déclaré en association avec un bloc fonction PIM et un PIM correctement configuré.

Attributs du bloc fonction

Type : 10 E2

Classe : ..... INPUTS

Default_Task: ..... Task_2

Liste récapitulative : ..... Total_Cnt, Delta_Cnt, Mode, Status

Besoins de capacité mémoire : 42 octets

Description des paramètres

Adresse (ADD)

Adresse associe la déclaration du bloc fonction aux entrées physiques sur le module matériel auxquelles elle se rapporte. Sa valeur doit être saisie manuellement par l'utilisateur. Elle prend la forme X:YY:ZZ, où X représente le numéro du rack où réside le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro du canal dans le module. Par exemple,

1:02:03 signifie que la déclaration du bloc fonction se rapporte au troisième canal d'un module qui se trouve dans le deuxième emplacement du premier rack du système PC3000. Le bloc fonction du canal fonctionne uniquement s'il y a un PIM dans l'emplacement spécifié et si un bloc fonction PIM a été défini avec l'adresse qui convient.

Capt_Period (CP)

Capt_Period définit la période d'échantillonnage pour les décomptes échantillonnés. Ce paramètre doit recevoir une valeur supérieure à la durée d'exécution du bloc fonction PIM. Si l'on fixe une valeur inférieure à la durée d'exécution du PIM, cela provoquera le passage du bloc à un état d'erreur.

Mode (M)

Mode sert à démarrer et à réinitialiser le compteur. Le passage de Mode sur Reset (0) provoque la remise à zéro du décompte. Le positionnement de Mode sur ON (1) provoque le démarrage du compteur.

Status (ST)

Status_Info est un paramètre de diagnostic qui sert à montrer la valeur de Status. Il peut avoir dix valeurs :

Reset (0) :	le programme utilisateur est en cours de réinitialisation
Ok (1) :	le Status est Go (1)
Mod_Err (2) :	le module est dans un état d'erreur. Le type d'erreur sera décrit dans le bloc fonction PIM associé.
Bad_Add (3) :	le paramètre Adresse n'est pas une adresse valable.
Dup_Add (4) :	cette adresse matérielle a été déjà affectée à un autre bloc fonction de canal.
Inv_Cfg (5) :	une erreur s'est produite dans la configuration du PIM. Se reporter au bloc fonction PIM pour avoir des informations supplémentaires sur la nature de l'erreur.

Total_Cnt (TC)

Total_Cnt sort le décompte total d'impulsions actuel provenant du module matériel.

Delta_Cnt (DC)

Delta_Cnt sort la différence entre le dernier décompte saisi et le précédent. décompte saisi

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Adresse	STRING		Oper	Oper		
Capt_Period	TIME	0	Oper	Oper	Lim. haute Lim. basse	23d_23h_59m 0
Delta_Cnt	DINT	0	Oper		Lim. haute Lim. basse	2 147 483 647 -2 147 483 647
Mode	BOOL	Reset (0)	Oper	Oper	Détection	Reset (0) ON(1)
Status	BOOL	Nogo (0)	Oper		Détection	Nogo (0) Go (1)
Status_Info	ENUM	Reset (0)	Oper		Détection	Reset (0) OK(1) Mod_Err(2) Bad_Add(3) Dup_Add(4) Inv_Cfg(5)
Total_Cnt	DINT	0	Oper		Lim. haute Lim. basse	2 147 483 647 -2 147 483 647

Tableau 6-7 PI_Smpl_Ctr Attributs des paramètres

Chapitre 7

SORTIES

Edition 1

Présentation

Vue d'ensemble

Des blocs fonctions de sortie sont automatiquement créés dans le cadre de la configuration matérielle du PC3000, c'est-à-dire la procédure de déclaration du type de module d'E/S qui se trouve à chacun des emplacements du rack. Chaque bloc fonction de sortie est "attaché" à une voie physique d'E/S. Une fois définis, ils peuvent être manipulés de la même façon que les autres types de blocs fonctions.

BLOC FONCTION Digital_Out

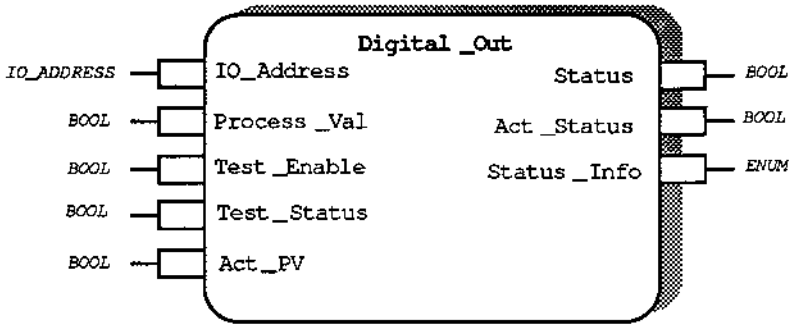


Figure 7-1BLoc fonction Digital_Out

Description fonctionnelle

Le bloc fonction Digital_Out assure l'interface de bloc fonction pour tout module matériel pouvant recevoir une sortie logique. Il dispose d'un paramètre d'entrée booléen qui définit l'état requis de la sortie logique physique associée.

Des possibilités de test sont prévues pour permettre de commander directement une sortie physique au moyen d'une valeur de test qui se substitue à la valeur normale de process. L'état du bloc peut également être forcé.

Attributs du bloc fonction

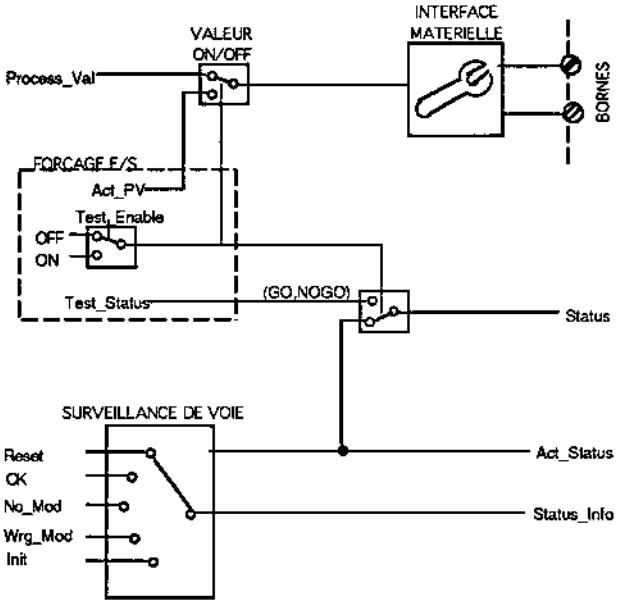
Type :..... 18 10

Classe :..... SORTIES

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Process_Val, Act_PV, Status, Status_Info

Mémoire requise :..... 12 octets



Description des paramètres

IO_Address (IOA)

Le paramètre IO_Address associe le bloc fonction concerné aux entrées physiques du module matériel auquel il se réfère. Sa valeur est définie automatiquement lorsque la déclaration du bloc fonction concerné est effectuée. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack dans lequel est placé le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro de la voie dans le module. Par exemple, 1:02:03 signifie que le bloc fonction concerné désigne la troisième voie d'un module placé dans le second emplacement du premier rack du système PC3000.

Process_Val (PV)

C'est le paramètre qui définit l'état de la voie matérielle dont l'adresse est indiquée par le bloc fonction.

> **Nota** : Lorsque Test_Enable est sur On (1), Process_Val n'est pas utilisé.

C'est l'entrée commandée par la stratégie de commande.

Test_Enable (TEN)

Test_Enable permet à l'utilisateur de basculer la sortie Process_Val à Act_PV. Si Test_Enable est sur Off (0), la sortie matérielle lira sa valeur dans Process_Val et Status indiquera l'état du module matériel. Si Test_Enable est sur On (1), la sortie matérielle lira sa valeur dans Act_PV et Status prendra la valeur indiquée par Test_Status.

Test_Status (TST)

La valeur de Test_Status est reportée dans Status lorsque Test_Enable est sur On (1). Si Test_Enable est sur Off (0), Test_Status n'est pas utilisé.

Act_PV (APV)

Le paramètre Act_PV est reporté dans la sortie matérielle lorsque Test_Enable est sur On (1). Si Test_Enable est sur Off (0), Act_PV n'est pas utilisé.

Status (ST)

Si Test_Status est sur Off (0), le paramètre Status indique l'état de la voie matérielle à laquelle se réfère le bloc fonction.

Si Test_Enable est sur On (1), Status prendra la valeur attribuée à Test_Status.

Act_Status (AST)

Le paramètre Act_Status indique toujours l'état de la voie matérielle. Si le matériel indique un défaut, Act_Status indiquera NOGO (0). Act_Status ne doit être utilisé que pour le diagnostic.

Status_Info (STI)

Status_Info est un paramètre de diagnostic utilisé pour préciser l'état de Status. Cinq états sont possibles:

Reset (0):

Le programme utilisateur n'est pas en cours d'exécution.

Ok (1):

La voie fonctionne normalement.

No_Mod (2):

Il n'y a pas de module à l'emplacement dont l'adresse est indiquée par le bloc fonction.

Wrg_Mod (3):

Un type de module incorrect a été mis à l'emplacement dont l'adresse est indiquée par le bloc fonction.

Init (4):

Le module ou la voie est en cours d'initialisation.

Attributs des paramètres

Nom	Type	Dém. à froid	Accès en lect.	Accès en écrit.	Informations propres au type	
Act_PV	BOOL	Off (0)	Config	Config	Détect.	Off (0) On (1)
Act_Status	BOOL	NOGO (0)	Config	Bloc	Détect.	NOGO (0) Go (1)
IO_Address	IO_ADDRESS		Config	Config		
Process_Val	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Status	BOOL	NOGO (0)	Oper	Bloc	Détect.	NOGO (0) Go (1)
Status_Info	ENUM	Reset (0)	Oper	Bloc	Détect.	Reset(0) Ok(1) No_Mod(2) Wrg_Mod(3) Init(4)
Test_Enable	BOOL	Off (0)	Config	Config	Détect.	Off (0) On (1)
Test_Status	BOOL	NOGO (0)	Config	Config	Détect.	NOGO (0) Go (1)

Tableau 7-1 Attributs des paramètres Digital_Out

BLOC FONCTION Analog_Out

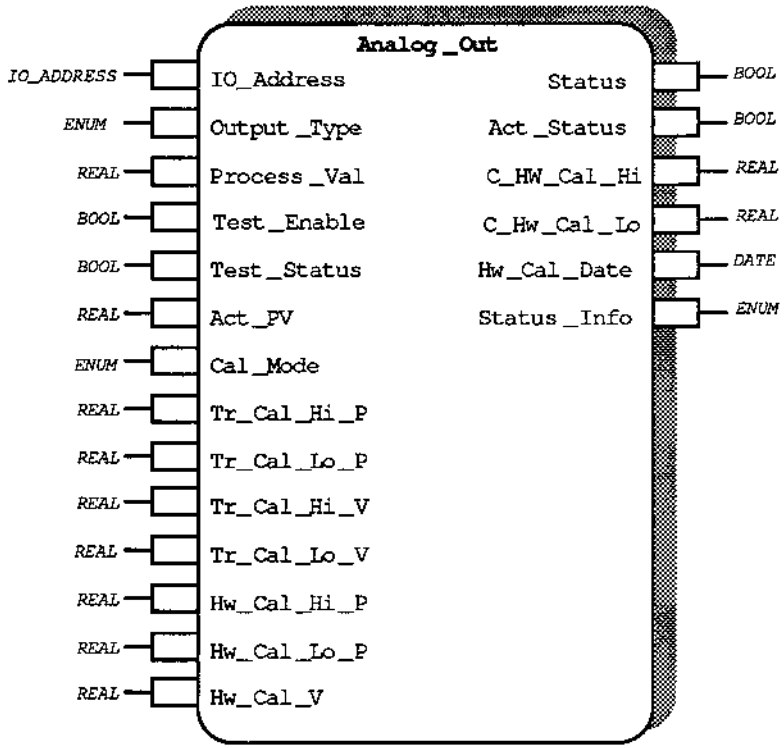


Figure 7-2 Diagramme du bloc fonction Analog_Out

Description fonctionnelle

Le bloc fonction Analog_Out assure l'interface avec une voie analogique.

Il dispose d'un paramètre d'entrée à virgule flottante (REEL) qui définit la valeur de sortie d'une sortie physique analogique. Des possibilités de test sont prévues pour permettre de commander directement une sortie physique au moyen d'une valeur de test qui se substitue à la valeur normale de process. L'état du bloc peut également être forcé.

Attributs du bloc fonction

Type : 18 20

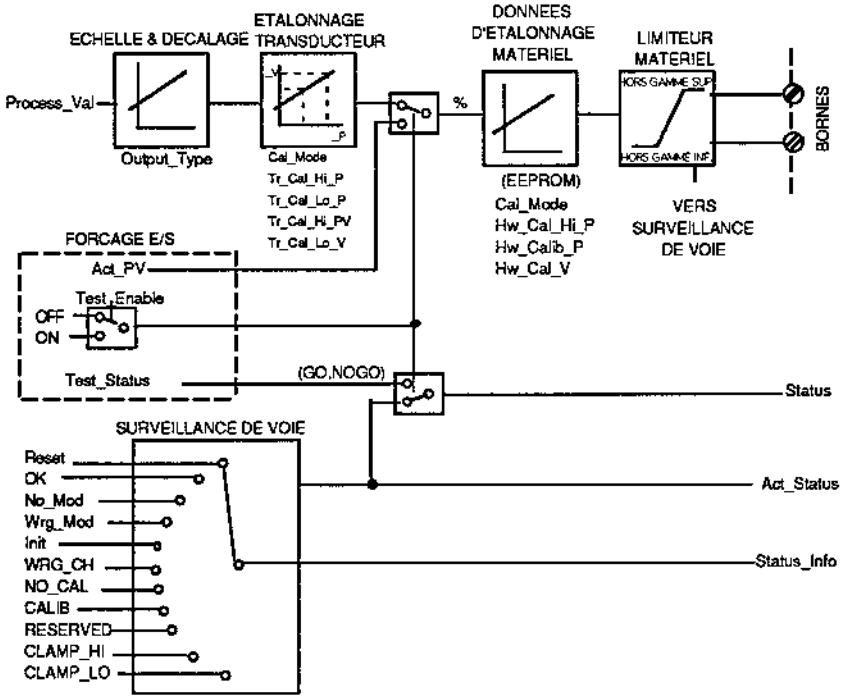
Classe : ..... SORTIES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Process_Val, Act_PV, Status, Status_Info

Capacité mémoire nécessaire : 82 octets

Temps d'exécution : ..... 154 µs



Description des paramètres

Les paramètres marqués par '*' doivent impérativement être confirmés avant de lancer le programme. Tous les autres paramètres sont optionnels.

IO_Address (IOA)

Le paramètre IO_Address associe le bloc fonction concerné aux raccordements physiques du module matériel auquel il se réfère. Sa valeur est attribuée automatiquement lorsque la déclaration du bloc fonction concerné est définie. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack dans lequel est placé le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro de la voie dans le module. Par exemple, 1:02:03 signifie que le bloc fonction concerné désigne la troisième voie d'un module placé dans le second emplacement du premier rack du système PC3000.

Output_Type (OT) *

Output_Type définit le type et la gamme de la sortie utilisée par le module matériel. Le paramètre peut être choisi parmi 6 options, en fonction du réglage correspondant des connexions matérielles de la carte qui sont décrites dans le Manuel d'Installation PC3000. Ces options sont :

mA0_20 (0):
: pour une sortie en courant CC, 0 - 20 mA @ 12 V

mA4_20 (1):
: pour une sortie en courant CC, 4 - 20 mA @ 12 V

V0_10 (2):
: pour une sortie en tension CC, 0 - 10 V @ 20 mA

V2_10 (3):
: pour une sortie en tension CC, 2 - 10 V @ 20 mA

V0_5 (4):
: pour une sortie en tension CC, 0 - 5 V @ 20 mA

V1_5 (5):
: pour une sortie en tension CC, 1 - 5 V @ 20 mA.

D'autres gammes peuvent être configurées en utilisant les possibilités de basculement de la caractéristique que permet le mode calibration du transducteur.

Process_Val (PV)

Process_Val est la valeur qui définit le niveau de sortie de la voie matérielle dont l'adresse est indiquée par le bloc fonction. Si l'entrée sur Process_Val dépasse 100 %, la sortie est bloquée à 100 % (activation totale). Si Process_Val est négatif, la sortie sera 0% (désactivation).

> **Nota:** Lorsque Test_Enable est sur On (1), Process_Val n'est pas utilisé.

Il s'agit de l'entrée commandée par la stratégie de commande.

Etalonnage du transducteur

L'étalonnage du transducteur est une méthode qui permet la mise à l'échelle et le décalage d'origine des gammes de sortie non standard. Il offre également la possibilité de corriger les erreurs des actionneurs, p.e. lorsque la sortie 100 % ou 10 V ne correspond pas à l'ouverture totale d'une vanne, du fait de l'imprécision de la vanne.

Le module de sortie analogique du PC3000 utilise un mode d'étalonnage du transducteur en deux points, afin de corriger à la fois les erreurs de gain et de décalage à l'origine. Deux points peuvent être spécifiés arbitrairement.

L'étalonnage du transducteur s'effectue sur la gamme ou la configuration sélectionnée actuellement, p.e. 0 à 10 V, 0 à 20 mA, etc. Chaque gamme / configuration dispose de données d'étalonnage spécifiques qui lui sont associées.

Les données d'étalonnage du transducteur sont enregistrées dans la mémoire de l'UC et non dans le module, afin d'assurer la compatibilité des modules de rechange. Les données d'étalonnage du transducteur sont téléchargées dans le module lorsque le PC3000 est à l'état Run (marche).

Tous les paramètres associés à l'étalonnage du transducteur sont inclus dans le bloc fonction du module analogique de sortie (Analog_Out).

Tr_Cal_Lo_P (TLP)

C'est la plus basse des deux valeurs d'étalonnage d'entrée, p.e. 0 %

Tr_Cal_Hi_P (THP)

C'est la plus haute des deux valeurs d'étalonnage d'entrée, p.e. 100 %

Tr_Cal_Lo_V (TLV)

C'est la valeur de sortie actuelle requise pour obtenir la valeur du point d'étalonnage bas, p.e. - 0,1 % assure la fermeture totale de la vanne.

Tr_Cal_Hi_V (THV)

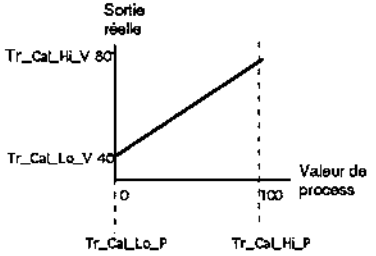
C'est la valeur de sortie réelle requise pour obtenir la valeur du point d'étalonnage haut, p.e. 100,2 % assure l'ouverture totale de la vanne.

Cal_Mode (CAL)

Ce paramètre indique le mode de fonctionnement de la voie. Normalement, la voie est en mode Run (Marche). Pendant l'étalonnage, les états TCal_Hi et TCal_Lo sont choisis à partir de la liste.

Exemple

Si l'on considère une sortie qui délivre 8 - 16 mA. Output_Type doit alors être réglé sur la gamme mA 0-20 (0)



La procédure est la suivante:

(1) Faire passer le paramètre Cal_Mode de Run (0) à TCal_Lo (2).

(2) Choisir Tr_Cal_Lo_P et entrer la valeur zéro qui correspond au pourcentage de pleine échelle requis pour le point d'étalonnage bas. Le module délivre en sortie la valeur qui correspond au pourcentage entré en tant que Tr_Cal_Lo_P.

(3) Entrer la valeur qui correspond à la sortie requise pour Tr_Cal_Lo_P. Si la sortie requise est par exemple de 8mA, entrer 40 % (8/20 x 100 %) au moyen du paramètre Tr_Cal_Lo_V. La sortie analogique prendra cette valeur.

(4) Faire passer le paramètre Cal_Mode de TCal_Lo (2) à TCal_Hi (3).

(5) Choisir Tr_Cal_Hi_P et entrer une valeur de 100%. Le module donnera en sortie la valeur qui correspond au pourcentage entré en tant que Tr_Cal_Hi_P.

(6) Entrer la valeur qui correspond à la sortie requise pour Tr_Cal_Hi_P. Si la sortie requise est par exemple de 16mA, entrer 80 % (16/20 x 100 %) au moyen du paramètre Tr_Cal_Hi_V. La sortie analogique prendra cette valeur.

(7) Faire passer le paramètre Cal_Mode de TCal_Hi (3) à Run (0) afin de mémoriser les nouvelles données d'étalonnage.

(8) En cas d'erreur en cours d'étalonnage, p.e. si une valeur erronée a été introduite pour le niveau de sortie voulu, il suffit d'entrer à nouveau la valeur au moyen de Tr_Cal_Hi_V ou de Tr_Cal_Lo_V.

Forçage des valeurs d'E/S

Ces paramètres permettent de dissocier la sortie de la valeur fournie par le programme. Ceci permet de tester la sortie indépendamment des valeurs de programme du PC3000, ce qui signifie qu'il est facile de tester le mécanisme de défaut ou bien de résoudre les problèmes de mise en service du genre "détecteur de fin de course pas encore installé".

Les paramètres sont utilisés en combinaison, comme suit :

Test_Enable (TEN)

Ce paramètre doit être à l'état On (1) pour forcer Process_Value. Ceci fait, les valeurs de Status et de Process_Value sont réglées par les paramètres suivants.

Test_Status (TST)

La voie étant placée en mode test, ce paramètre peut être utilisé pour contrôler directement le paramètre Status de la voie.

> **Nota :** La valeur du paramètre Status_Info indique toujours OK (1) dans ce mode de fonctionnement.

Act_PV (APV)

Lorsque la voie est placée en mode test, ce paramètre peut être utilisé pour contrôler directement le niveau de sortie de la voie.

Status (ST)

Lorsque Test_Status est sur Off (0), Status indique l'état de la voie matérielle de sortie analogique dont l'adresse est indiquée par le bloc fonction. Si Test_Status est sur On (1), Status prendra la valeur atttribuée à Test_Status.

Act_Status (AST)

Le paramètre Act_Status représente toujours l'état de la voie matérielle. Ce paramètre ne doit être utilisé qu'à des fins de diagnostic.

Status_Info (STI)

Status_Info est un paramètre de diagnostic servant à préciser l'état. Douze états sont possibles :

Reset (0):

Le programme utilisateur ne tourne pas.

Ok (1):

La voie fonctionne normalement.

No_Mod (2):

Il n'y a pas de module à l'emplacement dont l'adresse est indiquée par le bloc fonction.

Wrg_Mod (3):

Un type de module incorrect a été mis à l'emplacement dont l'adresse est indiquée par le bloc fonction.

Init (4):

Le module ou la voie est en cours d'initialisation.

Wrg_Ch (5):

Un type de voie erroné a été choisi sur le module. Pour corriger l'erreur, les cavaliers du module doivent être placés pour correspondre à la gamme choisie par le paramètre Output_Type.

No_Cal (6):

La gamme de sortie choisie par Output_Type n'a pas été étalonnée.

Calib (7):

Le module est actuellement en mode étalonnage.

_ (8)

Cette option n'a pas de fonction.

_ (9)

Cette option n'a pas de fonction.

Clamp_H (10):

Le niveau de sortie demandé est trop élevé pour être obtenu. La sortie est bloquée à sa valeur maximale.

Clamp_L (11):

Le niveau de sortie demandé est trop bas pour être obtenu. La sortie est bloquée à sa valeur minimale.

Paramètres d'étalonnage

Les paramètres ci-dessous sont utilisés pour étalonner la voie matérielle.

Cal_Mode
Tr_Cal_Hi_P
Tr_Cal_Lo_P
Tr_Cal_Hi_V
Hw_Cal_Hi_P
Hw_Cal_Lo_P
Hw_Cal_V
C_Hw_Cal_Hi
C_Hw_Cal_Lo
Hw_Cal_Date

Voir "Manuel d'Installation PC3000, HA022231" pour plus de détails sur la procédure d'étalonnage.

Tr_Cal_Lo_V

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Act_PV	**REAL**	0 %	Config	Config	Lim. haute Lim. basse	100 % 0%
Act_Status	**BOOL**	NOGO (0)	Config	Bloc	Détect.	NOGO (0) Go (1)
C_Hw_Cal_Hi	**REAL**	0%	Config	Bloc	Lim. haute Lim. basse	100 % C_Hw_Cal_Lo
C_Hw_Cal_Lo	**REAL**	0%	Config	Bloc	Lim. haute Lim. basse	C_Hw_Cal_Hi 0%
Cal_Mode	**ENUM**	Run (0)	Config	Config	Valeurs énumérées	Run (0) Save (1) Tcal_Lo (2) Tcal_Hi (3) Hcal_Lo (4) Hcal_Hi (5)
Hw_Cal_Hi_P	**REAL**	0%	Config	Config	Lim. haute Lim. basse	110 % Hw_Cal_Lo_P
Hw_Cal_Lo_P	**REAL**	0%	Config	Config	Lim. haute Lim. basse	Hw_Cal_Hi_P 0%
IO_Address	**IO_ADDRESS**		Config	aucun		
Output_Type	**ENUM**	mA0_20 (0)	Config	Config	Détect.	mA0_20 (0) mA4_20 (1) V0_10 (2) V2_10 (3) V0_5 (4) V1_5 (5)
Process_Val	**REAL**	0 %	Oper	Oper	Lim. haute Lim. basse	100 % 0%
Status	**BOOL**	NOGO (0)	Oper	Block	Détect.	NOGO (0) Go (1)

Tableau 7-2 Attributs des paramètres Analog_Out (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture.	Information propres au type	
Status_Info	ENUM	Reset (0)	Oper	Bloc	Détect.	Voir descript. paramètres
Test_Enable	BOOL	Off (0)	Config	Config	Détect.	Off (0) On (1)
Test_Status	BOOL	NOGO (0)	Config	Config	Détect.	NOGO (0) Go (1)
Tr_Cal_Hi_P	REAL	100 %	Config	Config	Lim. haute Lim. basse	100 % Tr_Cal_Lo_P
Tr_Cal_Hi_V	REAL	100 %	Config	Config	Lim. haute Lim. basse	100 % Tr_Cal_Lo_V
Tr_Cal_Lo_P	REAL	0%	Config	Config	Lim. haute Lim. basse	Tr_Cal_Hi_P 0%
Tr_Cal_Lo_V	REAL	0%	Config	Config	Lim. haute Lim. basse	Tr_Cal_Hi_V 0%

Tableau 7-2 Attributs des paramètres Analog_Out

BLOC FONCTION T_Prop_Out

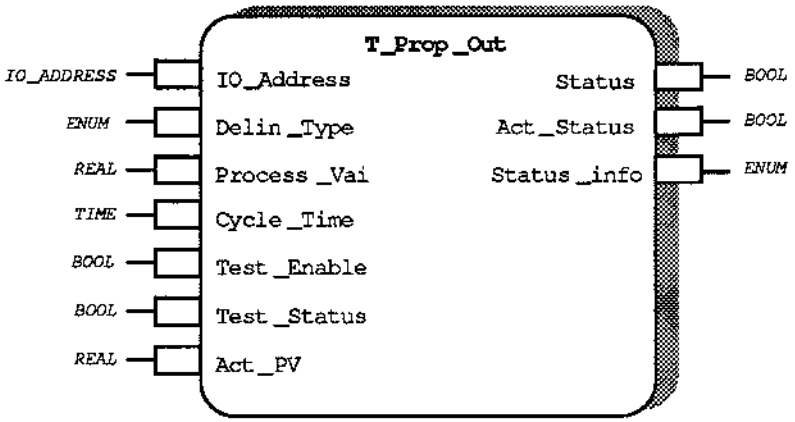


Figure 7-3 Diagramme du bloc fonction T_Prop_Out

Description fonctionnelle

Le bloc fonction T_Prop_Out assure l'interface de bloc fonction pour tout module matériel pouvant recevoir une sortie proportionnelle modulé

Il dispose d'un paramètre d'entrée à virgule flottante (REEL) qui définit un rapport cyclique d'ouverture d'une sortie logique. Des possibilités de test sont prévues pour permettre de commander directement le rapport cyclique au moyen d'une valeur de test. L'état du bloc peut également être forcé.

Attributs du bloc fonction

Type : 24 50

Classe : .....SORTIES

Tâche par défaut: .....Task_2

Liste récapitulative : .....Process_Val, Act_PV. Status, Status_Info

Capacité mémoire nécessaire : 54 octets

Description des paramètres

IO_Address (IOA)

Le paramètre IO_Address associe la déclaration du bloc fonction concerné aux entrées physiques du module matériel auquel il se réfère. Sa valeur est attrubée automatiquement lorsque le bloc fonction concerné est défini. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack dans lequel est placé le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro de la voie dans le module. Par exemple, 1:02:03 signifie que le bloc fonction concerné désigne la troisième voie d'un module placé dans le second emplacement du premier rack du système PC3000.

Delin_Type (DT)

Le paramètre Delin_Type définit le type de fonction de délinéarisation utilisée par la voie de sortie.

Des algorithmes de délinéarisation sont utilisés pour compenser l'effet de refroidissement non linéaire constaté dans les process de refroidissement par eau lorsqu'une vaporisation a lieu dans l'eau de refroidissement. Cet effet est fonction principalement de la valeur du débit et de la différence entre la température d'admission de l'eau et la température du serpentin de refroidissement. Si la température de l'eau monte jusqu'à 100 °C avant d'atteindre la sortie, une partie au moins de l'eau va se vaporiser et la quantité de chaleur extraite sera multipliée par environ 10, du fait de la chaleur latente de vaporisation.

Des algorithmes de délinéarisation doivent être utilisés pour contrôler le débit d'eau moyen dans les serpentins de refroidissement, qui est commandé par la sortie proportionnelle au temps qui pilote l'électrovanne d'admission d'eau. Si la commande et la vanne d'admission sont réglées correctement, la non-linéarité de la voie de sortie compense celle du système de refroidissement.

Delin_Type peut être réglé selon trois options, qui sont :

None (0):

Si None (aucun) est choisi, la sortie matérielle suit linéairement Process_Val.

D_800 (1):

C'est la loi de délinéarisation employée par les appareils Eurotherm de la série 800. La loi consiste en un point de rupture à 80 % de la valeur d'entrée maximale.

Si la valeur d'entrée est inférieure à 80 %, la valeur de sortie est égale à 1/4 de la valeur d'entrée. Si la valeur d'entrée est supérieure à 80 %, la valeur de sortie est égale à (4,0 * valeur d'entrée) - 300.

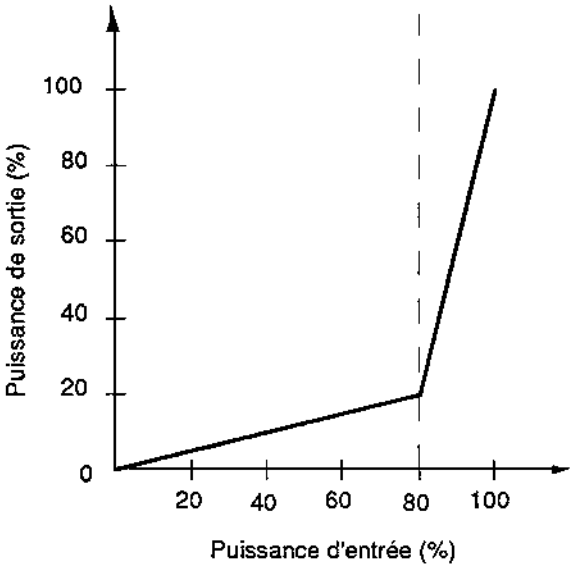


Figure 7-4 Caractéristique de délinéarisation D_800

D_EM1 (2):

C'est la loi de délinéarisation employée par les appareils série Eurotherm de la série EM1. La loi consiste en deux points de rupture à 33,3 % et 66,6 % de la valeur d'entrée maximale. Si la valeur d'entrée est inférieure à 33,3 %, la valeur de sortie est égale à 0,06 * valeur d'entrée. Entre 33,3 % et 66,6 % de la valeur d'entrée, la valeur de sortie est égale à (0,54 * valeur d'entrée) - 16. Au-dessus de 66,6 % de la valeur d'entrée, la valeur de sortie est égale à (2,4 * valeur d'entrée) - 140.

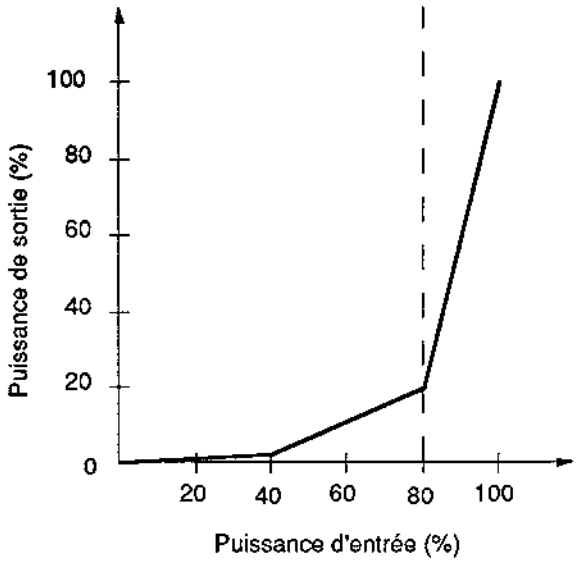


Figure 7-5 Caractéristique de délinéarisation D_EM1

Process_Val (PV)

Le paramètre Process_Val est la valeur qui définit le niveau de la voie à laquelle se réfère le bloc fonction de sortie en fonctionnement normal. Si Process_Val dépasse 100 %, la sortie est bloquée à 100 % (1 logique). Si Process_Val est négative, la voie de sortie est mise à 0 % (0 logique).

Nota : Lorsque Test_Enable est sur On (1), Process_Val n'est pas utilisé.

C'est l'entrée commandée par la stratégie de contrôle.

Cycle_Time (CT)

Le paramètre Cycle_Time définit le temps d'activation (1 logique) et le temps de désactivation (0 logique) de la sortie proportionnelle au temps. Le temps de cycle est défini comme étant le temps nécessaire à la sortie proportionnelle au temps pour effectuer un cycle complet à 50 % de la valeur maximale. Pour réduire l'usure de l'équipement contrôlé, la relation entre le temps d'activation, le temps de désactivation et le niveau de valeur demandé (Process_Val) n'est pas linéaire. Les temps d'activation et de désactivation sont définis par les relations suivantes :

temps d'activation = $\frac{25.0 \cdot Cycle\_Time}{Process\_Val}$

temps de désactivation = $\frac{25.0 \cdot Cycle\_Time}{(100\text{-}Process\_Val)}$

En dessous de 0,2 % de la valeur maximale, la sortie est en permanence désactivée (0 logique). Au-dessus de 99,8 %, la sortie est en permanence activée (1 logique).

Il y a lieu de noter, à propos de ces relations, que Cycle_Time n'est respecté qu'à 50% du cycle. Cycle_Time représente la période minimale de la sortie proportionnelle au temps. A tous les niveaux de sortie autres que 50 %, la somme cyclique temps d'activation + temps de désactivation est supérieure à Cycle_Time.

La relation entre le temps d'activation, le temps de désactivation, Cycle_Time et Process_Val est représentée par le diagramme ci-dessous :

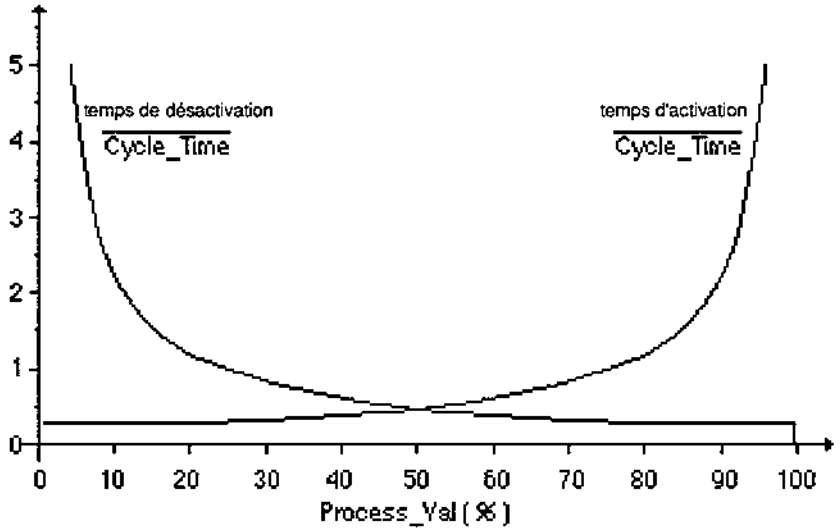


Figure 7-6 Relation entre le temps d'activation, le temps de désactivation, Cycle_Time et Process_Val

Forçage des valeurs d'E/S

Ces paramètres permettent de dissocier la sortie de la valeur fournie par le programme. Ceci permet de tester l'interface de l'installation indépendamment des valeurs de programme du PC3000, ce qui signifie qu'il est facile de tester les

mécanismes prévus en cas de défaillance, ou bien de résoudre les problèmes de mise en service du genre "détecteur de fin de course pas encore installé".

Se reporter au diagramme du bloc fonction pour avoir plus de détails.

Les paramètres sont utilisés en combinaison, comme suit :

Test_Enable (TEN)

Doit être à l'état On pour forcer la valeur aux bornes de sortie. Une fois ce paramètre réglé, la valeur de Status et le signal aux bornes sont réglés par les paramètres suivants.

Test_Status (TST)

La voie étant placée en mode test, ce paramètre peut être utilisé pour contrôler directement le paramètre Status de la voie.

Nota : La valeur du paramètre Status_Info indique toujours OK dans ce mode de fonctionnement.

Act_PV (APV)

La voie étant placée en mode test, ce paramètre peut être utilisé pour contrôler directement l'état de sortie de cette voie.

Status (ST)

Lorsque Test_Status est sur Off (0), Status indique l'état de la voie matérielle à laquelle se réfère le bloc fonction. Dans ce mode, si la puissance demandée est supérieure à 100 % ou est négative, Status sera mis à NOGO (0).

Si Test_Enable est sur On (1), Status prendra la valeur attribuée à Test_Status.

Act_Status (AST)

Le paramètre Act_Status représente toujours l'état de la voie matérielle. Si la puissance demandée est supérieure à 100 % ou est négative, Act_Status sera mis à NOGO (0). Act_Status ne doit être utilisé qu'à des fins de diagnostic.

Status_Info (STI)

Status_Info est un paramètre de diagnostic servant à préciser l'état. Cinq états sont possibles :

Reset (0)	Le programme utilisateur ne tourne pas.
Ok (1):	La voie fonctionne normalement.
No_Mod (2):	Il n'y a pas de module à l'emplacement dont l'adresse est indiquée par le bloc fonction.
Wrg_Mod (3):	Un type de module incorrect a été mis à l'emplacement dont l'adresse est indiquée par le bloc fonction.
Init (4):	Le module ou la voie est en cours d'initialisation.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture.	Informations propres au type	
Act_PV	REAL	0 %	Config	Config	Lim. haute Lim. basse	100 % 0%
Act_Status	BOOL	NOGO (0)	Config	Bloc	Détect.	NOGO (0) Go (1)
Cycle_Time	TIME	2 S	Super	Super	Lim. haute Lim. basse	02h_46m_40s 300ms (Logique) 02s (Relais)
Delin Type	ENUM	None (0)	Config	Config	Détect.	None (0) D_800 (1) D_EM1 (2)
IO_Address	IO_ADDRESS		Super	Super		
Process_Val	REAL	0 %	Oper	Oper	Lim. haute Lim. basse	100 % 0%
Status	BOOL	NOGO (0)	Oper	Bloc	Détect.	NOGO (0) Go (1)
Status_Info	ENUM	Reset (0)	Oper	Bloc	Détect.	Reset(0) Ok(1) No_Mod(2) Wrg_Mod(3) Init(4)
Test_Enable	BOOL	Off (0)	Config	Config	Détect.	Off (0) On (1)
Test_Status	BOOL	NOGO (0)	Config	Config	Détect.	NOGO (0) Go (1)

Table 7-3 Attributs des paramètres T_Prop_Out

BLOC FONCTION xFast_An_O (Pas valable pour la nouvelle version)

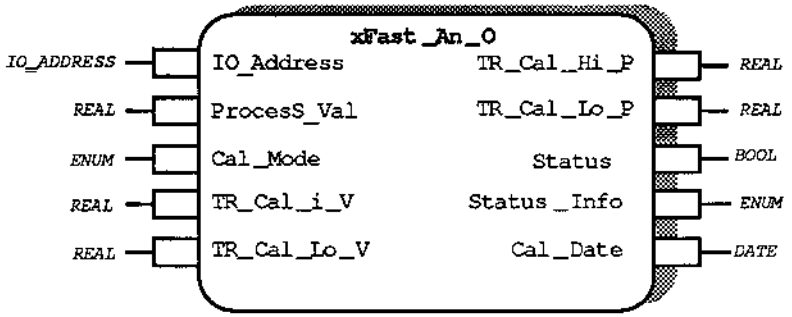


Figure 7-7 Diagramme du bloc fonction xFast_An_O

Description fonctionnelle

Le bloc fonction xFast_An_O assure l'interface de bloc fonction pour tout module matériel pouvant recevoir une voie de sortie analogique rapide, par exemple le module d'E/S analogique rapide (AI08).

Il doit être utilisé lorsqu'une fonctionnalité inférieure à celle qu'apporte le bloc fonction Analog_Out est acceptable, xFast_An_O nécessite moins de temps système que le bloc fonction Analog_Out et son temps d'exécution est par conséquent plus court.

Attributs du bloc fonction

Type : 18 40

Classe : ..... SORTIES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Process_Val, Status, Status_Info

Capacité mémoire nécessaire : 48 octets

Description des paramètres

Paramètres d'étalonnage

Les paramètres ci-dessous sont utilisés pour étalonner le module matériel dont l'adresse est indiquée par le bloc fonction concerné :

Cal_Mode

TR_Cal_Hi_V

TR_Cal_Lo_V

IO_Address (IOA)

Le paramètre IO_Address associe le bloc fonction concerné aux raccordements physiques du module matériel auquel il se réfère. Sa valeur est attribuée automatiquement lorsque la déclaration du bloc fonction concerné est définie. Sa valeur prend la forme X:YY:ZZ, où X représente le numéro du rack dans lequel est placé le module, YY représente le numéro de l'emplacement dans le rack et ZZ représente le numéro de la voie dans le module. Par exemple, 1:02:03 signifie que le bloc fonction concerné désigne la troisième voie d'un module placé dans le second emplacement du premier rack du système PC3000.

Process_Val (PV)

Process_Val est la valeur qui définit le niveau de sortie de la voie matérielle à laquelle se réfère le bloc fonction. Si la valeur d'entrée de Process_Val est supérieure à 102,4 %, l'état sera NOGO (0), Status_Info sera Clmp_Hi (6) et la voie de sortie sera bloquée à 100 % de la puissance. Si la valeur d'entrée de Process_Val est inférieure à -102,4 %, l'état sera NOGO (0), Status_Info sera Clmp_Lo (7) et la voie de sortie sera bloquée à -100 % de la puissance.

Status (ST)

Status indique l'état de la voie matérielle à laquelle se réfère le bloc fonction.

Status_Info (STI)

Status_Info est un paramètre de diagnostic utilisé pour préciser l'état de Status. Huit états sont possibles :

Reset (0):	Le programme utilisateur ne tourne pas.
Ok (1):	La voie fonctionne normalement.
No_Mod (2):	Il n'y a pas de module à l'emplacement dont l'adresse est indiquée par le bloc fonction.
Wrg_Mod (3):	Un type de module incorrect a été mis à l'emplacement dont l'adresse est indiquée par le bloc fonction.
Calib (4):	Le module est actuellement en mode étalonnage.
Init (5):	Le module ou la voie est en cours d'initialisation.
Clmp_Hi (6):	Le niveau de sortie demandé est trop élevé pour être obtenu.
Clmp_Lo (7):	Le niveau de sortie demandé est trop bas pour être obtenu.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Cal_Date	**DATE**	01-Jan-1970	Config	Config		
Cal_Mode	**ENUM**	Run (0)	Config	Config	Détect.	Run (0) Save (1) Hcal_Hi (2) Hcal_Lo (3) Tcal_Hi (4) Tcal_Lo (5) Dflt_TR (6)
IO_Address	**IO_ADDRESS**		Config	Config		
Process_Val	**REAL**	0%	Config	Config	Lim. haute Lim. basse	110 % -110 %
Status	**BOOL**	Go (1)	Oper	Block	Détect.	NOGO (0) Go (1)
Status_Info	**ENUM**	Ok (1)	Oper	Block	Détect.	voir descript. paramètres
TR_Cal_Hi_P	**REAL**	100%	Config	Config	Lim. haute Lim. basse	110 % TR_Cal_Lo_P
TR_Cal_Hi_V	**REAL**	100%	Config	Config	Lim. haute Lim. basse	110 % TR_Cal_Lo_V
TR_Cal_Lo_P	**REAL**	-100%	Config	Config	Lim. haute Lim. basse	TR_Cal_Hi_P -110 %
TR_Cal_Lo_V	**REAL**	-100%	Config	Config	Lim. haute Lim. basse	TR_Cal_Hi_V -110 %

Table 7-4 Attributs des paramètres Fast_An_Out

Chapitre 8

TRAITEMENT DES SIGNAUX

Edition 2

Sommaire (suite)

Sommaire (suite)

Présentation

Cette classe de blocs fonctions contient un ensemble de blocs fonctions généraux de traitement des signaux. Elle comporte également un générateur de nombres aléatoires.

BLOC FONCTION SCALE

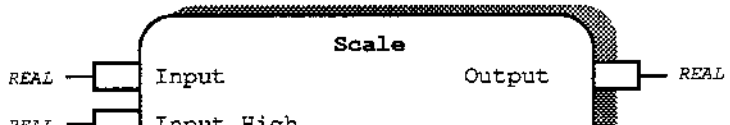


Figure 8-1 Schéma du bloc fonction Scale

Description fonctionnelle

Ce bloc fonction permet de modifier l'échelle d'une valeur à l'aide d'une étendue d'entrée et d'une étendue de sortie. Les valeurs d'entrée basse et haute sont fixées avec les valeurs de sortie basse et haute qui leur correspondent. Le bloc fonction fournit ensuite une sortie qui couvre un pourcentage de l'étendue de sortie identique au pourcentage de l'étendue d'entrée couvert par l'entrée. Cette relation s'exprime mathématiquement de la manière suivante :

$$\text{Scale\_fb.Output} = \text{Output\_Low} + \text{Scale\_fb.Input} * \frac{(\text{Output\_High} - \text{Output\_Low})}{(\text{Input\_High} - \text{Input\_Low})}$$

Aucun système de dépassement n'est intégré au bloc, par conséquent, si une valeur d'entrée sort de l'étendue d'entrée (c'est-à-dire est supérieure à **Input_High** ou inférieure à **Input_Low**), la même relation linéaire s'applique.

Attributs du bloc fonction

Type : 1C01

Classe : ..... TRAITEMENT DES SIGNAUX

Tâche par défaut : ..... Task_2

Liste résumée : ..... Input, Output

Mémoire nécessaire : ..... 24 octets

Description des paramètres

Input (IN)

Valeur dont il faut changer l'échelle.

Input_High (IHI)

Valeur supérieure d'Input. Cette valeur, si elle était appliquée comme Input, donnerait une Output égale à **Output_High.**

Input_Low (ILO)

Valeur inférieure d'Input. Cette valeur, si elle était appliquée comme Input, donnerait une Output égale à **Output_Low.**

Output_High (OHI)

Valeur supérieure d'Output correspondant à **Input_High.** Elle est égale à l'Output que l'on obtiendrait en appliquant **Input_High** comme Input.

Output_Low (OLO)

Valeur inférieure d'Output correspondant à **Input_Low.** Elle est égale à l'Output que l'on obtiendrait en appliquant **Input_Low** comme Input.

Output

Equivalent d'Input mis à l'échelle.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	REAL	0	Oper	Oper	Limite haute	+3·402823E+38
					Limite basse	-3·402823E+38
Input_High	REAL	100	Oper	Oper	Limite haute	+3·402823E+38
					Limite basse	-3·402823E+38
Input_Low	REAL	0	Oper	Oper	Limite haute	+3·402823E+38
					Limite basse	-3·402823E+38
Output_High	REAL	10	Oper	Oper	Limite haute	+3·402823E+38
					Limite basse	-3·402823E+38
Output_Low	REAL	-10	Oper	Oper	Limite haute	+3·402823E+38
					Limite basse	-3·402823E+38
Output	REAL	-10	Oper	Oper	Limite haute	+3·402823E+38
					Limite basse	-3·402823E+38

Tableau 8-1 Attributs des paramètress Scale

BLOC FONCTION MAXIMUM/MINIMUM/AVERAGE

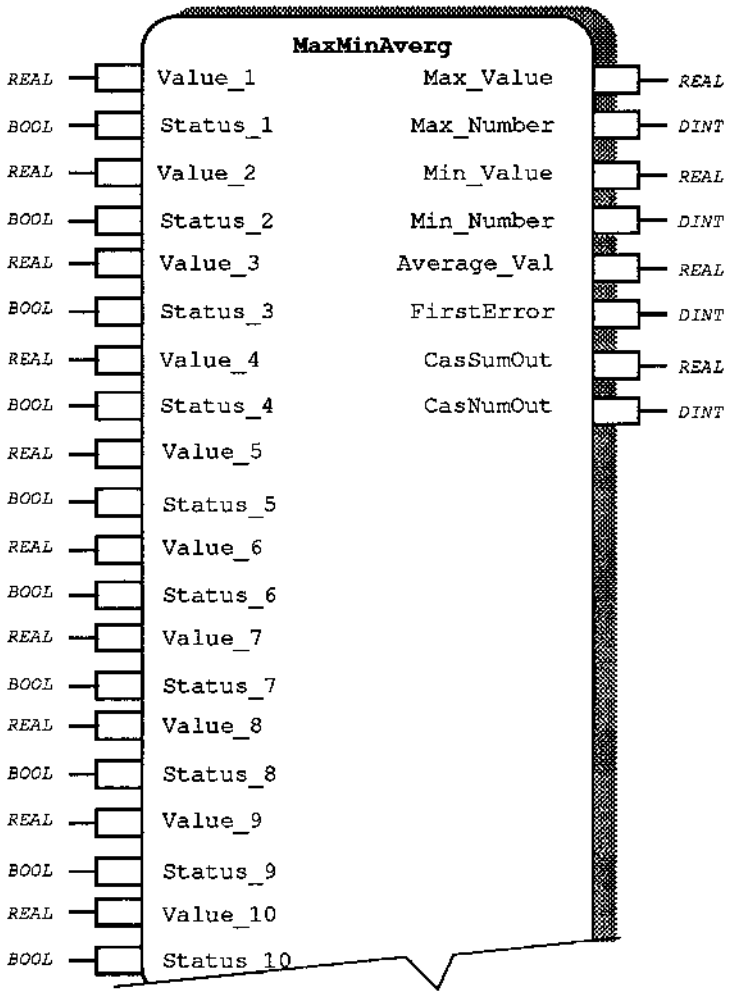


Figure 8-2 Schéma du bloc fonction MaxMinAverg

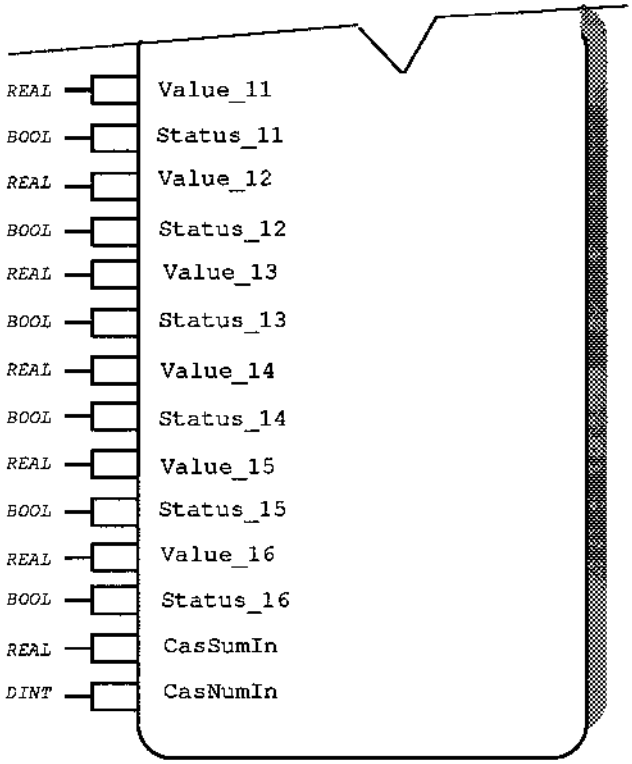


Figure 8-2 Schéma du bloc fonction MaxMinAverg (suite)

Description fonctionnelle

Le maximum, le minimum et la moyenne d'un maximum de seize entrées sont dérivés et ces valeurs calculées se présentent sous la forme de sorties. Les entrées sont numérotées de un à seize et le numéro de l'entrée qui a la valeur la plus faible ainsi que le numéro de l'entrée qui a la valeur la plus forte sont donnés comme deux sorties supplémentaires.

Des sorties sont également incluses pour donner en cascade la somme et le numéro des entrées dans une série de blocs, afin de permettre le calcul de la moyenne générale des entrées de l'ensemble des blocs.

Les entrées peuvent aussi être activées ou désactivées séparément et le numéro de la première voie désactivée est indiqué sur une sortie.

Comme exemple de manière d'utiliser ce bloc fonction, envisageons la situation dans laquelle un certain nombre d'entrées de thermocouple doivent servir à réguler

la température d'une pièce de grande taille. Les thermocouples seraient câblés sur les entrées du bloc fonction Maximum/Minimum/Average et la température moyenne pourrait servir de variable de procédé pour un bloc PID. Les sorties maximale et minimale pourraient servir d'alarmes ou, dans le cas de l'utilisation d'un bloc fonction rampe, elles pourraient servir à activer le maintien sur écart. La sortie d'état de chaque bloc fonction d'entrée analogique pourrait servir à activer l'entrée correspondante du bloc fonction Maximum/Minimum/Average. La sortie FirstError indiquerait l'entrée au numéro le plus faible sur laquelle s'est produit un défaut matériel.

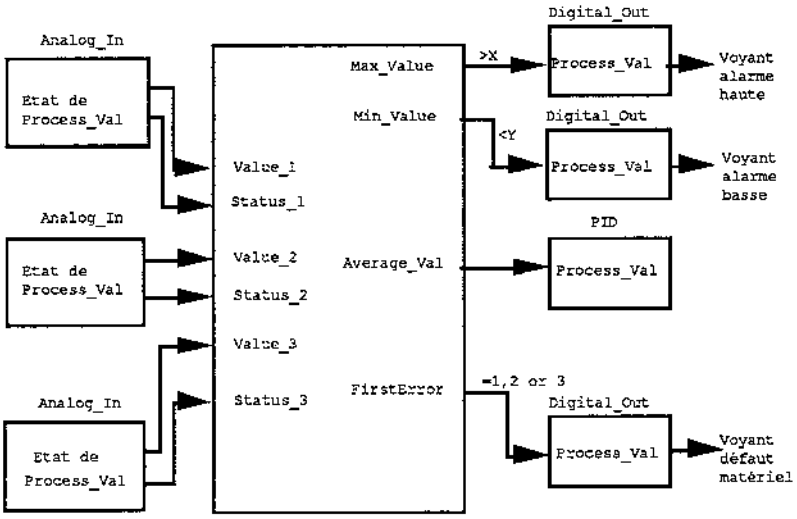


Figure 8-3 Exemple d'utilisation du bloc fonction MaxMinAverg pour fournir une entrée à la boucle PID.

Attributs du bloc fonction

Type : 1C08
Classe : .....TRAITEMENT DES SIGNAUX
Tâche par défaut : .....Task_2
Liste résumée : .....Max_Value, Min_Value, Average_Val,
.....First_Error.
Mémoire nécessaire : ..... 220 octets

Description des paramètres

Value_1 (V1) à Value_16 (V16)

Entrées du bloc fonction qui doivent être traitées.

Status_1 (S1) à Status_16 (S16)

Permet d'activer ou de désactiver séparément chaque entrée Value. **Status_1** active **Value_1**, **Status_2** active **Value_2**, etc.

CasSumIn (CSI)

Sert à insérer la somme d'un bloc précédent, afin que la moyenne calculée soit égale à la moyenne des entrées de ce bloc et des entrées d'un ou plusieurs bloc(s) précédent(s). Utilisé conjoitement avec CasNumIn.

CasNumIn (CNI)

Sert à insérer le nombre d'entrées activées d'un bloc précédent afin que la moyenne calculée soit égale à la moyenne des entrées de ce bloc et des entrées d'un ou plusieurs bloc(s) précédent(s). Utilisé conjoitement avec CasSumIn.

Max_Value (MXV)

Valeur maximale actuellement appliquée aux entrées activées.

Max_Number (MXN)

Numéro de l'entrée à laquelle est actuellement appliquée la valeur la plus élevée. Par exemple, si la valeur la plus élevée des entrées activées est 27 et si cette valeur est appliquée à l'entrée Value_8, **Max_Number** est égal à 8 et **Max_Value** est égal à 27.

Min_Value (MNV)

Valeur minimale actellement appliquée aux entrées activées.

Min_Number (MNN)

Numéro de l'entrée à laquelle est actuellement appliquée la valeur la plus faible. Par exemple, si la valeur la plus faible des entrées activées est 20 et si cette valeur est appliquée à l'entrée Value_11, **Min_Number** est égal à 11 et **Min_Value** est égal à 20.

Average_Val (AV)

Moyenne arithmétique de l'ensemble des entrées activées de ce bloc si CasNumIn et CasSumIn sont égaux à zéro :

$$\text{Average\_Val} = \frac{\Sigma \text{ (entrées activées)}}{\text{(nombre d' entrées activées)}}$$

Toutefois, si CasNumIn est positif et si CasSumIn n'est pas nul, Average_Val est égal à la moyenne des entrées activées de ce bloc et de l'ensemble des blocs précédents placés en cascade à l'aide de CasNumOut/CasNumIn et CasSumOut/CasSumIn. Pour le bloc actuel cette relation pourrait s'exprimer sous la forme :

$$\text{Average\_Val} = \frac{\Sigma \text{ (entrées activées)} + \text{CasSumIn}}{\text{(nombre d'entrées activées)} + \text{CasNumIn}}$$

Imaginons deux blocs mis en cascade. Si dix valeurs sont activées sur le premier bloc dont la somme est égale à 428, **Average_Val** est égal à 42,8, CasNumOut à 10 et CasSumOut à 428. Les deux sorties en cascade du premier bloc sont reliées aux deux entrées en cascade du deuxième bloc (first.CasSumOut à second.CasSumIn et first.CasNumOut à second.CasNumIn). Sur ce deuxième bloc, six autres valeurs sont activées et leur total est égal à 270. **Average_Val** est égal à 43,63 ((270+428)/(10+6)), CasNumOut à 16 et CasSumOut à 698.

FirstError (FER)

Le numéro de la première entrée désactivée est indiqué dans ce paramètre. Par exemple, si l'ensemble des entrées sont activées sauf **Value_12** et **Value_16**, FirstError est fixé à 12.

CasSumOut (CSO)

Somme de l'ensemble des entrées activées et de CasSumIn.

CasNumOut (CNO)

Somme du nombre d'entrées activées et de CasNumIn.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Value_1 à Value_16	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Status_1 à Status_16	**BOOL**	Disable (0)	Oper	Oper	Sens	Disable (0) Enable (1)
CasSumIn	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
CasNumIn	**DINT**	0	Oper	Oper	Limite haute Limite basse	2147483647 0
Max_Value	**REAL**	-3·4E + 38	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Max_Number	**DINT**	0	Oper	Block	Limite haute Limite basse	16 0
Min_Value	**REAL**	3·4E + 38	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Min_Number	**DINT**	0	Oper	Block	Limite haute Limite basse	16 0
Average_Value	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
FirstError	**DINT**	1	Oper	Block	Limite haute Limite basse	16 0
CasSumOut	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
CasNumOut	**DINT**	0	Oper	Block	Limite haute Limite basse	2147483647 0

Tableau 8-2 Attributs des paramètres Maximum/Minimum Average

BLOC FONCTION INPUT SWITCHOVER

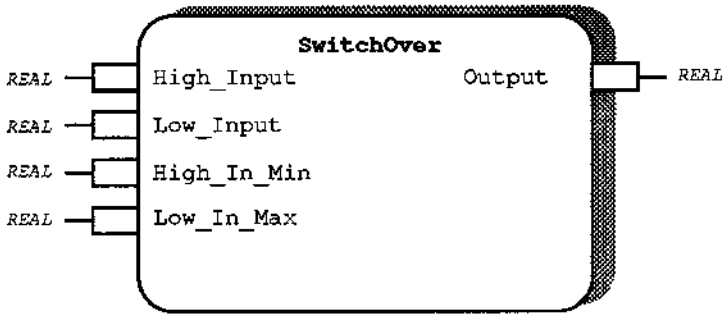


Figure 8-4 Schéma du bloc fonction Input Switchover

Description fonctionnelle

Lorsque des mesures doivent être effectuées sur une plage importante, il arrive qu'il soit nécessaire d'utiliser deux transducteurs : un pour la partie inférieure de la plage et un autre pour la partie supérieure. Il y a au milieu de la plage un ensemble de valeurs où l'on utilise un agrégat des deux valeurs, appelé plage de chevauchement. On utilise par exemple ce type de disposition pour les mesures de température avec un thermocouple et un pyromètre et pour les systèmes de vide utilisant des indicateurs de vide primaire et poussé.

Ce bloc fonction possède deux entrées qui peuvent être câblées sur les paramètres **Process_Val** des blocs fonctions d'entrée traitant le transducteur d'entrée de l'échelle supérieure (**High_Input**) et le transducteur d'entrée de l'échelle inférieure (**Low_Input**). Deux autres entrées du bloc fonction définissent la valeur maximale de validité de l'entrée de l'échelle inférieure (**Low_in_Max**) et la valeur minimale de validité de l'entrée de l'échelle supérieure (**High_In_Min**).

La sortie du bloc fonction provient de **High_Input** lorsque l'entrée est supérieure à la plage de chevauchement ou de **Low_Input** lorsque l'entrée est inférieure à la plage de chevauchement. Lorsque les deux entrées sont situées dans la plage de chevauchement, la sortie est égale à la moyenne arithmétique des entrées.

Attributs du bloc fonction

Type : 1C10
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... High_Input, Low_Input, Output.
Mémoire nécessaire : ..... 28 octets

Description des paramètres

High_Input (HI)

La valeur d'un transducteur représentant la partie supérieure de l'échelle, c'est-à-dire les valeurs supérieures à **High_In_Min.**, serait normalement reliée à **Process_Val** d'un bloc fonction **Analog_In.**

Low_Input (LI)

La valeur d'un transducteur représentant la partie inférieure de l'échelle, c'est-à-dire les valeurs inférieures à **Low_In_Max.**, serait normalement reliée à **Process_Val** d'un bloc fonction **Analog_In.**

High_In_Min (HMN)

Valeur en-dessous de laquelle les valeurs de **High_In** ne sont pas prises en compte.

Low_In_Max (LMX)

Valeur au-dessus de laquelle les valeurs de **Low_In** ne sont pas prises en compte.

Output (OP)

Lorsque **High_Input** est supérieur à **Low_In_Max**, Output provient exclusivement de **High_Input** (fig a). Lorsque **High_Input** devient inférieur à **Low_in_Max**, Output provient de **Low_Input** (si **Low_Input** est inférieur à **High_In_Min)** (fig b) ou de la moyenne arithmétique de **High_Input** et **Low_Input** (si **Low_Input** est compris entre **High_In_Min** et **Low_In_Max**) (fig c).

Dans la situation improbable où **High_Input** serait inférieur à **High_in_Min** et **Low_Input** supérieur à **Low_In_Max**, Output serait égal à la moyenne arithmétique de **High_Input** et de **Low_Input** (fig d).

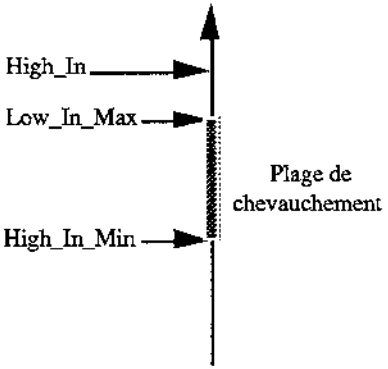


Fig (a)
Output = High_In
(La valeur de Low_In n'est pas prise en compte)

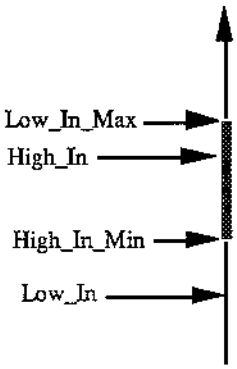


Fig (b)
Output = Low_In

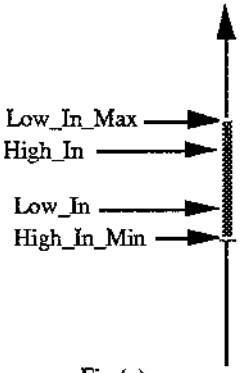


Fig (c)
Output = $\frac{\text{High\_In} + \text{Low\_In}}{2}$

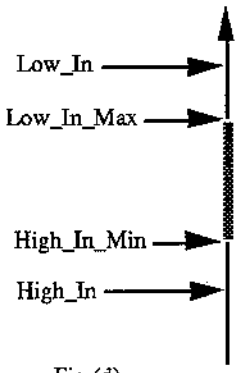


Fig (d)
Output = $\frac{\text{High\_In} + \text{Low\_In}}{2}$

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
High_Input	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Low_Input	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
High_In_Min	**REAL**	500	Oper	Oper	Limite haute Limite basse	+3·402823E+38 Low_In_Max
Low_In_Max	**REAL**	1000	Oper	Oper	Limite haute Limite basse	High_In_Min -3·402823E+38
Output	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 8-3 Attributs des paramètres SwitchOver

BLOC FONCTION PROCESS DELAY

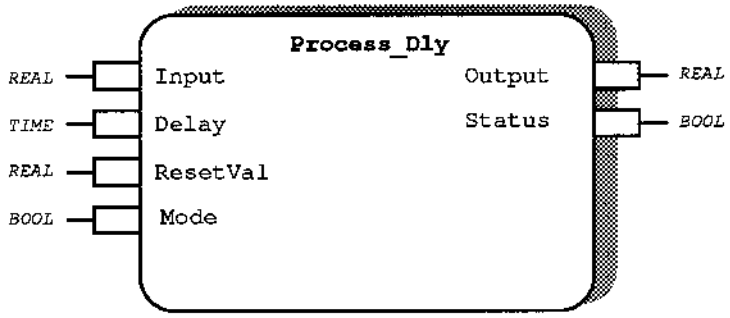
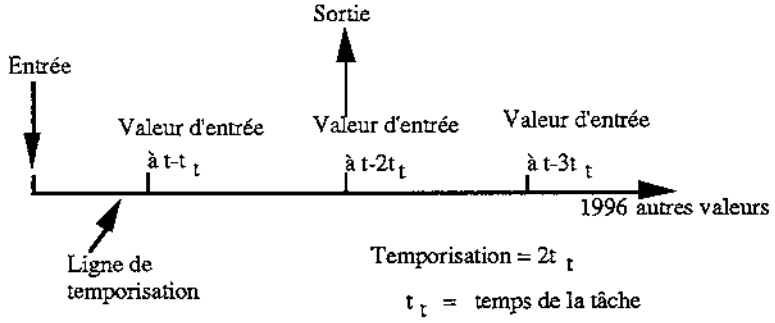


Figure 8-5 Schéma du bloc fonction Process Delay

Description fonctionnelle

Une "ligne de temporisation" sert à stocker la valeur de l'entrée de ce bloc fonction comme fonction du temps. La sortie affiche la valeur qui était sur l'entrée du bloc fonction à une période passée spécifiée (à condition que Mode soit positionné sur Run).

Un changement du paramètre d'entrée passe donc par cette ligne de temporisation et, après une durée spécifiée, la sortie affiche cette valeur d'entrée modifiée.



Les valeurs situées sur la ligne de temporisation se décalent d'une position à chaque exécution du bloc fonction et, avec les tâches par défaut toutes les 100 msec, toutes les valeurs se déplacent. La valeur actuelle est chargée de manière sélective sur la première position de la ligne de temporisation une fois que

l'ensemble des déplacements est terminée. Etant donné qu'il y a 2000 positions sur la ligne de temporisation, la temporisation maximale est de 3 minutes et 2 secondes avec les tâches par défaut. Il est possible d'obtenir des temporisations supérieures en faisant exécuter le bloc fonction à une fréquence inférieure.

Il est possible de replacer la totalité des valeurs de la ligne entière sur un ensemble de valeurs choisies à l'aide de ResetVal et en positionnant Mode sur Reset.

Etant donné que la ligne de temporisation est une longueur de temps constante, si la temporisation est augmentée, il est possible de voir le même changement sur la sortie qui a été déjà vue. On peut comparer cette situation au fait de voir passer une voiture en étant sur le bord de la route, puis d'être transporté par magie cinq cent mètres plus loin et de la voir passer une nouvelle fois.

Attributs du bloc fonction

Type : 1C20
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Input, Output, Delay, Status.
Mémoire nécessaire : ..... 8042 octets

Description des paramètres

Input (IN)

Valeur à placer sur la ligne de temporisation.

Delay (TD)

Intervalle nécessaire entre l'apparition d'un changement sur l'entrée et l'apparition de ce même changement sur la sortie. On peut aussi le considérer comme la distance temporelle de la sortie sur la ligne de temporisation.

ResetVal (R)

L'ensemble des valeurs situées sur la ligne de temporisation seront positionnées sur cette valeur lorsque Mode sera positionné sur Reset (réinitialisation).

Mode (M)

Commutable entre Run(exécution) et Reset (réinitialisation), cette entrée permet la mise à jour de la ligne de temporisation à partir de l'entrée (mode Run) ou le positionnement de l'ensemble des valeurs situées sur la ligne de temporisation sur ResetVal (mode Reset).

Output (OP)

Valeur qui était sur l'entrée à un moment du passé égal à Delay (temporisation). Si Delay est supérieur à la valeur maximale autorisée (2000 * temps de la tâche), la sortie affiche la dernière valeur de la ligne de temporisation, c'est-à-dire la valeur qui a subi la temporisation maximale.

Status (S)

Normalement, cette sortie affiche un Status (état) Go. Toutefois, si Delay est supérieur à la valeur maximale admissible (2000 * temps de la tâche), Status affiche TooLong, indiquant que la temporisation demandée n'est pas exécutée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	REAL	0	Oper	Oper	Limite haute Limite basse	+3402823E+38 -3·402823E+38
Delay	TIME	10s	Oper	Oper	Limite haute Limite basse	fonction du temps de la tâche 0ms
ResetVal	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Mode	BOOL	Run(0)	Oper	Oper	Sens	Run(0) Reset (1)
Output	REAL	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Status	BOOL	Go(0)	Oper	Block	Sens	Go (0) TooLong (1)

Tableau 8-4 Attributs des paramètres Process Delay

BLOC FONCTION HYSTERESIS REAL

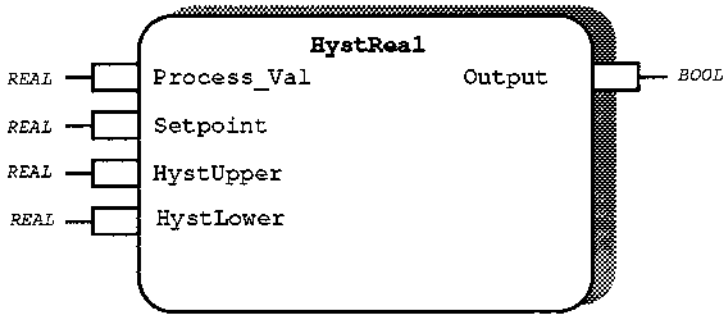


Figure 8-6 Schéma du bloc fonction Hysteresis Real

Description fonctionnelle

Lors du contrôle de valeurs analogiques, il existe de nombreuses situations dans lesquelles une certaine forme d'hystérésis est nécessaire. Ce bloc fonction permet de définir indépendamment un point d'hystérésis supérieur et un point d'hystérésis inférieur de chaque côté d'une consigne pour les valeurs de type réel. Lorsque **Process_Val** devient supérieur à la consigne plus la limite d'hystérésis supérieure HystUpper, Output passe à l'état On et y reste jusqu'à ce que **Process_Val** devienne inférieur à la consigne moins la limite d'hystérésis inférieure HystLower.

En réglant les valeurs d'hystérésis inférieure et supérieure, il est possible de mettre au point un certain nombre de stratégies différentes pour provoquer l'activation ou la désactivation de la sortie. En donnant à HystUpper la valeur **Setpoint,** il est possible de créer une hystérésis d'alarme haute classique : **Output** devient vrai lorsque **Setpoint** est dépassé et ne redevient faux que lorsque **Process_Val** est devenu inférieur à **Setpoint** moins HystLower.

En positionnant HystLower sur zéro et en inversant **Output** dans les instructions de câblage ultérieures, il est possible de créer une hystérésis d'alarme basse classique : **Output** devient faux lorsque **Process_Val** devient inférieur à la consigne et ne redevient vrai que lorsque l'entrée est devenue supérieure à **Setpoint** plus HystUpper. L'utilisation en tandem d'HystUpper et HystLower offre des possibilités supplémentaires.

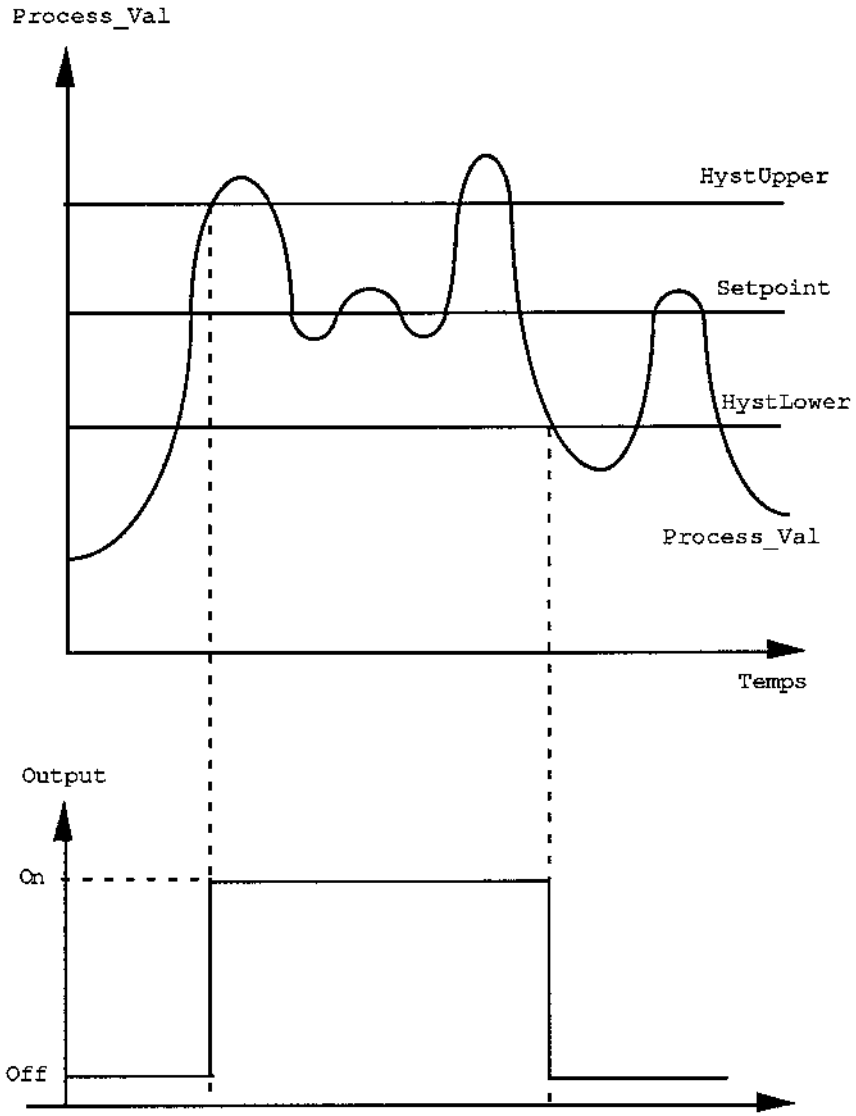


Figure 8-7 Hystérésis supérieure et inférieure

Attributs du bloc fonction

Type : 1C60
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Process_Val, Setpoint, Output
Mémoire nécessaire : ..... 18 octets

Description des paramètres

Process_Val (PV)

Valeur à comparer à **Setpoint.** La variable sur laquelle une alarme est nécessaire doit être câblée sur cette entrée.

Setpoint (SP)

Valeur principale par rapport à laquelle il faut comparer **Process_Val.**

HystUpper (HYU)

Décalage positif par rapport à la consigne qui est le niveau auquel **Output** va être activé. Les valeurs d'hystérésis sont relatives par rapport à la consigne.

HystLower (HYL)

Décalage négatif par rapport à la consigne qui est le niveau auquel **Output** va être désactivé. Les valeurs d'hystérésis sont relatives par rapport à la consigne.

Output (OP)

Cette sortie est activée lorsque **Process_Val** est supérieur à **Setpoint** plus HystUpper, désactivée lorsque **Process_Val** est inférieur à **Setpoint** moins HystLower et conserve sa valeur actuelle lorsqu'elle est située entre **Setpoint** plus HystUpper et **Setpoint** moins HystLower.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Proces_Val	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Setpoint	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
HystUpper	REAL	1	Oper	Oper	Limite haute Limite basse	+3·402823E+38 0
HystLower	REAL	1	Oper	Oper	Limite haute Limite basse	+3·402823E+38 0
Output	BOOL	Off (0)	Oper	Block	Sens	On (1) Off (0)

Tableau 8-5 Attributs des paramètres Hysteresis Real

BLOC FONCTION HYSTERESIS DINT

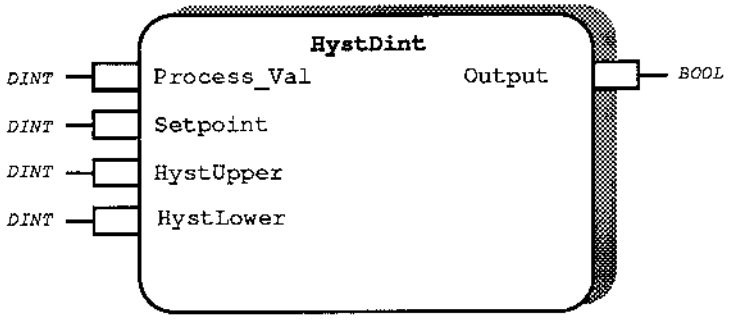


Figure 8-8 Schéma du bloc fonction Hysteresis Dint

Description fonctionnelle

Lors du contrôle de valeurs analogiques, il existe de nombreuses situations dans lesquelles une certaine forme d'hystérésis est nécessaire. Ce bloc fonction permet de définir indépendamment un point d'hystérésis supérieur et un point d'hystérésis inférieur de chaque côté d'une consigne pour les valeurs de type entier double. Lorsque **Process_Val** devient supérieur à la consigne plus la limite d'hystérésis supérieure HystUpper, **Output** passe à l'état On et y reste jusqu'à ce que **Process_Val** devienne inférieur à la consigne moins la limite d'hystérésis inférieure HystLower.

En réglant les valeurs d'hystérésis inférieure et supérieure, il est possible de mettre au point un certain nombre de stratégies différentes pour provoquer l'activation ou la désactivation de la sortie. En donnant à HystUpper la valeur **Setpoint,** il est possible de créer une hystérésis d'alarme haute classique : **Output** devient vrai lorsque **Setpoint** est dépassé et ne redevient faux que lorsque **Process_Val** est devenu inférieur à **Setpoint** moins HystLower.

En positionnant HystLower sur zéro et en inversant **Output** dans les instructions de câblage ultérieures, il est possible de créer une hystérésis d'alarme basse classique : **Output** devient faux lorsque **Process_Val** devient inférieur à la consigne et ne redevient vrai que lorsque l'entrée est devenue supérieure à **Setpoint** plus HystUpper. L'utilisation en tandem d'HystUpper et HystLower offre des possibilités supplémentaires.

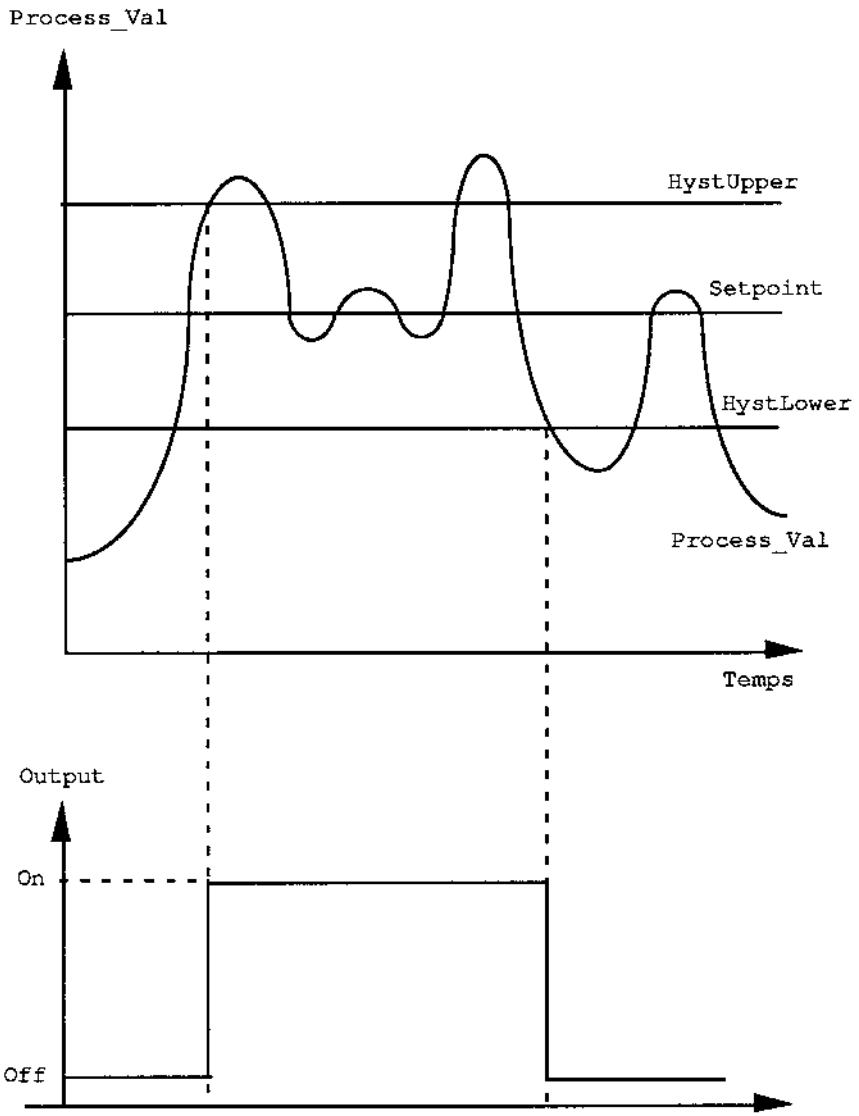


Figure 8-9 Hystérésis supérieure et inférieure

Attributs du bloc fonction

Type : 1C62
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Process_Val, Setpoint, Output
Mémoire nécessaire : ..... 18 octets

Description des paramètres

Process_Val (PV)

Valeur à comparer à **Setpoint.** La variable sur laquelle une alarme est nécessaire doit être câblée sur cette entrée.

Setpoint (SP)

Valeur principale par rapport à laquelle il faut comparer **Process_Val.**

HystUpper (HYU)

Décalage positif par rapport à la consigne qui est le niveau auquel **Output** va être activé. Les valeurs d'hystérésis sont relatives par rapport à la consigne.

HystLower (HYL)

Décalage négatif par rapport à la consigne qui est le niveau auquel **Output** va être désactivé. Les valeurs d'hystérésis sont relatives par rapport à la consigne.

Output (OP)

Cette sortie est activée lorsque **Process_Val** est supérieur à **Setpoint** plus HystUpper, désactivée lorsque **Process_Val** est inférieur à **Setpoint** moins HystLower et conserve sa valeur lorsqu'elle est située entre **Setpoint** plus **HystUpper** et **Setpoint** moins **HystLower.**

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Setpoint	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
HystUpper	**DINT**	1	Oper	Oper	Limite haute Limite basse	+2147483647 0
HystLower	**DINT**	1	Oper	Oper	Limite haute Limite basse	+2147483647 0
Output	**BOOL**	0	Oper	Block	Sens	On (1) Off (0)

Tableau 8-6 Attributs des paramètres Hysteresis Dint

BLOC FONCTION HYSTERESIS TIME

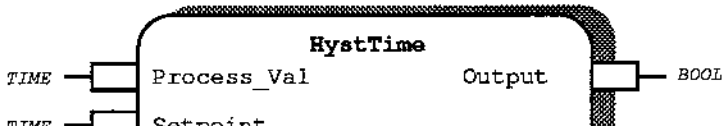


Figure 8-10 Schéma du bloc fonction Hysteresis Time

Description fonctionnelle

Lors du contrôle de valeurs analogiques, il existe de nombreuses situations dans lesquelles une certaine forme d'hystérésis est nécessaire. Ce bloc fonction permet de définir indépendamment un point d'hystérésis supérieur et un point d'hystérésis inférieur de chaque côté d'une consigne pour les valeurs de type temps. Lorsque **Process_Val** devient supérieur à la consigne plus la limite d'hystérésis supérieure HystUpper, **Output** passe à l'état On et y reste jusqu'à ce que **Process_Val** devienne inférieur à la consigne moins la limite d'hystérésis inférieure HystLower.

En réglant les valeurs d'hystérésis inférieure et supérieure, il est possible de mettre au point un certain nombre de stratégies différentes pour provoquer l'activation ou la désactivation de la sortie. En donnant à HystUpper la valeur **Setpoint,** il est possible de créer une hystérésis d'alarme haute classique : **Output** devient vrai lorsque **Setpoint** est dépassé et ne redevient faux que lorsque **Process_Val** est devenu inférieur à **Setpoint** moins HystLower.

En positionnant HystLower sur zéro et en inversant **Output** dans les instructions de câblage ultérieures, il est possible de créer une hystérésis d'alarme basse classique : **Output** devient faux lorsque **Process_Val** devient inférieur à **Setpoint** et ne redevient vrai que lorsque l'entrée est devenue supérieure à **Setpoint** plus HystUpper. L'utilisation en tandem d'HystUpper et HystLower offre des possibilités supplémentaires.

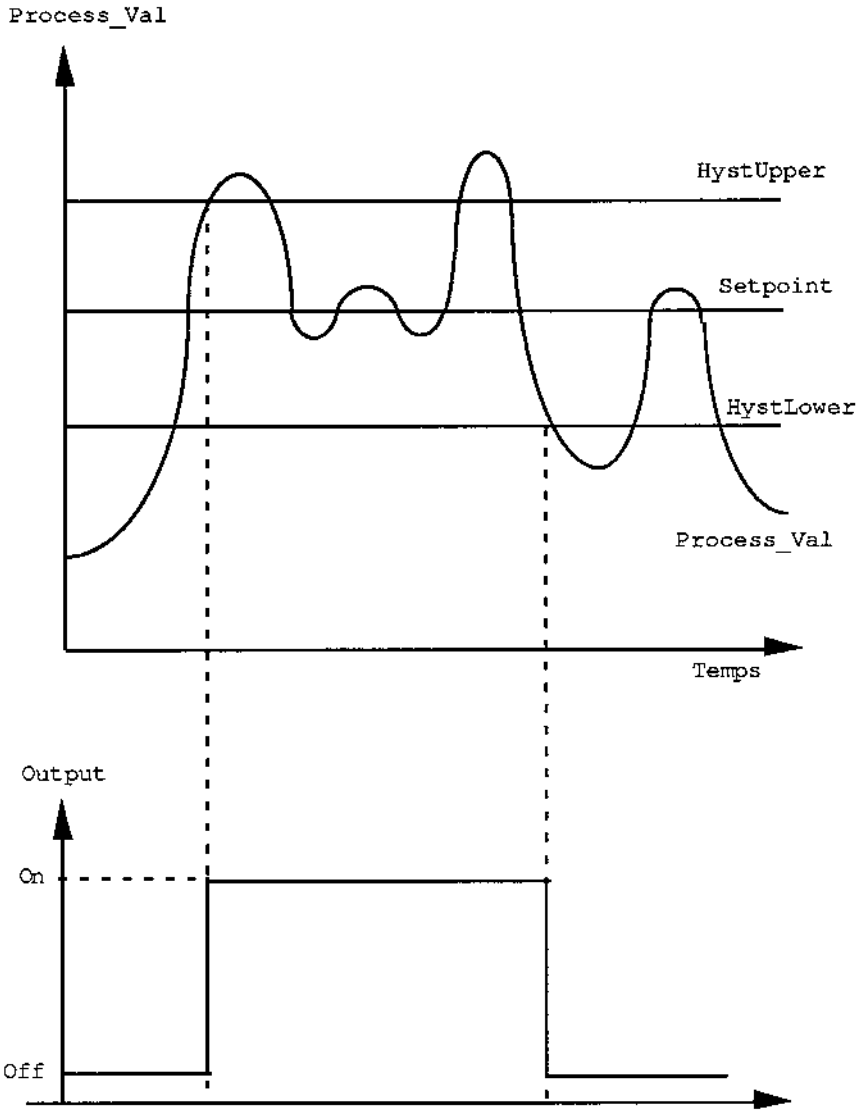


Figure 8-11 Hystérésis supérieure et inférieure

Attributs du bloc fonction

Type : 1C64
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : .....Process_Val, Setpoint, Output
Mémoire nécessaire : ..... 18 octets

Description des paramètres

Process_Val (PV)

Valeur à comparer à **Setpoint.** La variable sur laquelle une alarme est nécessaire doit être câblée sur cette entrée.

Setpoint (SP)

Valeur principale par rapport à laquelle il faut comparer **Process_Val.**

HystUpper (HYU)

Décalage positif par rapport à la consigne qui est le niveau auquel **Output** va être activé. Les valeurs d'hystérésis sont relatives par rapport à la consigne.

HystLower (HYL)

Décalage négatif par rapport à la consigne qui est le niveau auquel **Output** va être désactivé. Les valeurs d'hystérésis sont relatives par rapport à la consigne.

Output (OP)

Cette sortie est activée lorsque **Process_Val** est supérieur à **Setpoint** plus HystUpper, désactivée lorsque **Process_Val** est inférieur à **Setpoint** moins HystLower et conserve sa valeur actuelle lorsqu'elle est située entre **Setpoint** plus HystUpper et **Setpoint** moins HystLower.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	TIME	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Setpoint	TIME	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
HystUpper	TIME	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
HystLower	TIME	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Output	BOOL	Off (0)	Oper	Block	Sens	On (1) Off (0)

Tableau 8-7 Attributs des paramètres Hysteresis Time

BLOC FONCTION TOLERANCE REAL

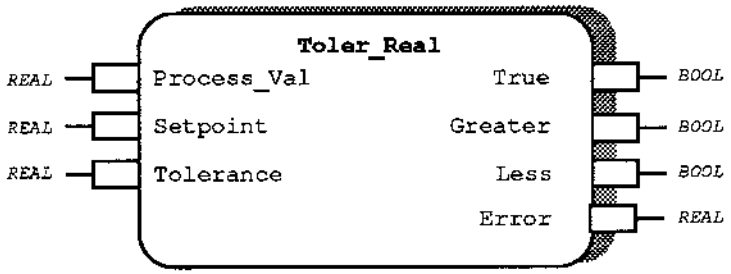


Figure 8-12 Schéma du bloc fonction Tolerance Real

Description fonctionnelle

La différence entre **Process_Val** entrant et une consigne est présentée sous la forme de la sortie Error. **Process_Val** est également comparée à **Setpoint** plus et moins **Tolerance.** Si **Process_Val** est supérieure à **Setpoint** plus Tolerance, seule la sortie Greater est vraie. Si **Process_Val** est inférieure à **Setpoint** moins Tolerance, seule la sortie Less est vraie. Si **Process_Val** est comprise entre **Setpoint** plus Tolerance et **Setpoint** moins Tolerance, la sortie True est vraie et Greater et Less sont fausses.

Ainsi, ce bloc fonction réunit en une seule fonction les informations nécessaires à une action de régulation simple : l'erreur absolue, une bande morte autour de la consigne et des voyants trop haut/trop bas.

Il est prévu pour être utilisé avec les variables de type réel.

Attributs du bloc fonction

Type : 1C70

Classe : ..... TRAITEMENT DES SIGNAUX

Tâche par défaut : ..... Task_2

Liste résumée : ..... Process_Val, Setpoint, Tolerance, True

Mémoire nécessaire : ..... 20 octets

Description des paramètres

Process_Val (PV)

Valeur à comparer avec **Setpoint** et **Tolerance.** La valeur **Process_Val** d'un bloc fonction d'entrée est normalement reliée à ce paramètre.

Setpoint (SP)

Valeur centrale de la bande de tolérance et valeur servant à calculer Error en liaison avec **Process_Val.**

Tolerance (TOL)

Largeur de la bande autour de **Setpoint** pour laquelle **True** est activé. Correspond à la bande morte en régulation simple augmentation/diminution.

True (T)

Cette sortie est vraie lorsque **Process_Val** est compris entre **Setpoint** plus Tolerance et Setpoint moins **Tolerance.**

True=Process_Val<(Setpoint+Tolerance)

ET Process_Val>(Setpoint-Tolerance)

Greater (G)

Lorsque **Process_Val** est supérieur ou égal à **Setpoint** plus **Tolerance,** cette sortie est vraie.

Greater=Process_Val>=(Setpoint+Tolerance)

Less (L)

Lorsque **Process_Val** est inférieur ou égal à **Setpoint** moins **Tolerance,** cette sortie est vraie.

Less=Process_Val<=(Setpoint-Tolerance)

Error (E)

Indique la différence absolue entre **Process_Val** et **Setpoint.**

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Setpoint	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Tolerance	**REAL**	1	Oper	Oper	Limite haute Limite basse	+3·402823E+38 0
True	**BOOL**	On (1)	Oper	Block	Sens	Off (0) On (1)
Grater	**BOOL**	Off (0)	Oper	Block	Sens	On (1) Off (0)
Less	**BOOL**	Off (0)	Oper	Block	Sens	On (1) Off (0)
Error	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 0

Tableau 8-8 Attributs des paramètres Tolerance Real

BLOC FONCTION TOLERANCE DINT

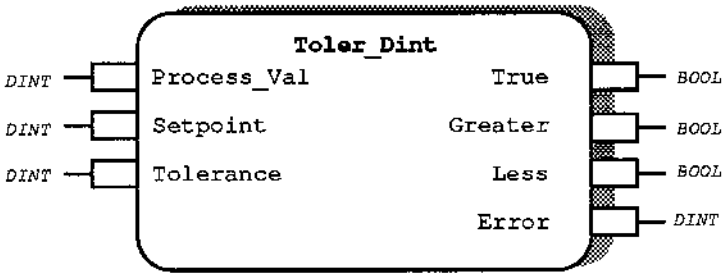


Figure 8-13 Schéma du bloc fonction Tolerance Dint

Description fonctionnelle

La différence entre **Process_Val** entrant et une consigne est présentée sous la forme de la sortie Error. **Process_Val** est également comparée à **Setpoint** plus et moins Tolerance. Si **Process_Val** est supérieure à **Setpoint** plus **Tolerance**, seule la sortie Greater est vraie. Si **Process_Val** est inférieure à **Setpoint** moins **Tolerance**, seule la sortie Less est vraie. Si **Process_Val** est comprise entre **Setpoint** plus **Tolerance** et **Setpoint** moins **Tolerance**, la sortie True est vraie et Greater et Less sont fausses.

Ainsi, ce bloc fonction réunit en une seule fonction les informations nécessaires à une action de régulation simple : l'erreur absolue, une bande morte autour de la consigne et des voyants trop haut/trop bas.

Il est prévu pour être utilisé avec les variables de type entier double.

Attributs du bloc fonction

Type : 1C72

Classe : .....TRAITEMENT DES SIGNAUX

Tâche par défaut : .....Task_2

Liste résumée : .....Process_Val, Setpoint, Tolerance, True

Mémoire nécessaire : .....20 octets

Description des paramètres

Process_Val (PV)

Valeur à comparer avec **Setpoint** et Tolerance. La valeur **Process_Val** d'un bloc fonction d'entrée est normalement reliée à ce paramètre.

Setpoint (SP)

Valeur centrale de la bande de tolérance et valeur servant à calculer Error en liaison avec **Process_Val.**

Tolerance (TOL)

Largeur de la bande autour de **Setpoint** pour laquelle True est activé. Correspond à la bande morte en régulation simple augmentation/diminution.

True (T)

Cette sortie est vraie lorsque **Process_Val** est compris entre **Setpoint** plus **Tolerance** et Setpoint moins **Tolerance.**

True=Process_Val<(Setpoint+Tolerance)

ET Process_Val>(Setpoint-Tolerance)

Greater (G)

Lorsque **Process_Val** est supérieur ou égal à **Setpoint** plus **Tolerance**, cette sortie est vraie.

Greater=Process_Val>=(Setpoint+Tolerance)

Less (L)

Lorsque **Process_Val** est inférieur ou égal à **Setpoint** moins **Tolerance**, cette sortie est vraie.

Less=Process_Val<=(Setpoint-Tolerance)

Error (E)

Indique la différence absolue entre **Process_Val** et **Setpoint.**

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Setpoint	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Tolerance	**DINT**	1	Oper	Oper	Limite haute Limite basse	+2147483647 0
True	**BOOL**	On (1)	Oper	Block	Sens	Off (0) On (1)
Grater	**BOOL**	Off (0)	Oper	Block	Sens	On (1) Off (0)
Less	**BOOL**	Off (0)	Oper	Block	Sens	On (1) Off (0)
Error	**DINT**	0	Oper	Block	Limite haute Limite basse	+2147483647 0

Tableau 8-9 Attributs des paramètres Tolerance Real

BLOC FONCTION TOLERANCE TIME

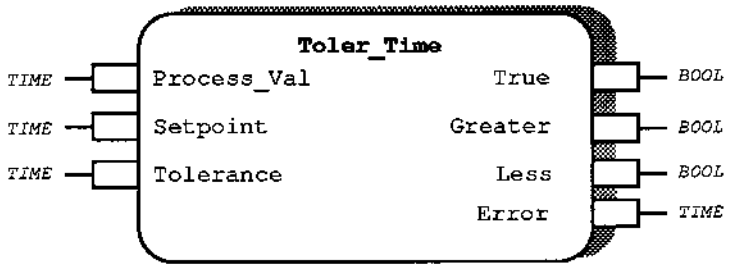


Figure 8-14 Schéma du bloc fonction Tolerance Time

Description fonctionnelle

La différence entre **Process_Val** entrant et une consigne est présentée sous la forme de la sortie Error. **Process_Val** est également comparée à **Setpoint** plus et moins Tolerance. Si **Process_Val** est supérieure à **Setpoint** plus Tolerance, seule la sortie Greater est vraie. Si **Process_Val** est inférieure à **Setpoint** moins Tolerance, seule la sortie Less est vraie. Si **Process_Val** est comprise entre **Setpoint** plus Tolerance et **Setpoint** moins Tolerance, la sortie True est vraie et Greater et Less sont fausses.

Ainsi, ce bloc fonction réunit en une seule fonction les informations nécessaires à une action de régulation simple : l'erreur absolue, une bande morte autour de la consigne et des voyants trop haut/trop bas.

Il est prévu pour être utilisé avec les variables de type temps.

Attributs du bloc fonction

Type : 1C74
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Process_Val, Setpoint, Tolerance, True
Mémoire nécessaire : ..... 20 octets

Description des paramètres

Process_Val (PV)

Valeur à comparer avec **Setpoint** et **Tolerance.** La valeur **Process_Val** d'un bloc fonction d'entrée est normalement reliée à ce paramètre.

Setpoint (SP)

Valeur centrale de la bande de tolérance et valeur servant à calculer Error en liaison avec **Process_Val.**

Tolerance (TOL)

Largeur de la bande autour de **Setpoint** pour laquelle **True** est activé. Correspond à la bande morte en régulation simple augmentation/diminution.

True (T)

Cette sortie est vraie lorsque **Process_Val** est compris entre **Setpoint** plus Tolerance et **Setpoint** moins Tolerance.

True=**Process_Val**<(Setpoint+Tolerance)
ET **Process_Val**>(Setpoint-Tolerance)

Greater (G)

Lorsque **Process_Val** est supérieur ou égal à **Setpoint** plus **Tolerance**, cette sortie est vraie.

Greater=Process_Val>=(Setpoint+Tolerance)

Less (L)

Lorsque **Process_Val** est inférieur ou égal à **Setpoint** moins **Tolerance**, cette sortie est vraie.

Less=Process_Val<=(Setpoint-Tolerance)

Error (E)

Indique la différence absolue entre **Process_Val** et **Setpoint.**

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Setpoint	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Tolerance	**TIME**	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
True	**BOOL**	On (1)	Oper	Block	Sens	Off (0) On (1)
Grater	**BOOL**	Off (0)	Oper	Block	Sens	On (1) Off (0)
Less	**BOOL**	Off (0)	Oper	Block	Sens	On (1) Off (0)
Error	**TIME**	0ms	Oper	Block	Limite haute Limite basse	23d23h59m59s999ms 0

Tableau 8-10 Attributs des paramètres Tolerance Time

BLOC FONCTION RELATIVE HUMIDITY AND DEW POINT

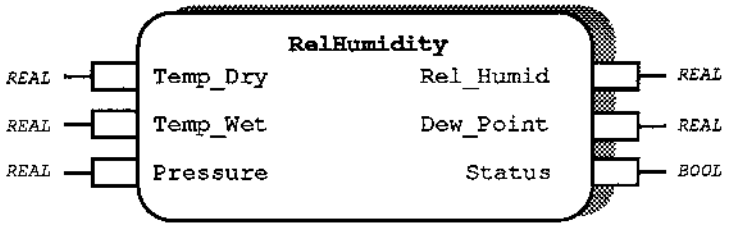


Figure 8-15 Schéma du bloc fonction Relative Humidity and Dew Point

Description fonctionnelle

Ce bloc fonction offre la possibilité de calculer l'humidité relative et le point de rosée à partir de deux températures en utilisant la technique standard qui fait appel à deux thermomètres sec et mouillé. Les deux températures sont normalement amenées dans le PC3000 par deux voies d'entrées analogiques quelconques. Des formules standard servent à calculer l'humidité relative et le point de rosée à partir des températures des thermomètres sec et mouillé. Pour avoir plus d'informations, consulter les normes BS 4833:1986, BS1399 et les manuels Eurotherm Automation 900EPC et Chessel Corporation 390 Recorder. Toutes les températures sont exprimées en degrés Celsius.

Différentes pressions atmosphériques sont prises en charge à l'aide de la broche d'entrée Pressure (en mBar).

Attributs du bloc fonction

Type : 1C80

Classe : .....TRAITEMENT DES SIGNAUX

Tâche par défaut : .....Task_2

Liste résumée : .....Temp_Dry, Temp_Wet, Rel_Humid, ...Dew_Point

Mémoire nécessaire : .....34 octets

Description des paramètres

Temp_Dry

La température du capteur du thermomètre sec doit être câblée sur cette entrée. Le traitement des signaux d'entrée utilisé doit donner une valeur en degrés Celsius sur cette entrée car les formules utilisées offrent uniquement cette possibilité.

Temp_Wet

La température du capteur du thermomètre humide doit être câblée sur cette entrée. Le traitement des signaux d'entrée utilisé doit donner une valeur en degrés Celsius sur cette entrée car les formules utiliées offrent uniquement cette possibilité.

Pressure

La valeur **Pressure** provenant d'un capteur de pression atmosphérique situé à proximité des thermocapteurs doit être reliée à cette entrée. Cette valeur doit être exprimée en millibars.

Rel_Humid

L'humidité relative de l'air entourant les thermocapteurs est indiquée par cette sortie sous forme d'un pourcentage. Se reporter aux normes BS 4833:1986 et BS1399 pour voir les formules utilisées pour ce calcul.

Si la température du thermomètre humide est supérieure à 100 degrés C (ébullition) ou inférieure à 0 degré C (gel), l'humidité relative est fixée respectivement à 100 % et 0 %. Si une combinaison de températures interdite donne une humidité relative supérieure à 100 % ou inférieure à 0 %, **Rel_Humid** est fixée respectivement à 100 % et 0 %.

Dew_Point

Température du point de rosée calculée à partir des températures des thermomètres sec et humide et de la pression ambiante. Si le thermomètre humide est supérieur à 100 degrés C (ébullition) ou inférieur à 0 degré C (gel), le point de rosée est fixé respectivement à la température du thermomètre humide ou à 0 degré C.

Si une combinaison de températures interdite donne une humidité relative inférieure à 0 %, le point de rosée est forcé à 0 degré C.

Status

Cette sortie indique si les valeurs d'entrée sont ou non acceptables.

Go(1) : les valeurs d'entrée sont situées dans les limites et les valeurs de sortie du point de rosée et de l'humidité relative sont valables.

Nogo(0) : une combinaison de températures ou une combinaison température/pression interdite est présentée aux entrées, provoquant l'échec des calculs.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Temp_Dry	**REAL**	26	Oper	Oper	Limite haute Limite basse	100 0
Temp_Wet	**REAL**	25	Oper	Oper	Limite haute Limite basse	100 0
Pressure	**REAL**	1013	Oper	Oper	Limite haute Limite basse	1200 800
Rel_Humid	**REAL**	92	Oper	Oper	Limite haute Limite basse	100 0
Dew_Point	**REAL**	25	Oper	Oper	Limite haute Limite basse	100 0
Status	**BOOL**	Go(1)	Oper	Oper	Sens	NOGO()) Go(1)

Tableau 8-11 Attributs des paramètres Relative Humidity and Dew Point

BLOC FONCTION RATE_LIMIT

Avant la version 3.00, ce bloc fonction se trouvait dans la classe de blocs DIVERS.

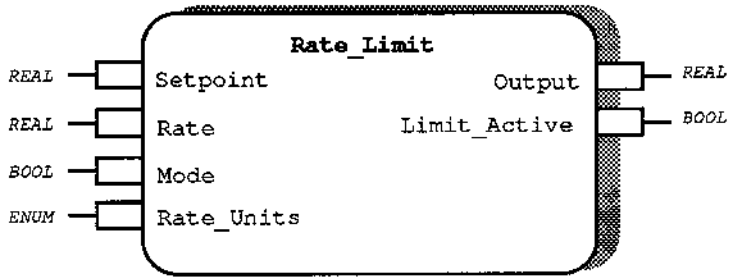


Figure 8-16 Bloc fonction Rate_Limit

Description fonctionnelle

Le bloc fonction **Rate_Limit** sert à limiter la vitesse maximale de changement d'un paramètre. Le paramètre dont la vitesse doit être limitée est entré dans **Setpoint** et la valeur limitée du paramètre est sortie par Output. La vitesse maximale de changement admissible d'Output est définie par Rate, les unités de Rate étant elles-mêmes définies par le paramètre **Rate_Units.** Lorsque la limitation de la vitesse a lieu, **Limit_Active** est positionné sur Limit (1). Ce bloc fonction possède deux modes de fonctionnement définis par le paramètre Mode.

Modes de fonctionnement

Track (0): en mode Track, la sortie Output suit Setpoint sans limitation de la vitesse.

Limit (1): en mode Limit, la vitesse maximale de changement d'Output est limitée à la valeur fixée par Rate.

Attributs du bloc fonction

Type : ICA0

Classe : ..... TRAITEMENT DES SIGNAUX

Tâche par défaut : ..... Task_2

Liste résumée : ..... Setpoint, Mode, Output

Mémoire nécessaire : ..... 32 octets

Durée d'exécution : ..... 298 µsec

Description des paramètres

Setpoint (SP)

Setpoint est l'entrée du bloc fonction dont la vitesse doit être limitée.

Rate (R)

Rate définit la vitesse maximale de changement à laquelle Output doit être limitée. Les unités de Rate sont définies par **Rate_Units.**

Mode (M)

Mode définit le mode de fonctionnement du bloc fonction, comme cela a été décrit antérieurement.

Rate_Units (RU)

Rate_Units définit les unités pour le paramètre **Rate. Rate_Units** peut être positionné sur quatre états :

/Seconde (0) :	vitesse en unités par seconde
/Minute (1):	vitesse en unités par minute
/Heure (2):	vitesse en unités par heure
/Jour (3):	vitesse en unités par jour

Output (OP)

Output est la sortie à vitesse limitée du bloc fonction. En mode Track (0), **Output** suit **Setpoint** sans limitation de vitesse. En mode Limit(1), **Output** suit **Setpoint** avec sa vitesse maximale de changement limitée à la valeur fixée par **Rate.**

Limit_Active (LA)

Limit_Output indique qu'il y a une limitation de la vitesse. Si le limiteur de vitesse est actif, **Output** est différent de Setpoint et **Limit_Output** est égal à Limit (1). Si **Output** est égal à **Setpoint,** le limiteur de vitesse n'est pas actif et **Limit_Output** est égal à Track (0).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Limit_Active	BOOL	Track (0)	Oper	Block	Sens	Track (0) Limit (1)
Mode	BOOL	Track (0)	Oper	Config	Sens	Track (0) Limit (1)
Output	REAL	0.0	Oper	Block	Limite haute Limite basse	10,000 -10,000
Rate	REAL	0.0	Oper	Oper	Limite haute Limite basse	1,000 0
Rate_Units	ENUM	/ Second (0)	Oper	Config	Sens	/ Seconde(0) / Minute (1) / Heure (2) / Jour (3)
Setpoint	REAL	0.0	Oper	Oper	Limite haute Limite basse	10,000 -10,000

Tableau 8-12 Attributs des paramètres Rate_Limit

BLOC FONCTION FREQUENCY COUNTER

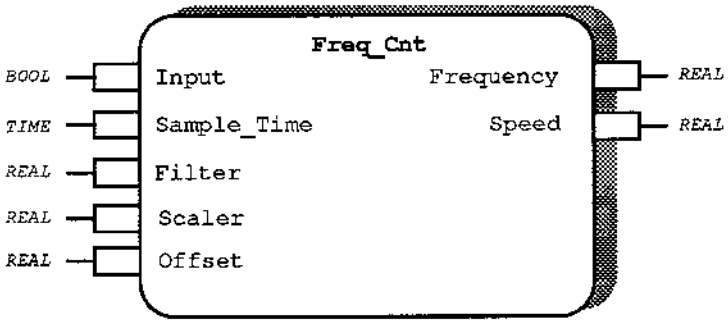


Figure 8-17 Bloc fonction Frequency Counter

Description fonctionnelle

Le bloc fonction **Freq_Cnt** accepte une entrée booléenne unique et fournit une sortie correspondant à la fréquence de l'entrée booléenne.

Des entrées numériques multiples (comme deux entrées en quadrature) peuvent être traitées dans le câblage de l'entrée de ce bloc.

Il existe aussi une possibilité de mise à l'échelle de la mesure de la fréquence pour donner une mesure de la vitesse en unités techniques.

Il faut noter que les entrées numériques ne peuvent être balayées que toutes les 5 msec au maximum, ce qui donne une fréquence théorique maximale de 100 Hz. Compte tenu des incertitudes du temps d'exécution, le maximum réel est de l'ordre de 60-70 Hz. Quelle que soit la fréquence de balayage de l'entrée numérique, le bloc fonction **Freq_Meter** doit être exécuté à la vitesse maximale possible afin de donner la résolution optimale des mesures de temps.

Attributs du bloc fonction

Type : ICC8

Classe : .....TRAITEMENT DES SIGNAUX

Tâche par défaut : .....Task_1

Liste résumée : .....Input, Sample_Time, Frequency, Speed

Mémoire nécessaire : .....48 octets

Description des paramètres

Input (IN)

Entrée d'impulsions dont la fréquence doit être mesurée.

Sample_Time (ST)

L'entrée **Sample_Time** spécifie la vitesse de mise à jour de la sortie Frequency et la vitesse de réaction au changement de fréquences. Il faut choisir **Sample_Time** soigneusement de façon à avoir un compromis entre la vitesse de mise à jour de la sortie, pour des raisons de régulation et d'ergonomie, et la régularité et la précision du résultat. Cette entrée **DOIT** être suffisamment longue pour garantir au moins une impulsion dans **Sample_Time** à la vitesse la plus faible mesurable. Si aucune impulsion n'arrive dans Sample_Time, le bloc fixe la sortie Frequency égale à zéro.

Filter (FLT)

L'entrée **Filter** spécifie le coefficient de filtrage qui sera utilisé dans le filtre de la sortie Frequency. Pour une première approximation, la constante de temps du filtre de sortie est donnée par l'expression

Constante de temps = Sample_Time * Filter

Scaler (SCL) et Offset (OFS)

Le bloc fonction utilise ces paramètres pour calculer la valeur de la sortie **Speed** qui est donnée par l'expression

Speed = Frequency * Scaler + Offset

Frequency (FRQ)

La sortie **Frequency** indique le nombre d'impulsions par seconde appliquées à **Input.** L'unité de ce paramètre est le Hz.

Speed (SPD)

Ce paramètre lie la fréquence aux unités techniques de la vitesse. Les paramètres **Scaler** et **Offset** servent à définir les coefficients de conversion. **Speed** est lié à la fréquence par l'expression

Speed = Frequency * Scaler + Offset

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Sample_Time	**TIME**	2s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0s
Filter	**REAL**	1.0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 0
Scaler	**REAL**	1.0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 0
Offset	**REAL**	0.0	Oper	Oper	Limite haute Limite basse	+3402823E+38 -3·402823E+38
Frequency	**REAL**	0.0	Oper	Block	Limite haute Limite basse	+3·402823E+38 0
Speed	**REAL**	0.0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 8-13 Attributs des paramètres Frequency Counter

BLOC FONCTION ASTABLE CYCLE TIME

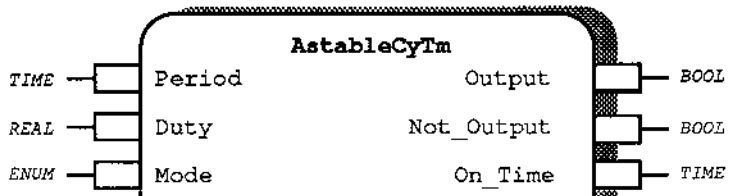


Tableau 8-18 Bloc fonction AsTable Cycle Time

Description fonctionnelle

Un train d'impulsions régulier est produit sur la sortie : il faut pour cela spécifier le temps de répétition (Period) et le pourcentage de cette période (**Period**) pendant lequel le bloc fonction **Output** doit être actif (Duty). Il est possible de faire varier **Period** et **Duty** pour obtenir des impulsions de la longueur souhaitée qui se répètent à intervalles réguliers.

A titre d'exemple, supposons que l'on choisisse une période (**Period**) de 4 sec puis que l'on fixe Duty à 50 % : **Output** sera actif pendant 2 sec puis inactif pendant 2 sec. Cette disposition va ensuite se répéter de manière continue. Si Duty passe à 25 %, **Output** sera actif pendant 1 sec puis inactif pendant 3 sec.

Attributs du bloc fonction

Type : 1CCA
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Period, Duty, Mode, Output
Mémoire nécessaire : ..... 32 octets

Description des paramètres

Period (PER)

Durée totale du cycle marche/arrêt. Peut également être considérée comme l'intervalle de répétition du bloc fonction.

Duty (DTY)

Pourcentage de **Period** où **Output** est actif et **Not_Output** est inactif.

Mode (M)

Le paramètre Mode sert à réguler le fonctionnement des blocs fonctions :

Reset : l'ensemble des horloges internes sont ramenées à zéro, **Output** est sur Off(0) et **Not_Output** sur On(1).

Run : **Output** et **Not_Output** sont mis à jour en fonction des réglages de **Period** et de **Duty.**

Hold : **Output** et **Not_Output** conservent les valeurs actuelles et les horloges internes sont maintenues. Le fait de ramener **Mode** sur **Run** provoque la poursuite de la mise à jour des sorties à partir de l'endroit qu'elles ont quitté.

Wait : la partie actuelle du cycle est terminée, **Output** et **Not_Output** changent encore une fois d'état puis **Output** et **Not_Output** conservent les nouvelles valeurs et les horloges internes sont maintenues. Le fait de ramener **Mode** sur **Run** provoque la poursuite de la mise à jour des sorties à partir du dernier changement d'état.

Output (OP)

Lorsque le bloc fonction passe en mode Run, la sortie reste désactivée pendant une durée égale à Period*(100-Duty) puis activée pendant une durée égale à Period*Duty. Cette séquence se répète ensuite.

Not_Output (NOP)

Not_Output est toujours la condition inverse d'**Output.**

On_Time (ONT)

Spécifie la durée pendant laquelle la sortie sera activée pendant cette période de marche. Lors de la modification de **Duty** ou de **Period**, cette valeur sera mise à jour lors du commencement de la prochaine période de marche.

Off_Time (OFT)

Spécifie la durée pendant laquelle la sortie sera désactivée pendant cette période d'arrêt. Lors de la modification de **Duty** ou de **Period**, cette valeur sera mise à jour au début de la prochaine période d'arrêt.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Period	**TIME**	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0ms
Duty	**REAL**	50%	Oper	Oper	Limite haute Limite basse	99% 1%
Mode	**ENUM**	Reset (0)	Oper	Oper	Cf. liste des paramètres	
Output	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Not_Output	**BOOL**	On (1)	Oper	Block	Sens	On (1) Off (0)
On_Time	**TIME**	0ms	Oper	Block	Limite haute Limite basse	23d23h59m59s999ms 0ms
Off_Time	**TIME**	0ms	Oper	Block	Limite haute Limite basse	23d23h59m59s999ms 0ms

Tableau 8-14 Attributs des paramètres AsTable Cycle Time

BLOC FONCTION ASTABLE ON/OFF TIME

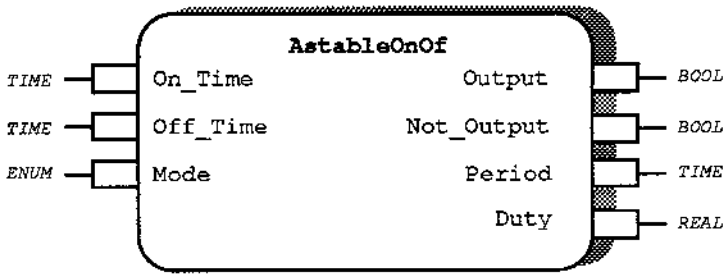


Figure 8-19 Bloc fonction AsTable Cycle Time

Description fonctionnelle

Un train d'impulsions régulier est produit sur la sortie : il faut pour cela spécifier le temps d'activation et le temps de désactivation de chaque impulsion. Les impulsions se répètent de manière régulière. La période (somme de **On_Time** et **Off_Time**) et la duré d'activation (**On_Time** sous la forme d'un pourcentage de la période) apparaissent sous la forme de sorties.

A titre d'exemple, supposons que l'on choisisse **On_Time** égal à 4 sec puis **Off_Time** égal à 2 sec. La sortie sera active pendant 4 sec puis inactive pendant 2 sec. Cette disposition va ensuite se répéter de manière continue. **Duty** apparaîtra comme égal à 66 % et **Period** comme égal à 6 sec.

Attributs du bloc fonction

Type : 1CCC

Classe : ..... TRAITEMENT DES SIGNAUX

Tâche par défaut : ..... Task_2

Liste résumée : ..... On_Time, Off_Time, Mode, Output

Mémoire nécessaire : ..... 32 octets

Description des paramètres

On_Time (ONT)

Spécifie la durée pendant laquelle la sortie sera active au cours de chaque cycle.

Off_Time (OFT)

Spécifie la durée pendant laquelle la sortie sera inactive au cours de chaque cycle.

Mode (M)

Le paramètre **Mode** sert à réguler le fonctionnement des blocs fonctions :

Reset : l'ensemble des horloges internes sont ramenées à zéro, **Output** est sur Off(0) et **Not_Output** sur On(1).

Run : **Output** et **Not_Output** sont mis à jour en fonction des réglages de **On_Time** et de **Off_Time.**

Hold : **Output** et **Not_Output** conservent les valeurs actuelles et les horloges internes sont maintenues. Le fait de ramener Mode sur Run provoque la poursuite de la mise à jour des sorties à partir de l'endroit qu'elles ont quitté.

Wait : la partie actuelle du cycle est terminée, **Output** et **Not_Output** changent encore une fois d'état puis **Output** et **Not_Output** conservent les nouvelles valeurs et les horloges internes sont maintenues. Le fait de ramener **Mode** sur Run provoque la poursuite de la mise à jour des sorties à partir du dernier changement d'état.

Output (OP)

Lorsque le bloc fonction passe en mode Run, la sortie reste désactivée pendant une durée égale à **Off_Time** puis activée pendant une durée égale à **On_Time.** Cette séquence se répète ensuite.

Not_Output (NOP)

Not_Output est toujours la condition inverse d'**Output.**

Period (PER)

Durée totale du cycle marche/arrêt. Peut également être considérée comme l'intervalle de répétition du bloc fonction.

Duty (DTY)

Pourcentage de **Period** où **Output** est actif et **Not_Output** est inactif.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
On_Time	**TIME**	500ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0ms
Off_Time	**TIME**	500ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0ms
Mode	**Enum**	Reset (0)	Oper	Oper	cf. liste des paramètres	
Output	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Not_Output	**BOOL**	On (1)	Oper	Block	Sens	Off(0) On (1)
Period	**TIME**	1s	Oper	Block	Limite haute Limite basse	23d23h59m59s999ms 0ms
Duty	**REAL**	50%	Oper	Block	Limite haute Limite basse	100% 0%

Tableau 8-15 Attributs des paramètres AsTable On/Off Time

BLOC FONCTION TOGGLER

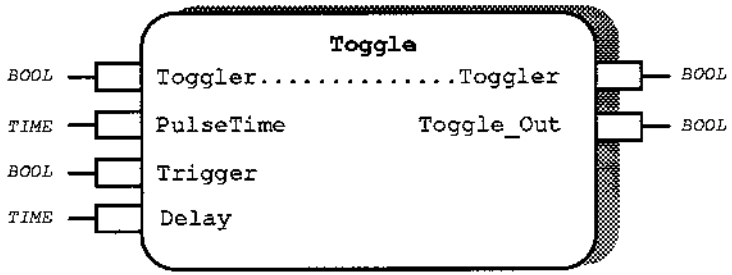


Figure 8-20 Bloc fonction Toggler

Description fonctionnelle

Le paramètre d'entrée/sortie **Toggler** peut être activé directement et peut rester actif pendant une période spécifiée par l'entrée **PulseTime**. Il devient aussi actif après une temporisation spécifiée par **Delay** si l'entrée **Trigger** est active. Dans ce cas, **Toggler** redevient inactif après avoir été actif pendant une période spécifiée par l'entrée **PulseTime**. **Trigger** provoque le retour de **Toggler** à l'état actif uniquement si **Trigger** redevient inactif puis actif ; en d'autres termes, l'action se produit sur le front ascendant de **Trigger.**

Attributs du bloc fonction

Type : 1CCE
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Toggler, Trigger, Toggle_Out
Mémoire nécessaire : ..... 26 octets

Description des paramètres

Toggler (TGR)

Il s'agit d'une entrée/sortie. Une fois activée, elle reste active pendant une période spécifiée par **PulseTime**. Elle peut être activée directement ou à l'aide de l'entrée Trigger. Dans ce dernier cas, **Toggler** devient actif une fois que **Trigger** est vrai, la temporisation étant spécifiée par **Delay**.

PulseTime (PT)

Période pendant laquelle **Toggler** reste actif, indépendamment de toute action externe.

Trigger (TRG)

Provoque le passage de **Toggler** à l'état vrai après une temporisation spécifiée par Delay. **Trigger** doit être ramené à l'état faux par le programme utilisateur et n'agit à nouveau sur Toggler que si **Trigger** revient à l'état faux puis à l'état vrai. Si **Toggler** est déjà actif lorsque Trigger est positionné sur "vrai", **Trigger** n'a aucun effet.

Delay (DLY)

Temporisation entre le passage de **Trigger** à l'état vrai et le passage de **Toggler** à l'état vrai.

Toggle_Out (TGO)

Cette sortie modifie son état à chaque fois que **Trigger** fait passer Toggler d'Off à On. Lorsque **Toggler** fait passer State d'On à Off, la sortie **Toggle_Out** ne change pas. Cette sortie bascule donc à chaque font de montée de Toggler.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Toggler	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
PulseTime	**TIME**	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0ms
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Delay	**TIME**	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0s
Toggle_Out	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On(1)

Tableau 8-16 Attributs des paramètres Toggler

BLOC FONCTION TOTALIZER

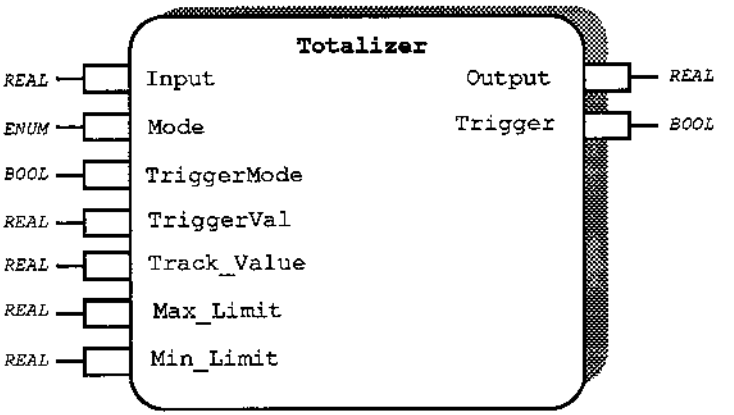


Figure 8-21 Bloc fonction Totalizer

Description fonctionnelle

Ce bloc offre un moyen de totaliser une entrée en fonction du temps. Il est entièrement bidirectionnel, c'est-à-dire que la sortie peut augmenter ou diminuer selon le signe de l'entrée. Ce bloc fonctionne de telle manière que, avec une entrée constante, une valeur égale à **Input** est ajoutée à **Output** toutes les secondes. L'intervalle de la tâche à laquelle le bloc fonction est affecté n'a aucune répercussion. La valeur de sortie se situe entre une limite supérieure et une limite inférieure.

Une broche de commande **Mode** permet de réinitialiser le bloc puis de provoquer son exécution. Il est également possible de figer **Output** sur sa valeur actuelle ou de lui faire suivre une autre entrée indépendante.

Une sortie booléenne verrouillable indique si **Output** dépasse une limite haute ou une limite basse.

Attributs du bloc fonction

Type : 1CD0
Classe : ..... TRAITEMENT DES SIGNAUX
Tâche par défaut : ..... Task_2
Liste résumée : ..... Input, Mode, Output, Trigger.
Mémoire nécessaire : ..... 46 octets

Description des paramètres

Input (IN)

Valeur à totaliser.

Mode (M)

Régule le fonctionnement du bloc fonction.

Reset: **Output** est remis à zéro et **Trigger** est remis sur Off.

Run: A chaque exécution du bloc fonction, la moyenne de la valeur de l'entrée lors de la dernière exécution du bloc et de la valeur actuelle de l'entrée est ajoutée à **Output** (intégration trapézoïdale). Lorsqu'Output repasse de Hold ou Track à **Run**, sa mise à jour commence à partir de sa valeur actuelle.

Hold: **Output** est figée sur sa valeur actuelle et aucune entrée n'a d'effet sur les sorties du bloc fonction.

Track: Output suit la valeur de l'entrée **Track_Value.** Les changements des entrées n'ont aucun effet.

TriggerMode (TM)

Détermine si la sortie **Trigger** devient vraie lorsqu'**Output** devient supérieur ou inférieur à **TriggerVal.**

TriggerVal (TVL)

Niveau auquel **Trigger** devient vrai lorsqu'**Output** lui devient supérieur ou inférieur (dépend de **TriggerMode**).

Track_Value (TRV)

La valeur de la sortie suit cette entrée lorsque **Mode** est positionné sur Track.

Max_Limit (MAX)

Output ne dépassera pas cette valeur, soit en raison de l'intégration d'**Input** soit parce qu'elle suit **Track_Value**.

Min_Limit (MIN)

Output ne sera pas inférieur à cette valeur, soit en raison de l'intégration d'**Input** soit parce qu'elle suit **Track_Value**.

Output (OP)

Lorsque **Mode** est réinitialisé, **Output** est égal à zéro.

En mode Run, à chaque cycle d'exécution, la valeur suivante est ajoutée à **Output** :

$$\frac{(I(last) + I(current)) / 2}{t(task)}$$

où : I(last) = valeur d'Input lors de la dernière évaluation du bloc fonction.

I(current) = valeur actuelle d'Input.

t(task) = intervalle de la tâche à laquelle le bloc fonction est affecté.

Il y a une seule exception : lorsque l'addition ferait sortir **Output** des limites imposées par **Max_Limit** ou **Min_Limit**. Dans ces conditions, Output reste sur **Max_Limit** ou **Min_Limit** (selon la limite qui ne serait pas respectée).

En mode Hold, **Output** est figé sur sa valeur actuelle.

En mode Track, **Output** affiche la valeur figurant sur l'entrée **Track_Value** à condition qu'elle se trouve entre **Max_Limit** et **Min_Limit**.

Trigger (TRG)

Trigger est une sortie booléenne verrouillable qui sert à indiquer qu'une condition d'alarme a lieu en mode Run. Si **TriggerMode** est sur Upper (supérieur), **Trigger** devient vrai lorsque **TriggerVal** est dépassé puis le reste jusqu'à ce que Mode passe à Reset ou Track : Trigger repasse alors sur Off.

Si **TriggerMode** est sur Lower (inférieur), **Trigger** devient vrai lorsqu'**Output** devient inférieur à **TriggerVal** puis reste vrai jusqu'à ce que **Mode** passe à Reset ou Track : **Trigger** repasse alors sur Off.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Mode	**ENUM**	Reset (0)	Oper	Oper	Cf. liste des paramètres	
TriggerMode	**BOOL**	Upper (0)	Oper	Oper	Sens	Upper (0) Lower (1)
TriggerVal	**REAL**	0	Oper	Oper	Limite haute Limite basse	Max_Limit Min_Limit
Track_Value	**REAL**	0	Oper	Oper	Limite haute Limite basse	Max_Limit Min_Limit
Max_Limit	**REAL**	100	Oper	Oper	Limite haute Limite basse	+3·402823E+38 Min_Limit
Min_Limit	**REAL**	-100	Oper	Oper	Limite haute Limite basse	Max_Limit -3·402823E+38
Output	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Trigger	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)

Tableau 8-17 Attributs des paramètres Totalizer

BLOC FONCTION CUSTOM LINEARISATION

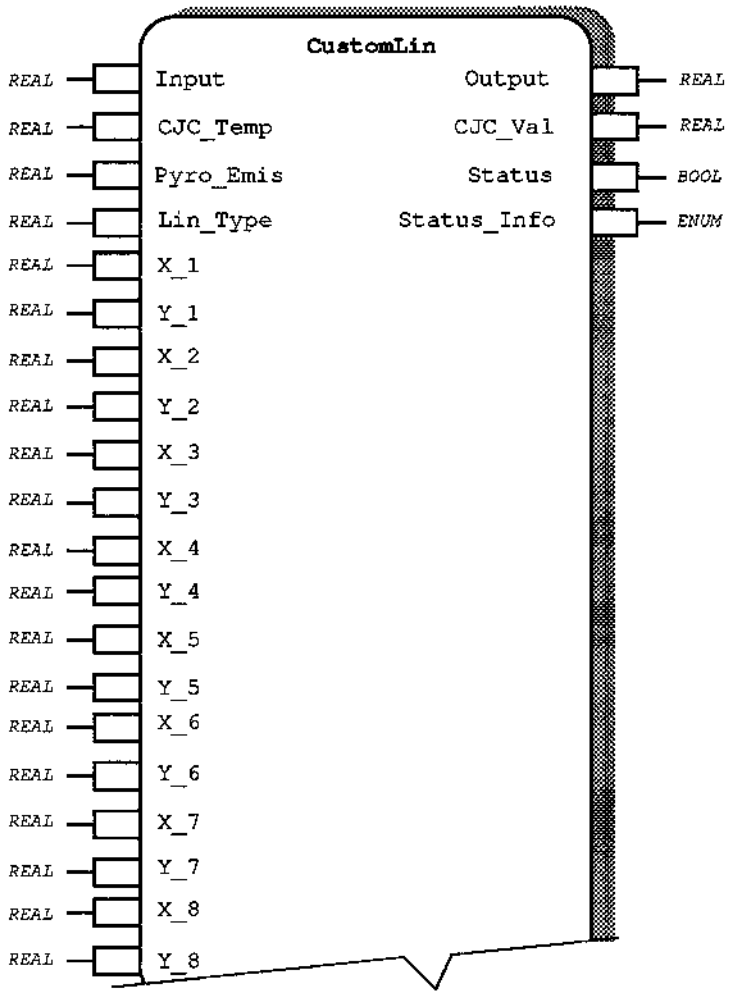


Figure 8-22 Bloc fonction Custom Linearisation

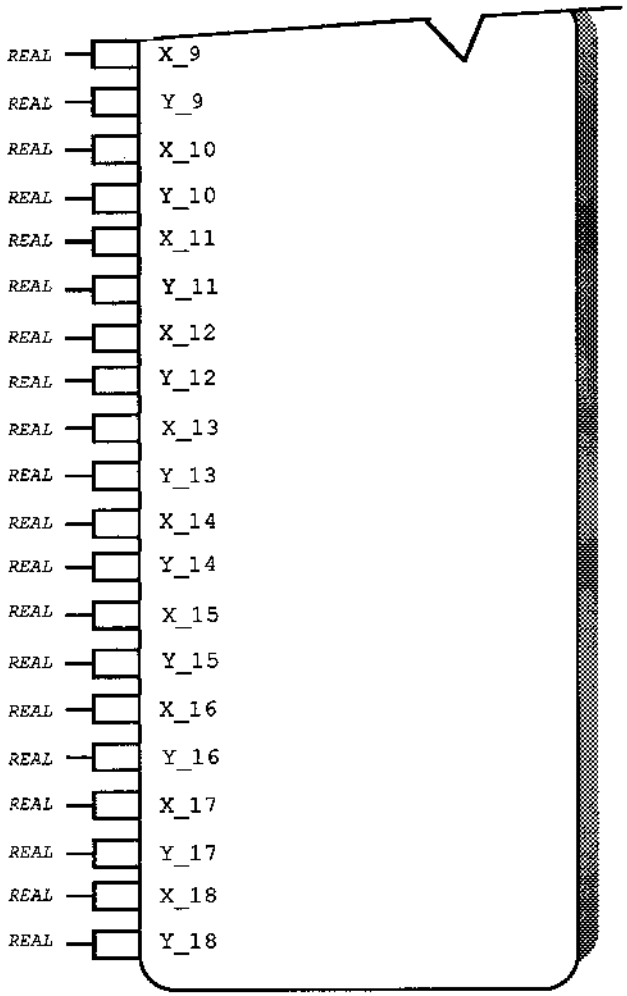


Figure 8-22 Bloc fonction Custom Linearisation (suite)

Description fonctionnelle

Les sorties de capteurs industriels sont souvent non linéaires par rapport à la variable physique mesurée. Les caractéristiques de linéarisation des transducteurs les plus courants sont fournies de manière standard avec les blocs fonctions **Analog_In**. Le bloc fonction Custom Linearisation permet d'utiliser un maximum de dix-huit paires de sorties de transducteur et la valeur de la variable physique pour définir une caractéristique de linéarisation pour les capteurs moins courants. Il est possible d'obtenir ces paires de données par étalonnage ou en se reportant aux caractéristiques fournies par le fabricant.

Un ajustage linéaire est utilisé entre les paires de mesures.

Les valeurs des sorties du transducteur (valeurs X) et les valeurs associées de la variable physique (valeurs Y) forment dix-huit paires d'entrées pour ce bloc fonction. Les valeurs X sont saisies par ordre de grandeur croissant. S'il y a moins de dix-huit paires de données, il est possible de mettre fin au tableau en saisissant une valeur X inférieure à la précédente. Par exemple, s'il n'y a que six paires de données avec la valeur X maximale égale à 300, l'entrée X_6 du bloc fonction serait 300 et X_7 devrait être une valeur quelconque inférieure à X_6 (la valeur par défaut zéro conviendrait parfaitement). Toutefois, si les six valeurs X sont négatives avec la valeur la plus élevée (c'est-à-dire la mois négative) égale à -90, X_6 serait égale à -90 et X_7 devrait être inférieure à -90, -91 par exemple. Si X_7 reste à la valeur par défaut zéro et si Y_7 est également égal à zéro, le tableau utilisera 0,0 comme paire de données valable.

Lors de la réalisation de mesures de la température, il est possible d'utiliser la compensation de soudure froide (CJC) ou l'émissivité de pyromètre comme élément du processus de linéarisation. Pour appliquer la compensation de soudure froide, il faut saisir la température de soudure froide. Cette valeur remonte ensuite le processus de linéarisation pour donner la compensation d'entrée qui est ensuite appliquée à l'entrée avant la linéarisation. L'émissivité sert à mettre à l'échelle la valeur de l'entrée avant la linéarisation.

Attributs du bloc fonction

Type : 1CDC

Classe : ..... TRAITEMENT DES SIGNAUX

Tâche par défaut : ..... Task_2

Liste résumée : ..... Input, Output, Status, Status_Info.

Mémoire nécessaire : ..... 198 octets

Description des paramètres

Input (IN)

Valeur de la sortie capteur industriel.

CJC_Temp (CJC)

Température de la soudure froide lorsqu'on utilise un thermocouple comme capteur. Cette valeur est uniquement utilisée si **Lin_Type** est positionné sur CJC.

Pyro_Emis (PEM)

Emissivité de l'élément examiné par le pyromètre. Cette valeur est uniquement utilisée si **Lin_Type** est positionné sur **PyrEms.**

Lin_Type (LT)

Détermine s'il faut effectuer des corrections sur la compensation de soudure froide ou l'émissivité du pyromètre.

No_CJC:	il ne faut apporter des corrections ni sur la compensation de soudure froide ni sur l'émissivité du pyromètre.
CJC:	**Input** doit être corrigé en fonction de la température de soudure froide spécifiée par **CJC_Temp.**
PyrEms:	il faut mettre l'entrée à l'échelle en utilisant le facteur spécifié par **Pyro_Emis.**

X_1 (X1)

Première valeur de la sortie capteur. Point de départ du tableau sorties capteur - valeurs des variables physiques mesurées.

Y_1 (Y1)

Première valeur de la variable physique donnant la sortie capteur X_1. Point de départ du tableau sorties capteur - valeurs des variables physiques mesurées.

X_2 (X2)

Deuxième valeur de la sortie capteur. Si cette valeur n'est pas supérieure à la valeur précédente, le tableau se termine par la valeur précédente.

Y_2 (Y2)

Deuxième valeur de la variable physique donnant la sortie capteur X_2.

Répétition pour chaque paire (X,Y) jusqu'à :

X_18 (X18)

Dix-huitième valeur de la sortie capteur. Si cette valeur n'est pas supérieure à la valeur précédente, le tableau se termine par la valeur précédente.

Y_18 (Y18)

Dix-huitième valeur de la variable physique donnant la sortie capteur X_18.

Output (OP)

Valeur linéarisée.

CJC_Val (CJV)

Décalage résultant de la compensation de soudure froide.

Status (ST)

Indique normalement un état Go. Toutefois, si **Input** se trouve en dehors des limites définies par le tableau (après correction de la compensation de soudure froide ou de l'émissivité, le cas échéant) ou si **CJC_Temp** se trouve en dehors des limites définies par le tableau, Status est sur be NoGo.

Status_Info (STI)

Explique les éventuels problèmes d'état.

Ok: l'état est sur Go.

Over_R:	**Input** est supérieur à la limite supérieure définie par le tableau.
Under_R:	**Input** est inférieur à la limite inférieure définie par le tableau.
CJCOver:	**CJC_Temp** est supérieur à la limite supérieure définie par le tableau.
CJCUndr:	**CJC_Temp** est inférieur à la limite inférieure définie par le tableau.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
CJC_Temp	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Pyro_Emis	**REAL**	0	Oper	Oper	Limite haute Limite basse	1 0
Lin_Type	**ENUM**	No_CJC (0)	Oper	Block	Cf. liste des paramètres	
X_1	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_1	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_2	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_2	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_3	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_3	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_4	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_4	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_5	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_5	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_6	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_6	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_7	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_7	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 8-16 Attributs des paramètres Custom Linearisation

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
X_8	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_8	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_9	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_9	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_10	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_10	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_11	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_11	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_12	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_12	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_13	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_13	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_14	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_14	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_15	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_15	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_16	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_16	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 8-16 Attributs des paramètres Custom Linearisation (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
X_17	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_17	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
X_18	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Y_18	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Output	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
CJC_Val	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Status	**BOOL**	NOGO (0)	Oper	Block	Sens	NOGO (0) Go (1)
Status_Info	**ENUM**	Over_R (1)	Block	Oper	Cf. liste des paramètres	

Tableau 8-16 Attributs des paramètres Custom Linearisation (suite)

BLOC FONCTION RANDOM

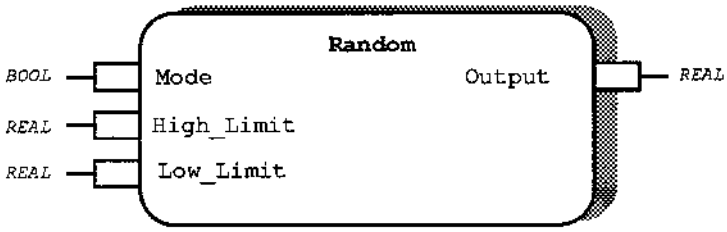


Figure 8-18 Schéma du bloc fonction Random

Description fonctionnelle

Ce bloc fonction produit en continu des nombres aléatoires dans une plage spécifiée. Une entrée permet de réinitialiser la valeur de départ de l'algorithme de nombres aléatoires.

Attributs du bloc fonction

Type : 1CFA
Classe : .....TRAITEMENT DES SIGNAUX
Tâche par défaut : .....Task_2
Liste résumée : .....Low_Limit, High_Limit, Mode, Output.
Mémoire nécessaire : .....26 octets

Description des paramètres

Mode (M)

En mode Run, le bloc fonction produit un nouveau nombre aléatoire à chaque cycle d'exécution du bloc fonction. En mode Reset, la valeur de départ de l'algorithme de nombres aléatoiresest réinitialisée

High_Limit (HL)

Les nombres aléatoires produits seront inférieurs ou égaux à la valeur de cette entrée.

Low_Limit (LL)

Les nombres aléatoires produits seront supérieurs ou égaux à la valeur de cette entrée.

Output (OP)

Le nombre aléatoire produit est présenté sur cette sortie lorsque Mode est sur Run. Lorsque Mode est réinitialisé, **Output** est égal à **Low_Limit**.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Mode	**BOOL**	Run (0)	Oper	Oper	Sens	Run (0) Reset (1)
High_Limit	**REAL**	1	Oper	Oper	Limite haute Limite basse	+3·402823E+38 Low_Limit
Low_Limit	**REAL**	0	Oper	Oper	Limite haute Limite basse	High_Limit -3·402823E+38
Output	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 8-19 Attributs des paramètres Random

Chapitre 9

REGULATION

Edition 1

Sommaire (suite)

Sommaire (suite)

Introduction

Le présent chapitre décrit les blocs fonctions de la classe REGULATION comprenant une gamme de blocs fonctions de régulation PID qui, lorsqu'ils sont utilisés avec des blocs fonctions INPUTS et des blocs fonctions OUTPUTS, permettent de configurer des boucles de régulation. Les régulations, comportant ou non des algorithmes de réglage automatique et adaptatif, ainsi que les dispositifs associés permettant la régulation de vannes motorisées, sont également inclus.

Des stratégies de régulations plus complexes, telles que la régulation en cascade ou la programmation du gain, peuvent être mises en oeuvre par interconnexion de plusieurs blocs fonctions de régulation.

Il est recommandé de lire la présentation de la régulation PC3000 du présent chapitre, en tant qu'introduction à la fonction de régulation assurée par le système PC3000. Ce chapitre comporte des informations sur les sujets concernés, tels que la mise en cascade, la régulation de ratio et la programmation du point de consigne.

PRESENTATION DE LA REGULATION

PREFACE

Etendue

Le présent document recouvre les domaines de fonctionnalité suivants :

Présentation des régulations tout ou rien et PID

Relation entre la fonctionnalité ci-dessus et la série de blocs fonctions de régulation Eurotherm

Configuration de boucle/câblage par soft élémentaires

Méthodes standard de réglage manuel de boucle

Mécanismes Eurotherm de réglage auto-réglant et adaptatif

Programmation du point de consigne

Programmation du gain

Régulation en cascade

Régulation de ratio

Régulation en boucle ouverte

Autres Stratégies

But

Le but du présent document est de décrire quelques méthodes et techniques standard utilisées pour la conduite de process au moyen des blocs fonctions intégrés au PC3000. Il peut être utilisé comme introduction préalable aux principes de la régulation. Quelques notions fondamentales relatives aux systèmes de régulation sont toutefois nécessaires pour comprendre certains chapitres plus techniques du présent document.

Introduction

La conduite d'un process ou d'une machine implique la coordination d'un grand nombre de paramètres. L'un de ces paramètres, qui est traditionnellement confié à des instruments discrets, est la régulation de quelques grandeurs mesurables du process. Il s'agit, en général, de variables comme la température, la pression, le débit, l'humidité, le niveau, la composition, etc. Les blocs fonctions PC3000 sont prévus pour réaliser des stratégies de régulation en continu adaptées au process, avec différentes caractéristiques de réponse.

En plus de la régulation de base, un grand nombre de paramètres de la stratégie de régulation sont concernés par la conduite des machines et l'instrumentation. Parmi ces paramètres, on peut citer les mises en marche, les mises à l'arrêt, les paramètres

en cas de rupture de capteur, la dégradation des performances en mode sécurité, le contrôle des séquences de fonctionnement pour s'assurer que l'opérateur est prévenu de la maintenance de routine, de l'émission des comptes rendus séquentiels, de la commutation sans à coup entre formulations, etc. En outre, le système de régulation gère les verrouillages, la logique séquentielle, la synchronisation avec les autres équipements, etc. Ces paramètres reposent, pour la plupart, sur des conditions "if then else" (si... alors.... sinon...) et peuvent être programmés en associant le câblage par soft et les grafcets.

La plupart des process sont, par principe, non linéaires. Pour pouvoir les conduire d'une façon optimale, il est parfois nécessaire de combiner la logique "conditionnelle" avec des fonctions de régulation du genre PID. En fait, dans de nombreuses applications, il est nécessaire de procéder ainsi. A titre d'exemple, on peut citer les procédures de mise en route et de mise à l'arrêt qui sont indispensables dans un processus de régulation. Ces schémas de régulation reposant sur des règles simples peuvent être établis sous forme de pas, de macros ou de câblage avec des programmes utilisateurs PC3000. Ces activités étant essentiellement spécifiques à l'application, elles ne sont pas reprises ici. Les informations relatives à la régulation séquentielle de process figurent dans le Guide utilisateur PC3000, Livre 2 : Langages.

Deux méthodes essentielles permettent la régulation en continu : la "régulation en boucle fermée" et la "régulation en boucle ouverte". Dans une régulation en boucle fermée, la variable (appelée habituellement "valeur de process") est mesurée et comparée à la valeur voulue, le "point de consigne". Une décision de régulation (par exemple réglage d'une valeur de sortie) résulte de cette comparaison.

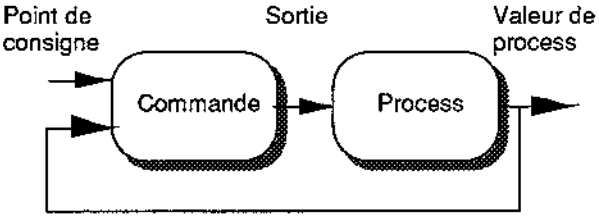


Figure 9-1 Régulation en boucle fermée

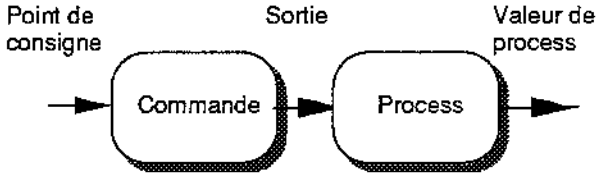


Figure 9-2 Régulation en boucle ouverte

La Figure 9-1 représente une régulation à rétroaction simple, par exemple pour un four électrique. La "Valeur de consigne" est une mesure de la température et le "Point de consigne", la température voulue pour le four. "Sortie" peut être un signal proportionnel au temps pour une sortie logique ou un triac. La régulation est en général un algorithme Proportionnel, Intégral et Dérivé (PID). En régulation boucle ouverte, la décision de régulation s'effectue sans mesurer la Valeur de process. La Figure 9-2 représente la régulation en boucle ouverte sous sa forme la plus simple. La régulation d'un four à sa pleine puissance pendant un temps donné, sans mesurer sa température dans un but de préchauffage, est un exemple simple de régulation en boucle ouverte. Pour une régulation précise des variables de process, ces deux stratégies doivent fréquemment être associées. Normalement, la régulation en boucle fermée demande moins de connaissances sur le process que la régulation en boucle ouverte Le but principal de la régulation en boucle fermée peut être résumé comme suit :

Réduction de l'incertitude du process (par exemple réduire les variations de température dues aux conditions d'environnement telles que l'ouverture ou la fermeture de la porte du four et les variations de son chargement)

Stabilisation des systèmes instables en boucle ouverte (par exemple les réactions exothermiques).

La régulation en boucle ouverte est utilisée, par contre, essentiellement pour faire face aux retards ou pour compenser l'effet des influences externes, telles que les signaux de régulation d'autres boucles du process.

TYPES DE RÉGULATION

Régulation tout ou rien

Une régulation tout ou rien, ou à deux positions, est la forme la plus simple de régulation en boucle fermée. Dans sa forme la plus simple, elle peut consister en :

```
DOP.Process_Val  :=  Setpoint.Val > ANIN.Process_Val;
```

où DOP et ANIN sont les exemples respectifs de blocs fonctions de sortie logique et d'entrée analogique. La sortie est totalement "On" (active) lorsque la valeur de process (mesurée par ANIN) est inférieure au point de consigne et "Off" (désactivée) lorsque la valeur est égale ou supérieure au point de consigne. Ce schéma de régulation oscille en permanence à une vitesse qui dépend du gain et de l'échelle de temps du process concerné. Voir la figure 9-3, qui montre un exemple de schéma de sortie d'une voie unique (dans ce cas de chauffage uniquement).

Pour résoudre le problème d'oscillation asymétrique représenté sur la figure 9-3, ainsi que le risque de battement du relais, une hystérésis (ou bande morte) peut être prévue. La puissance est coupée lorsque la valeur de processus atteint le point de consigne, mais n'est réenclenchée que si la valeur de process descend en dessous du point de consigne d'une quantité supérieure à la valeur d'hystérésis (ou de la bande morte) choisie par l'utilisateur. Ceci peut être réalisé par le câblage simple suivant, en utilisant la fonction SEL_BOOL:

```
DOP.Process_Val := SEL_BOOL( G := DOP.Process_Val,
                   IN0 := ANIN.Process_Val < Setpoint.Val - Hyst.Val,
                   IN1 := ANIN.Process_Val < Setpoint.Val);
```

Le résultat de l'algorithme ci-dessus est représenté sur la figure 9-4. Si les voies de chauffage ainsi que les voies de refroidissement sont présentes, l'algorithme peut être complété pour réaliser un réglage tel que représenté par la figure 9-5.

Une autre catégorie de régulation tout ou rien (parfois associée à la régulation de position de vanne) est l'utilisation de bandes mortes. Ici, il n'y a plus d'hystérésis. Dans le cas de deux voies, la sortie 1 est active lorsque l'écart est important ou négatif, la sortie 2 est active lorsque l'écart est important et positif ; enfin, les deux sorties sont inactives lorsque l'écart se situe dans une zone choisie par l'utilisateur. La figure 9-6 montre les caractéristiques d'une telle régulation tout ou rien.

```
DOP1.Process_Val := (Setpoint.Val - ANIN.Process_Val) > DB1.Val;
DOP2.Process_Val := (ANIN.Process_Val - Setpoint.Val) > DB2.Val;
```

Le câblage de la régulation tout ou rien avec bande morte est le suivant :

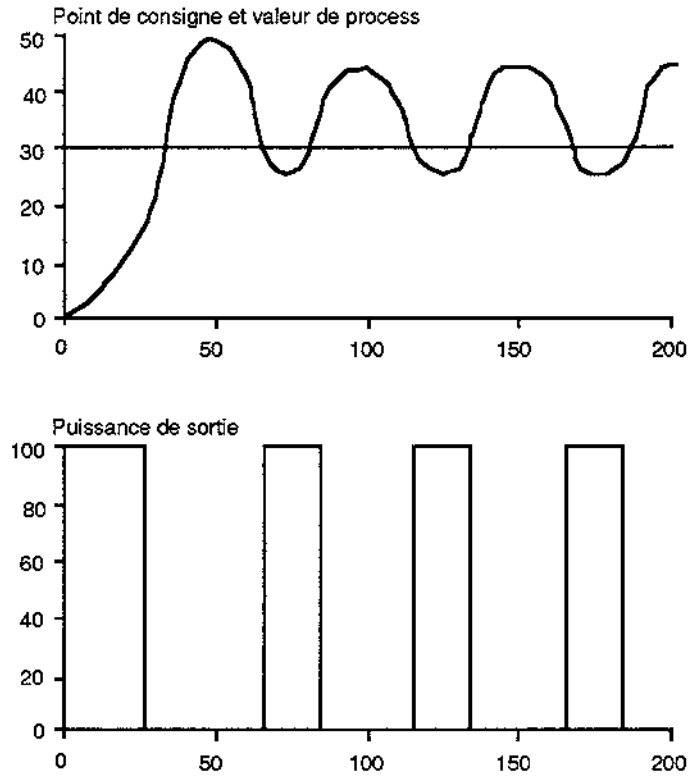


Figure 9-3 Régulation tout ou rien élémentaire

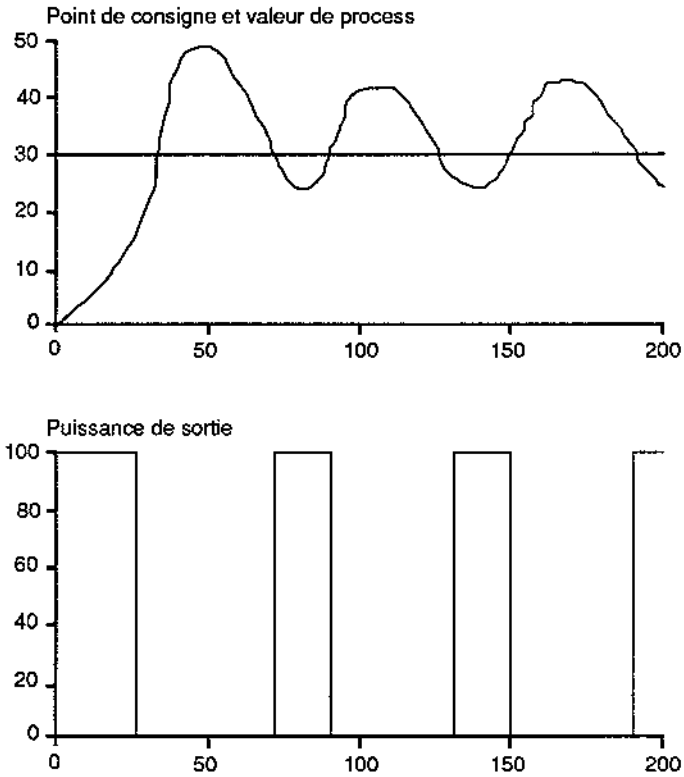


Figure 9-4 Régulation tout ou rien avec hystérésis

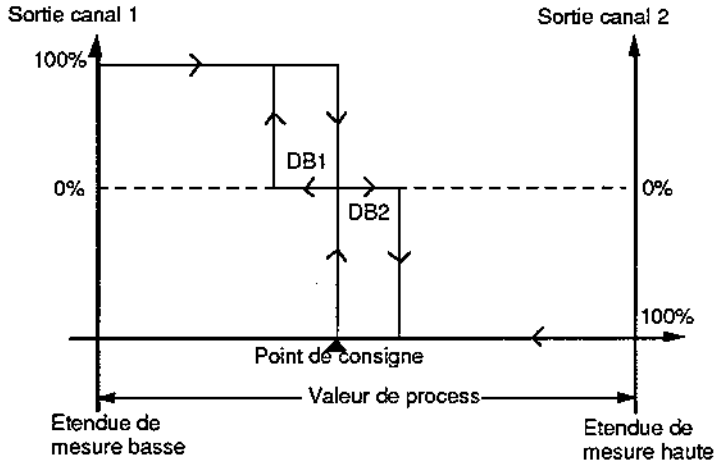


Figure 9-5 Régulation tout ou rien à deux voies avec hystérésis

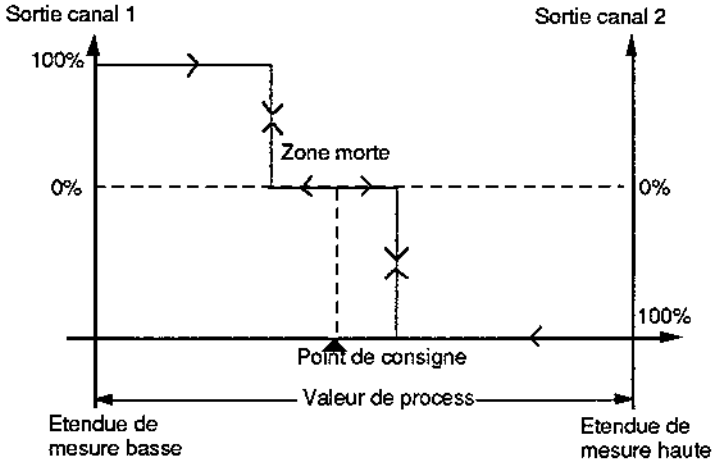


Figure 9-6 Régulation tout ou rien avec bande morte

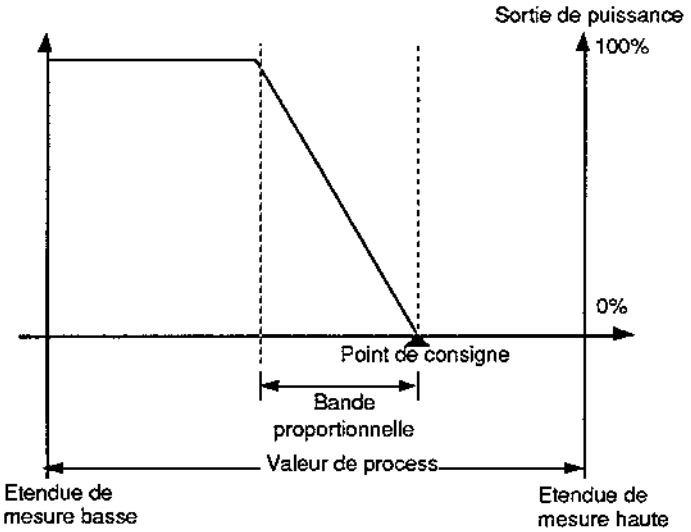


Figure 9-7 Régulation proportionnelle

Régulation PID

Régulation proportionnelle

En fait, le problème essentiel de la régulation tout ou rien est que la variable de process ne s'établit jamais au point de consigne. Une régulation plus précise peut être obtenue si, au lieu de prévoir une hystérésis ou une bande morte, une sortie proportionnelle à l'écart entre la valeur de process et le point de consigne pouvait être appliquée. C'est ce que représente la figure 9-7. Une sortie à 100 % est appliquée si la valeur de process dépasse le réglage de la bande proportionnelle située sous le point de consigne. Elle décroît alors linéairement de 100 % à 0 % lorsque la valeur de process se rapproche du point de consigne. La sortie est mise à zéro pour des valeurs de process supérieures au point de consigne. Si plusieurs voies de sortie sont utilisées, la bande proportionnelle peut être élargie en conséquence. Les limites habituelles sont appliquées à la valeur de process et au point de consigne entre Span_Low et Span_High. Le terme "bande proportionnelle" peut prêter à confusion, car il se désigne deux entités distinctes mais associées :

La zone ou bande dans laquelle la sortie est une fonction linéaire de l'écart

Le réglage (c'est-à-dire la largeur) de la zone de régulation linéaire.

Dans le présent document, la première entité est appelée la bande proportionnelle et la seconde le réglage - ou la largeur - de la bande proportionnelle. Dans l'ensemble de blocs fonctions du PC3000, la largeur est réglée par le paramètre "Prop_Band". La largeur de la bande proportionnelle dans le système PC3000 est réglée en tant que pourcentage de l'étendue de mesure (écart entre Span_High et Span_Low).

Prenons un four : à l'état stable, la chaleur totale fournie équilibre les pertes. Ceci implique que pour tout four ayant des pertes, il y aura un écart entre le point de consigne et la valeur de process à l'état d'équilibre. La valeur de l'erreur est visiblement fonction de la taille de la bande proportionnelle : plus la bande proportionnelle est étroite, plus l'erreur à l'état stable sera faible. Toutefois, plus la bande proportionnelle est étroite, plus la sortie se rapproche d'une régulation tout ou rien et tend ainsi à osciller. Ce qui signifie qu'il y a une limite au-delà de laquelle la diminution de la bande proportionnelle se fait au détriment des performances de la boucle.

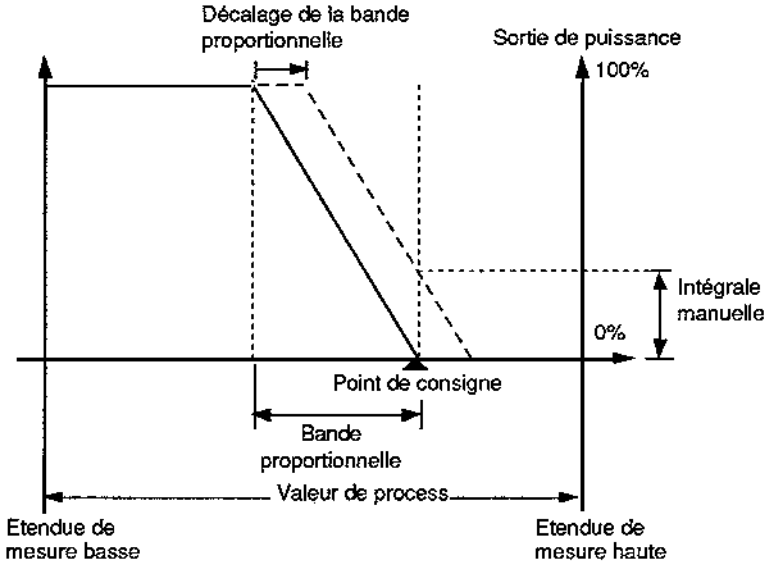


Figure 9-8 Influence de l'intégrale manuelle

Régulation proportionnelle et intégrale (PI)

Pour corriger les erreurs à l'état d'équilibre, l'une des deux stratégies suivantes peut être utilisée :

La fonction "intégrale manuelle" peut être utilisée. L'intégral ou le réajustage manuel, comme son nom l'indique, peut être utilisé pour annuler l'erreur manuellement. La valeur du réajustage manuel est choisie en pourcentage de la plage de sortie et est normalement de 50 %. Le réajustage manuel est ajouté au terme proportionnel. Le réajustage manuel du réglage peut être assimilé graphiquement à un décalage latéral de la bande proportionnelle de MR x PB x étendue de mesure / 10000, MR étant la valeur Manual_Reset et PB la largeur de la bande proportionnelle en pourcentage de l'étendue de mesure. La figure 9-8 représente la relation entre le réajustage manuel et la bande proportionnelle.

Un "réajustage automatique" ou une régulation intégrale peuvent être utilisés. Avec la régulation intégrale, la sortie de la régulation est proportionnelle à l'intégration de l'erreur dans le temps.

$$\text{Integral\_Out} = \frac{10000}{\text{Span} \times \text{PB} \times T_i} \int e(t)dt$$

$$e(t) = \text{Setpoint} - \text{Process\_Value}$$

Ti est le temps d'intégration choisi par l'utilisateur, et est en général défini comme étant le temps nécessaire à la bande proportionnelle pour se décaler d'une unité pour une erreur unitaire constante. La bande proportionnelle continue à se décaler jusqu'au moment où l'erreur est mise à zéro à l'état stable. Le terme intégral peut être considéré comme étant le réglage de la valeur moyenne de la régulation, tandis que le terme proportionnel réagit aux perturbations à court terme. La valeur du temps d'intégration peut être par conséquent choisie en fonction de la constante de temps du process en boucle ouverte et de la réponse souhaitée en boucle fermée. Pour plus de détails concernant le réglage des boucles de régulation, voir Réglage PID.

Pour la régulation proportionnelle et intégrale, la contribution de la régulation proportionnelle et la contribution de la régulation intégrale s'ajoutent. Il y a lieu de noter que le réajustage automatique et le réajustage manuel s'excluent mutuellement. La régulation proportionnelle et intégrale est la méthode de régulation en boucle fermée la plus courante.

Régulation proportionnelle, intégrale et dérivée (PID)

La régulation PI peut résoudre la plupart des problèmes de régulation. Toutefois, l'action de régulation de la régulation PI est toujours rétrospective. Les régulations PI montrent très souvent des dépassements et un temps de récupération important après une perturbation.

Considérons un process ayant une grande inertie thermique. Si le point de consigne de la régulation PI est augmenté, la séquence d'évènements suivante sera observée. La partie proportionnelle de la régulation va provoquer un saut initial de la sortie. La valeur de process va commencer à se déplacer très lentement en direction du nouveau point de consigne. Pendant ce temps, le terme intégral va continuer à intégrer l'erreur entre le point de consigne et la valeur de process. Ceci aura pour effet d'augmenter graduellement la sortie de la régulation. L'évènement le plus vraisemblable est que la sortie de régulation, ainsi que la contribution du terme intégral vont croître nettement plus que la nouvelle valeur d'équilibre recherchée. Le seul moyen pour la régulation de revenir à la nouvelle valeur d'équilibre est de changer le signe de l'erreur : la valeur de process doit donc dépasser son nouvel objectif. Pour les process ayant une inertie thermique importante, le dépassement peut ne pas être acceptable.

La régulation dérivée est un moyen permettant de traiter ce type de process. Prenons le terme d'erreur e(t). Pour résoudre le problème de dépassement, la contribution du terme proportionnel doit changer de signe avant que l'erreur instantanée e(t) ne le fasse. Une façon simple et efficace consiste à introduire un terme d'anticipation, de façon à ce que la régulation, au lieu de surveiller e(t), surveille une estimation de e(t + Td) (tendance en avance de Td secondes sur l'erreur).

En première approximation, ce terme est donné par :

$$e(t+T_d) \approx e(t)+T_d\frac{de(t)}{dt}$$

qui est, en fait, l'extrapolation linéaire de l'erreur sur la base de sa vitesse d'évolution instantanée. Si la régulation proportionnelle fonctionne avec cette prévision d'erreur, le problème du dépassement est en grande partie supprimé. Il en résulte que la régulation devient :

$$\text{Output} = \frac{10000}{\text{Span x Prop\_Band}}\left[e(t)+\frac{1}{T_i}\int e(t)+T_d\frac{de(t)}{dt}\right]$$

C'est la formule standard qui régit la régulation PID. Dans beaucoup d'applications, il est habituel de régler une valeur de gain K au lieu du réglage de bande proportionnelle. Le gain K est égal à :

$$K = \frac{10000}{\text{Span x Prop\_Band}}$$

La régulation PID a un autre effet bénéfique. Prenons un four n'ayant qu'une régulation PI. La chute de température due à l'ouverture de la porte peut être très rapide. Si la régulation comporte une bande proportionnelle large, la réponse de la régulation risque d'être très lente. Il est donc nécessaire d'ajuster la bande proportionnelle en fonction de cette vitesse de variation. Si la valeur de process s'écarte rapidement de l'objectif, la largeur de la bande proportionnelle réelle doit être réduite. Ceci améliore le temps de récupération du process régulé. Le terme dérivée du PID effectue cette opération de rétrécissement de la bande proportionnelle effective.

La figure 9-9 montre la différence entre les régulations PI et PID pour un changement de valeur de consigne et la réjection des perturbations. Les deux graphiques du haut représentent respectivement le point de consigne et la valeur de process, et la puissance de sortie de la régulation PI. Il y a lieu de noter le dépassement lors du changement initial de point de consigne. Les deux graphiques du bas représentent la réponse de la régulation PID dans le même process. Il y a lieu de noter l'amélioration du temps de récupération avec la régulation PID à la suite d'une perturbation, ainsi que l'élimination du dépassement initial. En contrepartie, il y a une activité accrue du signal de sortie.

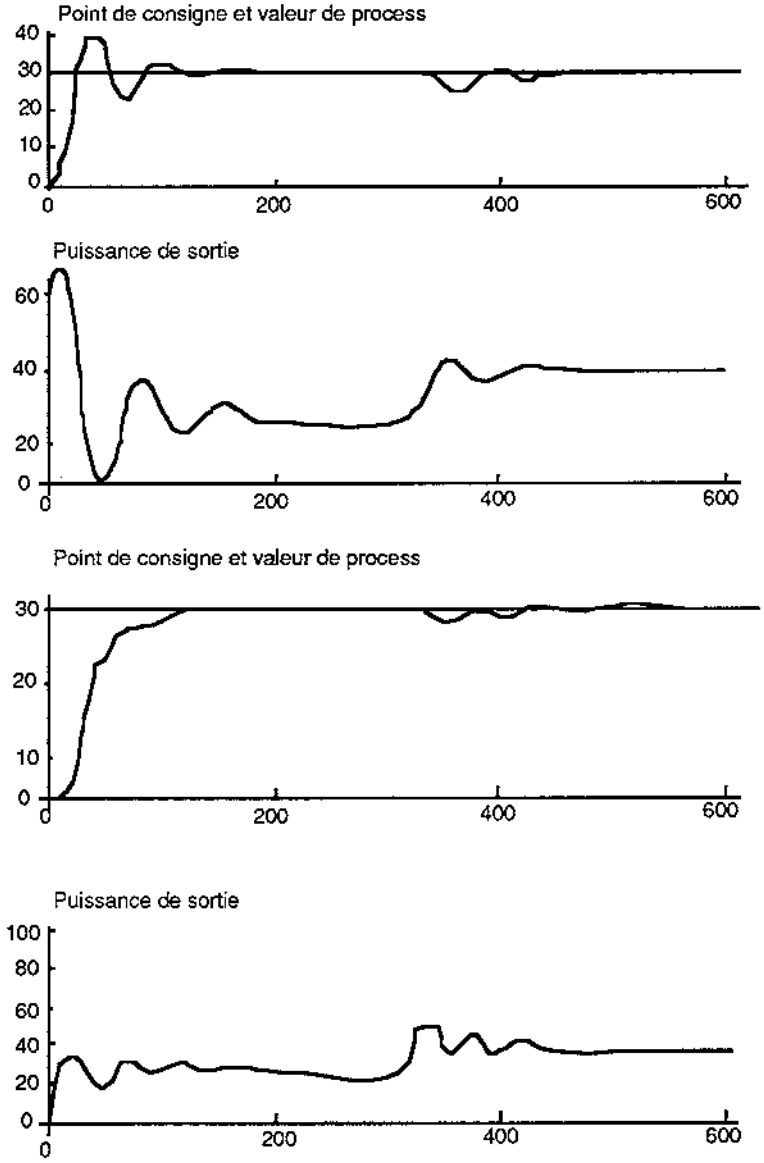


Figure 9-9 Commandes PI et PID

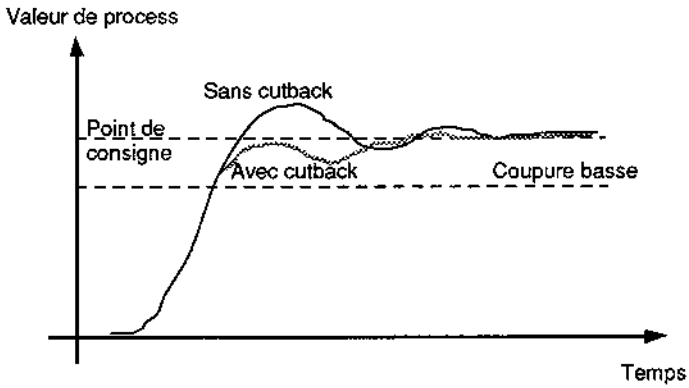


Figure 9-10 Cutback en régulation PID

En pratique, le terme dérivée est filtré par un filtre passe-bas effectuant un tri grossier pour éliminer les effets nuisibles du bruit d'amplification aux fréquences élevées. En outre, dans les applications de régulation de process, le terme dérivée est pris à partir de la valeur de process et non à partir du signal d'erreur.

L'algorithme Eurotherm est un PID non interactif. Si les valeurs sont connues pour une boucle d'interaction PID, la transformation en version non interactive est donnée par :

$$PB = T'_i \times PB' / (T'_i + T'_d)$$
$$Ti = T'_i + T'_d$$
$$Td = T'_i T'_d / (T'_i + T'_d)$$

Les (') indiquent les réglages du PID interactif correspondant. Il y a lieu de noter que PB et PB' doivent être exprimées soit en unités techniques, soit en pourcentage de l'étendue de mesure. Les blocs fonctions PC3000 nécessitent une bande proportionnelle égale à un pourcentage de l'étendue de mesure.

Cutback

Pour la plupart des process, les régulations PID élémentaires et leurs sous-ensembles (P, PI ou PD) conviennent, dans la mesure où les changements de paramètres de fonctionnement ne sont pas trop importants. Pour des écarts importants, une action énergique et immédiate est nécessaire. Le cutback assure cette fonction. Si la valeur de process est inférieure à (Point de consigne - Cutback bas), la puissance maximale sera appliquée en sortie, et si la valeur de process est supérieure à la valeur (Point de consigne + Cutback bas), la puissance minimale sera appliquée en sortie. Comme le montre la figure 9-10, si le dépassement est provoqué par une saturation de la régulation, un cutback peut être appliqué pour

compenser ce dépassement et donner la réponse représentée sur la figure 9-10. Lorsque la valeur de process franchit cette "bande de cutback", la régulation est commutée en PID standard sans saut. Un choix judicieux des valeurs de cutback peut améliorer la réponse sur un déplacement important du point de consigne ainsi que le temps de récupération après une perturbation du système régulé.

Les valeurs de cutback peuvent être ajustées en mode manuel ou en mode automatique dans le cas des blocs fonctions PID_Auto et VP_Auto. Une stratégie judicieuse d'ajustage consiste à :

Régler les valeurs de cutback haute et basse à 0 (désactivation)

Lorsque la réponse aux faibles perturbations (celles qui ne nécessitent pas de niveau de sortie maximal / minimal) est satisfaisante, régler les paramètres de cutback égaux à la bande proportionnelle

Mesurer la réponse aux déplacements importants du point de consigne et juger si les caractéristiques de dépassement sont satisfaisantes

Modifier le réglage des paramètres de cutback pour améliorer les caractéristiques de dépassement et vérifier le résultat. Le déplacement du point de consigne doit être considérablement supérieur au réglage du cutback. La modification du réglage de cutback doit être approximativement égale à l'augmentation ou à la diminution requise pour le dépassement. Normalement, une diminution du paramètre de cutback augmente le dépassement (et diminue le temps d'échauffement) et vice versa. Si le réglage du cutback est trop large, il est totalement inopérant. Une valeur de zéro est également équivalente à sa désactivation.

Le cutback est parfois considéré comme étant le point où la régulation commence à réduire la puissance de sortie à partir de sa valeur maximale. Toutefois, ce n'est le cas que si la vitesse d'évolution de la valeur de process reste dans une plage spécifique. Si l'on considère le cas où l'on s'approche par le bas de la zone de cutback, la valeur de process étant inférieure au point de consigne moins la valeur de cutback bas, la puissance de sortie est maximale. Ceci est parfois exprimé comme étant le blocage de la bande proportionnelle à la valeur de cutback bas. Lorsque la valeur de process coupe le point de cutback bas, la régulation est commutée en PID standard. La régulation ne commencera à réduire la puissance que si la vitesse de croissance du signal de régulation due à l'action d'intégration est compensée par la vitesse de décroissance due aux termes proportionnel et dérivé. En termes plus généraux, ceci signifie que la régulation ne va commencer à réduire la puissance au point de cutback que si la vitesse de décroissance de la valeur de process est supérieure à la valeur de cutback divisée par le temps d'intégration. Ce qui signifie à son tour que la vitesse de croissance de la valeur mesurée doit être supérieure au déplacement de la bande proportionnelle.

Considérons un process subissant des dépassements lors des déplacements importants du point de consigne. Si la réponse du process est acceptable pour les déplacements faibles du point de consigne mais ne l'est pas pour des déplacements importants, il est possible que la régulation ne réduise pas suffisamment vite la puissance de sortie. Un cutback peut être alors très utile. Régler le cutback sur la bande proportionnelle en unités physiques. Mesurer le dépassement éventuel du process sur un déplacement équivalent du point de consigne. Noter également la vitesse maximale de variation de la valeur de process. Régler la nouvelle valeur du cutback sur l'ancienne plus le dépassement. Si cette valeur divisée par le temps d'intégrale est toujours inférieure à la vitesse maximale de variation, le cutback éliminera le dépassement. Dans le cas contraire, régler la valeur de cutback sur le temps d'intégration multiplié par la vitesse maximale de variation, et essayer d'améliorer la réponse en utilisant, si possible, l'action dérivée.

Dans beaucoup de domaines d'application, le régleur doit trouver un compromis entre la vitesse de démarrage et le dépassement. Là aussi, le cutback peut être utilisé efficacement pour améliorer la réponse. Le régleur peut trouver un compromis simple entre le degré de dépassement et la vitesse de démarrage.

Le cutback est, par conséquent, un outil de base pour la régulation avec des écarts de signal importants, les réglages élémentaires PID étant utilisés pour la réponse de la régulation à de faibles variations de signal.

PID à deux voies

Prenons les zones limites d'une extrudeuse. Le chauffage est effectué par des éléments chauffants électriques et le refroidissement par des ventilateurs. Les signaux de chauffage et de refroidissement sont donnés sur deux voies distinctes. Il y a lieu de noter que les gains, normalement, sont différents pour le chauffage et pour le refroidissement. Ceci peut être compensé en utilisant la fonction de gain relatif de la voie 2. Il peut également être souhaitable, soit d'avoir une bande morte entre la désactivation de la voie 2 et l'activation de la voie 1, soit d'activer la voie 2 avant de désactiver complètement la voie 1.

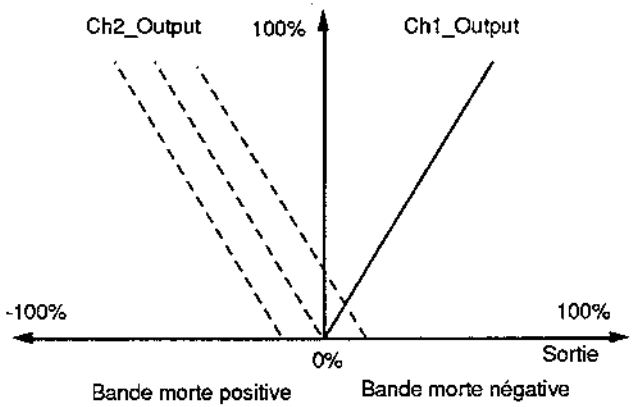


Figure 9-11 Relation graphique entre les gains des 2 voies et les bandes mortes

Ces deux fonctions peuvent être réalisées par l'utilitaire voie 1/voie 2/bande morte. Les sorties du PID à deux voies sont alors, par principe :

$$Ch1\_Output = Screen$$

$$Ch2\_Output = Rel\_Ch2\_Gain \times (Sortie + Ch1\_Ch2\_D\_B)$$

Bien entendu, seules des valeurs positives sont valables pour l'une ou l'autre des voies de sortie. La figure 9-11 montre la relation graphique entre la sortie PID de base et la sortie des différentes voies PID. Pour plus de détails, se reporter à la description du bloc fonction PID. Le PID à deux voies est également utile lors de l'utilisation de vannes à deux niveaux sur des réacteurs chimiques discontinus. Le PID à deux voies est sélectionné en réglant Output_low à une valeur négative quelconque (normalement -100%).

Extensions du PID de base

Dans de nombreux cas, le PID standard décrit précédemment ne convient pas et des modifications mineures sont nécessaires pour améliorer sa réponse. Quelques-unes d'entre elles sont décrites ci-après.

Modes de régulation

Pour annuler le terme intégral et le terme dérivée, les temps d'intégrale et de dérivée doivent être mis à zéro. Ainsi, le terme P peut être obtenu en annulant Ti et Td, PD en annulant Ti, et PI en annulant Td.

Limites de vitesse

Il est très fréquent de fixer des limites de vitesse pour les sorties de bloc fonction. Ceci est en particulier le cas pour obtenir une variation plus progressive des signaux de régulation ou pour prendre en considération le fait que certains éléments de la boucle de régulation (par exemple les vannes) sont limités en vitesse.

Saut proportionnel

Lorsque le déplacement du point de consigne d'un PID standard s'effectue selon un palier, le signal de régulation subit un saut proportionnel à la valeur de l'écart instantané, et inversement proportionnel à la largeur de la bande proportionnelle. Dans bien des cas, ce type de réponse peut ne pas être acceptable. Si "Debump" est mis au moment où le point de consigne change, la réponse du PID au déplacement du point de consigne sera moins brutale. Ceci peut être mis en oeuvre dans un pas du grafcet.

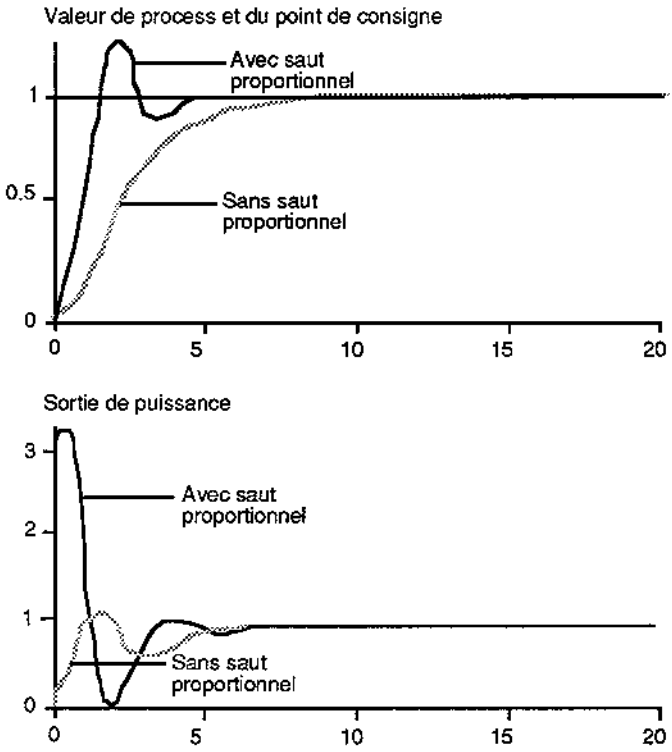


Figure 9-12 PID avec et sans saut proportionnel


```
Pid.Debump : = 1 (* Yes *);
Pid.Setpoint : = NewSP.Val;
```

Ceci assure que l'action proportionnelle seule agit sur la valeur de process. Cette méthode ne fonctionne que si l'écart instantané ne dépasse pas deux fois la largeur de la bande proportionnelle en unités physiques. Pour des changements plus importants du point de consigne, la fonction de limitation de vitesse de sortie peut être employée, la vitesse maximale de variation de la sortie pouvant être spécifiée et la limite de vitesse de sortie validée avant que le changement de point de consigne ne soit effectué. La figure 9-12 représente une régulation PI avec et sans saut proportionnel. Le PI avec saut proportionnel répond plus vite à un changement de point de consigne, mais il y a lieu de noter que la puissance de sortie "saute" instantanément. Si le saut proportionnel est éliminé, la puissance croît lentement jusqu'à sa nouvelle valeur. Le temps de réglage est plus long dans le dernier cas. Il est également possible d'utiliser la fonction de régulation en boucle ouvertek pour obtenir différentes formes de PID.

Saut dérivé

En plus du saut proportionnel décrit ci-dessus, l'action dérivée sur l'écart provoque un saut en raison de l'action dérivée lors d'un changement de point de consigne. Pour éliminer ceci, le terme dérivée peut être réglé pour agir sur la valeur de process et non sur l'écart. Pour cela, régler l'entrée Deriv_On_PV du bloc fonctionnel PID. La figure 9-13 représente le cas de la régulation PID avec et sans action dérivée sur l'écart. Dans le cas où l'action dérivée est sur l'écart, la sortie de puissance est réglée à une valeur élevée (hors de l'échelle du graphique) pendant un très court instant. Il en résulte un démarrage plus rapide et un dépassement plus faible. La sortie de puissance, dans le cas où l'action dérivée est sur la valeur de process, ne présente que le saut proportionnel initial et est moins brusque ensuite.

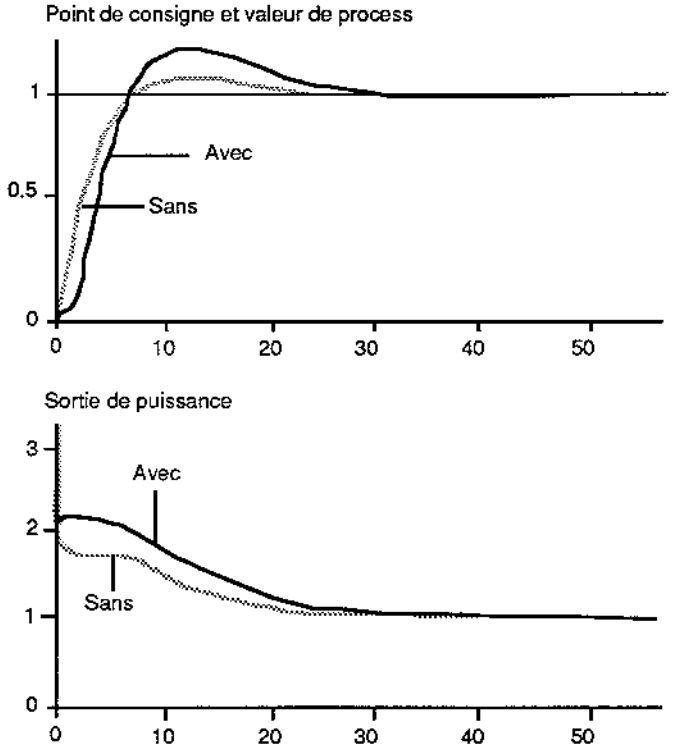


Figure 9-13 PID avec et sans saut dérivé

Suivi de point de consigne

Il est usuel, en particulier dans certaines boucles en cascade, que la valeur de process suive le point de consigne en manuel. Ceci peut être réalisé, bien entendu, au moyen d'un bloc de sélection externe. La sélection est donnée par :

```
Pid.Setpoint :=        SEL_REAL(G:= Pid.Manual, IN0:= SP.Val,
                IN1:= Pid.Process_Val);
```

SP.Val est positionné sur Pid.Setpoint en manuel.

Ceci est important pour passer entre les modes de régulation AUTO et MANUEL. Le passage se fait toujours sans à-coup, ce qui est normalement requis pour des boucles en cascade.

Régulation à action directe et inverse

La boucle fermée est utilisée pour rétablir la valeur de process au point de consigne en cas de perturbation externe ou de déplacement du point de consigne. Dans le cas d'un four à gaz, une chute de température due à une perturbation est compensée par une augmentation de débit du gaz (c'est-à-dire une régulation à effet inverse). Le contrôle de l'humidité dans un sécheur rotatif en utilisant le gaz comme grandeur de réglage, ou le contrôle de la température dans une application cryogénique en réglant le débit de l'azote ou de l'hélium liquide, nécessite cette fois une régulation par action directe, une diminution de la valeur de process étant compensée par une diminution du signal de sortie. Dans les blocs fonctions PC3000, une entrée booléenne permet de passer de la régulation à action inverse à la régulation à action directe.

Rupture de capteur

Dans une grande gamme d'entrées de niveau élémentaire (mesures en mV), le matériel est en mesure d'indiquer l'état de rupture d'un capteur. De plus, il est possible d'établir l'état logiciel de rupture d'un capteur en comparant la valeur de process à ses limites. Dans les deux cas, le PID doit être désactivé. Le bloc fonction a une entrée booléenne Sensor_Break et la valeur par défaut en sortie du PID (c'est-à-dire la valeur de sécurité de la sortie dans cette situation) est donnée par Break_Output. Il est clair que des stratégies plus complexes peuvent être utilisées dans le programme utilisateur, en fonction de la gravité de la rupture de capteur.

POSITIONNEURS DE VANNE

De temps en temps, l'entraînement d'une vanne ou d'un levier se fait au moyen d'un moteur réversible à vitesse constante. La vanne peut alimenter en gaz ou en air de combustion les brûleurs d'un four. Trois états sont possibles pour la régulation : envoyer une impulsion "Augmenter", une impulsion "Diminuer" ou pas d'impulsion du tout. Le but essentiel d'une régulation de four à gaz, par exemple, est de réguler la température avec précision. Ceci peut se faire de différentes manières en fonction de la disponibilité et de la fiabilité des signaux de positionnement de la vanne donnés par des potentiomètres.

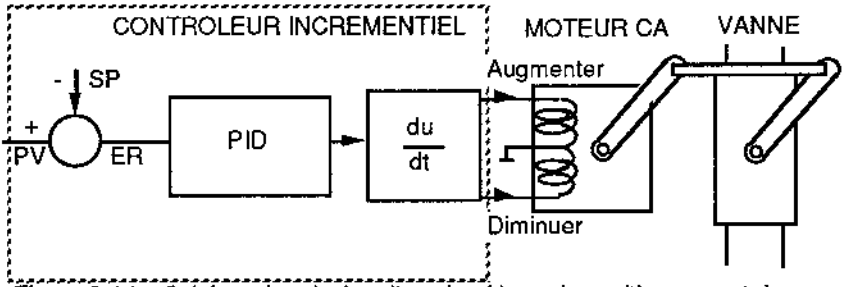


Figure 9-14 Schéma de principe d'un algorithme de positionnement de vanne

Positionnement de vanne sans retour de potentiomètre

Si aucun signal de potentiomètre n'est disponible, le PC3000 VP ou sa version à réglage automatique peut être utilisé. La figure 9-14 représente le principe de fonctionnement d'un positionneur de vanne.

En plus des réglages du PID classique, les caractéristiques suivantes de la vanne doivent être intégrées.

Temps de déplacement de la vanne

C'est le temps nécessaire à la vanne pour se déplacer d'une position extrême à l'autre. Dans bien des cas, le temps de déplacement de l'ouverture totale à la fermeture totale est différent du temps de déplacement de la fermeture totale à l'ouverture totale. Dans ce cas, la valeur moyenne est prise.

Temps d'activation minimal

C'est le temps minimal pendant lequel la régulation reste dans le même état d'ouverture, de fermeture ou de maintien en position de la vanne. Beaucoup de vannes ayant une course morte, le temps d'activation minimal est la durée nécessaire à l'impulsion pour que la vanne réagisse au signal envoyé par la régulation.

Délai de rafraîchissement

C'est le temps qui sépare deux mises à jour du filtre de mise à jour du positionneur de vanne, comme le représente la figure 9-15. Voir Réglage PID, qui décrit la méthode de réglage de ces valeurs pour l'application spécifique.

Positionnement de vanne avec retour de potentiomètre non fiable

Le VP (positionneur de vanne) est prévu pour utiliser des signaux de retour de potentiomètre. En règle générale, les signaux de retour de potentiomètre ne sont pas vraiment fiables. Ils ont tendance à présenter du bruit et il est fréquent que les signaux soient perdus. Pour ces raisons, l'algorithme VP les utilise comme indicateurs de position et pour le réglage de limites matérielles de fin de course (Pot_Limit_Hi et Pot_Limit_Lo) de la vanne, par opposition à la régulation de position réelle.

Positionnement de vanne avec retour de potentiomètre fiable

Si le signal de position de vanne est disponible et reconnu fiable, il peut être utilisé pour une régulation correcte en boucle fermée. Une structure simple en cascade, mettant en oeuvre une régulation tout ou rien élémentaire avec bande morte et hystérésis peut être utilisée pour l'étage de retour de position, et un PID standard peut donner le point de consigne nécessaire à cette boucle de positionnement.

La figure 9-16 représente cette régulation. Cette configuration convertit les caractéristiques d'intégration de la vanne en un résultat qui ressemble plutôt à un retard, ce qui améliore les performances de la boucle de régulation.

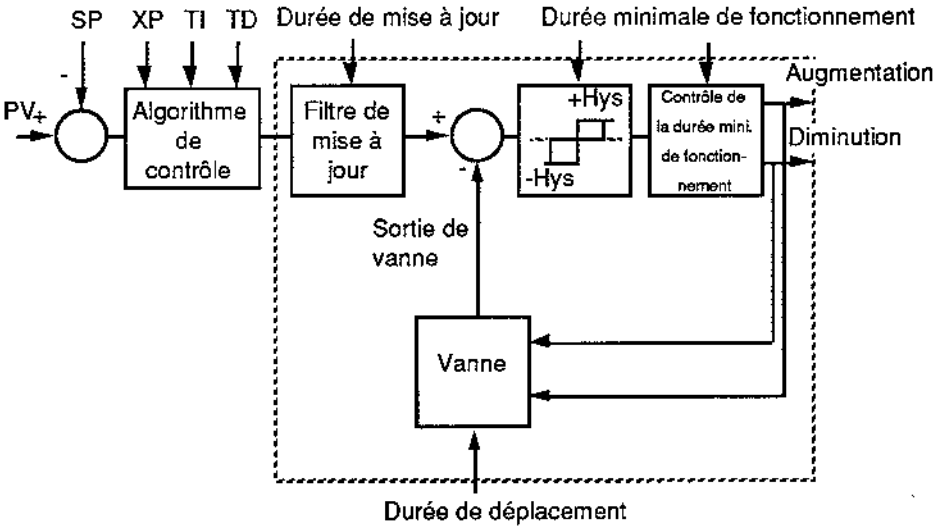


Figure 9-15 Schéma de principe d'un positionneur de vanne incrémentiel

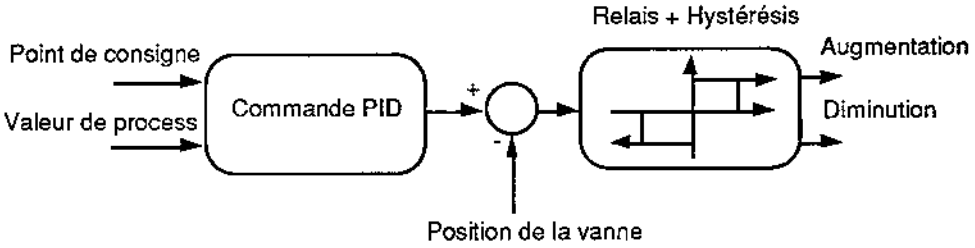


Figure 9-16 Positionneur de vanne en cascade

ENTREES, SORTIES ET BOUCLES DE REGULATION

Mise en forme du signal d'entrée

Les signaux analogiques mesurés par le PC3000 pour la régulation sont traités de la façon suivante :

Pour les cartes AIM2/AIM4, on dispose d'un filtre passe-bas analogique effectuant un premier tri avec une fréquence de coupure à 1,6 Hz (constante de temps de 100 ms), ainsi qu'un filtre de moyenne mobile sur 4 échantillons. La moyenne mobile sur 4 échantillons est liée à la tâche analogique. Un filtre passe-bas effectuant un premier tri à temps discontinu permet à l'utilisateur de choisir la constante de temps. Le choix de la constante de temps du filtre passe-bas est décrit au paragraphe Réglage. Les calculs nécessaires à la compensation de soudure froide et à la linéarisation sont effectués dans le module pour différentes entrées thermocouple et/ou de thermomètre à résistance.

Pour les cartes AI08, le filtre passe-bas est un filtre de Butterworth effectuant un deuxième tri dont la fréquence de coupure peut être, au choix, de 160 Hz, 80 Hz, 32 Hz ou 16 Hz. Il n'y a pas de filtrage à moyenne mobile. Le bloc fonction ne comporte pas non plus de filtre sélectionnable par l'utilisateur.

Pour les signaux dont la mesure fait appel à des liaisons de communication et pour ceux qui sont calculés par méthode inférentielle, il n'y a pas de filtrage. De même, il n'y a aucun filtrage de pics sur les signaux d'entrée. Ceci peut être réalisé dans le programme utilisateur. Le filtrage des pics peut s'avérer très utile. Il y a de nombreuses méthodes possibles, mais la plus simple consiste à utiliser le bloc fonction Rate_Limit.

Linéarisation d'entrée

La plus grande partie de la linéarisation s'efffectue dans le module lui-même. Il est possible de mettre à l'échelle et de décaler les mesures avant et après la linéarisation, pour les thermocouples. Une extraction de racine carrée est également disponible dans le cadre de la linéarisation. Ceci permet, par exemple, de convertir et d'étalonner la mesure de pression d'un capteur de pression dérivée, pour effectuer un contrôle de débit. D'autres conversions, comme la mesure de l'humidité relative à partir des températures sèches et humides selon BS 4883, doivent être effectuées par le programme utilisateur.

Les résultats du calcul polynomial peuvent être facilement utilisés dans un simple pas de câblage. Par exemple, si

$$\text{Inferred\_Var} = axPV^2 + bxPV + c$$

un simple pas de câblage du PC3000 peut être effectué avec

Inferred.Val := (a.Val * ANIN.Process_Val + b.Val) * ANIN.Process_Val + c.Val;

Il est évident que si d'autres fonctions, comme des exponentielles, des racines carrées ou des logarithmes sont également utilisées, elles peuvent être intégrées aux équations de linéarisation.

Une autre méthode de linéarisation consiste à effectuer une interpolation linéaire entre plusieurs points donnés. Si l'on considère, par exemple, les trois zones représentées sur la figure 9-17, la linéarisation peut s'effectuer par le câblage ci-après :

```
Y.Val := SEL_REAL(G  := (X.Val >= X1.Val) AND (X.Val < X2.Val),
IN0:= SEL_REAL(G  := (X.Val >= X2.Val) AND (X.Val < X3.Val),
IN0:= SEL_REAL(G  := (X.Val >= X3.Val) AND (X.Val < X4.Val),
IN0:= Y4.Val,
IN1:= Y3.Val +(X.Val-X3.Val)*(Y4.Val-Y3.Val)/(X4.Val-X3.Val))
IN1:= Y2.Val+(X.Val-X2.Val)*(Y3.Val-Y2.Val)/(X3.Val-X2.Val)),
     IN1:= Y1.Val +(X.Val-X1.Val)*(Y2.Val-Y1.Val)/(X2.Val-X1.Val));
```

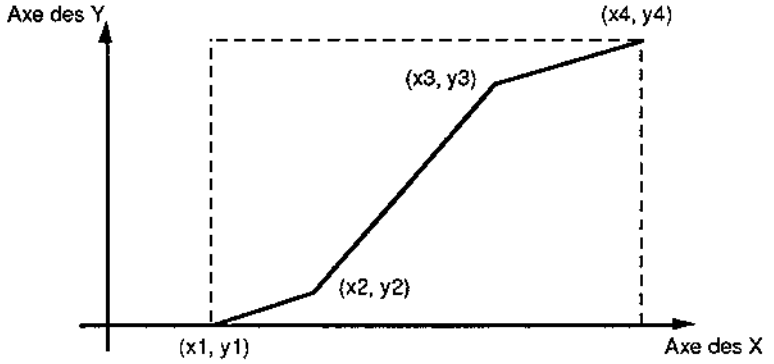


Figure 9-17 Caractéristiques d'entrée

Sorties

Normalement, la sortie du bloc fonction PID est câblée sur :

un module de sortie analogique

une sortie proportionnelle au temps

une variable déportée / esclave

un autre bloc fonction interne

Le module de sortie analogique permet différentes gammes de sortie, ainsi que le choix entre les sorties en intensité et en tension.

Dans la plupart des applications de régulation de température, on utilise des sorties proportionnelles modulées. La figure 9-18 en montre le principe. Le choix du temps de cycle est décrit dans le cadre du réglage PID.

Les variables déportées / esclaves sont utilisées en règle générale lorsque des stations déportées, telles que les appareils discrets de la série 900, sont utilisées soit en mode de régulation locale, soit en tant que stations de retransmission. Ce mode apporte à la fois l'intégrité et la flexibilité par l'utilisation d'une redondance dans la configuration du système de régulation.

D'autres blocs fonctions internes sont généralement utilisés lors de la conception de stratégies de régulation en cascade ou à plusieurs boucles.

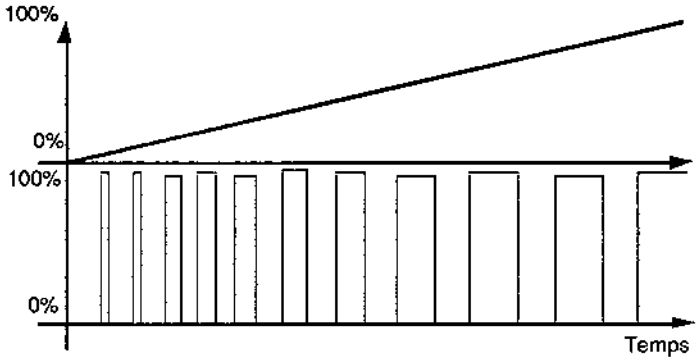


Figure 9-18 Sortie proportionnelle modulée

Boucles de régulation

La méthode la plus simple pour réaliser une boucle de régulation PID est d'avoir, par exemple :

un bloc fonction PID

un bloc fonction d'entrée analogique

un bloc fonction de sortie analogique

et le câblage par soft suivant

```
Pid.Process_Val    := ANIN.Process_Val;
ANOUT.Process_Val := Pid.Ch1_Output;
```

pour une voie PID unique. Si deux voies sont nécessaires, la seconde sortie est câblée, bien entendu, sur une autre sortie analogique. La gamme de sortie est choisie dans le bloc fonction de sortie.

Pour les sorties proportionnelles modulées, les temps de cycle doivent également être choisis. Dans le cas de PID_Auto (version réglage automatique du PID), le temps de cycle peut, si nécessaire, être câblé sur les valeurs réglées par le bloc PID_Auto. Les sorties proportionnelles modulées utilisées avec un refroidissement non linéaire (refroidissement par eau) sur les extrudeuses à plastique, sont sélectionnées à l'aide de l'entrée Delin_Type du bloc fonction.

AVERTISSEMENT !

Dans de nombreuses applications, il est important d'inclure un mécanisme de protection indépendant. La meilleure forme de protection est un "surveillant" totalement indépendant. C'est une alarme de température ayant son propre capteur ou thermocouple et, en cas de déclenchement, le contacteur principal est ouvert ou la vanne fermée pour assurer la sécurité de l'installation. La fonction normale du "surveillant" est de réagir en tant qu'alarme de dépassement de température dans le cadre de la stratégie de protection générale du process. Il est, par conséquent, essentiel que tous les éléments du système d'alarme soient contrôlés régulièrement pour s'assurer qu'ils sont en bon état de fonctionnement.

Mise en marche et mise à l'arrêt

Le bloc fonction PID effectue une remise à zéro à sa première exécution. Ceci implique que la puissance de sortie initiale sera à zéro (en cas de démarrage à froid), même si l'écart entre la valeur de process et le point de consigne de démarrage à froid est important. La puissance est alors augmentée progressivement à un taux qui dépend des réglages de la régulation et de la dynamique du process lui-même. Ceci peut prendre un temps assez long, surtout si le temps d'intégrale a une valeur élevée.

Si les conditions initiales du process sont connues, il est alors préférable de démarrer le PID en mode manuel. Le programme utilisateur peut alors régler librement la puissance de sortie initiale. La régulation sera mise en mode automatique. Le transfert se fera ainsi sans à-coup et le PID démarrera à partir d'un état connu. Si l'état initial n'est pas connu, la fonction cutback peut être utilisée. Une autre possibilité consiste à démarrer en régulation proportionnelle seule, puis de commuter en PI juste après le démarrage.

La situation est identique si, par exemple, la machine commandée est mise à l'arrêt ou bien si les actionneurs ne sont plus raccordés à la régulation, alors que le PID reste en mode automatique. La régulation se comportant comme si elle était toujours active, elle applique la puissance totale pour maintenir la valeur de process au point de consigne. Toutefois, l'actionneur n'étant plus raccordé, la valeur de process va dériver vers sa valeur "ambiante". Si la machine est redémarrée ou l'actionneur rebranché, la régulation va appliquer immédiatement la pleine puissance, ce qui peut avoir des conséquences désastreuses. Là encore, la

coupure peut s'avérer très utile. La meilleure solution, cependant, est de s'assurer que la régulation soit mise ou en mode manuel ou en mode poursuite dès que l'actionneur est débranché. Au rebranchement, comme dans les conditions de démarrage initial, le niveau de puissance doit être réglé sur une valeur de sécurité, pour ne reprendre qu'ensuite le gain normal.

Dans la plupart des cas, aucune précaution spéciale n'est à prendre pour la mise en marche ou la mise à l'arrêt. La plupart des boucles de température compensent ces situations. Toutefois, la prise en compte des conditions de mise en marche et de mise à l'arrêt fait partie de l'application, et doit être traitée comme telle dans les programmes utilisateurs. Se reposer sur le bloc fonction PID pour régler automatiquement les conditions initiales peut avoir des conséquences inattendues.

Positionneurs de vanne

Les signaux d'entrée d'un système de régulation de positionnement de vanne sont :

la valeur de process primaire (normalement une température)

le potentiomètre de rétroaction de position. Ce signal de rétroaction est donné par un potentiomètre. Une voie d'entrée analogique standard peut être utilisée pour la mesure de position du potentiomètre. La puissance nécessaire peut être fournie soit extérieurement, soit par une sortie analogique (si disponible). Il est aussi relativement facile de régler un simple circuit comportant une résistance de rappel au potentiel haut, de telle sorte que la rupture de capteur puisse être détectée par le logiciel.

La sortie de la régulation se fait par une sortie logique quelconque (sortie relais ou niveau logique).

REGLAGE PID

Les valeurs PID par défaut des blocs fonctions PC3000 sont la plupart du temps satisfaisantes dans le cas des extrudeuses ou des fours de taille moyenne. Toutefois, dans le cas général, pour optimiser les performances des machines, il est nécessaire de bien régler les boucles de régulation. Le lecteur est invité à se reporter aux divers manuels cités à la fin du présent document, pour tous détails concernant le réglage des boucles. Nous ne donnons ici que des règles générales applicables dans la plupart des cas.

Réglage manuel

La méthode la plus courante pour régler les régulations PID est la méthode empirique. En se servant de quelques lois empiriques simples, il est possible de régler les paramètres PID avec une précision tout à fait acceptable pour la plupart des process. Il y a lieu de noter que la précision de la régulation dépend dans une large mesure de la qualité du transducteur, de la puissance de l'actionneur et de la dynamique en boucle ouverte du process, comme elle est fonction des réglages de la régulation PID. C'est pourquoi il est nécessaire de prendre en considération les caractéristiques du process commandé pour choisir les paramètres de réglage de la régulation.

Méthode de la courbe de réaction

Dans leur grande majorité, les process sont stables en boucle ouverte. Ceci signifie que, si elle est laissée seule, la valeur de process se stabilisera à une valeur quelconque au lieu de dériver régulièrement jusqu'à l'une de ses limites. Il existe toutefois des exceptions. Le réacteur exothermique discontinu en est un exemple : il nécessite une régulation en boucle fermée précise dès que la réaction démarre. Ceci provient du fait que, dès que la réaction démarre, elle commence à dégager un excès de chaleur qui doit être éliminé par un refroidissement actif. Si elle est laissée seule, la chaleur de réaction continue à chauffer les produits réactifs, en accélérant la réaction, ce qui augmente encore la température. Un tel process est instable en boucle ouverte. La méthode de réglage décrite dans ce chapitre n'est applicable qu'aux process stables en boucle ouverte.

La méthode de la courbe de réaction est la suivante. Si le process est en régulation automatique, passer la régulation en mode manuel et attendre que la valeur de process se stabilise. Effectuer une variation de quelques % (en général 10 % environ) en sortie de la régulation et observer la courbe de réaction du process. La variation en sortie doit avoir une valeur suffisante pour se différencier d'un bruit normal, mais être limitée pour que l'effet reste linéaire. Lorsque la valeur de process s'est stabilisée à sa nouvelle valeur, ramener la sortie à sa valeur initiale. Examiner la réaction du process à cette variation inverse. Normalement, on peut s'attendre à l'une des réponses suivantes :

- Simple retard plus temps de retard
- Retards multiples plus temps de retard
- Comportement en phase non minimale
- Comportement oscillatoire

Un temps de retard est le délai pendant lequel aucun changement significatif de la valeur de process n'est visible après une variation de la sortie de régulation. Quatre exemples sont représentés sur la figure 9-19.

Les deux premiers sont fréquents, en pratique, dans les process industriels. Le comportement en phase non minimale est celui d'un process qui réagit en sens opposé, le rétrécissement et la dilatation des chaudières, par exemple, sont typiques d'un comportement en phase non minimale. Certains sécheurs rotatifs ont aussi ce comportement en régulation d'humidité. Le quatrième, le comportement oscillatoire, se rencontre en général soit avec des systèmes mécaniques, soit avec le comportement en boucle fermée des boucles esclaves des configurations en cascade. Si le temps de cycle de l'oscillation mécanique est inférieur à 100 fois le temps de cycle de tâche du bloc fonction PID, l'amortissement des oscillations par réglage PID n'est pas possible. Par exemple, pour un bloc fonction PID fonctionnant dans une tâche de 100 ms, des oscillations ayant un temps de cycle de 10 secondes ne peuvent être contrôlées.

Un comportement oscillatoire peut être dû à l'amortissement de la boucle interne d'une boucle en cascade trop basse, auquel cas la boucle interne doit être réglée en premier pour stabiliser le comportement.

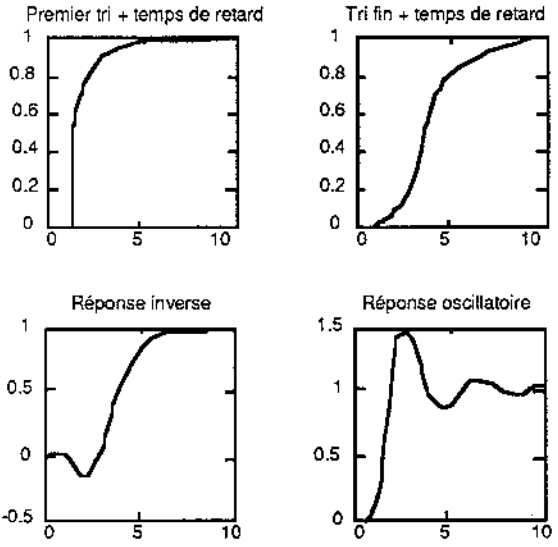


Figure 9-19 Quelques courbes de réaction typiques

Trois quantités sont calculées à partir de la courbe de réaction, voir figure 9-20.

Dead-Time (Dp): temps de retard estimé.

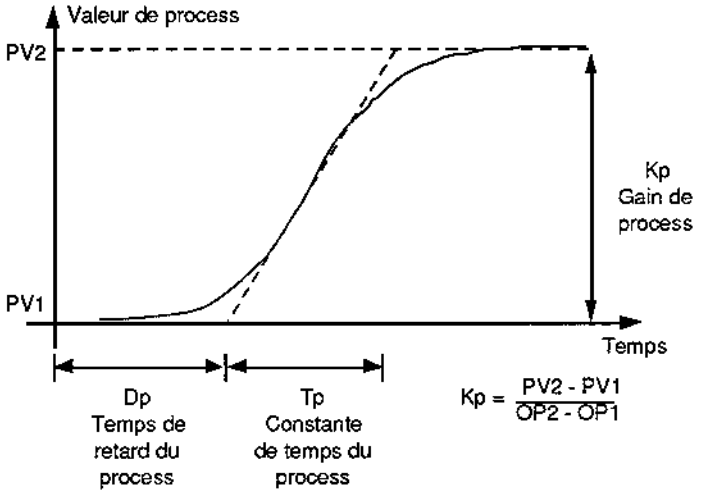


Figure 9-20 Calcul du gain, de la constante de temps et du temps de retard, à partir de la courbe de réaction

Régulation type	Bande prop.	Intégral	Dérivée
P	100KpNd		
PI	110KpNd	3Dp	
PID	80KpNd	2Dp	Dp/2

Table 9-1 Réglages PID à partir d'une courbe de réaction

Régulation type	Bande prop.	Intégral	Dérivée
P	2Pu		
PI	2,2Pu	0,8Tu	
PID	1,67Pu	0,5Tu	0,12Tu

Table 9-2 Principes de réglage Ziegler-Nichols

Constante de temps (Tp) : constante de temps estimée ou temps de réaction du process. Voir figure 9-20. C'est le temps qui sépare le point de rencontre de l'asymptote avec l'axe des temps et son point de rencontre avec le nouveau niveau de sortie.

Gain du process (Kp) : C'est le rapport entre la variation des états stables de la valeur de process et la variation de la sortie commandée.

Le temps de retard normalisé est alors donné par la formule :

$$N_d = \frac{Dp}{Tp}$$

De nombreuses règles permettent de définir la courbe de réaction en boucle ouverte du process, les formes les plus simples étant données par le tableau 9-1. Il y a lieu de noter que les réglages de la bande proportionnelle sont donnés dans le tableau 9-1 en unités physiques.

Des corrections sont apportées aux valeurs PID du tableau par Cohen and Coon, pour améliorer le cas des systèmes à dominante de retard. Shinskey donne également des valeurs de PID pour différents rapports entre les constantes de temps et les temps de retard. Son calcul donne une régulation avec un écart quadratique moyen minimal. Se reporter à l'annexe.

La méthode de la sensibilité maximale

La méthode de la sensibilité maximale est une méthode classique inspirée de Ziegler et Nichols. L'idée consiste à ne commander le process qu'en action proportionnelle et à diminuer le réglage de la bande proportionnelle jusqu'à ce que la boucle oscille. Dans le cas d'une régulation chauffage / refroidissement, le gain relatif de la voie 2 peut nécessiter également un réglage, jusqu'à ce que les oscillations soient raisonnablement symétriques. Ces valeurs de la bande proportionnelle Pu et de la période d'oscillation Tu sont notées. Une séquence de fonctionnement du type de celle représentée sur la figure 9-21 est possible. Le réglage de la bande proportionnelle est diminué progressivement jusqu'à obtenir une oscillation entretenue. Les valeurs PID peuvent alors se calculer à partir du Tableau 9-2.

Ces réglages sont acceptables pour une grande variété de systèmes et étaient conçus à l'origine pour obtenir un rapport de décroissance de 1/4 des perturbations. Ceci donne normalement une faible réponse au point de consigne et ne convient pas bien pour les process ayant un temps de retard important. Des améliorations à ces réglages sont apportées par Hang, Astrom et Ho [5]. Ces améliorations combinent la courbe de réaction et les tests de sensibilité maximale. Voir également le chapitre du présent document qui traite de stratégies évoluées, en ce qui concerne les règles et guides qu'indiquent ces méthodes améliorées.

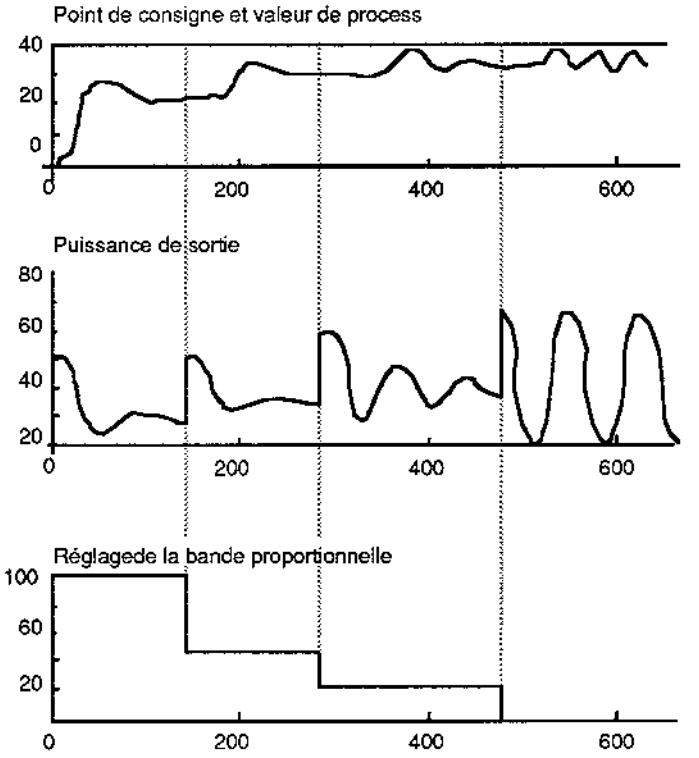


Figure 9-21 Exemple de test de sensibilité maximale

Le réglage fin de la régulation peut ensuite s'effectuer en mode manuel.

Méthode par approximations successives

Dans bien des exemples, aucune des méthodes décrites ci-dessus ne permet d'obtenir une valeur de départ pour les réglages PID. Certaines règles élémentaires peuvent aider à régler la boucle de régulation.

1. Le réglage s'effectue normalement dans l'ordre P, I, D, gain relatif de la voie 2 et temps de cycle, cutback (le cas échéant). Les cutbacks doivent être désactivés avant de démarrer le réglage manuel par approximations successives. Régler les temps de cycle au minimum et régler à 1 le gain relatif de la voie 2, à moins qu'il ne soit clair que leur réglage est correct. Régler Span_High et Span_Low sur les limites absolues, haute et basse, de la valeur de process. Il y a lieu de noter que les blocs fonctions signaleront une rupture de capteur si la valeur de process dépasse de 10 % la plage de mesure.

2. S'assurer que la forme correcte de régulation (c'est-à-dire action directe ou inverse) est bien utilisée.
3. Ne pas utiliser de termes intégral ou dérivé s'ils ne sont pas absolument nécessaires. Pour les capacités les plus simples, un réglage étroit de la bande proportionnelle est approprié. Le contrôle de niveau dans un réservoir d'équilibrage est un bon exemple. Les boucles internes d'une régulation en cascade sont un autre exemple. La régulation P seule ou PD (dans certains cas) fonctionne en général tout à fait bien.
4. La régulation PI est tout à fait acceptable dans la plupart des problèmes de régulation. La régulation dérivée n'est pas utilisée, ou très peu, dans :

   les process ayant peu de retard ou peu d'inertie, car l'action dérivée n'apporte rien dans ce cas

   les process ayant des temps de retard importants, car l'action dérivée n'apporte pas la forme correcte de tendance de la valeur de process

   les process ayant un niveau de bruit important, car l'action dérivée amplifie le niveau de bruit d'une façon significative et diminue les performances.
5. Le PID fonctionne extrêmement bien avec les process ayant des retards thermiques importants et pas - ou peu - de temps de retard.
6. Un accroissement de la bande proportionnelle diminue la vitesse de récupération après une perturbation et augmente la stabilité pour les process stables en boucle ouverte. Pour les systèmes instables, comme les réacteurs discontinus, la bande proportionnelle doit être réduite jusqu'au point d'amélioration de la stabilité.
7. Un accroissement du temps d'intégrale ralentit les oscillations dans les process stables en boucle ouverte.
8. Un accroîssement du temps de dérivée augmente la vitesse de récupération et la stabilité jusqu'à un certain point. Un rapport inférieur à 3:1 n'est pas recommandé entre le terme intégrale et le terme dérivée.
9. Dans un PID à deux voies, augmenter le gain relatif de la voie 2 si la régulation utilise une puissance excessive dans la voie 2 en provoquant des oscillations.
10. Lorsque des sorties proportionnelles modulées sont utilisées, si des sorties logiques le sont également, régler le temps de cycle à une valeur faible, par exemple de 1 seconde ou moins. Si des relais sont utilisés, les valeurs raisonnables sont de 1/10 à 1/20 du temps d'intégrale. Un temps de cycle trop court provoque une usure des relais et est généralement inutile. Un temps de cycle trop long augmente la tendance de la boucle à limiter le cycle (pompage).

11. Si les paramètres PID se montrent inadaptés pour de faibles variations et non pour les variations importantes, utiliser la fonction cutback. Régler les valeurs initiales de cutback en fonction du réglage de la bande proportionnelle en unités physiques. Si une régulation dérivée est utilisée, en augmentant les valeurs de cutback légèrement en dehors du réglage de la bande proportionnelle, il est possible d'améliorer le dépassement sur déplacement de point de consigne. Ne pas dévier trop loin vers l'extérieur ou vers l'intérieur du réglage de la bande proportionnelle de la régulation PID en ajustant les valeurs de cutback.
12. Le but du filtrage des signaux d'entrée analogiques est d'éliminer le bruit parasite. Une constante de temps de filtre trop grande peut créer des problèmes d'instabilité de boucle, car le retard apporté à la valeur de process peut devenir trop important. La valeur empirique pour le choix du filtre, la constante de temps du filtre doit être au moins d'un ordre de grandeur inférieur à l'échelle de temps du process. Si le bruit persiste pour cette valeur, la régulation ne peut pas apporter plus. Un appareil plus sensible peut s'avérer nécessaire.

Réglage des positionneurs de vanne et paramètres complémentaires

Les positionneurs de vanne nécessitent le réglage de trois autres paramètres. Ce sont : le temps de déplacement de la vanne, le temps minimal d'impulsion et le temps de rafraîchissement. Le temps de déplacement de la vanne, comme le définit ce chapitre, est le temps nécessaire à la vanne pour se déplacer en continu d'une fin de course de déplacement à l'autre.

En règle générale, pour mesurer ce temps, il suffit de mesurer le temps pour ouvrir à fond la vanne, et celui pour la refermer. Le temps de déplacement est alors la moyenne des deux temps de déplacement. Le temps minimal d'impulsion est la durée minimale qu'une impulsion doit avoir avant que la vanne ne commence à réagir. Si un signal de position de la vanne est disponible, le temps minimal d'impulsion peut être défini avec plus de précision en examinant ce signal.

Le signal de position doit probablement être filtré avec une constante de temps d'une seconde environ pour éliminer le bruit parasite d'une façon acceptable. Le rafraîchissement de la vanne, comme le temps de cycle des sorties proportionnelles modulées régule le niveau d'activité des impulsions d'augmentation et de diminution. Le temps de rafraîchissement de la vanne, comme son nom l'indique, est l'intervalle de temps qui sépare deux mises à jour de l'algorithme de "point de consigne" de position de la vanne. Cela signifie qu'à chaque "temps de rafraîchissement de la vanne", la partie PID de la régulation demande une "nouvelle" position à la partie positionneur de l'algorithme. Si le temps de rafraîchissement est mis à sa valeur minimale (c'est-à-dire l'intervalle de temps de la tâche du bloc fonction), l'activité de la vanne sera alors à son maximum. La régulation demandera presque toujours d'augmenter ou de diminuer légèrement la consigne de la vanne, à chaque scrutation.

Si le temps de rafraîchissement augmente, on obtient un intervalle de temps où la vanne reçoit peu d'impulsions pour augmenter ou diminuer dans la partie initiale du temps de rafraîchissement, et la période suivante est une période d'inactivité

totale. Si le temps de rafraîchissement est réglé trop long (comme le temps de cycle en sortie proportionnelle modulée) des effets de limitation de cycle peuvent apparaître (pompage). Une valeur raisonnable pour le temps de rafraîchissement de la vanne est la plus grande des 2 valeurs parmi Ti/24 et Td/4. Dans quelques applications exceptionnelles, en particulier lorsqu'il s'agit de process rapides, le temps de rafraîchissement peut avoir besoin d'être réglé à une valeur plus grande que les valeurs suggérées ici, afin de réduire l'activité de la vanne.

Pour le réglage PID au moyen des tests décrits, la situation est quelque peu différente. En clair, si un signal de position de la vanne n'est pas disponible, il n'est pas possible de savoir exactement quel signal de sortie est appliqué, et la taille du "saut" est choisie en réglant la durée d'une impulsion d'ouverture ou de fermeture. Si le temps de déplacement de la vanne est connu avec une précision raisonnable, la valeur de la sortie incrémentielle appliquée est alors connue.

Dans la mesure où le temps de réponse du process est raisonnablement plus long que celui de la vanne, les réglages PID à partir des tableaux sont probablement tout à fait appropriés. En règle générale, la bande proportionnelle a besoin d'être légèrement augmentée. La méthode de la sensibilité maximale peut être appliquée et les paramètres de réglage doivent être appropriés. Dans tous les cas, il est important d'avoir une idée raisonnable du temps de déplacement de la vanne.

Réglage automatique non répétitif

Le PC3000 associé aux appareils des séries 800 et 900 permet un auto-réglage automatique. Une description rapide de la méthode est donnée ici. Le lecteur peut également se reporter à [3] qui expose en détail les performances de l'algorithme sur quatre zones d'une extrudeuse de revêtement de fil.

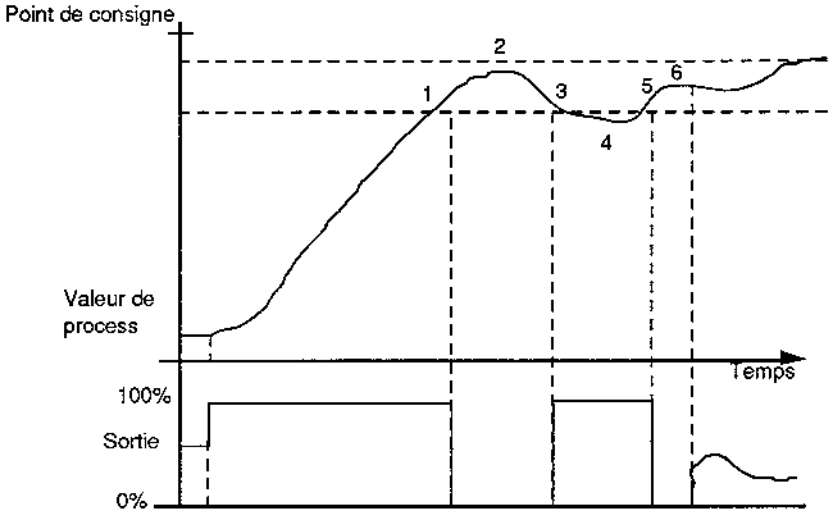


Figure 9-22 Séquence élémentaire PID_Auto

La méthode repose sur le procédé du cycle en boucle fermée (CLCM). Le principe utilisé consiste à donner une réponse en boucle fermée à amortissement critique. Le régleur peut agir sur l'une quelconque des configurations de régulation P, PI, PD ou PID. Cette méthode requiert un minimum de connaissances préalables -- la seule nécessité concerne le niveau maximum/minimum des valeurs de sortie disponibles sur le process. Sur certains process, un cycle en boucle fermé n'est pas autorisé : l'auto-réglage ne peut pas être utilisé dans ce cas. Par défaut, le niveau maximum/minimum des valeurs de sortie de la régulation lors d'un auto-réglage est réglé selon les valeurs Output_High et Output_Low. Dans les versions plus récentes du progiciel, AT_Out_High et AT_Out_Low peuvent être utilisées. Avant de démarrer un auto-réglage, l'utilisateur peut limiter ces valeurs pour réduire l'amplitude des évolutions du process pendant la phase de réglage.

Pour la mise en oeuvre de l'auto-réglage, la sortie de la régulation est, soit

gelée, si l'écart initial est inférieur à 1 % de la plage de mesure, soit

mise à zéro, si l'écart initial est supérieur à 1 % de la plage de mesure.

La sortie de la régulation est maintenue à ce niveau pour une durée déterminée qui dépend de la tâche du bloc fonction PID_Auto associé. Elle est de 1 minute pour des intervalles de tâche jusqu'à 5 secondes et est réglée en conséquence pour des intervalles plus longs, mais sans jamais dépasser 5 minutes. Pour la plupart des applications, il est peu vraisemblable que des intervalles de temps atteignant 5 secondes soient requis pour le bloc PID. Pendant la période d'attente, le bloc examine la tendance générale de la variable de process et calcule un niveau de déclenchement pour le réglage adaptatif suivant, si nécessaire.

Pendant la période d'attente initiale, le point de consigne peut être réglé selon les conditions de fonctionnement requises. Pour les points de consigne qui diffèrent de la valeur de process de plus de 1 % de l'étendue de mesure, l'algorithme applique la sortie de régulation maximale ou minimale pour ramener la valeur de process au point de consigne le plus rapidement possible. La valeur de process est surveillée en continu et une estimation du temps de retard du process est faite à partir de ces valeurs mesurées. Cette estimation est mise à jour en continu jusqu'à ce que le pas suivant de réglage soit entré. Cette estimation sert à fournir des délais de dépassement dans un algorithme de supervision, pour le cas où le process ne se terminerait pas comme prévu dans une phase ultérieure quelconque.

Pour éviter les dépassements, le réglage s'effectue par des tests en un point de consigne factice placé au-dessus ou en-dessous du point de consigne réel. Certains process à temps de retard important dépasseront ce point de réglage malgré la précaution prise avec ce point de consigne factice. Il est recommandé d'utiliser des valeurs de sortie limitées pour les process ayant des temps de retard importants. Lorsque la valeur de process atteint ce point de consigne factice, l'algorithme applique à cette valeur de process un état fini de machine de régulation tout ou rien. L'amplitude de ce pic et le temps nécessaire à son obtention sont stockés et l'algorithme passe à l'état suivant. L'algorithme continue les pas 1 à 6 (7 en cas de 2 voies de sortie) comme le montre la figure 9-22. L'auto-réglant vérifie la vraisemblance de toutes les temporisations, et lorsque l'étape finale est atteinte, les valeurs PID sont calculées selon une loi de Ziegler-Nichols modifiée.

L'autoréglant effectue un réglage au point de consigne si la valeur de process, à la fin de la période d'attente initiale, ne s'écarte pas de plus de 1 % du point de consigne. Dans ce cas, le temps de retard du process n'est pas évalué et les paramètres PID sont calculés uniquement à partir de l'amplitude et des périodes de l'oscillation.

La figure 9-23 montre un exemple de séquences d'auto-réglage pour monter jusqu'à un nouveau point de consigne, à un point de consigne, et pour descendre vers un nouveau point de consigne. Comme on peut le voir, la perturbation par l'auto-réglage n'est pas très significative.

Dans le cas d'un refroidissement par eau non linéaire, Ch2_Linear doit être réglé sur No(1) et la sortie maximale de refroidissement utilisée durant l'auto-réglage sera mise à 20 %. Il y a lieu de noter que le paramètre Delin_type de la sortie proportionnelle au temps pour cette voie doit être mis à D_800 ou D_EM1 (celui des deux qui convient le mieux pour l'application). Pour plus de détails, se reporter au Manuel des blocs fonctions du PC3000.

L'auto-réglant calcule les paramètres suivants :

1. P, I et D: si le terme intégrale ou le terme dérivée sont mis à zéro, l'autoréglant calcule les paramètres pour P seul (si les deux sont à zéro), PI (si D est mis à zéro), PD (si I est mis à zéro) et PID si aucun n'est à zéro. Si l'auto-réglant détecte un système à retard dominant, il coupe automatiquement le terme dérivée.
2. Gain relatif de la voie 2 : celui-ci est réglé si un PID à deux voies est utilisé (par exemple Output_Low est négatif). Ce paramètre est mis à 1 si l'auto-réglant ne termine pas correctement.

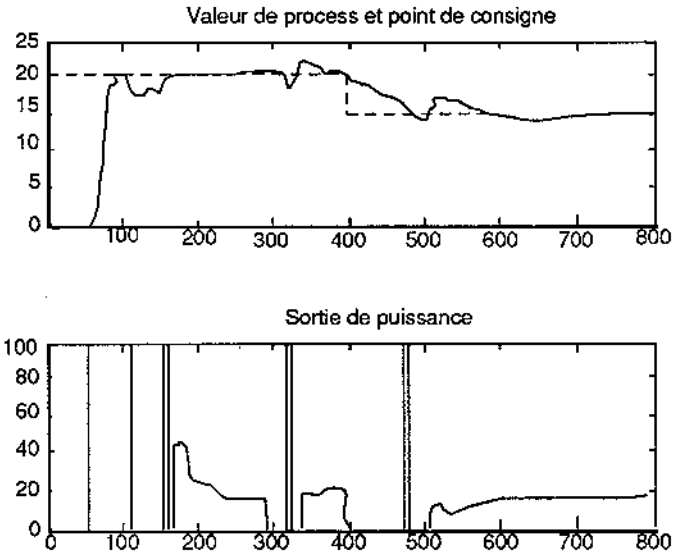


Figure 9-23 Séquence d'auto-réglage PID_Auto

3. Cutback : Cutback_Low est mis (si l'auto-réglant le juge nécessaire) sur un réglage en montant vers le point de consigne. De même, Cutback_High est mis (si nécessaire) sur un réglage en descendant du point de consigne. Pendant les réglages, aucun cutback n'est mis au point de consigne avant que la valeur de cutback ne soit inférieure à la bande proportionnelle nouvellement calculée.
4. Temps de cycle : les temps de cycle sont calculés si leur valeur initiale est mise à plus de 5 secondes. L'auto-réglant ne doit pas changer ces valeurs si elles sont réglées à moins de 5 secondes.
5. Paramètres de réglage adaptatif : l'auto-réglage règle aussi les 3 paramètres de réglage adaptatif : les paramètres Trigger_Val, MTC, et Q pour DRA (Analyse de réponse à une perturbation) et LSAT (Auto-réglant adaptatif des moindres carrés). Pour plus de détails, voir le chapitre Réglage adaptatif.

L'auto-réglant ne change pas le paramètre de temps de retard (zone morte). En mode manuel, des conditions de rupture de capteur ou de suivi sont entrées en un point quelconque, la séquence entière est stoppée et ne subit de modification sur aucun des paramètres.

N.B. : L'utilisateur doit s'assurer que la régulation mise en oeuvre est correcte (action directe ou inverse) : l'auto-réglant ne peut prendre la décision lui-même.

Les paramètres de diagnostic AT_State et Zn_Stage indiquent l'étape appropriée de l'auto-réglage (Autotune).

Comment obtenir les meilleurs résultats avec Autotune

Cette section donne quelques conseils sur la façon d'utiliser au mieux l'auto-réglage et comment éviter / minimiser quelques problèmes :

AT_State	Description
0	Remise à zéro
1	Initialisation
2	Bruit à l'état repos du moniteur
3	Fin du bruit du moniteur
4	Démarrage avec nouveau point de consigne
5	Fin démarrage avec nouveau point de consigne
6	Démarrage avec PV au point de consigne
7	Fin démarrage avec PV au point de consigne
8	Séquence Ziegler-Nichols
9	Calcul de nouveaux paramètres
10	Ecriture de l'état de mise à jour
11	Echec de l'auto-réglage
12	Auto-réglage réussi

Tableau 9-3 Variable d'auto-réglage AT_State

Zn_Stage	Description
0	Initialisation
1	Trouver pic PV & Sortie inverse
2	Trouver intersection PV / PV1
3	Trouver pic PV & Sortie inverse
4	Trouver intersection PV / PV1 & Test pour retard dominant
5	Trouver pic PV & Sortie inverse ou chang. PV
6	Trouver intersection PV / PV1 & Ajuster tendance et sortie
7	Trouver pic PV & Sortie inverse
8	Trouver intersection PV / PV1 & Calculer paramètres

Tableau 9-4 Variable de diagnostic d'auto-réglage Zn_Stage

1. Pour obtenir de bons résultats, ne passer en auto-réglage que si le niveau de sortie de l'instrument est stable et la variable de process à l'équilibre (choisir l'auto-réglage avec le point de consigne initial à la valeur de process, puis passer le point de consigne à la valeur voulue). Ne changer le point de consigne (si nécessaire) que pendant la première minute de l'étape d'attente.
2. Si les conditions de démarrage du process sont toujours très semblables et s'il n'y a pas de changement significatif des caractéristiques de charge, l'auto-réglage n'a besoin d'être effectué qu'une seule fois et non à chaque démarrage du process.
3. Le réglage en montant le point de consigne d'une régulation de température d'extrudeuse ou de four est en général plus efficace qu'en descendant la température ou en se plaçant au point de consigne lui-même.
4. L'auto-réglage n'est, en général, pas très efficace sur les process dont la constante de temps est inférieure à 20 secondes environ, à moins de régler PID_Auto avec un temps de tâche plus rapide. L'opération de réglage prend environ 5 ms du temps de tâche. Le temps de tâche ne peut pas être utilement divisé par un facteur supérieur à 2 ou 3, par rapport aux 100 ms initial. En augmentant le temps de tâche pour le bloc fonction PID_Auto, s'assurer que l'intervalle de temps de tâche est bien environ 100 fois inférieur à la constante de temps du process. Par exemple, si la constante de temps du process est de 100 s, la tâche interne ne doit pas être inférieure à 1 s.
5. Ne pas utiliser l'auto-réglage avec des process ayant des perturbations cycliques. Ceci risque d'induire le système de réglage en erreur, pour le réglage des valeurs PID.
6. Le système de réglage est inhibé si IntegralHold est actif.
7. Vérifier la cohérence des valeurs limites de sortie. Ne pas changer les limites pendant la phase d'auto-réglage.
8. S'assurer que la valeur de process se situe bien dans la plage utile du capteur. Certains capteurs (par exemple les pyromètres) ne sont utilisables que sur une plage limitée et ils indiquent leur valeur minimale même si la valeur de process est très en dessous de celle-ci. Si plusieurs capteurs sont utilisés pour plusieurs plages, s'assurer que l'auto-réglage se fait sur la bonne plage.
9. Il y a lieu de noter que les lois de réglage sont optimisées pour la régulation de température d'une extrudeuse ou d'un four et tendent à donner une régulation très serrée et active. Si ce type de réponse ne convient pas à certains systèmes, les paramètres doivent être modifiés en mode manuel.

10. Différents délais d'attente sont intégrés à la séquence d'auto-réglage. Si le process ne répond pas comme prévu, le système de réglage peut se dérouter sur l'un de ces temps d'attente avant de passer à la partie de la séquence qu'il est en mesure d'exécuter et donner sa meilleure estimation des réglages PID.

Réglage avec positionneurs de vanne

La fonction réglage non répétitif est également disponible pour le positionneur de vanne. Toutefois, le système ne règle ni le temps de rafraîchissement, ni le temps minimum d'activation, ni le temps de déplacement de la vanne. Il y a lieu de noter que le temps de déplacement de la vanne doit correspondre à la valeur réelle de déplacement d'une position extrême à l'autre. Sinon, le système risque de régler une bande proportionnelle incorrecte pour la régulation.

Réglage adaptatif

Le bloc fonction PC3000 dispose aussi de deux régleurs qui, contrairement au réglage non répétitif, fonctionnent en continu. C'est le DRA (Analyse de la réponse à une perturbation), qui est en fait un système expert simple qui examine la courbe de réponse de la valeur de process sur un changement de point de consigne ou sur perturbation externe, et le LSAT (Réglage adaptatif des moindres carrés) qui est, en fait, une variante de la stratégie de modélisation. DRA effectue le réglage approché et LSAT le réglage fin. Il y a aussi un superviseur qui coordonne l'activité de ces deux régleurs, en assurant également la surveillance de sécurité, par exemple :

pour le réglage des limites de paramètres PID

pour s'assurer de la cohérence du réglage du paramètre

pour contrôler l'activité de restauration de la régulation

Analyse de la réponse à une perturbation (DTA)

Cette technique repose sur une analyse continue des formes d'onde de la mesure en fonction du temps, pendant le fonctionnement de la régulation en boucle fermée. Les réponses réelles surveillées sont les signaux d'écart produits lorsque la boucle de régulation subit une perturbation. Des écarts de régulation pouvant être produits par un bruit du process, par des perturbations externes au process ou par des changements de point de consigne, il est judicieux de pouvoir différencier ces sources spécifiques pour évaluer la réponse à la perturbation. Un problème fondamental de la technique d'analyse de la forme d'onde est le fait que les informations relatives aux changements de la perturbation initiale peuvent être insuffisantes.

Le problème du bruit peut être résolu en disposant d'un moyen permettant de mesurer le bruit environnant, pendant le test d'auto-réglage, avant d'initialiser le réglage adaptatif et de paramétrer ensuite les filtres de bruit. Comme indiqué précédemment, l'auto-réglant règle le paramètre Trigger_Val qui doit être réglé approximativement sur l'amplitude crête-à-crête du niveau de bruit. Cette valeur peut également être ajustée par l'utilisateur. Si la valeur est réglée trop basse, l'auto-réglant adaptatif peut être brouillé par le bruit du process et il aura tendance à compenser en élargissant le réglage de la bande proportionnelle et en augmentant le temps d'intégrale. Dans la mesure du possible, il est préférable de procéder à un auto-réglage avant le réglage adaptatif. Sinon, s'assurer que Trigger_Val est au moins réglé sur l'amplitude crête-à-crête du bruit du process.

L'auto-réglant surveille les changements de point de consigne et agit en conséquence en cas de changement de point de consigne. Si les changements sont trop fréquents, le superviseur inhibe les changements de paramètres PID, car il est alors impossible de voir si la valeur de process oscille du fait des changements fréquents du point de consigne ou si le réglage de la boucle est mauvais.

Le DRA étudie la réponse du process sur une période de temps suffisante pour permettre des contrôles de cohérence des valeurs mesurées. Ceci s'étend, en général, sur une période de temps égale à 5 ou 6 fois le temps d'intégrale avant que le DRA ne soit sûr qu'un nouveau réglage est nécessaire.

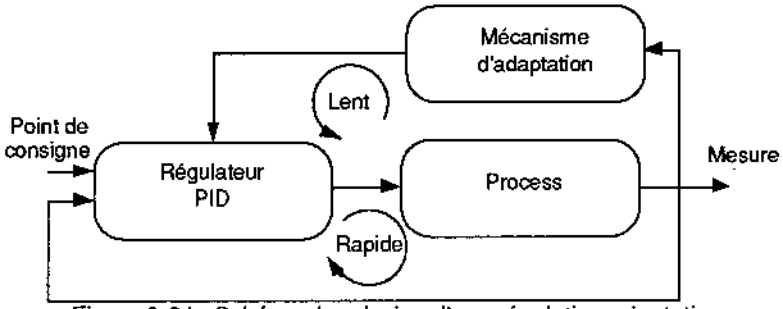
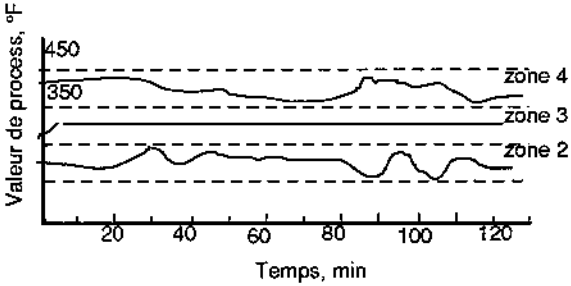


Figure 9-24 Schéma de principe d'une régulation adaptative



Zone 2	Auto-réglant	Réglage manuel	Sans changement	Réglage manuel	Adaptation
XP(%)	1,9	1,9		1,9	1,9
TI(sec)	930	240		30	960
TD(sec)	152	60		10	160

Zone 4	Auto-réglant	Réglage manuel	Adaptation	Manuel	Adaptation
XP(%)	1,7	0,8	1,4	1,4	1,4
TI(sec)	520	520	520	1800	613
TD(sec)	88	88	88	300	102

Figure 9-25 Réglage adaptatif DRA d'une extrudeuse multizone

Le caractère distinctif des "systèmes adaptatifs" est le fait que deux boucles de base sont présentes, comme le montre la figure 9-24. La boucle d'adaptation est plus lente, au mois d'un facteur 10, que la boucle fermée PID. Il ne serait, par conséquent, pas raisonnable de s'attendre à ce que la boucle d'adaptation réponde rapidement à des variations brusques du process. Par exemple, un process de pH subit des variations de gain rapides et importantes près de la zone de neutralisation.

Une régulation adaptative de type normal ne peut pas compenser ces changements -- le gain du process de pH varie presque aussi rapidement que les variables de débit d'effluent et doit par conséquent être traité en tant qu'état "de bonne foi" du process. Une table de paramétrage est ici la meilleure solution. Les méthodes adaptatives compensent toutefois un réglage initial insuffisant sur une période de temps suffisamment longue (par exemple quelques temps d'intégrale) dans la mesure où le process commandé n'évolue pas trop rapidement.

Lorsque le régleur DRA a été déclenché par un écart supérieur à Trigger_Val, les pics d'écart suivants sont mesurés comme pour l'auto-réglage (CLCM) décrit précédemment. Des boucles insuffisamment réglées peuvent avoir, après perturbation, une réponse oscillatoire amortie du genre de celles qui sont représentées sur la figure 9-25. Le DRA peut alors ajuster ces paramètres au moyen des tableaux décrits dans le Manuel des blocs fonctions du PC3000.

L'inconvénient du DRA et autres techniques d'analyse de formes d'onde est qu'une identification de réponse de régulation faible doit avoir lieu avant que la réponse de la régulation ne puisse être corrigée. Cette méthode est très efficace dans la mesure où la valeur de déclenchement est réglée judicieusement comme décrit précédemment.

Le DRA peut décider d'inclure une régulation intégrale même si celle-ci n'a pas été choisie à l'origine, mais elle ne règle pas le terme dérivée, qu'elle soit inactivée par l'utilisateur ou par l'auto-réglant.

Le DRA dispose de deux paramètres de diagnostic qui sont décrits dans les tableaux 9-5 et 9-6. DRA_State indique l'état en cours du DRA et DRA_Last le dernier changement effectué par le DRA sur les réglages de régulation.

DRA_State	Description
0	Laisser se stabiliser
1	Attendre le déclenchement
2	Trouver pic 1
3	Trouver zéro 1
4	Trouver pic 2
5	Trouver zéro 2
6	Trouver pic 3
7	Trouver zéro 3
8	Trouver pic 4
9	Trouver zéro 4
10	Trouver pic 5
11	Echec de fin sur zéro 4
12	Fin sur pic 4 trouvée
13	Echec de fin sur zéro 5
14	Fin sur pic 5 trouvée
15	Préparer mise à jour

Tableau 9-5 Les états de DRA

DRA_Last	Description
0	Refusée
1	Amortissement réduit
2	Gain augmenté
3	Temps diminué
4	Temps augmenté
5	Gain diminué

Tableau 9-6 L'adaptation de DRA

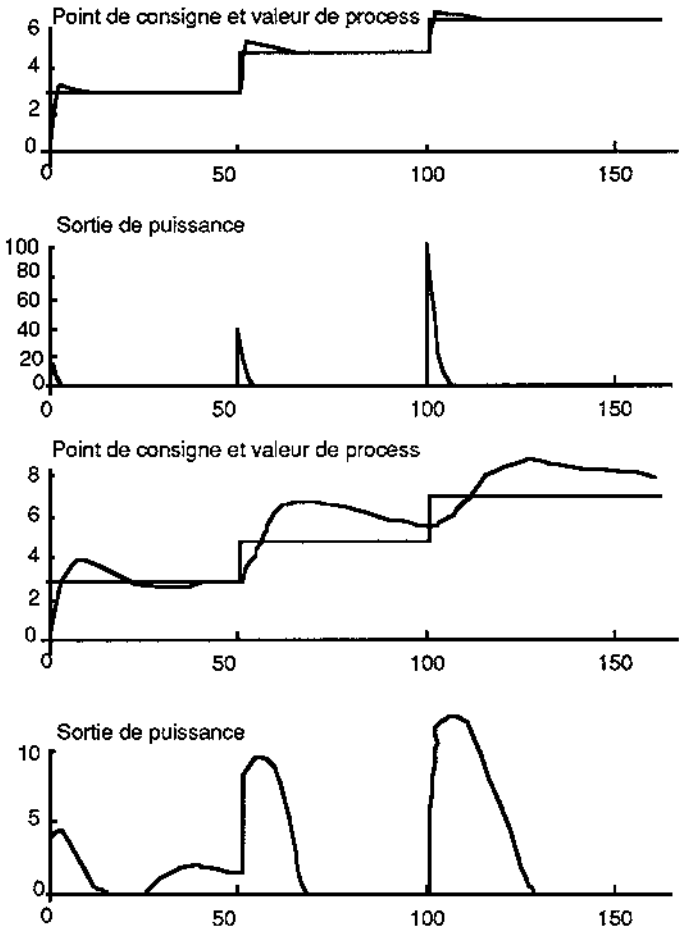


Figure 9-26 Schéma de principe du régleur adaptatif des moindres carrés

Le réglage adaptatif des moindres carrés (LSAT)

LSAT est une configuration qui inclut une variable de l'estimation récurrente par les moindres carrés et évalue les paramètres d'un modèle prédictif qui serviront ensuite à régler les paramètres PID. LSAT assure un réglage fin, en association avec le DRA décrit précédemment.

Deux paramètres sont utilisés avec LSAT : MTC (Constante de temps du modèle) et Q (facteur de déréglage de la régulation). L'auto-réglant et DRA règlent automatiquement ces paramètres. Lorsque la régulation est en mode manuel, ces valeurs sont réglées à 1/10 du temps d'intégrale et 8 % du réglage de la bande

proportionnelle (en unités physiques). Elles peuvent également être ajustées par l'utilisateur, bien que ce ne soit pas utile en pratique, à moins que l'utilisateur n'ait suffisamment d'expérience et de connaissance en LSAT. En bref, augmenter MTC a pour effet d'augmenter la constante de temps du modèle en boucle ouverte, et augmenter Q a un effet similaire mais en réduisant le gain de la boucle. LSAT est désactivée si MTC est inférieure à l'intervalle de temps de la tâche exprimée en ms divisé par 100. Par exemple, pour un PID_Auto avec une tâche de 100 ms, MTC doit être réglée au moins à 1, ce qui implique alors que le temps d'intégrale soit au moins de 10 secondes.

La structure de la régulation est représentée schématiquement sur la figure 9-34.

LSAT est utilisée pour le réglage fin et il ne faut pas, par conséquent, s'attendre à ce qu'il modifie notablement le paramètre réglé par l'auto-réglant ou par DRA. Contrairement à DRA, qui ne fonctionne que si l'écart est suffisamment important, LSAT fonctionne en continu et peut, en tant que telle, effectuer un changement à tout moment. Le superviseur s'assure que les paramètres sont bien dans les plages spécifiées.

Les trois algorithmes de réglage résultants constituent le "système de réglage Eurotherm polyvalent" qui est également disponible avec les instruments de la série 900HP.

LSAT, comme DRA, peut activer une régulation intégrale si nécessaire, mais ne peut pas activer une régulation dérivée si elle a été désactivée par l'utilisateur ou par l'auto-réglant.

Comment obtenir les meilleurs résultats avec le réglage adaptatif

Certaines précautions simples sont indispensables pour utiliser la régulation adaptative.

1. Dans la mesure du possible, procéder à un auto-réglage avant le réglage adaptatif, afin que les paramètres initiaux PID soient plus proches de leurs valeurs optimales qu'ils ne le seraient normalement. L'interruption due à la séquence d'auto-réglage n'est normalement pas pire qu'un PID insuffisamment réglé !
2. S'il n'est pas possible de procéder à un auto-réglage, s'assurer que la valeur de déclenchement soit réglée sur une valeur sensible (normalement, la valeur crête-à-crête du bruit).
3. Ne pas utiliser de réglage adaptatif sur les process à perturbations cycliques. Toutefois, s'il est nécessaire de l'utiliser ainsi, s'assurer que la valeur de déclenchement est bien réglée sur la valeur crête-à-crête de la perturbation.
4. Ne pas laisser la régulation en réglage adaptatif tout en modifiant le câblage, en débranchant les thermocouples ou en effectuant la maintenance de l'équipement. L'algorithme comporte un détecteur de perte d'action de restauration destiné à détecter ces situations, mais le meilleur remède est la prévention.

5. Comme pour l'auto-réglant, il est inutile d'essayer d'utiliser un réglage adaptatif sur les process dont la constante de temps n'est pas au moins 100 fois plus courte que l'intervalle de temps de la tâche de PID_Auto.

6. Il y a lieu de noter que le régleur adaptatif n'ajuste ni la zone morte d'un PID à deux voies, ni le temps minimum d'activation, ni le temps de déplacement, ni le temps de rafraîchissement du positionneur de vanne. C'est l'utilisateur qui doit les choisir.

Souvent, il suffit d'utiliser l'auto-réglant pour réduire le temps de mise en service. Cette régulation résoud 90 % des problèmes. Une adaptation n'est vraiment nécessaire que pour une petite classe de process. Par exemple, en utilisant une régulation PI adaptative sur un process ayant un temps de retard important, il n'est pas possible de réduire d'une manière significative les temps de réponse : une compensation du temps mort ou une nouvelle conception du système est nécessaire ici et non une régulation adaptative. Il est difficile de dire exactement dans quelles conditions la régulation adaptative donne les meilleurs résultats. Cependant, il faut savoir qu'un réglage non répétitif sera plus fiable sur une boucle bruyante et qu'un réglage continu, souvent, ne se justifie pas.

L'adaptation continue aux modifications des conditions de boucle est un problème différent. Ici, l'utilisateur doit comprendre que le réglage adaptatif, lorsqu'il est validé, a le contrôle absolu du process et qu'il faut faire attention jusqu'à ce que le process puisse bénéficier d'un certain degré de confiance. Les régulations adaptatives peuvent masquer les problèmes du process. Par exemple, le colmatage des tubes d'un échangeur thermique provoque une réduction progressive du coefficient d'échange thermique qui peut être compensé par la régulation pour obtenir des performances optimales dans les conditions données. Si ceci reste totalement inaperçu, le problème n'apparaîtra que lorsqu'il sera trop tard et la maintenance régulière en sera plus difficile !

TABLE DE PARAMETRAGE

Les tables de paramétrage apportent une solution simple mais efficace. La plupart des process industriels ne sont pas linéaires et par conséquent, un réglage fixe de paramètres PID ne permet pas d'obtenir des performances "optimales".

Il est parfois possible de trouver des variables auxiliaires qui sont en relation étroite avec les changements de dynamique du process. On peut les utiliser pour déclencher la sélection de nouvelles valeurs de paramètres qui correspondent mieux aux nouvelles conditions. La modification des paramètres peut se faire soit en continu, là où les paramètres PID (par exemple la bande proportionnelle, les temps d'intégrale et de dérivée) sont une fonction continue de la variable auxiliaire, soit par pas discontinus là ou la zone de fonctionnement est séparée en plusieurs zones, et les paramètres PID sont ajustés lorsque la variable auxiliaire passe d'une zone à la suivante.

Un inconvénient de la table de paramétrage est le fait que c'est une compensation en boucle ouverte de paramètres de contre-réaction. Il n'y a pas de contre-réaction pour compenser un réglage incorrect. Un autre inconvénient est le fait que l'étude peut demander du temps. Les paramètres du régulateur doivent être déterminés pour de nombreuses conditions de fonctionnement. La fonction auto-réglant décrite précédemment peut aider à réduire le délai de la phase d'étude.

L'avantage principal de la table de paramétrage est, par contre, une réponse rapide aux variations du process. Il n'y a pas d'estimation ni de transitoire de réglage qui ne soient, normalement, au moins 10 fois plus lents que le temps de réponse des boucles elles-mêmes. Il est ainsi possible de traiter avec une grande précision les variations très rapides du process.

Réglage continu

Considérons le cas d'une régulation de niveau dans un réservoir conique. On suppose, pour les besoins de l'exemple, que l'on désire utiliser une régulation PI pour des raisons de stabilité. En pratique, c'est très rarement le cas dans une régulation de niveau, mais ceci est pris pour les besoins de l'exemple. On voit aisément que :

la section transversale du réservoir conique augmente comme le carré du niveau du liquide,

le débit en sortie du réservoir augmente comme la racine carrée du niveau du liquide.

La figure 9-27 représente la réponse de la régulation de niveau PI à une série de changements identiques du point de consigne de 1 à 3, de 3 à 5, puis de 5 à 7 unités. Avec une régulation PI fixe (figure du bas), la réponse tend à devenir plus lente lorsque le niveau monte. En analysant cette régulation de niveau, on peut voir qu'il est possible d'effectuer une compensation en réduisant la largeur de la bande proportionnelle au fur et à mesure que la section transversale augmente. Se rappeler que le "gain" du réservoir varie en proportion inverse du niveau.

```
Pid.Prop_Band := 50 / (PV.Val * PV.Val);
Pid.Debump_Dis := 1;
```

Cela garantit d'avoir des réponses cohérentes à différents niveaux. Voir la moitié supérieure de la figure 9-27.

Il y a lieu de noter qu'il a fallu inhiber la suppression d'à-coup (Debump) lors de la table de paramétrage, à cause du fait qu'à chaque fois qu'un réglage de la bande proportionnelle est effectué, la régulation accomplit une procédure automatique de suppression d'à-coup. L'opération de suppression d'à-coup pourrait inhiber, dans ce cas, la sortie, et la régulation ne pourrait jamais être réglée correctement !

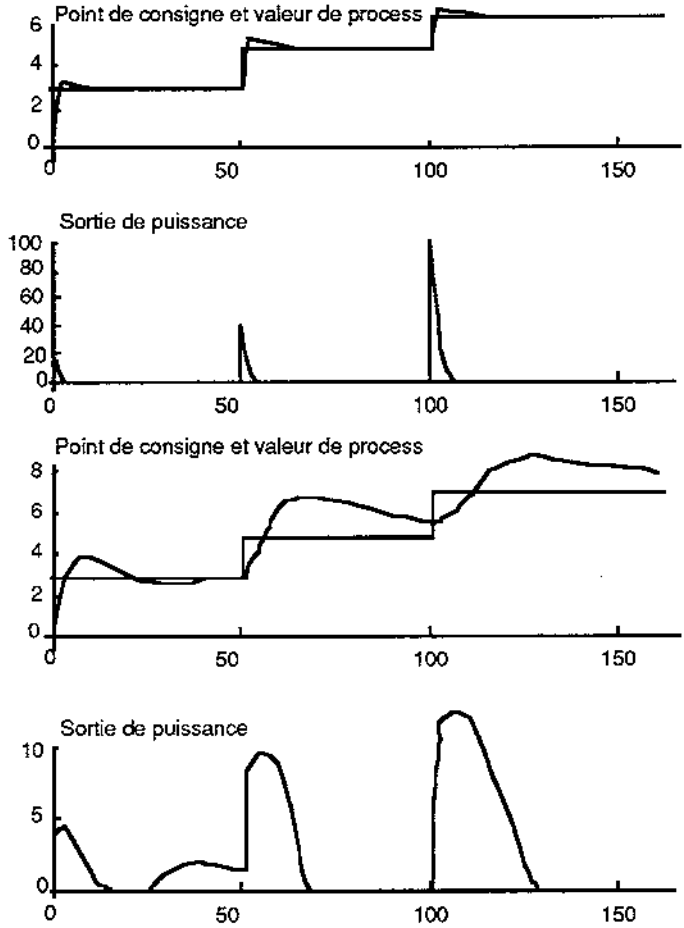


Figure 9-27 Réglage continu du gain

Table de paramétrage

Considérons un échangeur de chaleur. La température au secondaire de l'échangeur de chaleur est commandée par la position d'une vanne située du côté primaire. Le gain et la constante de temps du process d'échange thermique sont fonction du débit au secondaire.

Les variations du process peuvent être compensées en réglant la table de paramétrage en fonction du débit du côté secondaire de l'échangeur de chaleur. Lorsque le débit dans les tubes secondaires augmente, le gain de l'échange thermique diminue et la constante de temps augmente. Une façon simple de prévoir une programmation de gain pour régler la bande proportionnelle et le temps d'intégrale est l'auto-réglage en fonction de différentes conditions de fonctionnement et la mémorisation de ces valeurs dans un bloc fonction de sélection.

Trois zones différentes peuvent être prévues, pour un débit faible, moyen, ou fort. Les paramètres PI peuvent être mémorisés, par exemple dans un bloc Select_REAL, ou Select_TIME. Le câblage logiciel est alors :

```
Pid.Prop_Band := Pid_XP.Output;
Pid.Integral  := Pid_TI.Output;
Pid_XP.Index  := REAL_TO_DINT(IN := Flow.Process_Val/33.34) + 1;
Pid_TI.Index  := Pid_XP.Index;
```

Si tous les blocs fonctions sont sur la même tâche, l'ordre d'exécution est alors garanti et le PID est, par conséquent, sûr d'utiliser un réglage cohérent des paramètres. Les blocs fonctions de sélection agiront avant le PID.

Il y a lieu de noter que, contrairement à l'exemple précédent, la suppression d'à-coup n'est pas inhibée, afin de s'assurer que le passage d'un réglage de paramètres à l'autre ne crée pas d'à-coup en sortie. Bien entendu, le but recherché est que la variable de table de paramétrage (c'est-à-dire le débit) n'oscille pas rapidement entre ses limites. Il est possible d'utiliser un filtre passe-bas pour s'assurer que la sortie ne réagit qu'à des changements réels de débit.

Il y a lieu de noter également que la table de paramétrage doit nécessairement être limitée à P, I et D. Fréquemment, un autre paramètre doit être programmé, par exemple le gain relatif de refroidissement. Prenons une chambre climatique où la température est régulée de + 100 °C à - 190 °C. L'azote liquide assure un refroidissement efficace lorsque la valeur de process atteint -180 °C, par exemple. Pour obtenir des types de réponse du même ordre sur toute la plage de température, le gain relatif de refroidissement de la régulation doit être programmé, soit par rapport au point de consigne, soit par rapport à la valeur de process.

Il peut ainsi s'avérer nécessaire de commuter le type de la régulation. Par exemple : lorsque le rampage est actif, il est judicieux de faire agir la dérivée sur l'écart et de rebasculer, en phase de régulation, la dérivée sur la mesure. Il est clair que cette commutation est sans importance dans le cas du PC3000.

Une autre façon d'effectuer des tables de paramétrage consiste à utiliser différentes recettes. Si un même équipement est utilisé pour produire différentes sortes de produits, le système de formulation de recettes peut être utilisé pour commuter les valeurs PID ainsi que les points de consigne.

PROGRAMMATION DU POINT DE CONSIGNE

L'un des impératifs pour un système de régulation est un bon suivi du point de consigne. Dans beaucoup d'applications de régulation de température, il est nécessaire de suivre une séquence de rampes et de paliers de différentes longueurs. Dans le système PC3000, cela peut être réalisé très facilement au moyen du bloc fonction de rampe. Le point de consigne du PID est câblé en sortie du bloc fonction de rampe.

```
Pid.Setpoint := ramp.Output;
```

Le profil de référence est segmenté en une série de rampes et de paliers. Chaque segment peut alors constituer un pas de macro dans le programme séquentiel dont la fonction est de générer le profil de point de consigne.

On suppose que la sortie de la rampe est réglée sur SP1.Val et que la croissance linéaire jusqu'à SP2.Val doit se faire en T.Val secondes. Le taux de croissance linéaire est alors réglé simplement de la façon suivante :

```
ramp.Rate     := (SP2.Val - SP1.Val) / TIME_TO_REAL(IN:= T.Val);
ramp.Setpoint := SP2.Val;
```

L'ensemble de blocs fonctions Recipe fournit un excellent moyen de mémoriser une variété de programmes de points de consigne pour différentes sortes de produits. L'écriture d'une macro SFC qui sélectionne les entrées et sorties appropriées à partir des variables esclaves d'une série de recettes est relativement simple.

Inhibition de dépassement

L'un des problèmes associés aux régulations PI ou PID est le fait que la valeur de process peut dépasser la référence à la fin d'une rampe. Ceci est dû, en premier lieu, aux effets de l'action intégrale. Plusieurs solutions existent en fonction du domaine d'application, une ou plusieurs étant applicables au problème rencontré.

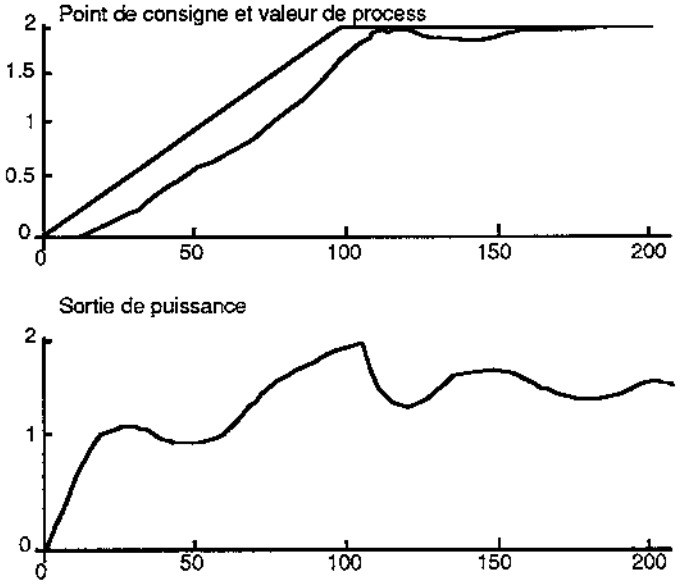
Utilisation de la fonction d'inhibition de dépassement de PID_Auto

La fonction d'inhibition de dépassement du bloc fonction PID_Auto est décrite dans le Manuel des blocs fonctions PC3000. Le câblage suivant active l'inhibition de dépassement :

```
Pid_Auto.Setpoint  := ramp.Output;
Pid_Auto.Ramp_Rate := ramp.Rate;
Pid_Auto.Ramp_Units:= ramp.Rate_Units;
Pid_Auto.Target_SP := ramp.Setpoint;
```

Le paramètre d'inhibition (inhibiter) peut alors être ajusté pour donner la réponse appropriée. Il doit normalement être de l'ordre de 0,5.

Le fonctionnement de la fonction d'inhibition est le suivant. Le dépassement étant dû, dans la plupart des cas, à l'effet cumulatif du terme intégral, la fonction inhibition supprime l'effet intégral en excès avant que la valeur de process n'atteigne le point de consigne recherché. Le point où cette suppression doit avoir lieu est fonction de la valeur d'inhibition. Cette valeur d'inhibition est réglée sur un nombre compris entre 0 et 1, zéro correspondant à l'absence d'inhibition. La figure 9-28 montre la réponse d'un PID à une rampe avec (moitié inférieure) et sans (moitié supérieure) activation de l'inhibition de dépassement. Comme on peut le voir, la réponse est identique, avec ou sans inhibition de dépassement, jusqu'à ce que la valeur de process se rapproche de la valeur recherchée.



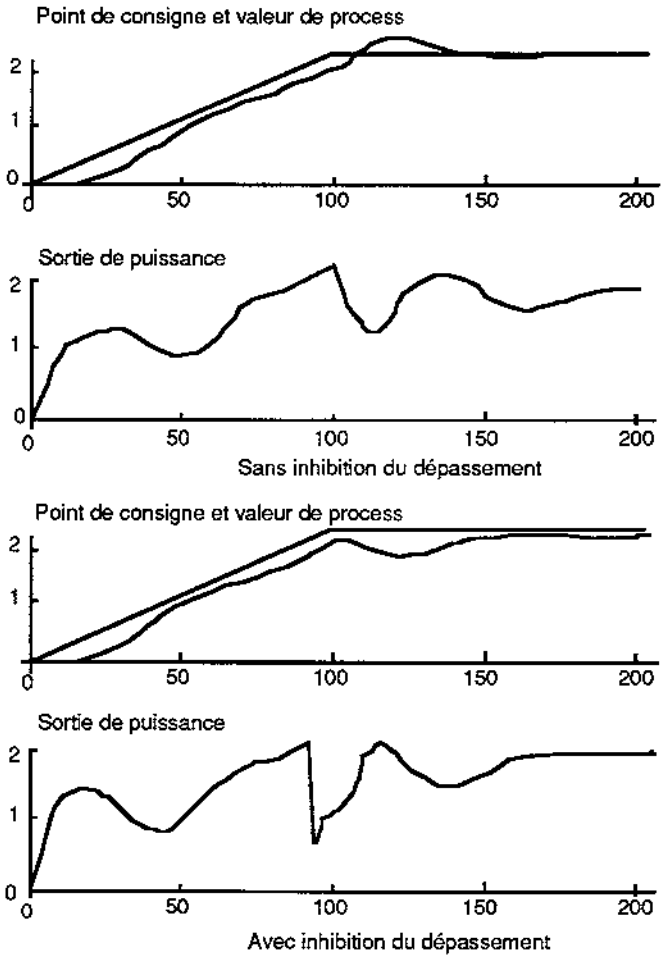


Figure 9-28 Avec inhibition du dépassement

Utilisation avec le bloc fonction PID

Les mesures prises par la fonction interne d'inhibition de dépassement du bloc fonction PID_Auto peuvent être trop sévères (par exemple si l'on demande à la régulation d'être limitée en vitesse) au moment de la suppression de l'intégral. Si l'action proportionnelle agit seule sur la mesure en sens opposé à l'écart, la réponse à la rampe ne risque pas de provoquer un dépassement, mais elle sera considérablement ralentie. Normalement, la dérivée doit également agir sur l'écart en sens opposé à la mesure pour réduire l'amplitude du dépassement. La figure 9-29 montre l'effet de l'utilisation de la pondération du point de consigne qui sera décrite ultérieurement. Il est possible d'obtenir la réponse appropriée pour un process en ajustant en conséquence les facteurs de changement d'échelle.

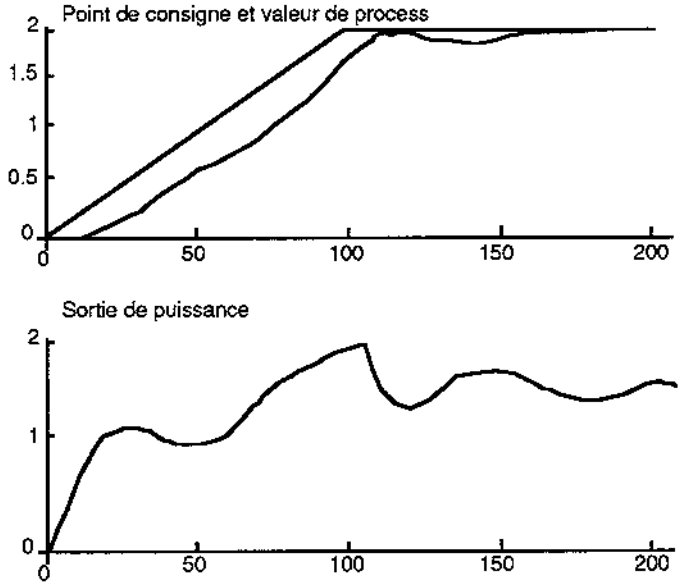


Figure 9-29 Dépassement et structure du PID

CONDUITE DE PROCESS

Régulation en cascade

L'un des outils les plus performants pour la conduite de process est la régulation en cascade. Il est tout à fait courant d'avoir à traiter des cas où il est possible de mesurer certains états d'un process, comme les températures, les pressions, etc. le long de la ligne. En extrusion, l'utilisation la plus courante de la configuration en cascade est la mise en oeuvre de thermocouples longs et étroits. De la sorte, l'effet des perturbations, comme la chaleur produite, ou bien l'effet de la granulométrie des pastilles de plastique ou de leur composition chimique peut être détecté plus rapidement par le thermocouple et peut ainsi être compensé plus vite. Il y a une grande variété d'applications, comme le séchage, les chambres climatiques, la conduite de process par des vannes (par exemple mise en cascade du débit et de la composition), où la régulation en cascade est particulièrement utile. Dans les fours, il est également tout à fait habituel de mettre en cascade les températures de la charge et de l'air.

Le câblage approprié de la régulation en cascade dans le système PC3000 est laissé à l'initiative du développeur de l'application, car une adaptabilité extrême donne plus de liberté pour configurer un système de régulation. Le présent document décrit quelques formes types de câblage de régulation en cascade de PID qui doivent être prises comme exemple et non comme solution unique à des applications de régulation en cascade.

Configuration de base :

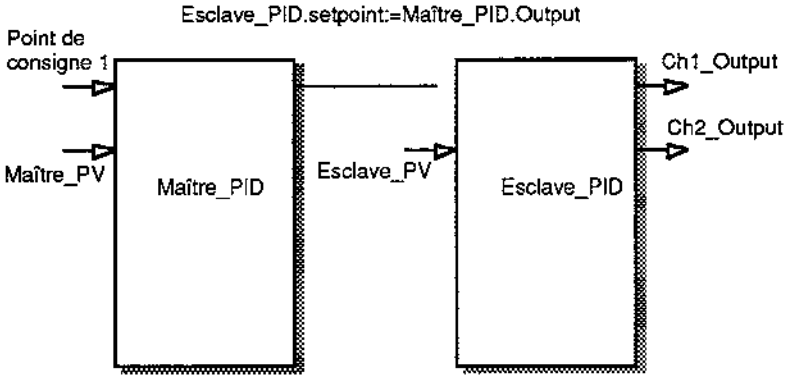


Figure 9-30 Configuration de base de la régulation en cascade

Dans son expresssion la plus simple, le PID en cascade est câblé selon la configuration représentée sur la figure 9-30. Le seul câblage est ici :

```
Esclave_PID.Setpoint := Maître_PID.Ch1_Output
```

Il faut noter que :

Il est impossible de désactiver le fonctionnement de la cascade.

La mesure esclave doit être exprimée en %, car la sortie du bloc fonction Maître_PID est en %.

Il n'y a pas de limites sur le point de consigne esclave.

Lorsque l'esclave est en mode manuel, la sortie du maître est sans effet et il existe un risque de dérive intégrale et d'à-coup (bumping) brutal lorsque la régulation repasse en gain automatique.

La figure 9-31 montre une configuration plus raisonnable pour une régulation en cascade. Les modifications suivantes ont été apportées à la configuration de base :

1. La sortie du PID maître est mise à l'échelle sur les unités de la boucle esclave de la façon suivante :

```
Esclave_Span.Val := Esclave_PID.Span_High -Esclave_PID.Span_Low;
Echel1_OP.Val := Maître_PID.Ch1_Output * Esclave_Span.Val/100 +
Esclave_PID.Span_Low;
```

2. Si la régulation en cascade est active, ce point de consigne est alors encadré par des limites haute et basse définies par l'utilisateur. Ceci peut se faire par le câblage suivant, par exemple :

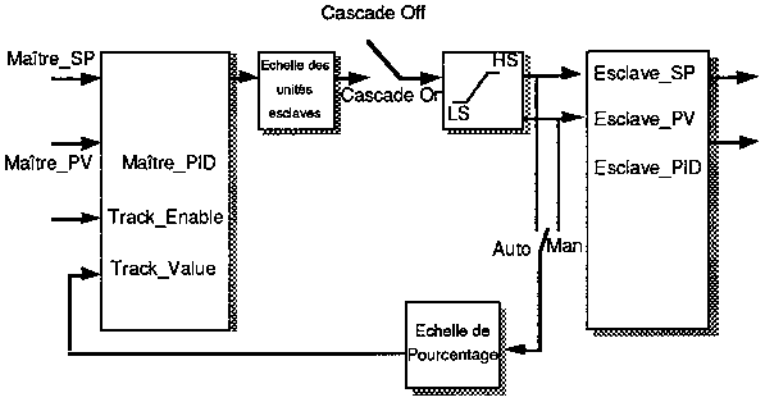


Figure 9-31 Configuration pratique de régulation en cascade

```
Esclave_PID.Setpoint := SEL_REAL( G:= Cascade.Val,
IN0:=Externe_SP.Val,
IN1:= SEL_REAL( G := Echel1_OP.Val > HS.Val,                    IN0 :=
MAX_REAL(IN1:= Echel1_OP.Val,                                  IN2 := LS.Val),
        IN1 := MN_REAL(IN1:= Echel1_OP.Val
        IN2:= HS.Val)));
```

Externe_SP est une variable définie par l'utilisateur qui peut être utilisée comme point de consigne local.

3. Pour assurer un transfert sans à-coup, le PID maître est réglé pour suivre la mesure ou le point de consigne de la boucle esclave, c'est-à-dire :

```
Maitre_PID.Track_Value := SEL_REAL(G := Esclave_PID.Manual,

            IN0(Esclave_PID.SetpointEsclave_PID.Span_Low)*100/Escla
ve_Span.Val,
            IN1 := (Esclave_PID.Process_Val-
Esclave_PID.Span_Low)*100/
            Esclave_Span.Val);
Maitre_PID.Track_Enable := Esclave_PID.Manual OR NOT Cascade.Val
                           OR SP_Limited.Val;
```

Le maître est réglé pour le suivi si :

l'esclave est en manuel

la cascade est désactivée

le point de consigne de l'esclave a atteint les limites.

Le contrôle du point de consigne en limite peut s'effectuer par :

```
SP_Limited.Val := Echel1_OP.Val>(HS.Val+EPS.Val)
                  OR Echel1_OP.Val<(LS.Val-EPS.Val);
```

EPS.Val est une variable utilisateur gardant une valeur faible, telle que 0,0001. Ceci est nécessaire pour s'assurer que l'arrondissement de l'écart ne bloque pas à l'infini le réglage en mode suivi.

Dans bien des circonstances (par exemple régulation en cascade de la température d'une zone de réservoir), la boucle interne (esclave) prend en charge toute l'activité de régulation et la boucle externe sert principalement de régulation en boucle ouverte avec un ajustage en boucle fermée pour s'assurer que les états stables sont corrects. Il est également nécessaire que l'écart de température entre variables primaire et secondaire ne devienne pas trop important. Pour cette raison, il est possible d'utiliser la fonction tendance pour le point de consigne ou la mesure.

1. La tendance du point de consigne s'effectue en réglant l'indicateur SP_FF_Enable dans la boucle principale et en réglant la valeur requise pour l'ajustage en %. Le reste du calcul est effectué par le bloc fonction. Il y a lieu de noter que la tendance du point de consigne n'est

disponible que pour des instruments uniquement thermiques. Ceci garantit que la sortie du bloc fonction varie de la valeur maximale de SP_FF_Trim autour de la valeur moyenne réglée par :

$$\frac{\text{Master\_PID.Setpoint} - \text{Master\_PID.Span\_Low}}{\text{Master\_PID.Span\_High} - \text{Master\_PID.Span\_Low}} \times 100$$

Cette fonction limite l'écart entre les points de réglage maître et esclave à une valeur maximale fixée par l'utilisateur. Prenons une boucle de température en cascade. La plage utile, à la fois pour la boucle primaire (maître) et la boucle secondaire (esclave), est de 600 degrés. Un point de mesure de 450 °C sur le maître est traduit par 450/600 ou 75 % du niveau de sortie de la boucle maître. Si un niveau d'ajustage de 5% est choisi, la sortie primaire peut alors varier entre 70 et 80 %. Cette mise à l'échelle pour le point de consigne de la boucle esclave (secondaire) implique une variation maximale de 420 à 480 °C pour le point de consigne esclave. Le déplacement du point de consigne maître sur 500 °C est immédiatement suivi par un décalage du point de consigne esclave d'une quantité appropriée.

2. Pour effectuer la tendance de la mesure, mettre l'indicateur PV_FF_Enable et régler PV_FF_Trim. Les calculs sont semblables à ceux de la tendance du point de consigne. La tendance de la mesure est utile pour la régulation de température dérivée.

   Considérons un réacteur discontinu. Il est typique de demander une différence de température maximale entre l'enveloppe et le produit. Une tendance de la mesure peut être utilisée dans ce cas, la régulation esclave contrôlant l'écart entre les mesures et la régulation maître réglant le niveau de base de la mesure principale. Il y a lieu de faire attention en réglant ces régulations car elles comportent des contre-réactions positives qui déstabilisent la boucle dans cette configuration.

Régleurs

Pour effectuer le réglage automatique des paramètres PID dans chacune des boucles, un câblage complémentaire est nécessaire pour s'assurer que la boucle externe (maître) soit au courant de l'état de la boucle interne (esclave). Cette fonction est prévue dans les régulations de la série 900EPC, le schéma de principe en étant représenté sur la figure 9-32.

Noter les différences par rapport à la figure 9-31.

Le suivi est également validé lorsque la boucle interne (esclave) est en auto-réglage (Autotune).

Si la boucle externe (maître) est à l'état stable, la séquence de bruit du moniteur ne doit provoquer aucun changement dans le niveau de sortie pendant les premières minutes pour éviter de perturber l'état stable. Pour cette raison, si la valeur absolue de l'écart est inférieure à 1 % de la plage utile, le point de consigne initial au début du bruit de moniteur de la boucle esclave est mis sur la mesure réelle . Puis, durant la première minute, le point de consigne peut être changé à volonté pour l'auto-réglant.

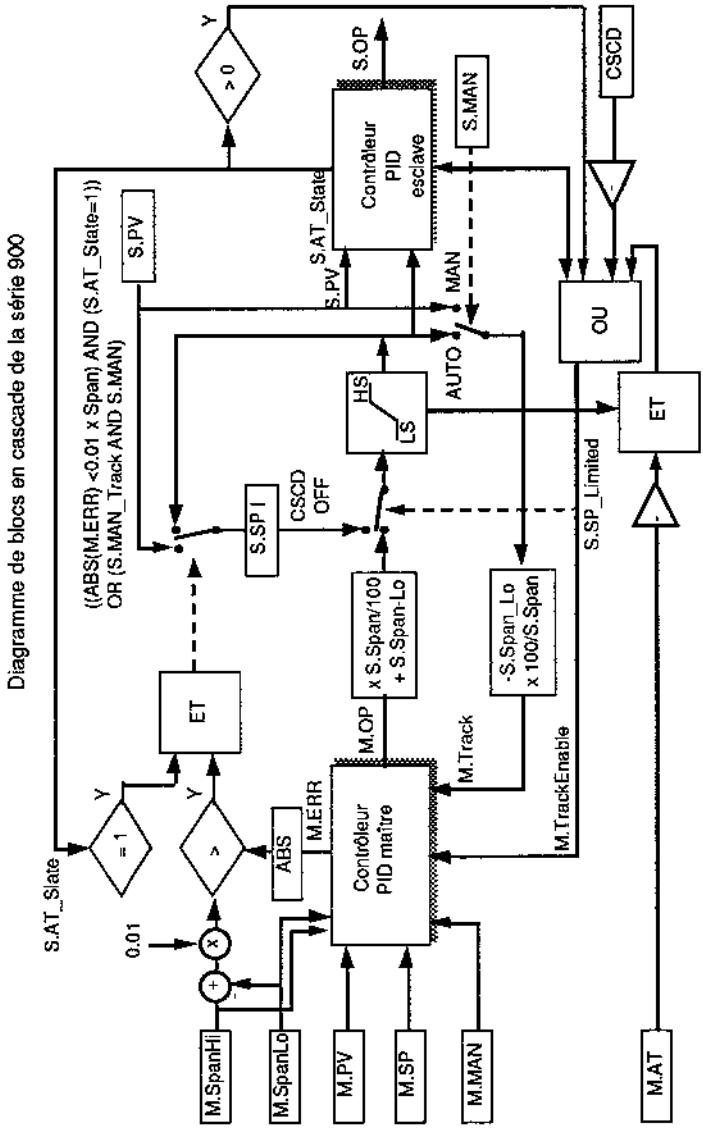


Figure 9-32 Mise en cascade de blocs fonctions PID_Auto

Dans certaines versions du progiciel, la séquence Autotune est fixe, de telle sorte qu'elle génère des sorties entre Output_High et Output_Low sans tenir compte d'aucun autre réglage. Pour une boucle maître ayant une possibilité d'ajustage limitée, ceci peut ne pas être acceptable. Pour résoudre ce problème pendant l'auto-réglage, les limites de sortie de puissance Output_High et Output_Low doivent être explicitement réglées sur des valeurs limites.

Il y a lieu de noter que :

1. Autotune (l'auto-réglant) doit toujours être exécutée en commençant par les boucles internes, avant les boucles externes.
2. Il est possible de régler à la fois les boucles interne et externe en mode adaptatif (Tune_Type = 4). Toutefois, par sécurité, le régleur adaptatif inhibe le rafraîchissement des paramètres si le point de consigne varie continuellement d'une façon aléatoire. Le régleur adaptatif de la boucle esclave ne doit par conséquent pas changer ses paramètres si la boucle maître est très active.
3. L'application du réglage adaptatif à la boucle maître doit se faire avec précaution, car le régleur adaptatif va essayer de compenser les instabilités de la boucle esclave en réglant les paramètres de la boucle maître.

Structure de la régulation

La configuration réelle de la régulation en cascade est beaucoup plus spécifique à l'application. Il y a toutefois certains points généraux qu'il faut prendre en considération.

1. Rebouclage intégral : tant que la sortie de la régulation secondaire n'a pas atteint l'une de ses limites de saturation, il n'y a pas de problème.

   Par contre, si la sortie de la régulation secondaire est à l'une de ses limites de saturation pendant un temps assez long, la partie intégrale de la boucle primaire risque fort de se reboucler. Une solution simple pour résoudre ce problème est d'inhiber l'intégration de la boucle primaire lorsque la régulation de la boucle secondaire est en saturation, en utilisant la fonction d'inhibition de la partie intégrale. Une autre solution possible est de placer la régulation primaire en mode suivi lorsque la régulation secondaire est en limite.

   Il est nécessaire de recalculer la valeur du point de consigne secondaire qui aurait provoqué la saturation de la régulation et de la reconvertir en pourcentage de la sortie primaire pour l'utiliser en tant que valeur de suivi. Ceci est plus compliqué que d'arrêter l'intégration qui donne le même résultat.
2. Action PI et PD : en général, il n'est pas nécessaire d'appliquer une régulation PID complète à la fois aux boucles de régulation primaire et secondaire. Normalement, le but recherché est d'avoir une boucle interne suffisamment serrée. Ceci peut être obtenu avec une régulation uniquement proportionnelle ou PD. Le terme dérivée doit toutefois agir

sur la mesure secondaire en sens opposé à l'écart secondaire. Si le terme dérivée agit sur l'écart secondaire, la régulation sera beaucoup trop active, en raison du changement continuel du point de consigne secondaire (c'est-à-dire, la régulation primaire). La régulation primaire, pour la même indication, doit être une régulation PI afin de réduire l'activité totale de la régulation par suite d'une action dérivée possible.

3. Rupture de capteur primaire ou secondaire : la stratégie en cas de rupture de capteur dans un système à boucles multiples peut être très différente de la stratégie en boucle simple où la sortie de régulation est réglée sur une valeur de sécurité. En cas de rupture du capteur de la boucle interne, il peut être possible de fonctionner avec la boucle externe agissant comme régulation à boucle unique à gain réduit, et vice versa. Le programme utilisateur peut prévoir une grande variété d'automatismes de sécurité.

Fonctionnement de base

Généralement, les boucles maître et esclave doivent être toutes deux configurées pour qu'en mode manuel les mesures suivent les points de consigne. Ceci peut être obtenu par la fonction SEL_REAL avec Maître_PID.Manual ou Esclave_PID.Manual en déclenchement. Il y a lieu de noter que cette prescription particulière vient en plus de celles qui ont été décrites dans les paragraphes précédents.

Le passage de l'esclave en Auto a pour effet de conserver au départ la dernière valeur de sortie et le point de consigne esclave est mis à la mesure esclave à cet instant, ce qui permet de passer sans à-coup du mode manuel au mode automatique. Un passage ultérieur en régulation automatique dans la boucle maître provoque l'ajustement de la sortie maître sur une valeur équivalente au point de consigne esclave en cours, tandis que le point de consigne maître est mis à la mesure maître. Lorsque le passage en régulation en cascade totalement automatique est achevé, tous les changements du process se font en modifiant le point de consigne maître.

Si une régulation du process en mode manuel est nécessaire, le passage de l'esclave en mode manuel provoque le gel de la sortie esclave sur la dernière valeur jusqu'à ce que la sortie esclave soit modifiée par l'opérateur. Pendant que l'esclave est en mode manuel, la sortie maître est mise en suivi de la mesure esclave mise à l'échelle, de telle sorte que lorsque l'esclave retourne en mode automatique, la sortie maître et le point de consigne esclave sont tous deux à la mesure esclave. Le point de consigne maître reste, toutefois, inchangé.

Si les points de consigne ne suivent pas les mesures en mode manuel, des "à-coups" peuvent se produire lors des changements de mode.

Pour des raisons de sécurité, le fonctionnement manuel doit prendre le pas sur toute autre activité de régulation (par exemple sur l'auto-réglage).

Régulation de rapport

Il est très rare que les boucles d'un système de régulation soient totalement indépendantes les unes des autres. Les applications de mélange en sont un exemple significatif. La plupart des applications de régulation de rapport commandent le rapport de deux débits entre eux -- il y a lieu de noter que de nombreuses mesures de débit sont soit bruyantes, soit difficiles à quantifier, voire les deux. Il en résulte que des filtres d'entrée doivent être normalement prévus pour toutes les applications de régulation de rapport. Ici, par principe, on utilise une boucle PID unique dont les points de consigne sont obtenus par :

```
Pid.Setpoint := PV_Lead * Ratio_Gain + Bias;
```

On voit bien que PV_Lead est la mesure pilote, le paramètre Ratio_Gain pouvant être soit fixe, soit ajustable, l'ensemble pouvant fonctionner comme un réglage simple à partir d'un terme de polarisation fixe.

Les applications de dosage sont une extension naturelle de la régulation de rapport de base, la régulation optionnelle de la variable de débit pilote nécessitant une seconde boucle PID. Le débit pilote est normalement commandé par rapport à un point de consigne fixe, bien qu'une programmation du point de consigne puisse être demandée en fonction du domaine d'application. Le point de consigne du débit de dosage est normalement réglé en proportion de la mesure pilote réelle avec un ajustage manuel externe en fonction de certaines mesures effectuées hors ligne.

La régulation de rapport air/carburant sur un brûleur est un autre exemple. Dans cet exemple, il est courant d'avoir la boucle de débit d'air proportionnelle à la boucle de débit de carburant. Normalement, la boucle d'air a un réglage serré pour réagir rapidement et la boucle de carburant est réglée de façon à répondre lentement aux changements du point de consigne, etc. Ceci implique que le point de consigne de la boucle de débit d'air soit toujours réglé avec un taux fixe par rapport à la mesure de la boucle de débit de carburant. La figure 9-33 représente une boucle élémentaire de taux air/carburant.

La combustion nécessite souvent d'être riche en air pour réduire le risque de production de fumée ou de suies dans le conduit d'évacuation, un ajustage de taux peut ainsi être utilisé pour régler le taux au-dessus de la valeur stoechiométrique. Toutefois, pour maintenir une atmosphère riche en air, la régulation doit commuter entre l'air pilote, lorsqu'une augmentation de température est nécessaire, et le gaz pilote lorsque la température doit être réduite. Dans ce cas, les deux boucles sont réglées avec des temps de réaction semblables. Normalement, le point de consigne est donné par une régulation maître qui commande la température du four, un analyseur d'oxygène étant utilisé pour ajuster le taux réel de façon à avoir la quantité nécessaire d'oxygène en excès dans le conduit d'évacuation.

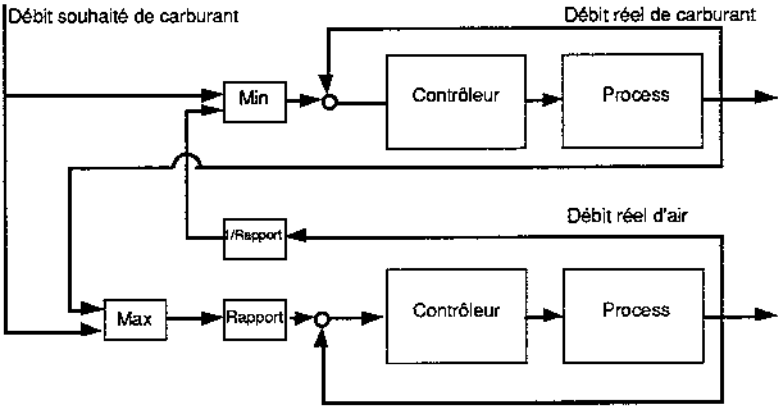


Figure 9-33 Configuration simple d'une régulation de rapport

Les fabricants de chaudières prévoient les brûleurs pour qu'ils puissent utiliser aussi bien le fuel que le gaz, avec possibilité de commutation entre ces deux carburants. Pour résoudre ce problème, il suffit de commuter le taux et le gain au moyen de fonctions de sélection dans le cas le plus simple, ou bien au moyen de recettes, dans les cas les plus complexes.

Attention

Il y a lieu de faire attention aux changements rapides de débits d'air et de carburant, car les taux qui s'écartent fortement du point de consigne sont potentiellement explosibles. Les brûleurs, en général, comportent des dispositifs internes de sécurité, mais un certain niveau de sécurité peut également être intégré au niveau du logiciel, par le programme utilisateur.

Réglage

Comme dans la situation de la régulation en cascade, le programme utilisateur doit prévoir la possibilité d'activer et de désactiver la régulation de rapport. L'auto-réglage doit, normalement, être effectué en désactivant le rapport. Dans la plupart des circonstances, ceci signifie que l'auto-réglage doit précéder le fonctionnement normal. En règle générale, ceci doit s'effectuer avec des limites plus serrées sur la sortie du bloc fonction pendant l'auto-réglage.

Tendance

La tendance, comme la contre-réaction, est un outil très puissant pour la conduite de process. La contre-réaction est, par principe, un type de régulation à effet contre-réactif : l'influence des perturbations est connue avant que le système de régulation ne puisse réagir pour compenser le déséquilibre. Dans bien des cas, il est possible de prédire l'effet des perturbations externes et de les compenser par avance. C'est le cas des régulations de chauffage, de ventilation ou de conditionnement d'air dans les grands bâtiments. De nombreuses influences externes sont connues ou mesurables. La période normale d'occupation du bâtiment est relativement bien connue, l'apport solaire peut être évalué dans une certaine mesure, la température et le degré hygrométrique de l'air pris à l'extérieur sont facilement mesurables. Au lieu de tout faire reposer sur la contre-réaction, il est possible de compenser ces influences en faisant appel à la tendance, les signaux de régulation étant fonction de ces influences externes, tout comme les signaux normaux de contre-réaction de la température et du degré hygrométrique de la pièce. Pour résumer, si la source de perturbation externe est connue et peut se mesurer avec précision, une combinaison de tendance et de contre-réaction donne les meilleurs résultats.

Contrairement à la contre-réaction, la tendance n'engendre pas d'instabilité. Il y a toutefois un prix à payer. Tout d'abord, la complexité de la configuration de la régulation augmente en même temps que le nombre de signaux de tendance : chacun nécessite un transducteur, un transmetteur, probablement une mise en forme du signal -- chacun ajoutant à la complexité. En outre, il est possible que les performances soient moins bonnes avec la tendance que sans elle. Ceci vient du fait que l'action prédictive est une régulation en boucle ouverte. Considérons l'exemple d'un système HVAC (Chauffage, ventilation et conditionnement d'air). Si l'entrée d'air est froide, il sera demandé au compresseur frigorifique de réduire son activité. Si la compensation prédictive demande d'augmenter l'activité du compresseur frigorifique, l'humidificateur et le chauffage devront augmenter leur activité pour compenser cette action prédictive incorrecte ! Ceci implique que pour avoir une tendance raisonnable, il est nécessaire d'avoir une bonne connaissance du process et une modélisation du process est nécessaire. Un autre point important est le fait qu'aucun modèle n'étant parfait et la tendance étant dépourvue de mécanisme auto-correcteur, il faut toujours associer la tendance à la contre-réaction.

En règle générale, deux cas de figure sont possibles :

contre-réaction avec ajustage prédictif, ou

tendance avec ajustage contre-réatif.

Le type de modèle à construire est différent pour l'un et l'autre de ces cas. Considérons une régulation de température simple pour un sécheur rotatif. La teneur en eau du produit entrant peut être mesurée très simplement par le courant consommé par l'entraînement des convoyeurs qui amènent le produit à sécher : plus la marchandise est humide, plus elle est lourde et plus elle pèse sur les convoyeurs, plus le couple résistant est important pour les moteurs et plus le

courant du moteur est élevé. Ce signal peut donc servir pour une tendance. Celle-ci est potentiellement très utile car il y a généralement un retard significatif entre l'alimentation en gaz et l'augmentation de la température dans le sécheur.

Si la contre-réaction seule est utilisée, la marchandise humide à l'arrivée va d'abord provoquer une chute importante de température dans le sécheur et la régulation ne va commencer à réagir qu'après. Mais l'effet sur l'action corrective de la régulation ne commence à se faire sentir qu'après un certain temps, etc. Avec une tendance, l'action corrective peut s'effectuer dès que la matière entre dans le sécheur. Les corrections à court terme doivent être judicieuses, car la régulation à contre-réaction n'est pas en mesure de les corriger et il y a un risque d'entrer en oscillation si les corrections de tendance à court terme provoquent une surcompensation.

Il est important de définir correctement le modèle à court terme. La situation à moyen et à long terme est quelque peu différente. Le régulateur de température est prévu pour compenser des variations à long ou à moyen terme dans un sécheur - c'est à cela que sert la régulation dans le premier exemple. Par conséquent, il est sans importance que les modèles à long ou à moyen terme soient incorrects dans ce cas, car la contre-réaction s'en occupera. Par long/moyen terme, on désigne des constantes de temps de l'ordre d'une fois la constante de temps de boucle ouverte à 5 fois la constante de temps en boucle fermée (noter que le rapport entre le temps d'intégration et la constante de temps en boucle fermée est d'environ 3:1).

Si la tendance est la régulation principale et la contre-réaction l'ajustage, il est important d'avoir également une bonne compensation de la tendance pour le moyen terme. Le niveau d'état stable peut être laissé à l'ajustage de contre-réaction. Dans ce cas, l'ajustage PI a normalement une action limitée (c'est-à-dire que le signal de sortie de régulation a des limites étroites).

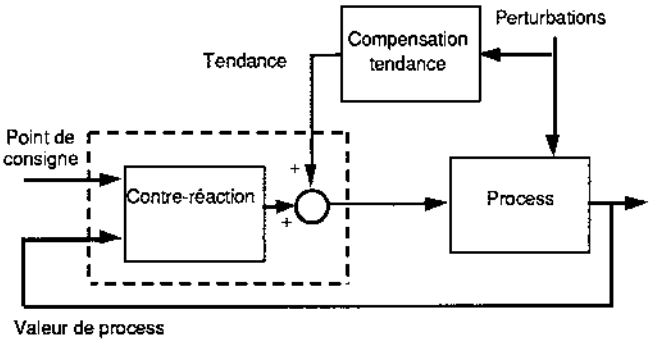


Figure 9-34 Association d'une contre-réaction et d'une tendance

Souvent, il est possible d'obtenir de bons résultats avec des compensateurs prédictifs très simples -- les avantages marginaux suceptibles d'être obtenus par des améliorations complémentaires sont si faibles qu'il n'est pas économique de poursuivre cette voie.

Tendance de base

La figure 9-34 montre le schéma de principe d'une tendance type. Il y a lieu de noter que les signaux de régulation de la tendance et de la contre-réaction s'ajoutent et sont traités ensuite pour obtenir les effets tels que le transfert manuel/auto sans à-coup, la désaturation intégrale, etc. En fonction de la nécessité ou non d'avoir une limite d'ajustage interne, le câblage suivant peut être utilisé pour une tendance et pour un bloc PID.

Pas de limite d'ajustage PID requise :

```
Pid.Feedforward := FeedFW.Val;
```

Limite d'ajustage PID requise :

```
Pid.Feedforward := FeedFW.Val;
Pid.Output_High := MIN_REAL(IN1:= FeedFW.Val + Trim.Val,
    IN2:=High_Limit.Val);
Pid.Output_Low  := MAX_REAL(IN1:= FeedFW.Val - Trim.Val,
    IN2:=Low_Limit.Val);
```

Il y a lieu de noter que la limitation décrite ci-dessus ne prend pas en compte l'influence du gain relatif de la voie 2 .

FeedFW est le signal de compensation calculé pour la tendance. Il y a plusieurs points importants dans l'étude et le réglage des compensateurs prédictifs.

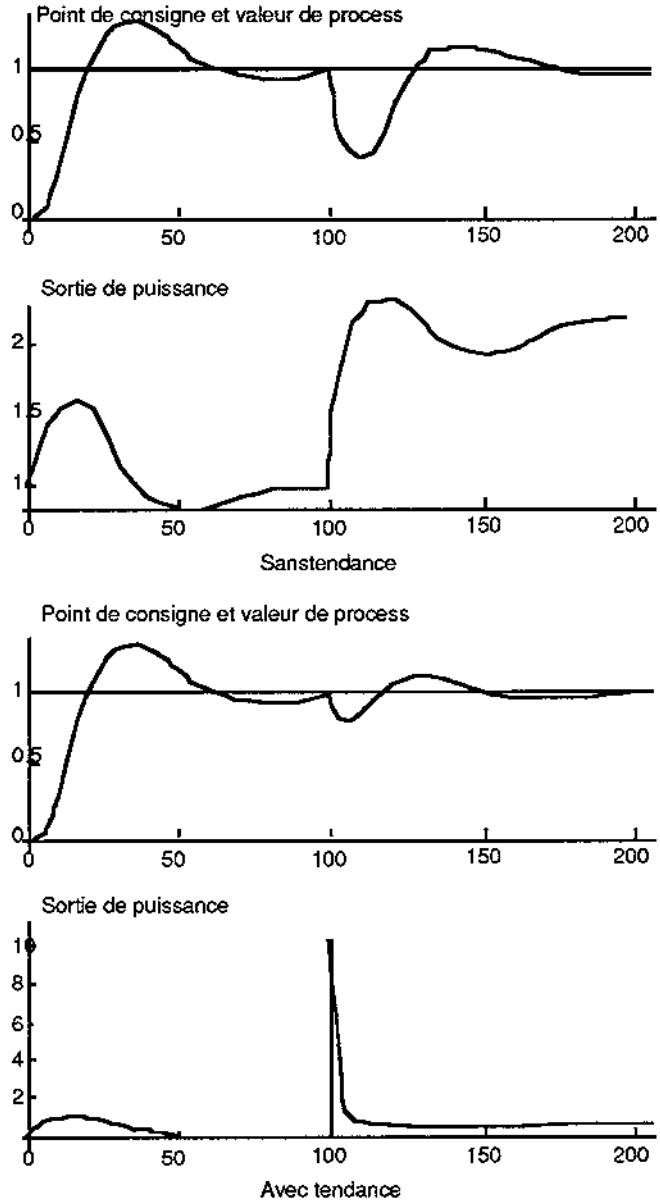


Figure 9-35 Influence des perturbations et de la tendance

Considérons une tendance dans une enceinte climatique. L'influence d'une chute de température de l'entrée d'air sur la température finale n'est probablement pas différente de celle de l'exemple de la figure 9-32. La moitié supérieure de la figure montre la réponse de la tendance à un changement de la demande et à une perturbation. La moitié inférieure montre la même tendance assistée d'un compensateur servant de tendance. Il y a lieu de noter la nette amélioration des propriétés de réjection des perturbations.

La réponse est typique d'une grande quantité de process. Il y a un temps mort pendant lequel l'effet du changement n'a pas encore atteint le capteur. Ensuite vient une période transitoire pouvant être assimilée à un retard pendant laquelle la température atteint sa nouvelle valeur d'équilibre. Chaque réponse peut être définie par trois nombres : un gain (K), un temps mort (D) et une constante de temps de retard (T). La méthode pour les calculer à partir de la courbe de réaction est identique à celle décrite précédemment. La compensation de la tendance doit, à cet effet, effectuer un calcul contre-réactif pour la détermination de son signal de régulation. Il y a trois cas élémentaires : le temps mort de la perturbation du process (Df) est supérieur au temps mort de la sortie de la régulation pour la mesure (Dp), Df est égal à Dp et enfin Df est plus petit que Dp. Chaque cas est envisagé successivement.

Lorsque les retards sont exactement les mêmes, il n'y a pas d'autre impératif que d'essayer de compenser différents taux de réaction (c'est-à-dire de constantes de temps). Ce qui signifie que le compensateur doit être de la forme :

$$u_{ff}(t) = -\frac{Kf \times (1+sTp)}{Kp \times (1+sTf)} v(t)$$

c'est-à-dire un compensateur de type avance ou retard. Voir la partie suivante qui traite de la réalisation de cette fonction de transfert dans le PC3000. Tp est la constante de temps du signal de régulation de la mesure et Tf est la constante de temps de la tendance pour la mesure. Kf et Kp sont les gains correspondants, leur valeur pouvant être aussi bien positive que négative.

Si Dp est plus petit que Df, alors la meilleure méthode consiste à retarder la compensation pendant une durée égale à la différence entre les deux temps morts, puis de procéder comme dans le cas précédent. La fonction de transfert du compensateur est alors :

$$u_{ff}(t) = -\frac{Kf \times (1+sTp)}{Kp \times (1+sTf)} e^{-s(Df-Dp)} v(t)$$

Le dernier cas est celui pour lequel l'effet de la tendance est plus rapide que le signal de contre-réaction. On ne peut rien faire en ce qui concerne la transition initiale : le signal de régulation ne peut pas atteindre à temps le point de détection. Le reste de la transition peut toutefois être légèrement amélioré. Plus généralement, la constante de temps du numérateur doit être le temps de réaction de la mesure et le dénominateur la tendance sur le temps de réaction de la mesure. Dans ce cas, la fonction tendance a une influence importante.

Dans bien des cas, en particulier lorsque les constantes de temps sont comparables, un gain sur la tendance est suffisant. Comme indiqué dans l'introduction, il est en général préférable de pêcher par excès de précaution. Une correction excessive est en général pire qu'une correction insuffisante. Lorsque le gain sur la tendance appropriée a été déterminé à partir des données du process, on peut réduire légèrement ce gain sur la tendance : de très bons résultats pouvent être obtenus de la sorte.

Comme pour toutes les stratégies reposant sur une modélisation, meilleur est le modèle, meilleures sont les performances générales, et la plupart du temps, des modèles simples peuvent donner de très bons résultats.

Une remarque peut être ajoutée concernant les méthodes d'obtention de ces modèles.

Signal de tendance mesurable et manipulable : c'est un cas classique dans la plupart des configurations de conduite de process à boucles multiples. La tendance est utilisée dans ce cas pour découpler les boucles le plus possible. La tendance est prise alors à partir soit des points de mesure soit de la sortie de la régulation. Il est possible d'obtenir des modèles tout à fait acceptables (si le niveau de bruit et les conditions du process le permettent).

Signal de tendance non manipulable : dans ce cas, il est possible de faire tourner le système avec uniquement une régulation par contre-réaction pendant un temps suffisamment long de façon à recueillir assez de données qui contiennent des décalages de niveau dans le signal de tendance. Il est alors possible de faire passer les données acquises par un filtre passe-bas et d'en déduire raisonnablement le gain en présence. Le gain et la constante de temps du signal de régulation de la mesure peuvent s'obtenir par des essais en boucle ouverte. Il reste la constante de temps du dénominateur (retard) qui doit être obtenue par approximations successives ou par des considérations physiques.

L'obtention de données de réponse à un échelon comme indiqué ici peut conduire à des erreurs importantes. L'approche qui est, de loin, la plus fiable, est l'analyse statistique correcte des séries de temps et l'application de signaux de test appropriés. Ceci nécessite toutefois des outils très élaborés, et une longue période d'étude est nécessaire pour qu'elle ait un tant soit peu de valeur.

Stratégies de régulation avancées

Variantes de configurations PID

Il est très difficile de concevoir une régulation PID ayant de bonnes propriétés de réjection de perturbations (par exemple taux d'amortissement 1/4) ainsi qu'un bon suivi de la grandeur de référence. Cette étude nécessite, normalement, deux degrés de liberté, alors que le PID standard n'en permet qu'un seul. Avec le système PC3000, il est relativement facile de retrouver le degré de liberté supplémentaire.

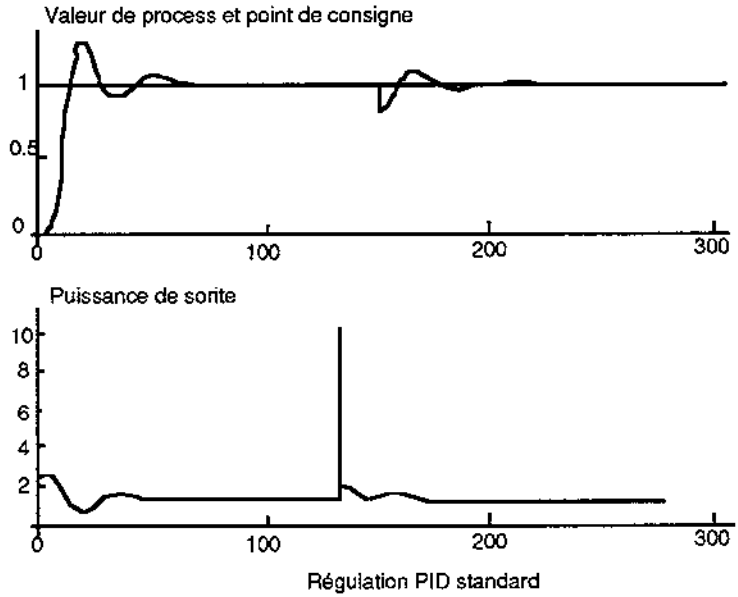
La régulation est réglée normalement pour donner de bonnes performances de régulation (par exemple par les lois de Ziegler-Nichols). Les régulations PID réglées selon Ziegler et Nichols sont connues pour donner un mauvais suivi de référence - souvent avec un dépassement inacceptable. Trois solutions sont possibles :

1. Dérégler les paramètres de la régulation pour obtenir une réponse qui soit un compromis des deux situations.
2. Utiliser des rampes ou des pré-filtres sur le point de consigne pour réduire l'amplitude du dépassement.
3. Utiliser une pondération du point de consigne. La pondération du point de consigne est décrite par Hang, Astrom and Ho [5] et la loi de régulation est donnée par :

$$\text{Output} = \frac{1000}{\text{Span} \times \text{PB}}\left((a \times \text{SP} - \text{PV}) + \frac{1}{\text{Ti}}\int e(t)dt + \text{Td}\frac{de}{dt}\right)$$

Une dérivée peut être choisie pour agir sur la mesure au lieu de l'écart, comme cela est plus fréquent dans les process industriels. Il y a lieu de noter que la différence entre la configuration PID normale et cette version appelée "à deux degrés de liberté" est le choix du terme de pondération "a". Cette configuration peut être obtenue par le câblage de la tendance indiqué ci-après.

```
Pid.Feedforward := b.Val * Pid.Setpoint;
b.Val       := (a.Val-1.0)*10000/(Pid.Prop_Band*(Pid.Span_Low-
                                  Pid.Span_High));
```



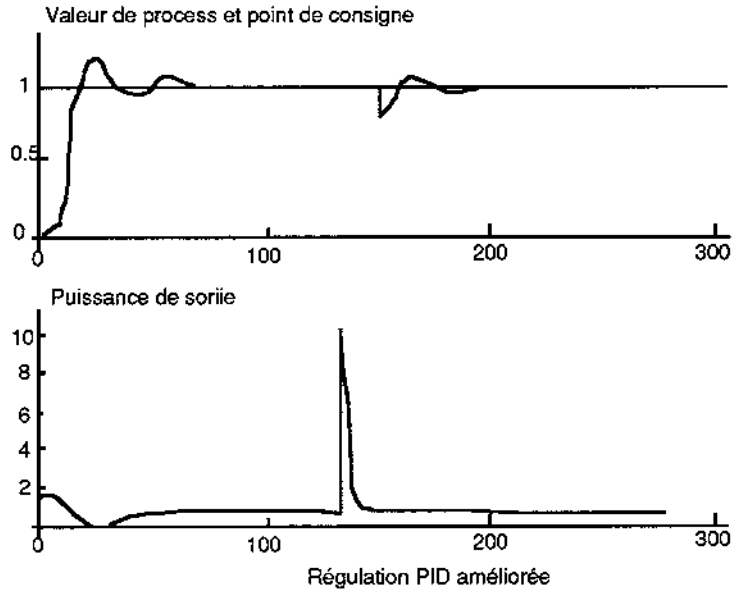


Figure 9-36 Régulations PID standard et améliorée

La pondération "b" n'a besoin d'être évaluée qu'en cas de modification de "a", du réglage de la bande proportionnelle ou des valeurs de la plage utile. Si "a" est mis à zéro, ceci équivaut à ne faire agir le terme proportionnel que sur la mesure.

La figure 9-36 montre un exemple de régulation PID réglée selon Ziegler-Nichols et selon la méthode améliorée. Il y a lieu de noter que la réponse au point de consigne avec la méthode améliorée est bien meilleure, tout en gardant inchangée la bonne réjection des perturbations. En clair, il est possible d'appliquer la même technique à la dérivée (c'est-à-dire qu'une partie de la dérivée pourrait agir sur l'écart et une partie sur la mesure). En fait, ceci n'est pas aussi utile dans le cas de la dérivée.

Il est courant d'utiliser le taux de variation du point de consigne comme terme de tendance lors d'une rampe de suivi. Ceci peut être fait très simplement en câblant le paramètre de taux de la rampe et la tendance avec un gain ajustable.

Commutation et suivi de la mesure

Il est très courant d'avoir à réguler une variable spécifique (par exemple une pression) pendant un cycle particulier d'une machine, puis de commuter sur une autre variable (par exemple une tension) durant le cycle suivant. Ceci est également tout à fait courant avec des capteurs qui ont une plage limitée. Selon la situation, diverses solutions sont possibles.

Commutation matérielle de mesure : s'il s'agit simplement de commuter entre des batteries de capteurs qui mesurent tous la même caractéristique du process, la solution la plus simple est d'effectuer la commutation avec une fonction SEL_REAL. Pour éviter un à-coup lors de la commutation, il est important de régler en même temps le paramètre Debump.

Commutation logicielle de mesure :Le plus souvent, la commutation ne peut pas s'effectuer par le moyen matériel décrit ci-dessus. Dans ce cas, il est possible de définir une zone où les deux mesures sont amenées proportionnellement à la régulation.

```
Pid.Process_Val := Weight.Val*ANIN1.Process_Val + (1.0-Weight.Val)*
ANIN2.Process_Val;
```

De cette manière, la commutation peut s'effectuer en douceur. La variable utilisateur Weight peut être ajustée linéairement avec changement de l'une des mesures.

Fonction de suivi PID : si le réglage de la régulation, le point de consigne ainsi que les mesures sont commutés, il est normalement plus facile d'avoir deux régulations, l'une suivant l'autre quand elle est inactive. Ceci est obtenu en câblant les régulations PID en opposition avec un interrupteur relié au signal de validation de suivi. On suppose ici que les deux contrôleurs sont à un seul canal.

```
Pid_1.Track_Enable := NOT Pid_1_Enable;
Pid_1.Track_Value  := Pid_2.Ch1_Output (* Feedback *);
Pid_2.Track_Enable := Pid_1_Enable; Pid_2.Track_Value
      := Pid_1.Ch1_Output;
```

Le plus souvent, la sélection s'effectue automatiquement. Prenons le cas où deux variables, la température de l'enveloppe et la température de la charge (PV1) d'un réacteur discontinu doivent être régulées. Le premier PID sert à réguler la température de la charge au point de consigne voulu et le second PID sert à réguler la température de l'enveloppe à PV1 plus une différence donnée. L'actionneur de la régulation (élément chauffant) est piloté par le PID ayant le signal de sortie minimal.

Une situation semblable a lieu avec une vanne utilisée pour commander un débit ainsi qu'une pression en amont d'une chaudière cylindrique. Une façon simple pour y parvenir est d'utiliser le câblage suivant :

```
Pid_1.Track_Enable := SEL_BOOL(G:= Pid_1.Track_Enable,
                                IN1:= 0,
                                IN0:= Pid_1.Ch1_Output > MINOP.Val
            (*Feedback *) + EPS.Val OR
    Pid_1.Ch1_Output < MINOP.Val      (* Feedback *) - EPS.Val);
Pid_1.Track_Value := MINOP.Val (* Feedback *);
Pid_2.Track_Enable := SEL_BOOL(G:= Pid_2.Track_Enable,
                                IN1:=0,
                                IN0:= Pid_2.Ch1_Output > MINOP.Val(* Feedback
*) + EPS.Val OR          Pid_2.Ch1_Output < MINOP.Val(* Feedback *) -
            EPS.Val);
Pid_2.Track_Value := MINOP.Val (* Feedback *);
MINOP.Val := MIN_REAL(IN1 := Pid_1.Ch1_Output, IN2 :=Pid_2.Ch1_Output);
```

EPS est un petit nombre positif et est utilisé pour éviter les problèmes d'arrondissement en arithmétique à virgule flottante. La séquence de fonctionnement sera alors la suivante : si la sortie du PID n'a pas été sélectionnée à la précédente scrutation, le bloc fonction passe en mode suivi et est réglé pour suivre la dernière valeur minimale. Si la sortie de la régulation a, par contre, été sélectionnée, le PID est réglé pour choisir sa sortie en conséquence. Si la régulation a été réglée en suivi à cette scrutation, elle sera sortie du mode suivi à la scrutation suivante. Cette configuration simule une version incrémentielle de régulation PID, où chacune suit l'autre.

Il aurait été peut-être plus facile d'effectuer quelques unes de ces décisions dans un pas SFC au lieu d'utiliser un câblage de blocs fonctions.

Compensation d'avance, de retard et d'avance-retard

Dans de nombreux domaines d'application, spécialement avec la tendance décrite au chapitres 8-9, il est nécessaire d'avoir des fonctions qui effectuent une compensation d'avance, de retard ou d'avance-retard.

Un bloc d'avance ou de retard peut se représenter sous la forme :

$$OP = \frac{1+sT_1}{1+sT_2} \times PV$$

Ceci peut se réaliser avec le câblage suivant mettant en oeuvre le bloc fonction Lag1.

```
lag1.Input    := PV.Val;
Gain.Val      := TIME_TO_REAL(IN:= T1.Val)/TIME_TO_REAL(IN:=T2.Val);
OP.Val        := Gain.Val * (PV.Val - lag1.Output) + lag1.Output;
```

Le paramètre "Gain" n'a pas besoin d'être calculé en câblage car il peut être calculé dans le SFC chaque fois que les constantes de temps sont ajustées. Le bloc fonction Lag1, par exemple, doit normalement être associé à une tâche dont l'intervalle de temps est inférieur à T2/10. Le réglage de la condition initiale pour ceci est très simple. La sortie du bloc fonction Lag1 est en suivi dans l'état Track, de telle sorte que l'activation du compensateur avance-retard est équivalent au fait de passer de l'état Track à l'état Limit.

Un bloc avance, retard ou avance-retard est simplement une mise en cascade d'un retard suivi d'une avance, ou vice versa.

Régulation modélisée et compensation d'attente

Les régulations PI et PID ne conviennent pas pour commander des système à dominance de temps d'attente. Si une régulation serrée est fréquemment requise, une stratégie reposant sur une modélisation doit être utilisée. La régulation à modèle interne (IMC) de Morari et Zafiriou [7] fournit une structure permettant de concevoir des stratégies de régulation modélisée dans le cadre du système PC3000.

Schéma de tendance de Smith

Le schéma de tendance de Smith traduit dans le modèle interne de structure de régulation est représenté par le schéma fonctionnel de la figure 9-37.

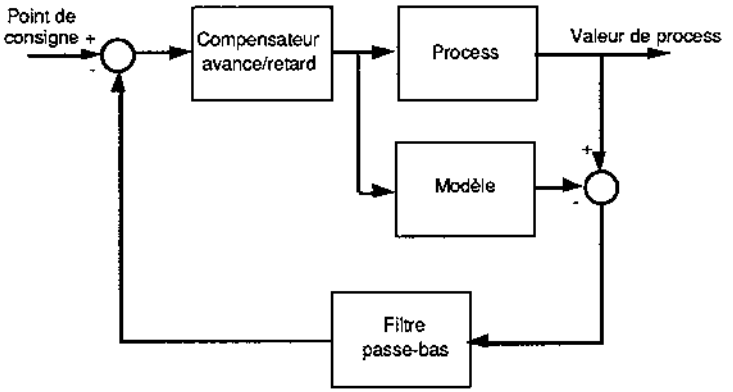


Figure 9-37 Schéma de tendance de Smith

Il y a lieu de noter que la régulation, dans la configuration IMC, se compose de trois parties distinctes :

Un modèle interne - Le modèle interne a pour but de reproduire la mesure le plus précisément possible. Deux causes de non-concordance entre la mesure modélisée et la mesure réelle sont la non-concordance entre l'installation et le modèle, et l'influence des perturbations en charge. Ces deux causes sont à l'origine du signal d'écart en retour qui traverse le filtre passe-bas. En l'absence de perturbation et de différence de modélisation, le signal en retour est nul, ce qui implique que le suivi de référence est alors directement ajustable au moyen du compensateur avance-retard.

Dans la mesure où il n'y a pas d'écart de modélisation, la stabilité en boucle fermée est assurée si le process, son modèle interne et le compensateur sont stables. De plus, l'action intégrale de la configuration de régulation obtenue est assurée si le gain à l'état stable du compensateur avance-retard est égal à l'inverse du gain du modèle à l'état stable. Le modèle est en général du premier ou du second ordre avec temps mort, bien qu'il n'y ait aucune raison, en principe, pour que le modèle soit d'ordre inférieur ou linéaire. En fait, des modèles très élaborés peuvent être construits avec les outils simples disponibles.

Un filtre passe-bas - Ce filtre passe-bas est prévu pour filtrer le bruit de la variable process et pour fournir une certaine robustesse à la dynamique non modélisée du process, comme l'effet de la non-linéarité et de la dynamique d'ordre élevé. La constante de temps du filtre doit être suffisamment importante pour réduire le niveau de bruit et suffisamment faible pour ne pas trop dégrader les propriétés de réjection des perturbations. Le gain à l'état stable du filtre doit être unitaire.

Un compensateur avance-retard - Le compensateur avance-retard est utilisé pour régler le temps de réponse en "boucle fermée". Le compensateur se présente normalement sous la forme d'un compensateur de tendance d'avance ou de retard. L'avance est utilisée pour accélérer la réponse en boucle fermée par rapport à la boucle ouverte et le retard pour ralentir la réponse en boucle fermée par rapport à la boucle ouverte. Normalement, une avance de facteur 2 entre les constantes de temps est utilisée, le numérateur ayant la même constante de temps que le process en boucle ouverte et le dénominateur la moitié de la constante de temps.

Une méthode simple pour obtenir un modèle raisonnable est la méthode de la courbe de réaction décrite précédemment. Considérons la réponse à un échelon, selon la figure 9-38. Le modèle interne du système est défini par :

$$M(s) = \frac{K_m e^{-sDm}}{(1+sT_m/2)^2}$$

La mise en oeuvre d'un retard peut se faire au moyen du bloc fonction Shift_Real. Dans cet exemple, le retard estimé est divisé par 10, ce qui définit la vitesse d'horloge en sortie du circuit pilote vers le bloc fonction Shift_Real. Le câblage de chacune des parties distinctes de la configuration est le suivant.

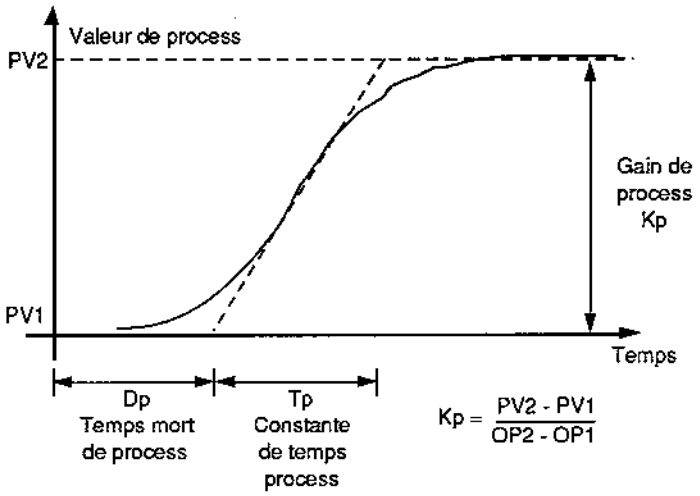


Figure 9-38 Réponse générale à un échelon

Modèle interne

```
OnDelay.Input    := OffDelay.Q_Output;
OffDelay.Input   := NOT OnDelay.Q_Output (* Feedback *);
Delay.Clock      := OnDelay.Q_Output;
Delay.Process_Val := Lead.Val;
Model_1.Input    := Delay.Output_11;
Model_2.Input    := Model_1.Output;
```

Le temps programme de chacune des temporisations d'activation et de désactivation est pris égal à 1/10 du retard estimé. La constante de temps de chacun des retards du modèle est prise égale à la moitié de la constante de temps estimée. Lead.Val est la valeur limite du compensateur d'avance indiqué ci-après.

Compensateur d'avance :

```
lag1.Input       := Setpoint.Val - FilterErr.Output;
Gain.Val         := TIME_TO_REAL(IN:= T1.Val)/TIME_TO_REAL(IN:=T2.Val);
OP.Val           := (Gain.Val * (PV.Val - lag1.Output) + lag1.Output)/
    Mod_Gain.Val;
Lead.Val         := MIN_REAL(IN1:= MAXOP.VAL,
                 IN2:= MAX_REAL(IN1:=MINOP.VAL,IN2:=OP.Val));
```

Normalement, T1 est réglé sur Tm et T2 sur Tm/2

FilterErr est un exemple de bloc fonction lag1 dont la constante de temps est normalement égale à (Tm + Dm)/2

Filtre passe-bas :

```
FilterErr.Input := ActualPV.Val - Mod_Gain.Val * Model_2.Output;
```

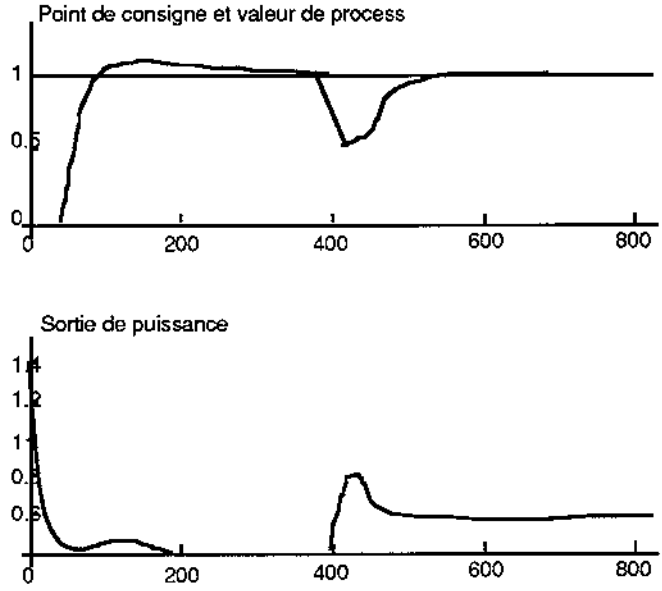


Figure 9-39 Régulation à modèle interne d'un process à temps de retard

La figure 9-39 montre la réponse d'un système utilisant la stratégie IMC simple. Dans cet exemple, le modèle du process ne concorde pas avec le process réel, mais la réponse de la régulation est très bonne. La réponse est comparable à un PID bien réglé pour le même process que celui représenté sur la figure 9-40. Il est clair que la réponse à un échelon représente une grande amélioration pour la régulation IMC, mais les propriétés de réjection de perturbation des deux configurations sont semblables.

Régulation inférentielle

Fréquemment, il n'est pas possible de mesurer directement les variables de régulation. Un exemple des plus évidents est celui de la culture de "bactéries" dans les process de fermentation pour produits manufacturés, comme la pénicilline. La mesure de la couleur et de l'éclat des céramiques en est un autre exemple. Dans bien des cas, il est possible d'établir une relation dynamique entre la variable qui ne peut pas être mesurée en ligne et d'autres variables d'environnement, comme la température, la pression, le degré hygrométrique, la teneur en oxygène, etc. Dans ce cas, il est possible d'utiliser le PC3000 pour une régulation inférentielle de cette variable secondaire.

L'inférence de la variable non mesurable peut être à la base de relations statiques ou dynamiques. Les relations statiques sont identiques aux calculs servant à déduire un débit massique à partir d'un débit volumétrique calculé au moyen de mesures de pression dérivée sur une plaque d'orifice et de la relation température / densité. Un autre exemple est celui des mesures de l'humidité relative et de la température au thermomètre mouillé / sec. Considérons le cas où il existe une relation dynamique entre la température et la composition d'un produit. Si le modèle dynamique est calculé empiriquement ou à partir de quelques séries d'études, les blocs fonctions Lag1 et Shift_Real peuvent alors être utilisés pour réaliser les équations d'émulation / tendance.

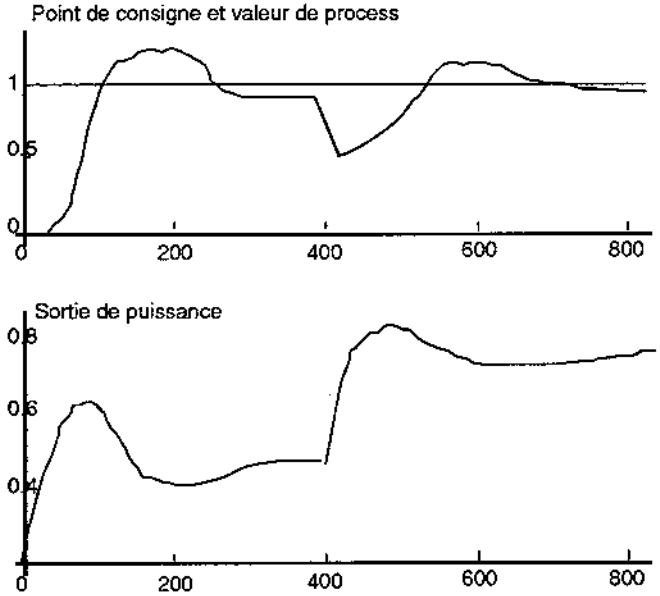


Figure 9-40 Régulation PI sur un process à temps de retard

Autres techniques de régulation

Strategies de régulation en temps discontinu

La méthode de régulation en temps réel discontinu, ou par échantillonnage de données, a suscité beaucoup d'intérêt avec les calculateurs de régulation de process. La plupart des problèmes de régulation par contre-réaction peuvent être résolus avec les configurations PID bien connues et éprouvées, avec adaptation, table de paramétrage, boucles multiples, etc. Dans le cas contraire, des stratégies reposant sur la modélisation peuvent faire face aux problèmes restants. Il y a pourtant certaines catégories de problèmes qui nécessitent des stratégies d'échantillonnage de données. De nombreux appareils de mesure comme les

analyseurs chimiques, les chromatographes en phase gazeuse, certains instruments de mesure d'épaisseur ou de profils, ne fournissent leurs mesures qu'à des intervalles de temps fixes. Beaucoup d'autres mesures sont transmises par des réseaux de communication et sont donc, par leur nature même, des données échantillonnées. D'autres encore, comme la pression maximale atteinte dans une machine à mouler par injection, sont des données échantillonnées du fait même des caractéristiques du process. Certains algorithmes de calcul, comme l'optimisation, l'analyse statistique ou même certains systèmes de détection de défaut sont intrinsèquement à temps discontinu.

Avec le PC3000, le développement d'algorithmes de régulation à temps discontinu est une tâche relativement prédictive.

Considérons un filtre passe-bas à temps discontinu de réponse d'entrée infinie (IIR) de premier ordre. Il peut se réaliser très simplement par le câblage suivant :

```
PVf.Val := PVf.Val + 0.1 * (Input.Val - PVf.Val);
```

C'est un filtre passe-bas avec une constante de temps de 1/0,1 ou 10 échantillons. L'intervalle réel d'échantillonnage du filtre est fixé en associant PVf à un intervalle de tâche approprié.

Un grand nombre de régulations en temps discontinu correspondent à une équation dérivée de la forme :

$$R(q^{-1})u(t) = T(q^{-1})w(t) - S(q^{-1})y(t)$$

dans laquelle R, T, et S sont des polynomes en décalage arrière de l'opérateur q-1 et y(t), w(t) et u(t) sont respectivement la mesure, le point de consigne et le signal de régulation. Il y a de nombreuses façons de réaliser de telles configurations. Une façon simple consiste à utiliser trois blocs fonctions Shift_Real, un pour chaque flot de données de la mesure, du point de consigne et de la sortie.

```
OnDelay.Input   := OffDelay.Q_Output;
OffDelay.Input  := NOT OnDelay.Q_Output (* Feedback *);
Y.Clock         := OnDelay.Q_Output;
Y.Process_Val   := PV.Val;
W.Clock         := OnDelay.Q_Output;
W.Process_Val   := Setpoint.Val;
U.Clock         := OnDelay.Q_Output;
U.Process_Val   := OP.Val (* Feedback *);
OP.Val          := (T1.Val * W.Output_1 + T2.Val * W.Output_2 +
                    R2.Val * U.Output_1 + R3.Val * U.Output_2-
                    S2.Val * Y.Output_1 + R3.Val * Y.Output_2)/R1.Val;
```

L'intervalle d'échantillonnage est défini, dans ce cas, par la variable Prog_Time dans les temporisations d'activation et de désactivation. Des régulations de type Dahlin peuvent être ainsi réalisées.

Un domaine où la régulation par données échantillonnées est particulièrement utile est celui des systèmes à taux de production variable. En clair, lorsque le taux production augmente ou diminue, les échelles de temps, et parfois le gain du process commandé, changent également. Les temps morts variables sont fréquemment la source de sérieux problèmes pour l'étude d'un système de régulation. Avec les stratégies de régulation en temps discontinu, il est possible d'échantillonner en fonction du taux de production et non plus en fonction du temps. Ceci signifie qu'avec une conception judicieuse, il est possible de convertir un problème qui est, par principe, variable dans le temps, en un problème sans variation en position, par exemple. Une régulation de concentration dans une conduite est un exemple où l'intervalle d'échantillonnage peut être réglé pour être inversement proportionnel au débit. Lorsque le débit augmente, l'intervalle d'échantillonnage diminue, et vice versa. La régulation de base en temps discontinu n'a pas besoin de changer et un problème potentiellement difficile est ainsi résolu par un moyen simple.

Mécanismes adaptatifs simples

Normalement, le système PC3000 ne fournit pas de bloc fonction d'usage général pour constituer des dispositifs de régulation adaptative. Il est toutefois possible de programmer des algorithmes de gradient spécifiques à l'application pour l'adaptation des paramètres.

L'équation générale pour l'adaptation d'un paramètre discontinu est la suivante :

$$\theta(t) = \theta(t-1) + k\varnothing(t)\in(t)$$

dans laquelle q est le paramètre réglé, k le gain d'adaptation, Δ est le vecteur de données, Œ est l'écart entre la valeur actuelle et la valeur prévue. On peut tout à fait démarrer directement de cette simple équation de rafraîchissement.

Considérons les équations de tendance suivantes :

$$y(t+1) = g_1u(t) + g_2u(t-1) + f_1y(t) + f_2y(t-1)$$

Le calcul peut s'effectuer, normalement, en plusieurs pas de macro successifs, de telle sorte que la charge de calcul soit répartie plus uniformément. Le ST généré par la station de programmation pour l'exemple ci-dessus est le suivant :

```
(* MACRO : MAIN *)
INITIAL_STEP  MAIN : MAIN__ACTION(N) ; END_STEP
(*
S  Start
        T1
        |============i============i============i                    m Monitor
m Control    m  Alarm      m UsrInfce
        !============!============!========== T1

End
*)

(* MACRO : Control *)
STEP  Control : Control__ACTION(N) ; END_STEP
ACTION  Control__ACTION :
(*
C GetData
        T
        [
ResetTim
        T1
        |
NewPars
        T1
        |
CompPred
        T1
        [
> GetData
*)
(* CONTINUOUS *)
INITIAL_STEP  GetData : GetData__ACTION(N) ; END_STEP
ACTION  GetData__ACTION :

        Y_Val      := load.Main_PV ;
END_ACTION
```

```
TRANSITION
FROM GetData
TO   ResetTim
:=
Watch.Elapsed_Time >= SampTime.Val ;
END_TRANSITION

(* SINGLE SHOT *)
STEP  ResetTim : ResetTim__ACTION(P) ; END_STEP ACTION  ResetTim__ACTION :
      Watch_Reset       := 1(*On*) ;
      (* Prediction Error Equation *)
      Error_Val         := Y.Val - Yhat.Val ;
      (* Denominator of the Update Equation *)
      Denom_Val         := U1.Val * U1.Val + U2.Val * U2.Val + Y1.Val *
                                 Y1.Val + Y2.Val * Y2.Val + 1.0 ;
      (* Compute Update Gains *)
      K_Val             := SEL_REAL( G :=  ABS_REAL( IN :=  Error.Val ) <
      Error.Max_Val AND ABS_REAL( IN :=  Error.Val )                    >
Error.Min_Val , IN0 :=  0.0 , IN1 :=                                    Error.Val * Gain.Val
/ Denom.Val ) ;
END_ACTION

TRANSITION
      FROM ResetTim
      TO   NewPars
:= 1; (* NULL transition - default TRUE *)
END_TRANSITION

(* SINGLE SHOT *)
STEP  NewPars : NewPars__ACTION(P) ; END_STEP
ACTION  NewPars__ACTION :

Watch_Reset    := 0(*Off*) ;
(* Update the Parameters *)
G1_Val         := G1.Val + K.Val * U1.Val ;
G2_Val         := G2.Val + K.Val * U2.Val ;
F1_Val         := F1.Val + K.Val * Y1.Val ;
F2_Val         := F2.Val + K.Val * Y2.Val ;
```

```
(* Limit the Parameters *)
G1_Val          := MAX_REAL( IN1 := G1.Min_Val , IN2 := MIN_REAL(
                   IN1 := G1.Max_Val , IN2 := G1.Val ) ) ; G2_Val          :=
MAX_REAL( IN1 := G2.Min_Val , IN2 := MIN_REAL(
                   IN1 := G2.Max_Val , IN2 := G2.Val ) ) ; F1_Val          :=
MAX_REAL( IN1 := F1.Min_Val , IN2 := MIN_REAL(
                   IN1 := F1.Max_Val , IN2 := F1.Val ) ) ; F2_Val          :=
MAX_REAL( IN1 := F2.Min_Val , IN2 := MIN_REAL(
                   IN1 := F2.Max_Val , IN2 := F2.Val ) ) ; END_ACTION

TRANSITION
      FROM NewPars
      TO   CompPred
:= 1; (* NULL transition - default TRUE *)
END_TRANSITION


(* SINGLE SHOT *)
STEP  CompPred : CompPred__ACTION(P) ; END_STEP
ACTION  CompPred__ACTION :

 (* Compute the Control *)
      Yhat_Val          := G2.Val * U1.Val + F1.Val * Y.Val + F2.Val * Y1.Val
                           ;
      U_Val             := ( W.Val - Yhat.Val ) / G1.Val ;
      U_Val             := MIN_REAL( IN1 := MAX_REAL( IN1 := U.Val
                           , IN2 := U.Min_Val ) , IN2 := U.Max_Val ) ; (*
Store the Data *)
      Yhat_Val          := Yhat.Val + G1.Val * U.Val ;
      U2_Val            := U1.Val ;
      U1_Val            := U.Val ;
      Y2_Val            := Y1.Val ;
      Y1_Val            := Y.Val ;
END_ACTION

TRANSITION
      FROM CompPred
      TO   GetData
```

```
:= 1; (* NULL transition - default TRUE *)
END_TRANSITION

END_ACTION (* Control__ACTION *)
```

Régulations à variables multiples

La réalisation d'un système de régulation à variables multiples à partir de blocs fonctions PC3000 est une tâche relativement prédictive. Les blocs fonctions PC3000 peuvent être reliés entre eux pour constituer des systèmes de régulation complexes.

La régulation à variables multiples n'a pas besoin de se limiter à une série de régulations à boucle unique, bien que, en pratique, un grand nombre de problèmes soient résolus de la sorte. Un panachage de technique classique et de modélisation peut être pratiqué ici pour obtenir les meilleurs résultats. La tendance peut également être utilisée pour les boucles à problèmes. Il est également possible de traiter les configurations à taux multiple en utilisant le système de tâches PC3000. La capacité de communication du PC3000 permet d'intégrer des instruments discrets à la configuration d'ensemble. Ceci augmente également l'intégrité totale du système.

Quelle que soit la stratégie de réalisation finale, il peut s'avérer intéressant d'examiner le niveau d'interaction dans le process, avant de tenter de réaliser un système de régulation complexe. Ceci s'effectue, en général, en calculant la "matrice de gain relatif de Bristol". Ce calcul est décrit en détail dans la référence [6,7,8]. Le calcul des gains relatifs statique et dynamique peut s'effectuer de différentes façons. Le gain relatif de Bristol est défini comme étant approximativement :

Gain incrémentiel de la voie en supposant toutes les autres boucles ouvertes

Gain incrémentiel de la voie en supposant toutes les autres boucles fermées

Pour une définition plus précise, voir [8]. En clair, en l'absence d'interaction entre canaux, le gain relatif est de 1 : il n'y a pas de différence si d'autres mesures sont maintenues constantes ou sont autorisées à varier -- elles n'ont aucune influence sur ce canal. Ceci signifie mathématiquement que :

$$PV_j = \sum_{i=1}^{N} K_{ij} \times OP_i \,(\text{In Open Loop})$$

$$OP_j = \sum_{i=1}^{N} H_{ij} \times PV_i \,(\text{In Closed Loop})$$

$$\lambda_{ij} = \frac{K_{ij}}{H_{ji}}$$

Sans compensation de variable multiple, il y a lieu de faire attention aux effets possibles des interactions. En résumé :

Un appariement de boucles doit être effectué avec des éléments correspondant à des gains relatifs proches de l'unité.

Si le gain relatif approprié est inférieur à l'unité, les réglages du temps d'intégration et de la bande proportionnelle doivent être augmentés.

Si le gain relatif approprié est supérieur à l'unité, seul le réglage de la bande proportionnelle a besoin d'être augmenté.

Si le gain relatif approprié est négatif, alors il existe une forte probabilité de comportement en réponse inverse et l'action de la régulation peut avoir besoin d'être inversé. Il y a lieu de faire très attention, dans ce cas, car l'intégrité de la boucle risque d'être perdue.

Les canaux à gain relatif très important (en valeur absolue), présentent un risque potentiel. L'implication est que les mesures ne peuvent pas être réglées indépendamment les unes des autres. Une directionnalité significative est présente dans le process.

Une matrice de gains relatifs indique d'une manière immédiate et claire :

si la régulation est susceptible de poser des problèmes

si l'appariement de boucles est sensible.

Pour les techniques plus avancées traitant les domaines à problèmes, comme la compensation de la tendance, il peut s'avérer nécessaire d'utiliser des configurations reposant sur l'observation ou l'une ou l'autre des nombreuses méthodologies de conception de régulation. La réalisation de configurations prédictives ou modélisées a été décrite précédemment.

Nous précisons ici le cas d'une régulation simple reposant sur l'observation, pour un process à 2 entrées, 2 sorties et 3 états, prise à titre d'exemple.

On suppose que les équations de la régulation sont :

$$x(t) = Ax(t) + Bu(t) + K_f y(t)$$

$$u(t) = U_c(t) + K_c x(t)$$

x étant l'état observé, y la matrice de mesures, et u la sortie de régulation.

Les paramètres Kc sont normalement définis hors ligne au moyen d'un progiciel de CACSD (Conception de système de régulation assistée par ordinateur), et sont transférés dans le PC3000 pour leur mise en oeuvre, comme le sont les éléments des matrices d'état. Kc and Kf doivent, normalement, être définis pour apporter une caractéristique souhaitée, comme la stabilité, la robustesse et un couplage

réduit à haute fréquence, et Uc est la sortie des régulations PI de chacun des canaux pour obtenir une bonne précision à l'état stable. Si, en plus, une intégration en trapèze est utilisée, on obtient :

```
SX1.Val:= a11.Val * X1.Val + a12.Val * X2.Val + a13.Val * X3.Val
                + Dummy.Val * Old_SX1.Val + b11.Val * OP1.Val(* Feedback *)
                +b12.V11 * OP2.Val(* Feedback *) + KF11.Val * PV1.Val
                + KF12.Val * PV2.Val;
Old_SX1.Val:= SX1.Val(* Feedback *);
X1.Val := X1.Val + h.Val * (SX1.Val + Old_SX1.Val) / 2.0;
OP1.Val:= OP1_Bar.Val +  KC11.Val * X1.Val + KC12.Val * X2.Val +
      KC13.Val * X3.Val;
```

La variable Dummy (Simulation) est utilisée pour créer une boucle algébrique dans le calcul, ce qui permet de forcer un ordre par câblage, (dans la plupart des exemples, les calculs sont effectués par pas, de telle sorte que l'ordre est fixé). La variable Dummy est mise à zéro. La variable h est l'intervalle d'échantillonnage de la tâche. En clair, si la conception est faite en temps discontinu, des tâches peuvent alors être utilisées pour fixer l'intervalle d'échantillonnage. L'erreur supplémentaire et les contraintes de sécurité normalement disponibles pour la régulation sont absentes, mais peuvent être ajoutées par l'utilisateur, si nécessaire.

Le PC3000 est prévu pour la mise en oeuvre de méthodologies de système de régulation, d'une façon simple et sûre. Il est possible, par exemple, de concevoir des systèmes de régulation utilisant une configuration reposant sur l'observation, en ayant une contre-réaction PID standard, ladite PID appropriée étant en mode suivi lorsque la régulation à entrées multiples et sorties multiples (MIMO) est active. S'il arrive que la "nouvelle" stratégie ne soit pas suffisamment efficace, l'opérateur peut commuter sur la configuration "conventionnelle", dont on sait qu'elle fonctionne.

BLOC FONCTION DE RÉGULATION PID

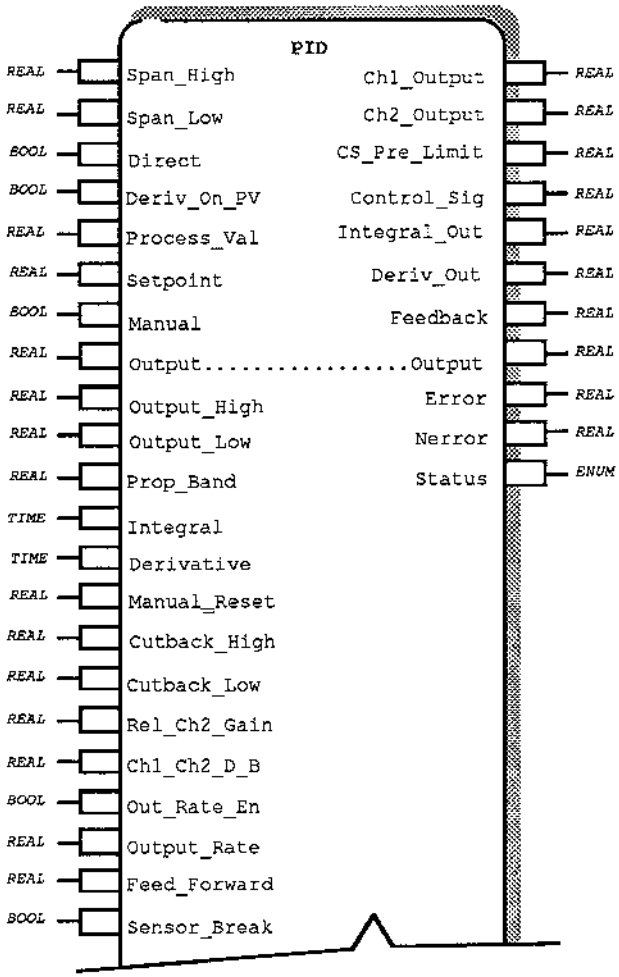


Figure 9-41 Schéma du bloc fonction PID

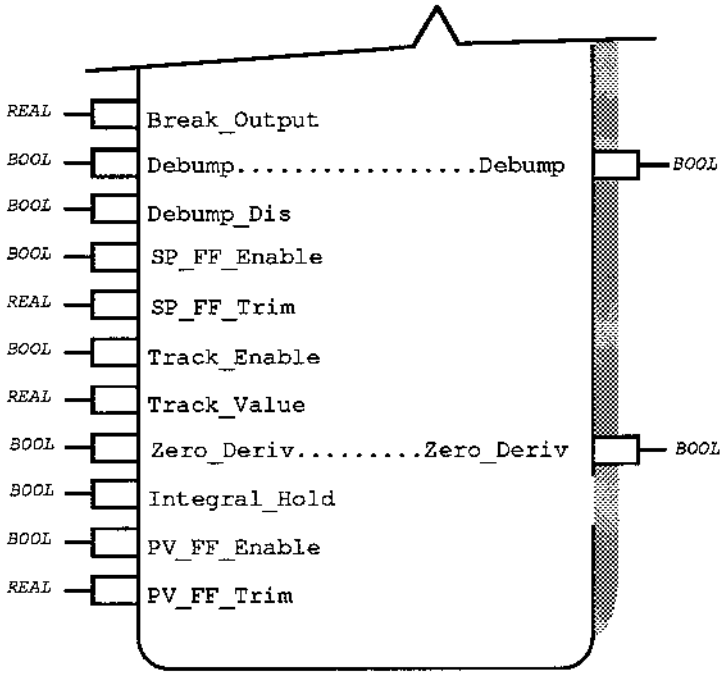


Figure 9-41 Schéma du bloc fonction PID (Suite)

Description fonctionnelle

Le bloc fonction PID met en oeuvre un algorithme de régulation proportionnel, intégral et dérivé qui est également utilisé dans les instruments de régulation Eurotherm série 900. L'algorithme de base PID peut être représenté par l'équation :

$$\text{Sortie} = \left(\frac{10000}{\text{Span} * \text{Prop\_Band}}\right)\left(E(t) + \frac{1}{\text{Integral}}\int E(t).dt + \text{Dérivée} \cdot \frac{d.E\,(t)}{dt}\right)$$

dans laquelle E (t) est donné par (Setpoint - Process_Val) et Echelle par (Span_High - Span_Low). Dans le PC3000, cet algorithme élémentaire est soutenu par une fonctionnalité complémentaire, pour améliorer les performances de régulation et pour permettre au bloc fonction d'être configuré pour réguler une grande variété de systèmes. Dans la description ci-après du fonctionnement du bloc fonction, les paramètres ont été regroupés selon une classification fonctionnelle classique en : paramètres de configuration, d'entrée dynamique, de régulation, paramètres en rapport avec la sortie et de diagnostic.

Les paramètres de configuration sont ceux qui affectent la structure de la régulation et n'ont besoin d'un réglage que lors de la configuration initiale.

Attributs du bloc fonction

Type : 20 38

Classe : .....RÉGULATION

Tâche par défaut : .....Task_2

Liste récapitulative : .....Setpoint, Process_Val, Manual, Output

Besoins de capacité mémoire :268 octets

Durée d'exécution : .....1,17 ms pour fonction. simple sortie

.....1,40 ms pour fonction. double sortie

Les entrées dynamiques sont celles qui, normalement, changent fréquemment pendant l'exécution du bloc fonction. Les paramètres en rapport avec la régulation affectent le déroulement de la boucle. Les paramètres en rapport avec la "Sortie" sont ceux qui sont directement concernés par l'étage de sortie du bloc. Les paramètres de diagnostic, enfin, fournissent des informations relatives à diverses étapes de calcul à l'intérieur du bloc.

Description des paramètres

Break_Output :	Configuration	Manual :	Régulation
Ch1	Sortie	Manual_Reset :	Régulation
Ch2	Sortie	Nerror :	Diagnostic
Control_Sig :	Diagnostic	Output :	Sortie
CS_Pre_Limit :	Diagnostic	Process_Val :	Entrée dynamique
Cutback_High :	Régulation	Prop_Band :	Régulation
Cutback_Low :	Régulation	PV_FF_Enable :	Entrée dynamique
Debump :	Entrée dynamique	PV_FF_Trim :	Entrée dynamique
Debump_Dis :	Entrée dynamique	Sensor_Break :	Entrée dynamique
Deriv_On_PV :	Configuration	Setpoint :	Entrée dynamique
Deriv_Out :	Diagnostic	Span_High :	Configuration
Derivative :	Régulation	Span_Low :	Configuration
Direct :	Configuration	SP_FF_Enable :	Entrée dynamique
Error :	Diagnostic	SP_FF_Trim :	Entrée dynamique
Feed_Forward :	Entrée dynamique	Status :	Diagnostic
Feedback :	Diagnostic	Zero_Deriv :	Régulation
Integral :	Régulation		
Integral_Hold :	Régulation		
Integral_Out :	Diagnostic		

Tableau 9-7 Classification des paramètres

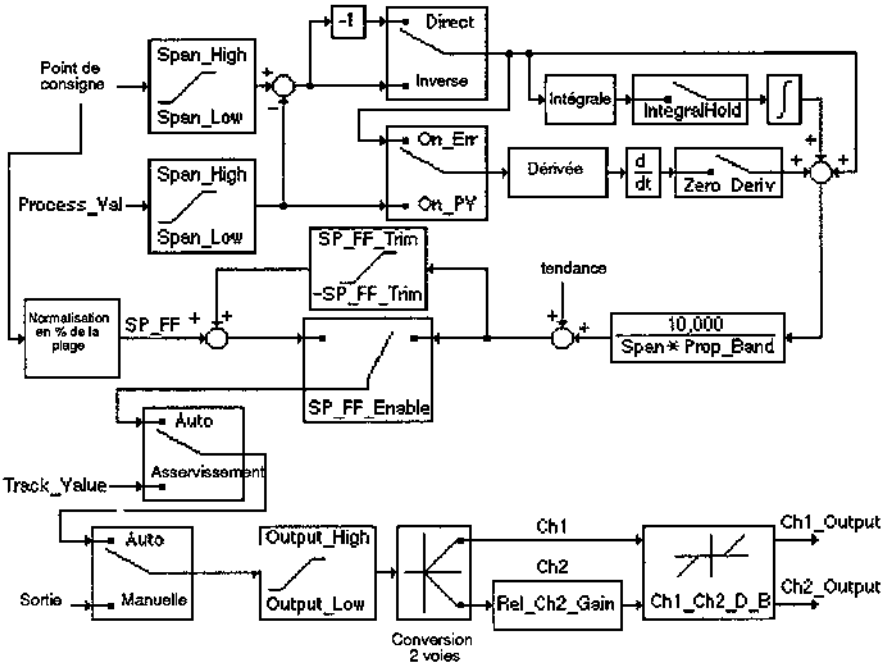


Figure 9-42 Schéma de principe du bloc fonction PID

Paramètres de configuration

Pour un fonctionnement optimal, le bloc fonction PID doit être configuré selon le type de problème de régulation auquel il doit être appliqué. Cette configuration est obtenue en réglant les paramètres d'entrée d'une façon appropriée.

Span_High et Span_Low.

Span_High et Span_Low définissent les limites maximale et minimale de la plage utile du bloc fonction. Généralement, celles-ci sont réglées sur des valeurs qui représentent des limites physiques dans le fonctionnement du process, comme la plage d'étalonnage d'un capteur ou les limites de sécurité d'un réservoir sous pression. La bande proportionnelle d'un algorithme PID est définie sous forme d'un pourcentage de la plage utile du bloc fonction, qui s'obtient en soustrayant Span_Low de Span_High. PV (mesure) est soumis aux limites de Span High +10% et Span Low -10%. Le point de consigne est soumis aux limites de Span High et de Span Low -. Si Process_Val ou le point de consigne sort de la plage utile, le bloc fonction se trouve en situation de rupture de capteur. Output prend alors la valeur définie par le paramètre Break_Output.

Direct

Direct définit si le bloc fonction est en action directe ou en action inverse. Si le bloc fonction est en action directe, la sortie va tendre à augmenter si Process_Val est supérieur au point de consigne. Si le bloc fonction est en action inverse, la sortie va tendre à augmenter si Process_Val est inférieur au point de consigne.

Deriv_On_PV

Deriv_On_PV définit si l'action dérivée doit répondre aux changements de Process_Val seulement (On_PV (1)) ou aux changements d'écart entre le point de consigne et Process_Val (On_Err (0)).

Break_Output

Break_Output définit le niveau de sortie par défaut du bloc fonction lorsqu'une rupture capteur a été détectée. Il est déclenché soit lorsque Sensor_Break est mis sur Break (1), soit lorsque Process_Val sort de la plage utile du bloc fonction de ± 10 %, soit si le point de consigne sort de la plage utile.

Paramètres d'entrée dynamique

Les entrées dynamiques du bloc fonction sont celles dont les valeurs sont censées changer d'une façon continue, comme Process_Value, et celles pour lesquelles on peut s'attendre à tout moment à un changement de valeur, soit par le programme séquentiel, soit par changement des conditions du process, comme Sensor_Break.

Setpoint (Point de consigne) et Process_Val (Mesure)

Process_Val est la variable régulée du bloc fonction et Setpoint est la valeur cible par rapport à laquelle est régulée Process_Val. L'algorithme PID agit de façon à réduire à zéro l'écart entre le point de consigne et la mesure.

Sensor_Break (Rupture de capteur)

L'entrée "sensor break" constitue un déclenchement qui peut être utilisé pour placer le bloc fonction dans un état de rupture capteur. Lorsque le bloc fonction est en rupture capteur, la sortie prend la valeur Break_Output et l'algorithme PID est désactivé. Lorsqu'il quitte l'état "sensor break", l'algorithme maintient la sortie à Break_Output pendant 16 cycles d'échantillonnage, pour s'assurer que le capteur a bien repris sa tâche d'acquisition.

Debump et Debump_Dis

La fonction Debump est utilisée par le bloc fonction pour éviter que les changements de conditions de fonctionnement ou de paramètres de régulation ne provoquent de brusques déviations du signal de sortie. Dans le PC3000, la suppression d'à-coup (debumping) s'effectue automatiquement lorsque des changements sont apportés aux paramètres Prop_Band, Derivative, Span_High, Span_Low, Ch1_Ch2_D_B, Rel_Ch2_Gain, ou lors des commutations entre les modes Manual et Auto. Le paramètre Debump peut être utilisé pour supprimer les à-coups d'autres changements, par exemple ceux du point de consigne. Debump_Dis peut être utilisé pour désactiver la fonctionnalité Debump, permettant ainsi à la sortie de faire un "saut" en réponse à la modification de l'un des paramètres indiqués ci-dessus.

SP_FF_Enable et SP_FF_Trim

SP_FF_Enable est utilisé pour activer la fonction de tendance du point de consigne, ce qui est généralement le cas lorsque le bloc fonction fait partie d'une application de régulation en cascade. SP_FF_Trim sert à limiter le niveau maximal de correction du point de consigne. SP_FF_Trim s'exprime en % de la plage utile de la sortie.

PV_FF_Enable et SP_FF_Trim

S'utilisent lorsqu'une tendance de PV est requise en cascade. C'est la même fonction que la tendance du point de consigne, mais appliquée à la mesure (PV).

Feed_Forward (Tendance)

S'ajoute directement en sortie de l'algorithme PID, avant d'effectuer la limitation de sortie et les conversions de sortie double.

Paramètres de régulation

Le bloc fonction dispose de différents paramètres qui ne sont pas prévus pour changer d'une façon dynamique, mais qui régulationnt le fonctionnement de l'algorithme. Ils incluent les paramètres de réglage, comme Prop_Band, ainsi que des paramètres booléens servant à modifier le mode de fonctionnement.

Manual (Manuel)

Manual est utilisé pour placer le bloc fonction soit en mode manuel, soit en mode automatique. En mode Manuel, l'algorithme PID est désactivé et la valeur Output (Sortie) est prise directement sur l'entrée. En mode Auto, la totalité de la fonction du bloc est active.

Prop_Band

Prop_Band est la bande proportionnelle de l'algorithme de régulation PID, exprimée en % de la plage utile. La bande proportionnelle est définie comme étant le pourcentage de la plage utile de la régulation où un écart de régulation produit un signal de sortie égal à la valeur de sortie maximale de l'instrument. Par exemple, si Span_High = 1000, Span_Low = 0, Setpoint = 500, Process_Val = 490, Prop_Band = 1 %, une régulation à action proportionnelle inverse seulement aura une sortie de +100 %, car l'erreur est de 10 unités, ce qui correspond à la bande proportionnelle.

N.B. : Le gain proportionnel est donné par :

$$\text{Gain} = \frac{10{,}000}{(\text{Span\_High} - \text{Span\_Low}) * \text{Prop\_Band}}$$

Integral (Intégrale)

Integral est la constante de temps d'intégrale de l'algorithme de régulation PID. Le temps d'intégrale est défini comme étant la période de temps sur laquelle la partie du signal de sortie due à l'action intégrale augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, pour un état d'écart constant.

Derivative (Dérivée)

Derivative est la constante de temps dérivée de l'algorithme de régulation PID. Le temps de dérivée est défini comme étant l'intervalle de temps sur lequel la partie du signal de sortie due à l'action dérivée augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, lorsque l'écart de régulation change à une vitesse constante.

Manual_Reset (Intégrale manuelle)

Manual_Reset n'est actif que si Integral est mis à zéro. Il effectue un décalage de la sortie pouvant servir à annuler l'écart de régulation lorsque l'action intégrale n'est pas utilisée.

Cutback_High et Cutback_Low

Le "Cutback" peut être utilisé pour réduire le délai de réponse de Process_Val à des changements importants de point de consigne et pour limiter le dépassement éventuel au cours de la phase transitoire. Cutback_High agit lorsque Process_Val est initialement supérieur au point de consigne recherché et Cutback_Low agit

lorsque Process_Val est initialement inférieur au point de consigne recherché. Cutback_High et Cutback_Low sont exprimés en unités physiques. Cutback agit en forçant la sortie à sa limite appropriée, maximale ou minimale, en réponse à un changement de point de consigne supérieur à la bande de cutback. Lorsque Process_Val approche du point de consigne, l'écart de régulation (Setpoint - Process_Val) devient inférieur à la valeur de cutback et le fonctionnement normal reprend.

Track_Enable et Track_Value

Track_Enable permet à l'algorithme PID d'être désactivé et au signal de sortie d'être mis directement à Track_Value. Ces paramètres sont généralement utilisés lorsque le bloc fonction fait partie d'une application de régulation en cascade.

Zero_Deriv

Zero_Deriv peut servir à forcer à zéro la sortie dérivée. Il est automatiquement effacé par le bloc fonction après un échantillonnage.

Integral_Hold

Integral_Hold permet de "geler" la sortie intégrale à sa valeur en cours. Celle-ci reste constante tant que Integral_Hold est actif.

Paramètres de sortie

Output, Ch1_Output et Ch2_Output

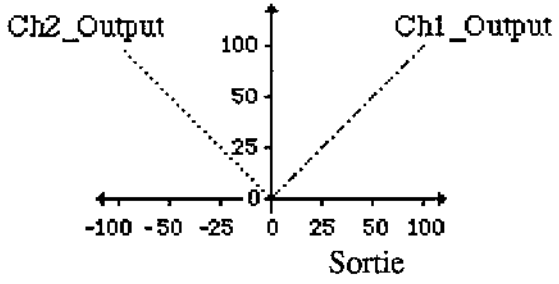


Figure 9-43 Relations entre les deux sorties, avec Rel_Ch2_Gain = 1, et Ch1_Ch2_D_B = 0

Le fonctionnement du canal du bloc fonction PID peut être configuré, au choix, en simple canal ou en double canal. En simple canal, la sortie de la régulation Output est limitée à la plage + 100 % à 0 % par les paramètres Output_High et Output_Low. Le fonctionnement en double canal est prévu pour les applications telles que les systèmes de chauffage / refroidissement, pour lesquelles des valeurs négatives doivent être disponibles en sortie pour l'unité de refroidissement, en valeur absolue pour augmenter la vitesse de refroidissement . En fonctionnement double canal, Output_Low est mis à - 100 %.

Output_High et Output_Low

Output_High et Output_Low définissent les limites supérieure et inférieure de Output. Ces paramètres doivent être situés tous deux dans la plage -100 % à +100 %, Output_High étant supérieur ou égal à Output_Low. Si le bloc fonction est utilisé en double canal, Ch1_Output et Ch2_Output sont liés à Output comme le montre la figure 9.2, Output étant limité par Output_High et Output_Low.

Rel_Ch2_Gain et Ch1_Ch2_D_B

Si le bloc fonction est utilisé pour une régulation double canal, la relation complète entre Ch2_Output et Output est donnée par :

Ch2_Output = (Output + Ch1_Ch2_D_B) * Rel_Ch2_Gain

Rel_Ch2_Gain est prévu pour les applications de régulation à double sortie non linéaire, comme les systèmes de chauffage / ventilation, pour compenser la différence de gain des équipements commandés par les deux canaux de sortie. Ch1_Ch2_D_B introduit entre les deux canaux de sortie une bande morte dont la valeur peut être soit positive, pour avoir une zone dans laquelle aucun canal n'est actif, soit négative, pour avoir une zone de recoupement dans laquelle les deux sorties sont actives.

Output_Rate et Out_Rate_En

Output_Rate peut être utilisé pour limiter la vitesse de changement de la sortie par seconde. Il est activé en réglant Out_Rate_En à Oui (1). Il y a lieu de noter que Output_Rate agit directement sur la vitesse de changement de Output, de telle sorte que la vitesse de changement de Ch2_Output est modifiée par Rel_Ch2_Gain.

Paramètres de diagnostic

{1} CS_Pre_Limit

C'est la sortie de la régulation PID après addition de Feed_Forward, mais avant la limitation de la sortie.

Control_Sig (Signal de régulation)

Control_Sig est la somme des composantes proportionnelle, intégrale, dérivée et de tendance, après limitation haute et basse et limitation de la vitesse de changement, mais avant adjonction de la bande morte et du gain relatif de double canal.

Integral_Out et Deriv_Out

Integral_Out et Deriv_Out sont respectivement les sorties des composantes intégrale et dérivée.

Feedback (Contre-réaction)

Feedback est la valeur du signal de retour au PID pour indiquer le signal réel en sortie de la régulation.

Error (Ecart)

Error est la différence entre la valeur de consigne et le point de consigne.

(Process_Val - Setpoint).

Nerror

Nerror est la sortie de la composante proportionnelle du bloc fonction.

Status (Etat)

Status fournit une indication relative à l'état de fonctionnement du bloc fonction PID. Huit états sont possibles :

Ok (0) :	Le bloc fonction fonctionne normalement
SnsrBrk (1) :	Détection de rupture d'un capteur externe
PV_High (2) :	Process_Val est supérieur à Span_High +10 %
PV_Low (3) :	Process_Val est inférieur à Span_Low -10 %
SP_High (4) :	Setpoint est supérieur à Span_High
SP_Low (5) :	Setpoint est inférieur à Span_Low
GainNeg (6) :	Prop_Band a été réglé avec une valeur négative ou Span_High a été réglé plus bas que Span_Low
GainHi (7) :	La valeur donnée à Prop_Band est trop faible ou la plage utile du bloc fonction est trop étroite, ce qui donne un gain très important.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Break_Output	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Output_High Output_Low
Ch1_Ch2_D_B	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	10 -10
Ch1_Output	**REAL**	0	Oper		Lim. haute Lim. basse	100 Le plus grand de 0 ou Output_Low
Ch2_Output	**REAL**	0	Oper		Lim. haute Lim. basse	Le plus petit de 0 ou Output_High -100
Control_Sig	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
CS_Pre_Limit	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Cutback_High	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Cutback_Low	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Debump	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Debump_Dis	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Deriv_On_PV	**BOOL**	On_Err (0)	Config	Config	Détect.	On_Err (0) On_PV (1)
Deriv_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000

Tableau 9-8 Attributs des paramètres du PID

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Derivative	TIME	50 s	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Direct	BOOL	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Error	REAL	0	Super		Lim. haute Lim. basse	100 000 -100 000
Feedback	REAL	0	Config		Lim. haute Lim. basse	100 000 -100 000
Integral	TIME	5 m	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Integral_Out	REAL	0	Config		Lim. haute Lim. basse	100 000 -100 000
Manual	BOOL	Manual (1)	Oper	Oper	Détect.	Auto (0) Manual (1)
Manual_Reset	REAL	0	Oper	Oper	Lim. haute Lim. basse	100 -100
Nerror	REAL	0	Super		Lim. haute Lim. basse	100 000 -100 000
Out_Rate_En	BOOL	Non (0)	Oper	Config	Détect.	Non (0) Oui (1)
Output	REAL	0	Oper	Oper	Lim. haute Lim. basse	Output_High Output_Low
Output_High	REAL	100	Oper	Oper	Lim. haute Lim. basse	100 Output_Low
Output_Low	REAL	0	Oper	Oper	Lim. haute Lim. basse	Output_High -100
Output_Rate	REAL	10	Oper	Oper	Lim. haute Lim. basse	10 000 0,1

Tableau 9-8 Attributs des paramètres du PID (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
Prop_Band	**REAL**	5 %	Oper	Oper	Lim. haute Lim. basse	10 000 0,1
Rel_Ch2_Gain	**REAL**	1	Oper	Oper	Lim. haute Lim. basse	10 0,1
Sensor_Break	**BOOL**	Ok (0)	Oper	Super	Détect.	Ok (0) Break (1)
Setpoint	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
SP_FF_Enable	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
SP_FF_Trim	**REAL**	0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
PV_FF_Enable	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
PV_FF_Trim	**REAL**	0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Span_High	**REAL**	100	Config	Config	Lim. haute Lim. basse	100 000 Span_Low
Span_Low	**REAL**	0	Config	Config	Lim. haute Lim. basse	Span_High -100 000
Status	**ENUM**	Ok (0)	Oper		Détect.	Ok (0) SnsrBrk (1) PV_High (2) PV_Low (3) SP_High (4) SP_Low (5) GainNeg (6) GainHi (7)
Track_Enable	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Track_Value	**REAL**	0	Super	Config	Lim. haute Lim. basse	Output_High Output_Low
Zero_Deriv	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)

Tableau 9-8 Attributs des paramètres du PID (Suite)

BLOC FONCTION VP

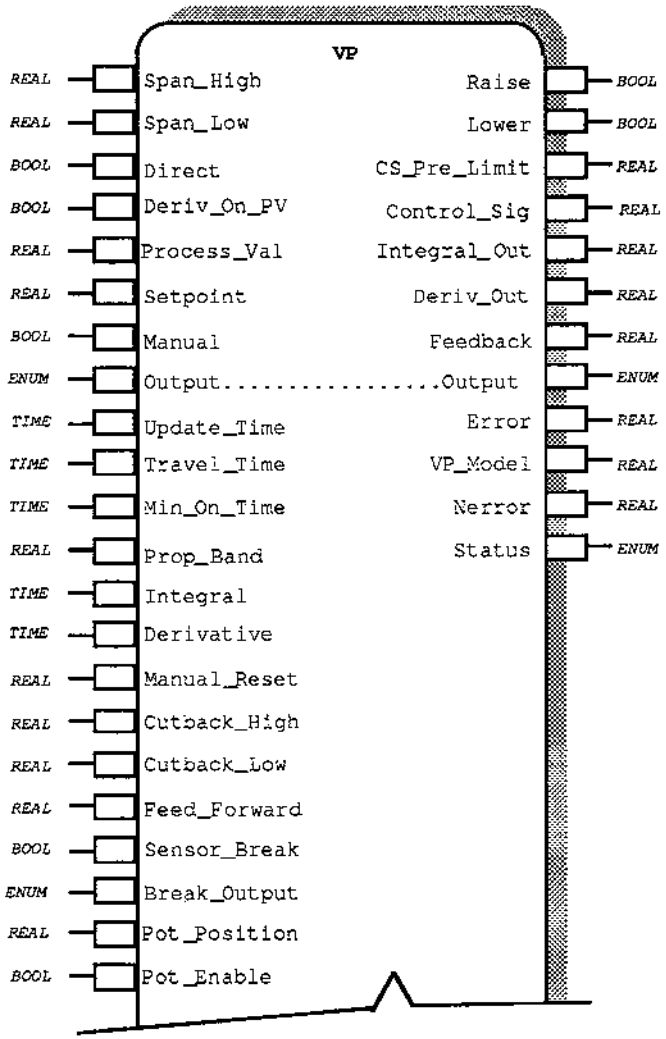


Figure 9-44 Schéma du bloc fonction VP

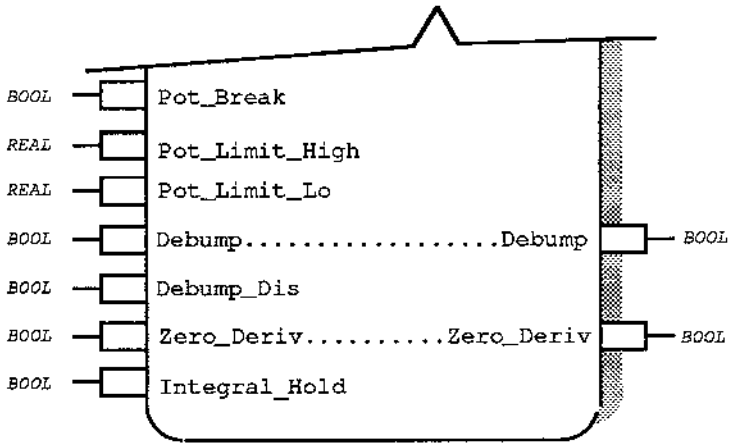


Figure 9-44 Schéma du bloc fonction VP (Suite)

Description fonctionnelle

Le bloc fonction VP est une version améliorée du bloc fonction PID qui met en oeuvre un algorithme de régulation proportionnelle, intégrale et dérivée, qui est également utilisé dans les instruments de régulation Eurotherm série 900 et il comporte, en plus, des étages de sortie permettant de réguler des boucles ayant pour actionneur des vannes motorisées. L'algorithme de base PID peut être représenté par l'équation :

$$\text{Sortie} = \left(\frac{10000}{\text{Span} * \text{Prop\_Band}}\right)\left(E(t) + \frac{1}{\text{Integral}}\int E(t).dt + \text{Dérivée} \cdot \frac{d.E\ (t)}{dt}\right)$$

dans laquelle E (t) est donné par (Setpoint - Process_Val) et Echelle par (Span_High - Span_Low). Dans le PC3000, cet algorithme élémentaire est soutenu par une fonction complémentaire pour améliorer les performances de régulation et pour permettre au bloc fonction d'être configuré pour réguler une grande variété de systèmes. Dans la description ci-après du fonctionnement et de l'utilisation du bloc fonction, les paramètres ont été regroupés selon une classification fonctionnelle classique en : paramètres de configuration, d'entrée dynamique, de régulation, paramètres en rapport avec la sortie et de diagnostic. Les paramètres de configuration sont ceux qui affectent la structure de la régulation et n'ont besoin d'un réglage que lors de la configuration initiale.

Attributs du bloc fonction

Type : 20 40

Classe: .....RÉGULATION

Tâche par défaut : .....Task_2

Liste récapitulative : .....Setpoint, Process_Val, Manual, Output

Besoins de capacité mémoire : 264 octets

Durée d'exécution : .....1,9 ms

Les entrées dynamiques sont celles qui, normalement, changent fréquemment pendant l'exécution du bloc fonction. Les paramètres en rapport avec la régulation affectent le déroulement de la boucle. Les paramètres en rapport avec la "Sortie" sont ceux qui sont directement concernés par l'étage de sortie du bloc. Les paramètres de diagnostic, enfin, fournissent des informations relatives à diverses étapes de calcul à l'intérieur du bloc.

Description de l'étage de sortie du positionneur de vanne

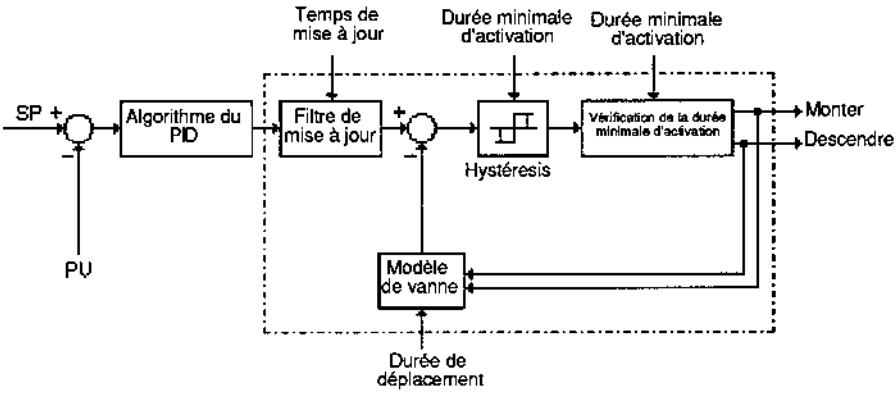


Figure 9-45 Schéma du bloc fonction positionneur de vanne

L'étage positionneur de vanne du bloc fonction utilise un algorithme dépourvu de limites, prévu pour ne pas nécessiter de signal de retour de position de la vanne, car celui-ci est souvent peu fiable. Un modèle de la vanne est prévu pour réguler la sortie positionnement de la vanne. Un potentiomètre de retour de position peut être utilisé avec le modèle de la vanne : dans ce mode de fonctionnement, la position du modèle de vanne est utilisée pour la régulation. La position réelle n'est utilisée

qu'à des fins de limitation. Ceci permet à la régulation de continuer à fonctionner en cas de défaut du potentiomètre de retour de position.

Description des paramètres

Break_Output:	Configuration	Min_On_Time	Configuration
Control_Sig:	Diagnostic	Nerror:	Diagnostic
CS_Pre_Limit:	Diagnostic	Output:	Sortie
Cutback_High:	Régulation	Pot_Break:	Retour potentiom.
Cutback_Low:	Régulation	Pot_Enable:	Retour potentiom.
Debump:	Entrée dynamique	Pot_Limit_Hi:	Retour potentiom.
Debump_Dis:	Entrée dynamique	Pot_Limit_Lo:	Retour potentiom.
Deriv_On_PV:	Configuration	Pot_Position:	Retour potentiom.
Deriv_Out:	Diagnostic	Process_Val:	Entrée dynamique
Derivative:	Régulation	Prop_Band:	Régulation
Direct:	Configuration	Raise:	Sortie
Error:	Diagnostic	Sensor_Break:	Entrée dynamique
Feed_Forward:	Entrée dynamique	Setpoint:	Entrée dynamique
Feedback:	Diagnostic	Span_High:	Configuration
Integral:	Régulation	Span_Low:	Configuration
Integral_Hold:	Régulation	Status:	Diagnostic
Integral_Out	Diagnostic	Travel_Time:	Configuration
Lower:	Sortie	Update_Time:	Configuration
Manual:	Régulation	VP_Model	Diagnostic
Manual_Reset:	Régulation	Zero_Deriv:	Régulation

Tableau 9-9 Classification des paramètres

Paramètres de configuration du bloc fonction

Pour un fonctionnement optimal, le bloc fonction VP doit être configuré selon le type de problème de régulation auquel il doit être appliqué. Cette configuration est obtenue en réglant les paramètres d'entrée d'une façon appropriée. L'utilisation de ces paramètres de configuration est décrite ci-après.

Span_High et Span_Low.

Span_High et Span_Low définissent les limites maximale et minimale de la plage utile du bloc fonction. Généralement, celles-ci sont réglées sur des valeurs qui représentent des limites physiques dans le fonctionnement du process, comme la plage d'étalonnage d'un capteur ou les limites de sécurité d'un réservoir sous pression. La bande proportionnelle de l'algorithme PID est définie sous forme d'un pourcentage de la plage utile du bloc fonction, qui s'obtient en soustrayant Span_Low de Span_High. Si Process_Val sort de la plage utile, le bloc fonction se trouve en situation de rupture capteur.

Direct

Direct définit si le bloc fonction est en action directe ou en action inverse. Si le bloc fonction est en action directe, la sortie va tendre à augmenter si Process_Val est supérieur au point de consigne. Si le bloc fonction est en action inverse, la sortie va tendre à augmenter si Process_Val est inférieur au point de consigne.

Deriv_On_PV

Deriv_On_PV définit si l'action dérivée doit répondre aux changements de Process_Val seulement (On_PV (1)) ou aux changements d'écart entre le point de consigne et Process_Val (On_Err (0)).

Update_Time

Update_Time définit le temps d'échantillonnage des étages de sortie du positionneur de vanne. Généralement, celui-ci doit être réglé au 1/10 de Travel_Time. Si Update_Time augmente, l'activité de la vanne diminue, mais cela risque de réduire la précision de la régulation. De même, en diminuant Update_Time, les performances de la régulation sont meilleures mais l'activité de la vanne est augmentée.

Travel_Time

Travel_Time est le temps nécessaire à la vanne pour se déplacer de sa position basse de fonctionnement à sa position haute de fonctionnement.

Min_On_Time

Min_On_Time définit le temps minimum pendant lequel la vanne doit maintenir son fonctionnement d'ouverture ou de fermeture ou bien rester immobile.

Break_Output

Break_Output définit le niveau de sortie par défaut du bloc fonction lorsqu'une situation de rupture capteur a été détectée. Il est déclenché soit lorsque Sensor_Break est mis sur Break (1), soit lorsque Process_Val ou le point de consigne sort de la plage utile du bloc fonction.

Paramètre de retour du potentiomètre

Pot_Position

Pot_Position est l'entrée à laquelle est raccordé le potentiomètre de retour de position de la vanne.

Pot_Enable

Si Pot_Enable est mis à Oui (1), l'algorithme de contre-réaction du potentiomètre est activé. Si Pot_Enable est mis sur Non (0), le bloc fonction fonctionne sans retour de potentiomètre.

Pot_Break

Pot_Break fournit un déclenchement externe pour indiquer que le retour de potentiomètre a été débranché ou est en défaut. Si Pot_Break est mis sur Oui (1), cela signifie qu'une situation de défaut a eu lieu. Si Pot_Break est mis sur Non (0), cela signifie que le capteur fonctionne correctement.

Pot_Limit_Hi

Pot_Limit_Hi définit la position haute du fonctionnement de la vanne.

Pot_Limit_Lo

Pot_Limit_Lo définit la position basse du fonctionnement de la vanne.

Paramètres d'entrée dynamique du bloc fonction

Setpoint (Point de consigne) et Process_Val (mesure)

Process_Val est la variable régulée du bloc fonction et Setpoint est la valeur cible par rapport à laquelle est commandée Process_Val. L'algorithme PID agit de façon à réduire à zéro l'écart entre le point de consigne et la mesure.

Sensor_Break

L'entrée "Sensor-Break" constitue un déclenchement qui peut être utilisé pour placer le bloc fonction dans un état de rupture capteur. Lorsque le bloc fonction est en rupture capteur, la sortie prend la valeur Break_Output et l'algorithme PID est désactivé. Lorsqu'il quitte l'état "Sensor-Break", l'algorithme maintient la sortie à Break_Output pendant 16 cycles d'échantillonnage, pour s'assurer que le capteur a bien repris sa tâche d'acquisition.

Debump et Debump_Dis

La fonction Debump est utilisée par le bloc fonction pour éviter que les changements de conditions de fonctionnement ou de paramètres de régulation ne provoquent de brusques déviations du signal de sortie. Dans le PC3000, la suppression d'à-coup (debumping) s'effectue automatiquement lorsque des changements sont apportés aux paramètres Prop_Band, Derivative, Span_High, Span_Low, Ch1_Ch2_D_B, Rel_Ch2_Gain, ou lors des commutations entre les modes Manual et Auto. Le paramètre Debump peut être utilisé pour supprimer les à-coups d'autres changements, par exemple ceux du point de consigne. Debump_Dis peut être utilisé pour désactiver la fonctionnalité Debump, permettant ainsi à la sortie de faire un "saut" en réponse à la modification de l'un des paramètres indiqués ci-dessus.

SP_FF_Enable et SP_FF_Trim

SP_FF_Enable est utilisé pour activer la fonction de tendance du point de consigne, ce qui est généralement le cas lorsque le bloc fonction fait partie d'une application de régulation en cascade. SP_FF_Trim sert à limiter le niveau maximal de correction du point de consigne. SP_FF_Trim s'exprime en % de la plage utile du signal de sortie du port PID.

Paramètres de régulation du bloc fonction

Le bloc fonction dispose de différents paramètres qui ne sont pas prévus pour changer d'une façon dynamique, mais qui commandent le fonctionnement de l'algorithme. Ils incluent les paramètres de réglage, comme Prop_Band, ainsi que des paramètres booléens servant à modifier le mode de fonctionnement. Ces paramètres sont décrits ci-après.

Manual (Manuel)

Utilisé pour placer le bloc fonction soit en mode manuel, soit en mode automatique. En mode Manuel, l'algorithme PID est désactivé et la valeur Output (Sortie) est prise directement sur l'entrée. En mode Auto, la totalité de la fonction du bloc est active.

Prop_Band

Prop_Band est la bande proportionnelle de l'algorithme de régulation PID, exprimée en % de la plage utile. La bande proportionnelle est définie comme étant le pourcentage de la plage utile de la régulation où un écart de régulation produit un signal de sortie égal à la valeur de sortie maximale de l'instrument. Par exemple, si Span_High = 1000, Span_Low = 0, Setpoint = 500, Process_Val = 490, Prop_Band = 1 %, une régulation à action proportionnelle inverse aura une sortie de +100 %, car l'erreur est de 10 unités, ce qui correspond à la bande proportionnelle.

N.B. : Le gain proportionnel est donné par :

$$\text{Gain} = \frac{10{,}000}{(\text{Span\_High} - \text{Span\_Low}) * \text{Prop\_Band}}$$

Integral (Intégrale)

Integral est la constante de temps d'intégrale de l'algorithme de régulation PID. Le temps d'intégrale est défini comme étant la période de temps sur laquelle la partie du signal de sortie due à l'action intégrale augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, pour un état d'écart constant.

Derivative (Dérivée)

Derivative est la constante de temps dérivée de l'algorithme de régulation PID. Le temps de dérivée est défini comme étant l'intervalle de temps sur lequel la partie du signal de sortie due à l'action dérivée augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, lorsque l'écart de régulation change à une vitesse constante.

Feed_Forward (Tendance)

Feed_Forward est ajouté directement en sortie de l'algorithme PID, avant d'effectuer la limitation de sortie et les conversions de sortie double.

Manual_Reset (Intégrale manuelle)

Manual_Reset n'est actif que si Integral est mis à zéro. Il effectue un décalage de la sortie pouvant servir à annuler l'écart de régulation lorsque l'action intégrale n'est pas utilisée.

Cutback_High et Cutback_Low

Le "Cutback" peut être utilisé pour réduire le délai de réponse de Process_Val à des changements importants de point de consigne et pour limiter le dépassement éventuel au cours de la phase transitoire. Cutback_High agit lorsque Process_Val est initialement supérieur au point de consigne recherché et Cutback_Low agit lorsque Process_Val est initialement inférieur au point de consigne recherché. Cutback_High et Cutback_Low sont exprimés en unités physiques. Cutback agit en forçant la sortie à sa limite appropriée, maximale ou minimale, en réponse à un changement de point de consigne supérieur à la bande de cutback. Lorsque Process_Val approche du point de consigne, l'écart de régulation (Setpoint - Process_Val) devient inférieur à la valeur de cutback et le fonctionnement normal reprend.

Zero_Deriv

Zero_Deriv peut servir à forcer à zéro la sortie dérivée. Il est automatiquement effacé par le bloc fonction après un échantillonnage.

Integral_Hold

Integral_Hold permet de "geler" la sortie intégrale à sa valeur en cours. Celle-ci reste constante, tant que Integral_Hold est actif.

Paramètres de sortie du bloc fonction

Output, Raise et Lower

Lorsque le bloc fonctionne en mode Manuel, Output (Sortie) sert d'entrée pour réguler directement les sorties Raise (Monter) ou Lower (Descendre) de la vanne.

Paramètres de diagnostic du bloc fonction

CS_Pre_Limit

C'est la sortie de la régulation PID après addition de Feed_Forward, mais avant la limitation de la sortie.

Control_Sig (Signal de régulation)

Control_Sig est la somme des composantes proportionnelle, intégrale, dérivée et de tendance, après limitation haute et basse et limitation de la vitesse de changement, mais avant adjonction de la bande morte et du gain relatif de double canal.

Integral_Out et Deriv_Out, Nerror

Nerror, Integral_Out et Deriv_Out sont respectivement les sorties des composantes proportionnelle, intégrale et dérivée.

Feedback (Contre-réaction)

Feedback est la valeur du signal de retour au PID pour indiquer le signal réel en sortie de la régulation.

Error (Ecart)

Error est la différence entre la valeur de consigne et le point de consigne.

VP_Model

VP_Model est la sortie du modèle interne de la vanne.

Etat

Etat fournit une indication relative à l'état de fonctionnement du bloc fonction PID. Huit états sont possibles :

Ok (0) : Le bloc fonction fonctionne normalement

SnsrBrk (1) : Détection de rupture d'un capteur externe

PV_High (2) : Process_Val est supérieur à Span_High +10 %

PV_Low (3) : Process_Val est inférieur à Span_Low -10 %

SP_High (4) : Setpoint est supérieur à Span_High

SP_Low (5) : Setpoint est inférieur à Span_Low

GainNeg (6) : Prop_Band a été réglé avec une valeur négative ou Span_High a été réglé plus bas que Span_Low

GainHi (7) : La valeur donnée à Prop_Band est trop faible ou la plage utile du bloc fonction est trop étroite, ce qui donne un gain très important.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Break_Output	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Output_High Output_Low
Control_Sig	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
CS_Pre_Limit	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Cutback_High	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Cutback_Low	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Debump	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Debump_Dis	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Deriv_On_PV	**BOOL**	On_Err (0)	Config	Config	Détect.	On_Err (0) On_PV (1)
Deriv_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Derivative	**TIME**	50 s	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Direct	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Error	**REAL**	0	Super		Lim. haute Lim. basse	100 000 -100 000
Feed_Forward	**REAL**	0	Super	Super	Lim. haute Lim. basse	100 -100
Feedback	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000

Tableau 9-10 Attributs des paramètres de VP

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Integral	**TIME**	5 m	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Integral_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
IntegralHold	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Lower	**BOOL**	Off (0)	Oper		Détect.	Off (0) Lower (1)
Manual	**BOOL**	Manual (1)	Oper	Oper	Détect.	Auto (0) Manual (1)
Manual_Reset	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	100 -100
Min_On_Time	**TIME**	100 ms	Oper	Oper	Lim. haute Lim. basse	5 s 100 ms
Nerror	**REAL**	0	Super		Lim. haute Lim. basse	100 000 -100 000
Output	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Output_High Output_Low
Pot_Break	**BOOL**	Non (0)	Oper	Super	Détect.	Non (0) Oui (1)
Pot_Enable	**BOOL**	Non (0)	Oper	Super	Détect.	Non (0) Oui (1)
Pot_Limit_Hi	**REAL**	100	Oper	Super	Lim. haute Lim. basse	100 0
Pot_Limit_Lo	**REAL**	0	Oper	Super	Lim. haute Lim. basse	100 0
Pot_Position	**REAL**	0	Oper	Super	Lim. haute Lim. basse	100 0

Tableau 9-10 Attributs des paramètres de VP (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Prop_Band	**REAL**	5 %	Oper	Oper	Lim. haute Lim. basse	10 000 0,1
Raise	**BOOL**	Off (0)	Oper		Détect.	Off (0) Raise (1)
Sensor_Break	**BOOL**	Ok (0)	Oper	Super	Détect.	Ok (0) Break (1)
Setpoint	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
Span_High	**REAL**	100	Config	Config	Lim. haute Lim. basse	100 000 Span_Low
Span_Low	**REAL**	0	Config	Config	Lim. haute Lim. basse	Span_High -100,000
Status	**ENUM**	Ok (0)	Oper		Détect.	Ok (0) SnsrBrk (1) PV_High (2) PV_Low (3) SP_High (4) SP_Low (5) GainNeg (6) GainHi (7)
Travel_Time	**TIME**	20 s	Oper	Oper	Lim. haute Lim. basse	16 m_40 s 5 s
Update_Time	**TIME**	1 s	Oper	Oper	Lim. haute Lim. basse	20 s 100 ms
VP_Model	**REAL**	50	Config		Lim. haute Lim. basse	100 0
Zero_Deriv	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)

Tableau 9-10 Attributs des paramètres de VP (Suite)

BLOC FONCTION PID_AUTO

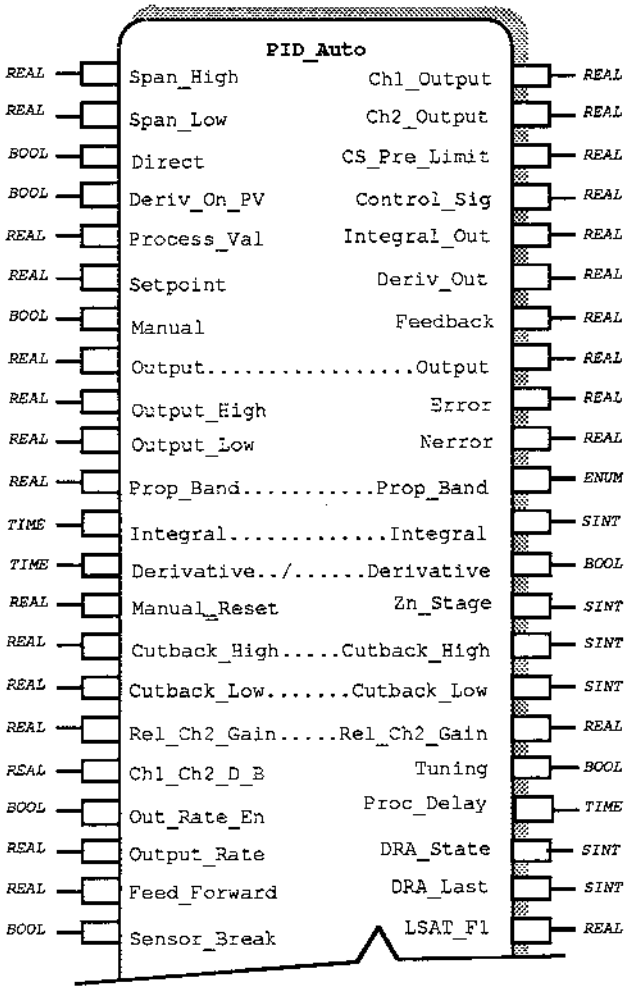


Figure 9-46 Schéma du bloc fonction PID_Auto

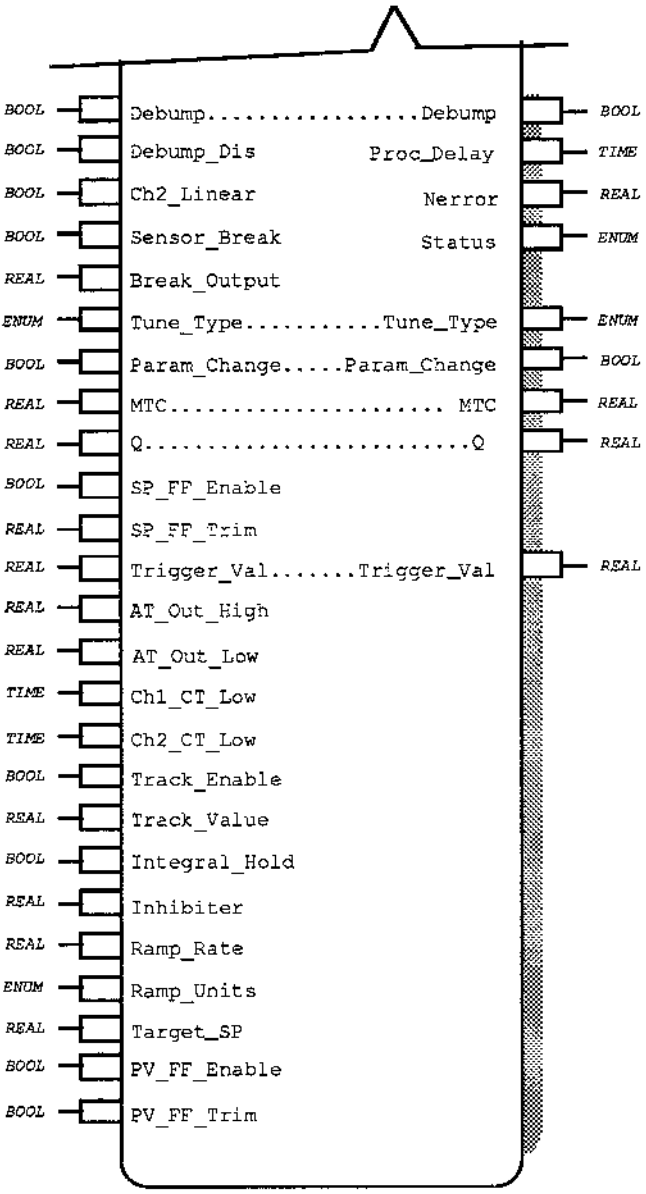


Figure 9-46 Schéma du bloc fonction PID_Auto (Suite)

Description fonctionnelle

Le bloc fonction PID_Auto met en oeuvre un algorithme de régulation proportionnelle, intégrale et dérivée, ainsi que les algorithmes de réglage Auto et Adaptive qui sont également utilisés dans les instruments de régulation Eurotherm série 900. C'est un bloc fonction complexe qui comporte trois algorithmes de réglage pouvant servir à régler les paramètres de la régulation PID, plus un algorithme d'inhibition de dépassement à utiliser lorsque le bloc fonction suit des profils de rampe de point de consigne.

L'algorithme de régulation utilisé en PID_Auto est identique à celui qu'utilise le bloc fonction PID. L'algorithme de base PID peut être représenté par l'équation :

$$\text{Sortie} = \left(\frac{10000}{\text{Span} * \text{Prop\_Band}}\right)\left(E(t) + \frac{1}{\text{Integral}}\int E(t).dt + \text{Dérivée} \cdot \frac{d.E\,(t)}{dt}\right)$$

dans laquelle E (t) est donné par (Setpoint - Process_Val) et Echelle par (Span_High - Span_Low). Dans le PC3000, cet algorithme élémentaire est soutenu par une fonction complémentaire pour améliorer les performances de régulation et pour permettre au bloc fonction d'être configuré pour réguler une grande variété de systèmes.

L'obtention de performances optimales avec ce bloc implique que la régulation et le régleur soient correctement configurés et activés. La description des différents éléments qui composent le bloc fonction est donnée ci-après.

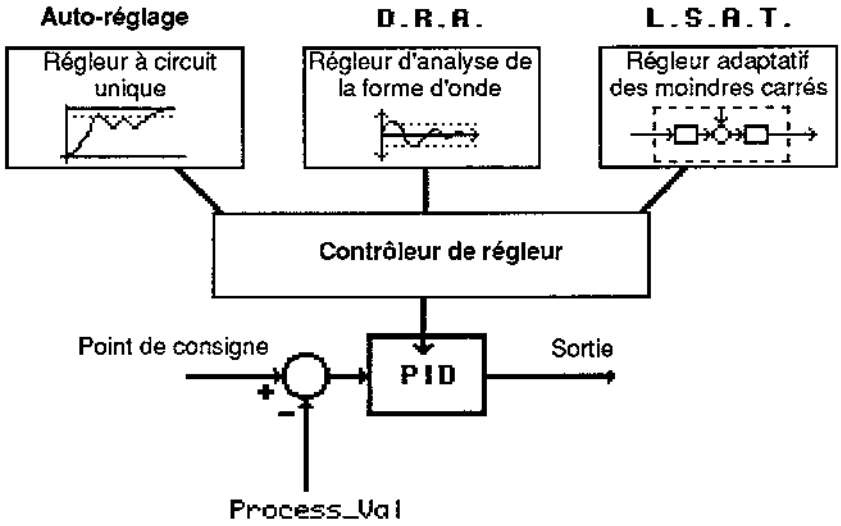


Figure 9-47 Schéma des principales fonctions de PID_Auto

Attributs du bloc fonction

Type : 20 50
Classe : .....RÉGULATION
Tâche par défaut : .....Task_2
Liste récapitulative : .....Setpoint, Process_Val, Manual, Output
Besoins de capacité mémoire : 2030 octets
Durée d'exécution : .....1,63 ms en l'absence de réglage
3,61ms maxi en cas de réglage

Description des paramètres

AT_Out_High:	Réglage	Manual_Reset:	Régulation
AT_Out_Low:	Réglage	MTC:	Réglage
Break_Output:	Configuration	Nerror:	Diagnostic
Ch1_Ch2_D_B:	Sortie	Out_Rate_En:	Sortie
Ch1_CT_Low:	Réglage	Output:	Sortie
Ch1_Cycle_T:	Réglage	Output_High:	Sortie
Ch1_Output:	Sortie	Output_Low:	Sortie
Ch2_CT_Low:	Réglage	Output_Rate:	Sortie
Ch2_Cycle_T:	Réglage	Param_Change:	Réglage
Ch2_Linear:	Réglage	Process_Val:	Entrée dynamique
Ch2_Output:	Sortie	Prop_Band	Régulation
Control_Sig:	Diagnostic	PV_FF_Enable	Entrée dynamique
CS_Pre_Limit:	Diagnostic	PV_FF_Trim	Entrée dynamique
Cutback_High:	Régulation	Q:	Réglage
Cutback_Low:	Régulation	Ramp_Rate:	Régulation
Debump:	Entrée dynamique	Rel_Ch2_Gain:	Sortie
Debump_Dis:	Entrée dynamique	Sensor_Break:	Entrée dynamique
Deriv_On_PV:	Configuration	Setpoint:	Entrée dynamique
Derivative:	Régulation	SP_FF_Enable:	Entrée dynamique
Direct:	Configuration	SP_FF_Trim:	Entrée dynamique
Error:	Diagnostic	Span_High:	Configuration
Feed_Forward:	Entrée dynamique	Span_Low:	Configuration
Feedback:	Diagnostic	Status:	Diagnostic
Inhibiter:	Régulation	Track_Enable:	Régulation
Integral:	Régulation	Track_Value:	Régulation
Integral_Hold:	Régulation	Trigger_Val:	Réglage
Manual:	Régulation	Tune_Type:	Réglage

Tableau 9-11 Classification des paramètres

Paramètres de configuration du bloc fonction

Pour un fonctionnement optimal, le bloc fonction doit être configuré selon le type de problème de régulation auquel il doit être appliqué. Cette configuration est obtenue en réglant les paramètres d'entrée d'une façon appropriée. L'utilisation de ces paramètres de configuration est décrite ci-après.

Span_High et Span_Low.

Span_High et Span_Low définissent les limites maximale et minimale de la plage utile du bloc fonction. Généralement, celles-ci sont réglées sur des valeurs qui représentent des limites physiques dans le fonctionnement du process, comme la plage d'étalonnage d'un capteur ou les limites de sécurité d'un réservoir sous pression. La bande proportionnelle de l'algorithme PID est définie sous forme d'un pourcentage de la plage utile du bloc fonction, qui s'obtient en soustrayant Span_Low de Span_High. Si Process_Val ou Setpoint sort de la plage utile, le bloc fonction se trouve en situation de rupture capteur.

Direct

Direct définit si le bloc fonction est en action directe ou en action inverse. Si le bloc fonction est en action directe, la sortie va tendre à augmenter si Process_Val est supérieur au point de consigne. Si le bloc fonction est en action inverse, la sortie va tendre à augmenter si Process_Val est inférieur au point de consigne.

Deriv_On_PV

Deriv_On_PV définit si l'action dérivée doit répondre aux changements de Process_Val seulement (On_PV (1)), ou aux changements d'écart entre le point de consigne et Process_Val (On_Err (0)).

Break_Output

Break_Output définit le niveau de sortie par défaut du bloc fonction lorsqu'une situation de rupture capteur a été détectée. Il est déclenché soit lorsque Sensor_Break est mis sur Break (1), soit lorsque Process_Val sort de la plage utile du bloc fonction de ± 10 %, soit si le point de consigne sort de la plage utile.

Paramètres de régulation

Le bloc fonction dispose de différents paramètres qui ne sont pas prévus pour changer d'une façon dynamique, mais qui commandent le fonctionnement de l'algorithme. Ils incluent les paramètres de réglage, comme Prop_Band, ainsi que des paramètres booléens servant à modifier le mode de fonctionnement. Ces paramètres sont décrits ci-après.

Manual (Manuel)

Manual est utilisé pour placer le bloc fonction soit en mode manuel, soit en mode automatique. En mode Manuel, l'algorithme PID est désactivé et la valeur Output (Sortie) est prise directement sur l'entrée. En mode Auto, la totalité de la fonction du bloc est active.

Prop_Band

Prop_Band est la bande proportionnelle de l'algorithme de régulation PID, exprimée en % de la plage utile. La bande proportionnelle est définie comme étant le pourcentage de la plage utile de la régulation où un écart de régulation produit un signal de sortie égal à la valeur de sortie maximale de l'instrument. Par exemple, si Span_High = 1000, Span_Low = 0, Setpoint = 500, Process_Val = 490, Prop_Band = 1 %, une régulation à action proportionnelle inverse aura une sortie de +100 %, car l'erreur est de 10 unités, ce qui correspond à la bande proportionnelle.

N.B. : le gain proportionnel est donné par :

$$\text{Gain} = \frac{10{,}000}{(\text{Span\_High} - \text{Span\_Low}) * \text{Prop\_Band}}$$

Integral (Intégrale)

Integral est la constante de temps d'intégrale de l'algorithme de régulation PID. Le temps d'intégrale est défini comme étant la période de temps sur laquelle la partie du signal de sortie due à l'action intégrale augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, pour un état d'écart constant.

Derivative (Dérivée)

Derivative est la constante de temps dérivée de l'algorithme de régulation PID. Le temps dérivée est défini comme étant l'intervalle de temps sur lequel la partie du signal de sortie due à l'action dérivée augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, lorsque l'écart de régulation change à une vitesse constante.

Feed_Forward (Tendance)

Feed_Forward est ajouté directement en sortie de l'algorithme PID, avant d'effectuer la limitation de sortie et les conversions de sortie double.

Manual_Reset (Intégrale manuelle)

Manual_Reset n'est actif que si Integral est mis à zéro. Il effectue un décalage de la sortie pouvant servir à annuler l'écart de régulation lorsque l'action intégrale n'est pas utilisée.

Cutback_High et Cutback_Low

Le "Cutback" peut être utilisé pour réduire le délai de réponse de Process_Val à des changements importants de point de consigne et pour limiter le dépassement éventuel au cours de la phase transitoire. Cutback_High agit lorsque Process_Val est initialement supérieur au point de consigne recherché et Cutback_Low agit lorsque Process_Val est initialement inférieur au point de consigne recherché. Cutback_High et Cutback_Low sont exprimés en unités physiques. Cutback agit en forçant la sortie à sa limite appropriée, maximale ou minimale, en réponse à un changement de point de consigne supérieur à la bande de cutback. Lorsque Process_Val approche du point de consigne, l'écart de régulation (Setpoint - Process_Val) devient inférieur à la valeur de cutback et le fonctionnement normal reprend.

Track_Enable et Track_Value

Track_Enable permet à l'algorithme PID d'être désactivé et au signal de sortie d'être mis directement sur Track_Value. Ces paramètres sont généralement utilisés lorsque le bloc fonction fait partie d'une application de régulation en cascade.

Integral_Hold

Integral_Hold permet de "geler" la sortie intégrale à sa valeur en cours. Celle-ci reste constante, tant que Integral_Hold est actif.

Paramètres d'entrée dynamique et de sortie

Setpoint (Point de consigne) et Process_Val (Mesure)

Process_Val est la variable régulée du bloc fonction et Setpoint est la valeur cible par rapport à laquelle est régulée Process_Val. L'algorithme PID agit de façon à réduire à zéro l'écart entre le point de consigne et la mesure.

Zero_Deriv

Zero_Deriv peut servir à forcer à zéro la sortie dérivée. Il est automatiquement effacé par le bloc fonction après un échantillonnage.

Integral_Hold

Integral_Hold permet de "geler" la sortie intégrale à sa valeur en cours. Celle-ci reste constante tant que Integral_Hold est actif.

Paramètres de sortie

Output, Ch1_Output et Ch2_Output

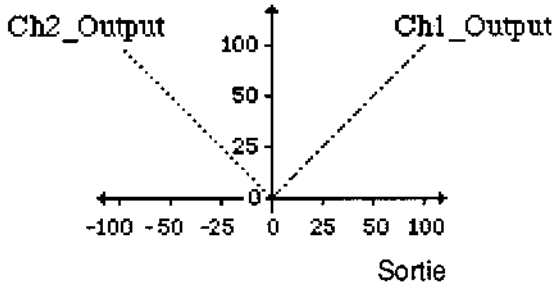


Figure 9-48 Relations entre les deux sorties, avec Rel_Ch2_Gain = 1, et Ch1_Ch2_D_B = 0

Le fonctionnement du canal du bloc fonction PID peut être configuré, au choix, en simple canal ou en double canal. En simple canal, la sortie de la régulation Output est limitée à la plage + 100 % à 0 % par les paramètres Output_High et Output_Low. Le fonctionnement en double canal est prévu pour les applications telles que les systèmes de chauffage / refroidissement, pour lesquelles des valeurs négatives doivent être disponibles en sortie pour l'unité de refroidissement, en valeur absolue pour augmenter la vitesse de refroidissement. En fonctionnement double canal, Output_Low est mis à - 100 %.

Output_High et Output_Low

Output_High et Output_Low définissent les limites supérieure et inférieure de Output. Ces paramètres doivent être situés tous deux dans la plage -100 % à +100 %, Output_High étant supérieur ou égal à Output_Low. Si le bloc fonction est utilisé en double canal, Ch1_Output et Ch2_Output sont liés à Output comme le montre la figure 9.2, Output étant limité par Output_High et Output_Low.

Rel_Ch2_Gain et Ch1_Ch2_D_B

Si le bloc fonction est utilisé pour une régulation double canal, la relation complète entre Ch2_Output et Output est donnée par :

Ch2_Output = (Output + Ch1_Ch2_D_B) * Rel_Ch2_Gain

Rel_Ch2_Gain est prévu pour les applications de régulation à double sortie non linéaire, comme les systèmes de chauffage / ventilation, pour compenser la différence de gain des équipements commandés par les deux canaux de sortie. Ch1_Ch2_D_B introduit entre les deux canaux de sortie une bande morte dont la valeur peut être soit positive, pour avoir une zone dans laquelle aucun canal n'est actif, soit négative, pour avoir une zone de recoupement dans laquelle les deux sorties sont actives.

Output_Rate et Out_Rate_En

Output_Rate peut être utilisé pour limiter la vitesse de changement de la sortie par seconde. Il est activé en réglant Out_Rate_En sur Oui (1). Il y a lieu de noter que Output_Rate agit directement sur la vitesse de changement de Output, de telle sorte que la vitesse de changement de Ch2_Output est modifiée par Rel_Ch2_Gain.

Ch1_Cycle_T et Ch2_Cycle_T

Ch1_Cycle_T et Ch2_Cycle_T sont les temps de cycle des canaux 1 et 2 qui peuvent être utilisés lorsque le bloc fonction est relié à des sorties proportionnelles modulées. Les valeurs des deux temps de cycle sont automatiquement réglées par l'auto-réglant en fin de séquence de réglage, uniquement si leur valeur initiale est supérieure à 5 s.

Ch2_Linear

Ch2_Linear est un paramètre booléen qui indique aux algorithmes de réglage et de régulation que la sortie sur le canal 2 est linéaire. Si Ch2_Linear est sur Oui (1), les séquences de réglage fonctionneront avec un niveau de sortie maximal identique sur les deux canaux de sortie. Si Ch2_Linear est sur Non (0), la sortie sur le canal 2 sera limitée à 20 % et le système de réglage fixera les paramètres de régulation pour se conformer à la relation avec les sorties d'eau de refroidissement.

Ch1_CT_Low et Ch2_CT_Low

Ch1_CT_Low et Ch2_CT_Low définissent les limites inférieures des temps de cycle des canaux de sortie 1 et 2 qui peuvent être fixées par l'auto-réglant, lorsque le bloc fonction est utilisé pour réguler des sorties proportionnelles modulées.

Sensor_Break (Rupture de capteur)

L'entrée "sensor break" constitue un déclenchement qui peut être utilisé pour placer le bloc fonction dans un état de rupture capteur. Lorsque le bloc fonction est en rupture capteur, la sortie prend la valeur Break_Output et l'algorithme PID est désactivé. Lorsqu'il quitte l'état "sensor break", l'algorithme maintient la sortie à Break_Output pendant 16 cycles d'échantillonnage, pour s'assurer que le capteur a bien repris sa tâche d'acquisition.

Debump et Debump_Dis

La fonction Debump est utilisée par le bloc fonction pour éviter que les changements de conditions de fonctionnement ou de paramètres de régulation ne provoquent de brusques déviations du signal de sortie. Dans le PC3000, la suppression d'à-coup (debumping) s'effectue automatiquement lorsque des changements sont apportés aux paramètres Prop_Band, Derivative, Span_High, Span_Low, Ch1_Ch2_D_B, Rel_Ch2_Gain, ou lors des commutations entre les modes Manual et Auto. Le paramètre Debump peut être utilisé pour supprimer les à-coups d'autres changements, par exemple ceux du point de consigne. Debump_Dis peut être utilisé pour désactiver la fonction Debump, permettant ainsi à la sortie de faire un "saut" en réponse à la modification de l'un des paramètres indiqués ci-dessus.

SP_FF_Enable et SP_FF_Trim

SP_FF_Enable est utilisé pour activer la fonction de tendance du point de consigne, ce qui est généralement le cas lorsque le bloc fonction fait partie d'une application de régulation en cascade. SP_FF_Trim sert à limiter le niveau maximal de correction du point de consigne. SP_FF_Trim s'exprime en % de la plage utile de la sortie.

PV_FF_Enable et PV_FF_Trim

S'utilisent lorsqu'une tendance de PV est requise en cascade. C'est la même fonction que la tendance du point de consigne, mais appliquée à la mesure (PV).

Les algorithmes de réglage PID_Auto

L'algorithme Autotune (auto-réglage)

L'algorithme d'auto-réglage est un régleur non répétitif qui avait été prévu, à l'origine, pour les instruments Eurotherm 818 et 815. Le principe de fonctionnement de ce régleur est décrit en détail dans la Vue d'ensemble de la régulation PC3000. La description qui en est faite ici ne reprend que les généralités permettant à l'utilisateur de comprendre son rôle dans PID_Auto.

Lorsque l'algorithme d'auto-réglage est activé, l'algorithme de régulation PID est déconnecté du process et l'auto-réglant pilote directement les sorties de la régulation au moyen d'ordres Marche / Arrêt. Au moment où l'auto-réglage est activé, si la mesure en cours ne s'écarte pas de plus de 1 % du point de consigne, les sorties sont gelées à leur valeur réelle; dans le cas contraire, les sorties sont forcées à zéro. Cette valeur est maintenue pendant une minute, durant laquelle l'auto-réglant surveille le process pour déterminer le niveau de bruit et l'action de la mesure. Pendant cette première minute, l'opérateur peut choisir le point de consigne pour lequel l'auto-réglage doit être effectué. Ce peut être le point de consigne en cours, si nécessaire, ou bien un point de consigne supérieur ou inférieur à la valeur en cours.

Lorsque la phase de surveillance d'une minute est écoulée, le cycle de l'auto-réglant passe à sa phase active. Si le point de consigne en cours est supérieur à la mesure réelle de plus de 1 % de la plage utile, un réglage en montant est effectué. Si le point de consigne en cours est inférieur à la mesure réelle de plus de 1 % de la plage utile, un réglage en descendant est effectué. Si le point de consigne en cours ne s'écarte pas de la mesure de plus de 1 % de la plage utile, un réglage sur le point de consigne est effectué. Ceci est décrit en détail dans le document cité ci-dessus. La mesure, ainsi que les courbes de sortie des trois types de cycle d'auto-réglage, dans le cas d'un chauffage seul, sont représentées sur la figure 9-49.

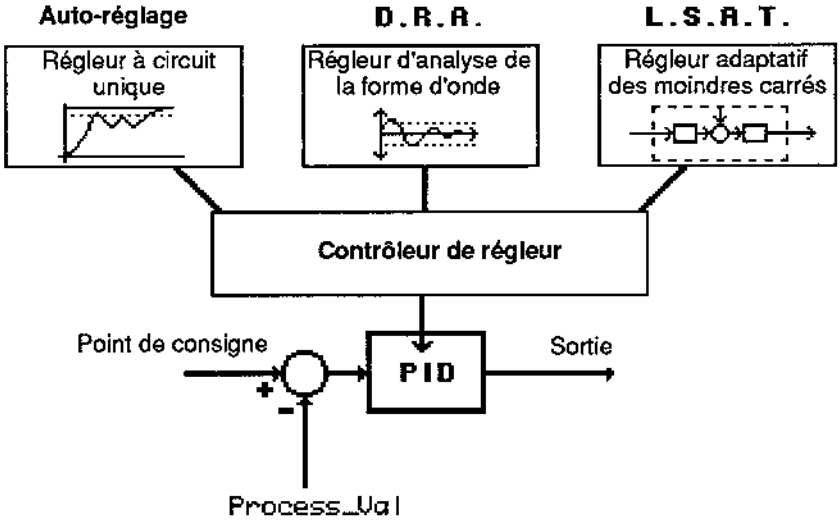


Figure 9-49 Cycles d'auto-réglage en chauffage seul

A la fin du cycle d'auto-réglage, les paramètres suivants sont calculés et chargés dans la régulation :

Bande proportionnelle	
Temps d'intégrale	
Temps de dérivée	
Cutback haut	(réglage en descendant au point de consigne uniquement)
Cutback bas	(réglage en montant au point de consigne uniquement)
Temps de cycle Ch1	(si sortie proportionnelle modulée)
Temps de cycle Ch2	(si sortie proportionnelle modulée et sélection de canal de refroidissement)
Gain réel Ch2	(si sélection de canal de refroidissement)
Trigger Val	(utilisé par DRA et LSAT)
MTC	(utilisé par LSAT)
Q	(utilisé par LSAT)

Algorithme d'analyse de réponse à une pertubation (DRA)

Le DRA est un régleur adaptatif, prévu initialement pour l'instrument Eurotherm 818. C'est un système expert élémentaire, qui agit en identifiant des modèles dans la réponse d'écart (point de consigne - mesure) et ajuste les paramètres de la régulation pour compenser les réponses qui sont trop lentes ou qui oscillent. Contrairement à l'auto-réglant, DRA ne pilote pas directement les sorties de la régulation.

Le DRA est déclenché pour surveiller la réponse de la régulation lorsque l'écart absolu dépasse Trigger_Value, qui peut être réglé manuellement, mais qui est fixé automatiquement par l'auto-réglant. Une fois déclenché par une perturbation, comme un changement de point de consigne ou une variation de la charge, le DRA enregistre jusqu'à deux cycles d'oscillation de l'écart, avant de décider quelle modification éventuelle des termes de régulation est nécessaire. Le rapport des maxima de l'écart (taux d'amortissement) est le principal critère utilisé pour décider si un nouveau réglage est nécessaire. Si l'amplitude des maxima est faible, l'auto-réglant se désactive lui-même et ne refait pas de réglage des paramètres de la régulation.

Dans la version 818 du DRA, si l'algorithme détermine qu'un nouveau réglage est nécessaire, la bande proportionnelle ainsi que les temps d'intégrale et de dérivée sont ajustés. Dans la mise en oeuvre du bloc PID_Auto, DRA ajuste également les termes MTC et Q qui référencent le LSAT. MTC est ajusté en série avec les temps d'intégrale et de dérivée, Q est ajusté avec la bande proportionnelle.

Si l'analyse de la réponse montre des oscillations, le temps d'oscillation est utilisé pour déterminer s'il est nécessaire d'ajuster le gain, les temps d'intégrale et de dérivée, ou bien à la fois le gain et les temps.

Choix de la stratégie de réglage adaptatif DRA

Algorithme du régleur adaptatif des moindres carrés (LSAT)

L'algorithme LSAT est un régleur adaptatif qui, comme l'algorithme de DRA, ne pilote pas directement les sorties de la régulation mais règle les paramètres de régulation PID. Le LSAT doit être considéré comme étant constitué de deux éléments principaux :

{1} Un identifieur de modèle de process

{2} Un configurateur de régulation.

L'algorithme surveille la sortie de la régulation et la mesure. Il envoie en permanence la sortie de la régulation dans un modèle interne du process et compare la mesure réelle à la mesure prédite par ce modèle.

L'algorithme récurrent des moindres carrés est utilisé pour ajuster les paramètres du modèle, de telle sorte que la mesure prévue corresponde à la mesure réelle. Ainsi, le modèle est réglé en continu avec précision de telle sorte qu'il corresponde au process réel commandé. Le modèle est ensuite utilisé par le configurateur de régulation pour déterminer et régler les paramètres PID optimisés pour le process régulé. C'est une approche modèle / référence, dans laquelle les performances en boucle fermée du système sont optimisées par rapport à celles du modèle de référence interne en boucle fermée. Le schéma fonctionnel de l'algorithme du LSAT est représenté ci-dessous.

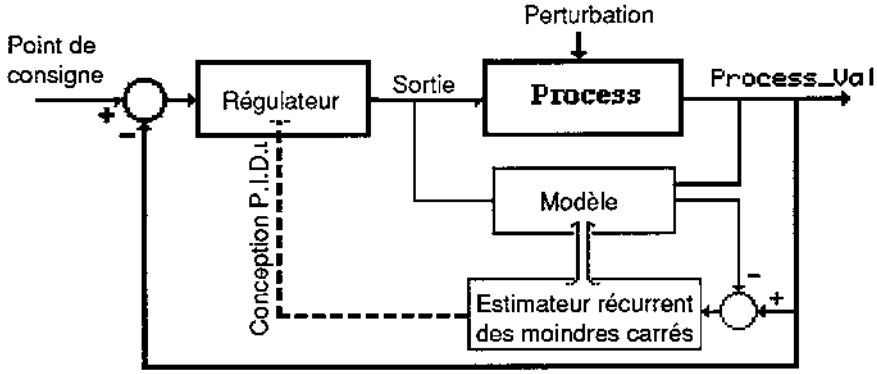


Figure 9-50 Schéma fonctionnel du LSAT

Le LSAT présente l'avantage, contrairement au DRA, de ne pas nécessiter de fortes perturbations de process ou de changements importants du point de consigne pour pouvoir agir. Il peut effectuer un réglage fin de la régulation en cours de régulation, puisqu'il est en mesure de discriminer les composantes du signal de bruit du process et de les utiliser pour l'identification du modèle.

L'action de réglage fin du LSAT est une caractéristique importante de la mise en oeuvre du système de réglage adaptatif PID_Auto. Il permet à la fois de compléter l'action de réglage approché du DRA et de réaliser le réglage fin des paramètres réglés par l'auto-réglant

La procédure d'identification du modèle par tendance et de configuration par référence au modèle permet au LSAT un réglage en continu des paramètres de la régulation au cours du fonctionnement de la régulation en ligne. Toutefois, l'algorithme du LSAT n'est pas en mesure de déterminer la structure du modèle du process : il fixe les paramètres d'un modèle dont la structure a été définie au préalable. Cette restriction peut limiter sa possibilité

de mise en oeuvre en tant que régleur adaptatif autonome d'usage général, c'est pourquoi l'algorithme a été étendu pour adapter son identification et la détermination des paramètres, en réponse à des références externes. Cela se présente sous la forme de deux paramètres : "MTC", mise à l'échelle de la constante de temps process, qui détermine la fréquence d'échantillonnage de LSAT et "Q", qui fournit une indication des paramètres non modélisés (haute fréquence) du process. Bien que cette information puisse être entrée manuellement, elle nécessite un niveau élevé de savoir-faire et de connaissance du process, c'est pourquoi les algorithmes Autotune et DRA ont été étendus pour identifier MTC et Q et pour initialiser le LSAT avec ces valeurs. L'interaction résultante de ces trois auto-réglants constitue le système de réglage "Polyvalent".

Fonctionnement des régleurs PID_Auto

Le système de réglage polyvalent associe les fonctionnalités des trois auto-réglants décrits précédemment : Autotune, DRA et LSAT. Il a été conçu pour permettre d'utiliser les possibilités des trois algorithmes pour fournir les performances optimales de réglage, dans les conditions d'utilisation les plus variées de la régulation. Les fonctions assurées par les trois algorithmes de réglage du système de réglage polyvalent peuvent être résumées comme suit :

{i} Autotune : réglage initial non répétitif des paramètres PID, identification approximative du modèle du process, identification des niveaux de bruit du process.

{ii} DRA : adaptation approchée aux changements de condition de fonctionnement du process, ajustement approximatif du modèle de référence du LSAT, ajustement éventuel de la régulation aux perturbations importantes du process.

{iii} LSAT : réglage fin des paramètres PID après la séquence d'auto-réglage et pendant le réglage adaptatif, réglage fin lors des rampes de consigne et en réponse à de faibles modifications du process ou à de faibles pertubations.

La validation de l'auto-réglant a pour effet d'activer la séquence du régleur non répétitif. La validation du réglage adaptatif active à la fois les deux algorithmes de réglage adaptatif DRA et LSAT. Il y a lieu de noter que **les performances optimales de réglage sont obtenues quand AUTO/TUNE précède ADAPT/TUNE.** L'algorithme Autotune a été amélioré pour fournir l'initialisation que nécessite le LSAT. Si Autotune ne précède pas Adaptive, le DRA est alors nécessaire pour initialiser le LSAT, ce qui peut nécessiter plusieurs séquences d'adaption de DRA et peut entraîner en un réglage très lent de la régulation.

Si Autotune suivi par Adaptive est choisi, une séquence Autotune est effectuée, suivie d'une commutation automatique sur le réglage par Adaptive. La séquence de réglage par Adaptive commence par inhiber le LSAT jusqu'à ce que l'estimateur ait convergé sur le modèle du process. Une fois cette convergence sur le LSAT effectuée, la séquence Adaptive Tune se poursuit avec le réglage de la régulation, par les algorithmes, à la fois de DRA et de LSAT, sous le contrôle du logiciel du superviseur de réglage. La séquence AUTO/TUNE suivie par ADAPT/TUNE est représentée ci-dessous pour un réglage Autotune en montant au point de consigne.

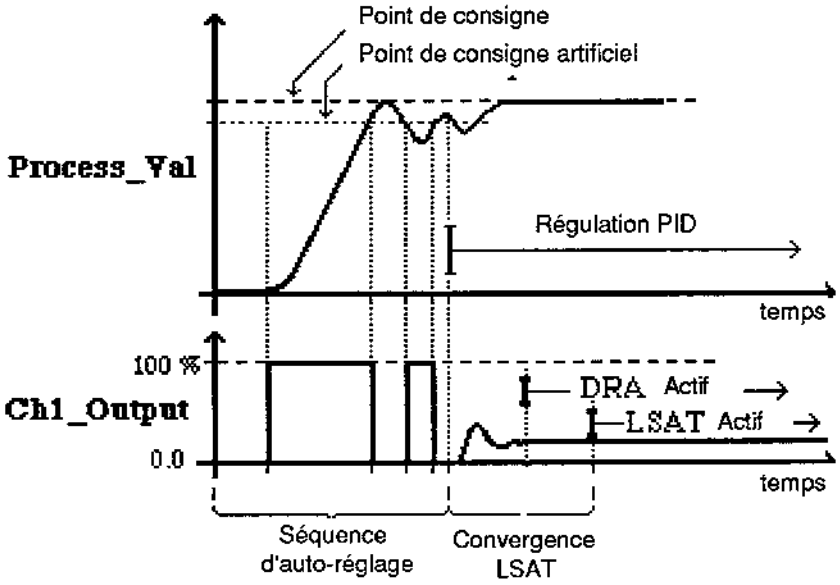


Figure 9-51 Séquence de convergence de réglage Autotune–Adaptive lors d'un réglage en phase de démarrage

Paramètres de réglage du bloc fonction

{1} Tune_Type

Tune_Type définit le régleur éventuellement actif. Il peut prendre huit valeurs différentes :

Aucun (0) :..... Aucun régleur actif

AT (1) : ..... Auto-réglant actif

DRA (2) : ..... Réglage adaptatif (DRA) actif

LSAT (3) : ..... Réglage adaptatif (DRA et LSAT) actif

DRA_LS (4) : ..... Réglage adaptatif (DRA et LSAT) actif

AT_DRA (5) : ..... Auto-réglant suivi d'un réglage adaptatif (DRA) actif

AT_LSAT (6) : ..... Auto-réglant suivi d'un réglage adaptatif (DRA et LSAT) actif

AT_D_L (7) :..... Auto-réglant suivi d'un réglage adaptatif (DRA et LSAT) actif

Param_Change

Param_Change est positionné par les systèmes de réglage pour indiquer que l'un des paramètres a été modifié. Une fois mis sur Oui (1) par un des systèmes, il doit être remis sur Non (0) en externe.

Trigger_Val

Trigger_Val fournit au réglage adaptatif une indication sur la quantité de bruit de process présent dans Process_Val. Il est automatiquement indiqué par l'auto-réglant pendant la séquence de réglage, mais l'opérateur peut l'introduire en mode manuel.

AT_Out_High et AT_Out_Low

AT_Out_High et AT_Out_Low limitent les oscillations de la sortie pendant l'auto-réglage (version 2-27 et suivantes du progiciel). Pour les versions antérieures, la limitation des oscillations en sortie peut se faire pendant l'auto-réglage par Output_High et Output_Low.

Tuning

Tuning fournit l'indication que les systèmes de réglage sont en fonctionnement. Si des réglages automatiques ou adaptatifs sont actifs, Tuning est actif (1), sinon Tuning est Off (0).

Fonction inhibition de dépassement du bloc fonction

Le bloc fonction PID_Auto est en mesure d'inhiber le dépassement de Process_Val susceptible de se produire lors des rampes de Setpoint. Pour utiliser la fonction d'inhibition, il est nécessaire de fournir au bloc fonction les informations concernant le point de consigne visé et la pente de la rampe et de régler le niveau d'inhibition de dépassement requis. Utiliser, à cet effet, les quatres entrées d'inhibition.

Inhibiter (Inhibiteur)

Inhibiter définit le niveau d'inhibition de dépassement requis. Il peut être réglé sur une valeur comprise entre 0 et 1, la valeur O correspondant à l'absence d'inhibition.

Ramp_Rate et Ramp_Units

Ramp_Rate définit la vitesse d'évolution du point de consigne sur la rampe. Les unités sont fixées par Ramp_Units.

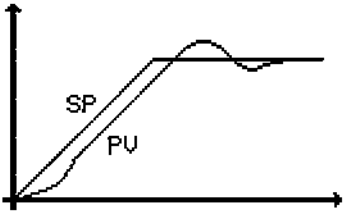
Target_SP

Target_SP est la valeur finale visée par la rampe du point de consigne.

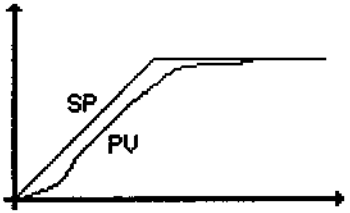
Dans une régulation PID correctement réglée, un dépassement du point de consigne a généralement lieu en fin de rampe, car l'intégrateur est remonté pendant la rampe. La fonction inhibition de dépassement agit en réduisant la remontée de l'intégrateur vers la fin de la rampe, ce qui supprime la "force d'inertie" de la mesure et lui permet de s'approcher du point de consigne visé, sans dépassement.

La fonction d'inhibition est initialisée au départ de la rampe de point de consigne, lorsque la valeur Target_SP change. Le point auquel l'intégrateur remonte, et la valeur de la suppression sont alors déterminés en fonction du gain et de la constante de temps de la régulation, ainsi que de Ramp_Rate et de la valeur d'Inhibiter qui peut être fixée entre (0) et (1). Des exemples de caractéristiques de dépassement sont représentés sur la figure ci-dessous pour les cas suivants : pas d'inhibition, réglage idéal de l'inhibiteur, et réglages trop fort et trop faible de l'inhibiteur.

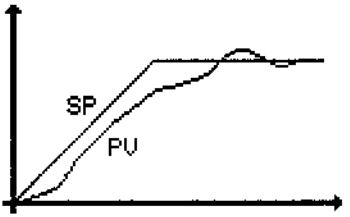
L'algorithme a été prévu pour donner une caractéristique de suramortissement avec un réglage de 0,5 pour l'inhibiteur. Quand la rampe du point de consigne atteint le point approprié, proche de Target_SP, l'intégrateur est éliminé, ce qui inhibe le dépassement. Ceci est représenté par la courbe {2}. Si le réglage de l'inhibiteur est trop fort, l'intégrateur est supprimé trop tôt et recommence à monter, ce qui donne un dépassemnt retardé, comme le montre la courbe (3). Si le réglage de l'inhibiteur est trop faible, l'intégrateur est supprimé trop tard et un dépassement se produit en raison de l'inertie de la mesure, comme le montre la courbe {4}.



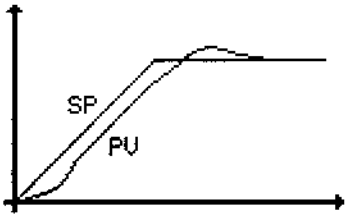
{1} Absence d'inhibition du dépassement



{2} Inhibiteur positionné correctement



{3} Inhibiteur positionné sur une valeur trop élevée



{4} Inhibiteur positionné sur une valeur trop faible

Figure 9-52 Réponses à une rampe de point de consigne montrant l'influence de l'inhibition.

Paramètres de diagnostic du bloc fonction

MTC
Constante de temps du modèle de référence de LSAT.

Q
Q est le facteur de déréglage de référence de LSAT.

CS_Pre_Limit

C'est la sortie de la régulation PID après addition de Feed_Forward, mais avant d'effectuer la limitation de Output.

Control_Sig (Signal de régulation)

Control_Sig est la somme des composantes proportionnelle, intégrale, dérivée et de tendance, après limitation haute et basse et limitation de la vitesse de changement, mais avant adjonction de la bande morte et du gain relatif de double canal.

Nerrer, Integral_Out et Deriv_Out,

Nerror, Integral_Out et Deriv_Out sont respectivement les sorties des composantes proportionnelle, intégrale et dérivée.

Feedback (Contre-réaction)

Feedback est la valeur du signal de retour au PID pour indiquer le signal réel en sortie de la régulation.

Error (Ecart)

Error est la différence entre la valeur de consigne et le point de consigne.

AT_State

AT_State indique l'état de l'auto-réglant.

0	Remise à zéro
1	Initialisation
2	Bruit à l'état repos du moniteur
3	Fin du bruit du moniteur
4	Démarrage avec nouveau point de consigne
5	Fin de démarrage avec nouveau point de consigne
6	Démarrage avec mesure au point de consigne
7	Fin de démarrage avec mesure au point de consigne
8	Séquence de Ziegler-Nichols
9	Calcul de nouveaux paramètres
10	Ecriture de la mise à jour de l'état
11	Echec d'Autotune
12	Autotune réussi

Tableau 9-12 Valeurs d'AT_State

ZN_Stage

ZN_Stage indique l'état des étages de réglage Ziegler-Nichols de l'Auto-réglant.

3	Trouver pic PV & Sortie inverse
4	Trouver intersection PV / PV1 & Test pour retard dominant
5	Trouver pic PV & Sortie inverse ou chang.ement inverse PV
6	Trouver intersection PV / PV1 & Ajuster tendance et sortie
7	Trouver pic PV & Sortie inverse
8	Trouver intersection PV / PV1 & Calculer nouveaux paramètres

Tableau 9-13 Valeurs de Zn_State

DRA_State

DRA_State indique l'état du régleur adaptatif DRA.

0	Autorise un temps de réglage
1	Attente de déclenchement
2	Trouver pic 1
3	Trouver zéro 1
4	Trouver pic 2
5	Trouver zéro 2
6	Trouver pic 3
7	Trouver zéro 3
8	Trouver pic 4
9	Trouver zéro 4
10	Trouver pic 5
11	Echec de fin sur zéro 4
12	Fin sur pic 4 trouvée
13	Echec de fin sur zéro 5
14	Fin sur pic 5 trouvée
15	Préparer mise à jour

Tableau 9-14 Valeurs de DRA_State

DRA_Last

DRA_Last enregistre la dernière stratégie de réglage effectuée par le régleur adaptatif DRA.

0	Pas de changement
1	Réduire l'amortissement
2	Augmenter le gain
3	Temps diminué
4	Temps augmenté
5	Gain diminué

Tableau 9-15 Valeurs de DRA_Last

LSAT_F1

LSAT_F1 indique la forme du modèle de process qui a été identifiée par le LSAT.

Proc_Delay

Proc_Delay est le temps de retard du système en boucle ouverte estimé par l'auto-réglage.

Status

Status fournit une indication du fonctionnement du bloc PID_Auto. Huit états différents sont possibles :

OK (0) : Le bloc fonction fonctionne normalement

SnsrBrk (1) : Détection de rupture d'un capteur externe

PV_High (2) : Process_Val est supérieur à Span_High +10 %

PV_Low (3) : Process_Val est inférieur à Span_Low -10 %

SP_High (4) : Setpoint est supérieur à Span_High

SP_Low (5) : Setpoint est inférieur à Span_Low

GainNeg (6) : Prop_Band a été réglé avec une valeur négative ou Span_High a été réglé plus bas que Span_Low

GainHi (7) : La valeur donnée à Prop_Band est trop faible ou la plage utile du bloc fonction est trop étroite, ce qui donne un gain très important.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
AT_Out_High	**REAL**	0	Config	Config	Lim. haute Lim. basse	Output_High AT_Out_Low
AT_Out_Low	**REAL**	0	Config	Config	Lim. haute Lim. basse	AT_Out_High AT_Out_Low
AT_State	**SINT**	0	Config		Lim. haute Lim. basse	255 0
Break_Output	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Output_High Output_Low
Ch1_Ch2_D_B	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	10 - 10
Ch1_CT_Low	**TIME**	0	Super	Super	Lim. haute Lim. basse	1 d 0
Ch1_Cycle_T	**TIME**	5 s	Super	Super	Lim. haute Lim. basse	1 d 0
Ch1_Output	**REAL**	0	Oper		Lim. haute Lim. basse	100 Le plus grand de 0 ou Output_Low
Ch2_CT_Low	**TIME**	0	Super	Super	Lim. haute Lim. basse	1 d 0
Ch2_Cycle_T	**TIME**	5 s	Super	Super	Lim. haute Lim. basse	1 d 0
Ch2_Linear	**BOOL**	Oui (1)	Super	Config	Détect.	Non (0) Oui (1)
Ch2_Output	**REAL**	0	Oper		Lim. haute Lim. basse	Le plus petit de 0 ou Output_High -100

Tableau 9-16 Attributs des paramètres de PID_Auto

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Control_Sig	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
CS_Pre_Limit	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Cutback_High	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Cutback_Low	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Debump	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Debump_Dis	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Deriv_On_PV	**BOOL**	On_Err (0)	Config	Config	Détect.	On_Err (0) On_PV (1)
Deriv_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Derivative	**TIME**	50 s	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Direct	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
DRA_Last	**SINT**	0	Config		Lim. haute Lim. basse	255 0
DRA_State	**SINT**	0	Config		Lim. haute Lim. basse	255 0
Error	**REAL**	0	Super		Lim. haute Lim. basse	100 000 -100 000
Feed_Forward	**REAL**	0	Super	Super	Lim. haute Lim. basse	100 - 100
Feedback	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000

Tableau 9-16 Attributs des paramètres de PID_Auto (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Inhibiter	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	1 0
Integral	**TIME**	5 m	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Integral_Hold	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Integral_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
LSAT_F1	**REAL**	0	Config		Lim. haute Lim. basse	100 000 0
Manual	**BOOL**	Manual (1)	Oper	Oper	Détect.	Auto (0) Manual (1)
Manual_Reset	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	100 -100
MTC	**REAL**	30	Config	Config	Lim. haute Lim. basse	100 000 0,1
Nerror	**REAL**	0	Super		Lim. haute Lim. basse	100 000 -100 000
Out_Rate_En	**BOOL**	Non (0)	Oper	Config	Détect.	Non (0) Oui (1)
Output	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Output_High Output_Low
Output_High	**REAL**	100	Oper	Oper	Lim. haute Lim. basse	100 Output_Low
Output_Low	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Output_High -100
Output_Rate	**REAL**	10	Oper	Oper	Lim. haute Lim. basse	10 000 0,1

Tableau 9-16 Attributs des paramètres de PID_Auto (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Param_Change	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Proc_Delay	**TIME**	0	Config		Lim. haute Lim. basse	1d 0
Process_Val	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
Prop_Band	**REAL**	5 %	Oper	Oper	Lim. haute Lim. basse	10 000 0,1
Q	**REAL**	0,4	Config	Config	Lim. haute Lim. basse	100 000 0
Ramp_Rate	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	10 000 0
Ramp_Units	**ENUM**	/ Seconde (0)	Oper	Super	Détect.	/ Seconde (0) / Minute (1) / Heure(2) / Jour (3)
Rel_Ch2_Gain	**REAL**	1	Oper	Oper	Lim. haute Lim. basse	10 0,1
Sensor_Break	**BOOL**	Ok (0)	Oper	Super	Détect.	Ok (0) Break (1)
Setpoint	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
SP_FF_Enable	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
SP_FF_Trim	**REAL**	0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
PV_FF_Enable	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
PV_FF_Trim	**REAL**	0	Config	Config	Lim. haute Lim. basse	100 000 -100 000
Span_High	**REAL**	100	Config	Config	Lim. haute Lim. basse	100,000 Span_Low

Tableau 9-16 Attributs des paramètres de PID_Auto (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Span_Low	**REAL**	0	Config	Config	Lim. haute Lim. basse	Span_High - 100 000
Status	**ENUM**	Ok (0)	Oper		Détect.	Ok (0) SnsrBrk (1) PV_High (2) PV_Low (3) SP_High (4) SP_Low (5) GainNeg (6) GainHi (7)
Target_SP	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
Track_Enable	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Track_Value	**REAL**	0	Super	Config	Lim. haute Lim. basse	Output_High Output_Low
Trigger_Val	**REAL**	0,1	Config	Config	Lim. haute Lim. basse	100 000 0
Tune_Type	**ENUM**	Aucun (0)	Config	Config	Détect.	Aucun (0) AT (1) DRA (2) LSAT (3) DRA_LS (4) AT_DRA (5) AT_LSAT (6) AT_D_L (7)
Tuning	**BOOL**	Off (0)	Super		Détect.	Off (0) Active (1)
Zn_Stage	**SINT**	0	Config		Lim. haute Lim. basse	255 0

Tableau 9-16 Attributs des paramètres de PID_Auto (Suite)

BLOC FONCTION VP_AUTO

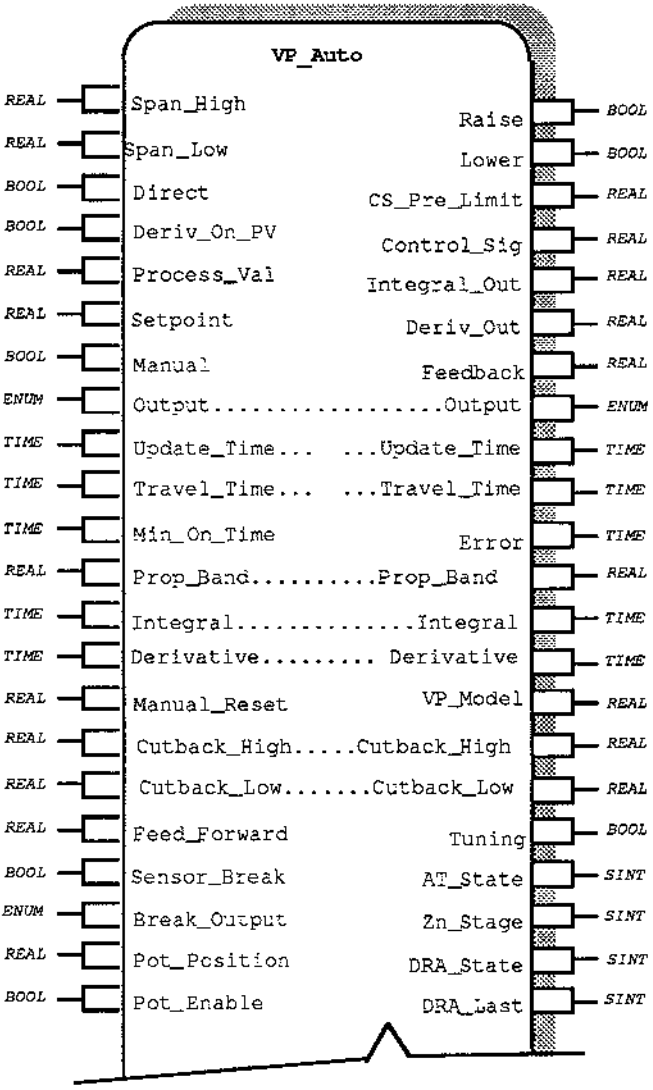


Figure 9-53 Schéma du bloc fonction VP_Auto

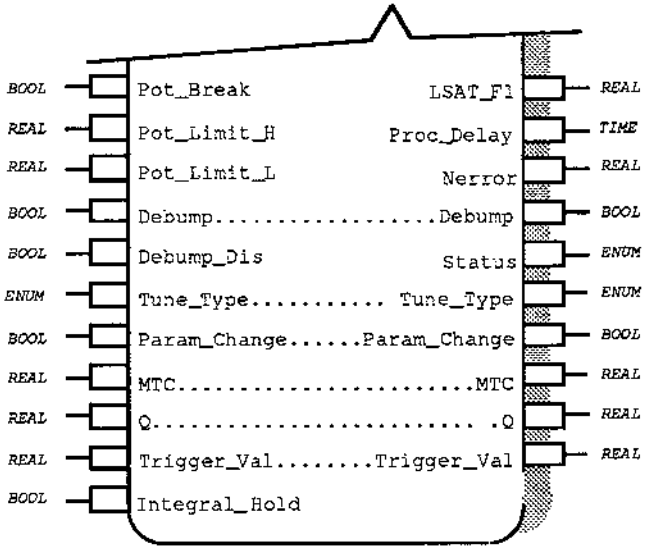


Figure 9-53 Schéma du bloc fonction VP_Auto (Suite)

Description fonctionnelle

Le bloc fonction VP_Auto met en oeuvre un algorithme de régulation proportionnelle, intégrale et dérivée, ainsi que les algorithmes de réglage Automatique et Adaptatif qui sont également utilisés dans les instruments de régulation Eurotherm série 900. L'algorithme de régulation est une version améliorée de celle qui est utilisée dans le bloc fonction PID. L'algorithme de base PID peut être représenté par l'équation :

$$\text{Sortie} = \left(\frac{10000}{\text{Span} * \text{Prop\_Band}}\right)\left(E(t) + \frac{1}{\text{Integral}}\int E(t).dt + \text{Dérivée} \cdot \frac{d.E\,(t)}{dt}\right)$$

dans laquelle E (t) est donné par (Setpoint - Process_Val) et Echelle par (Span_High - Span_Low). Dans le PC3000, cet algorithme élémentaire est soutenu par une fonction complémentaire pour améliorer les performances de régulation et pour permettre au bloc fonction d'être configuré pour réguler une grande variété de systèmes.

Les étages du bloc fonction positionneur de vanne mettent en oeuvre un algorithme dépourvu de limites, prévu pour ne pas nécessiter de signal de retour de position de la vanne, car celui-ci est souvent peu fiable. Un modèle de la vanne est prévu pour réguler la sortie positionnement de la vanne. Un potentiomètre de retour de position peut être utilisé avec le modèle de la vanne : dans ce mode de fonctionnement, la position du modèle de vanne est utilisée pour la régulation. La position réelle n'est utilisée qu'à des fins de limitation. Ceci permet à la régulation de continuer à fonctionner en cas de défaut du potentiomètre de retour de position.

VP_Auto est un bloc fonction complexe. L'obtention de performances optimales avec ce bloc implique que la régulation, l'étage de sortie et les auto-réglants soient correctement configurés et activés. La description des différents éléments qui composent le bloc fonction est donnée ci-après.

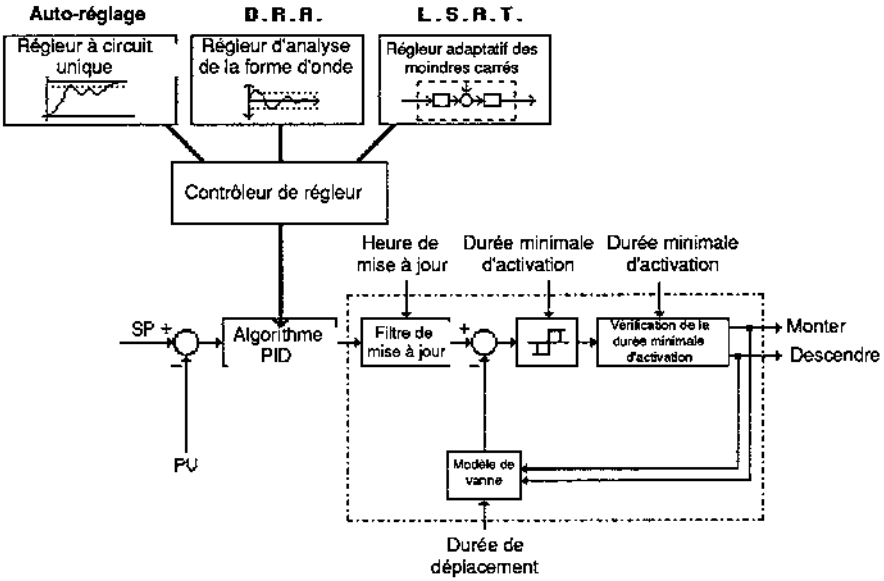


Figure 9-54 Schéma fonctionnel des principales fonctions de VP_Auto

Attributs du bloc fonction

Type : 20 60
Classe : .....RÉGULATION
Tâche par défaut : .....Task_2
Liste récapitulative : .....Setpoint, Process_Val, Manual, Output
Besoins de capacité mémoire : 1930 octets
Durée d'exécution : .....1,8 ms en l'absence de réglage
3,73 ms maximum avec réglage

Description des paramètres

AT_State:	Diagnostic	Nerror:	Diagnostic
Break_Output:	Configuration	Output:	Sortie
Control_Sig:	Diagnostic	Param_Change:	Auto-réglage
CS_Pre_Limit	Diagnostic	Pot_Break:	Retour potentiom.
Cutback_High:	Régulation	Pot_Enable:	Retour potentiom.
Cutback_Low:	Régulation	Pot_Limit_Hi:	Retour potentiom.
Debump:	Entrée dynamique	Pot_Limit_Lo:	Retour potentiom.
Debump_Dis:	Entrée dynamique	Pot_Position:	Retour potentiom.
Deriv_On_PV:	Configuration	Process_Val:	Entrée dynamique
Deriv_Out:	Diagnostic	Proc_Delay:	Diagnostic
Derivative:	Régulation	Prop_Band:	Régulation
Direct:	Configuration	Q:	Auto-réglage
DRA_Last:	Diagnostic	Raise:	Sortie
DRA_State:	Diagnostic	Sensor_Break:	Entrée dynamique
Error:	Diagnostic	Setpoint:	Entrée dynamique
Feed_Forward:	Entrée dynamique	Span_High:	Configuration
Feedback:	Diagnostic	Span_Low:	Configuration
Integral:	Régulation	Status:	Diagnostic
Integral_Hold:	Régulation	Travel_Time:	Configuration
Integral_Out:	Diagnostic	Trigger_Val:	Auto-réglage
Lower:	Sortie	Tune_Type:	Auto-réglage
LSAT_F1:	Diagnostic	Tuning:	Auto-réglage
Manual:	Régulation	Update_Time:	Configuration
Manual_Reset:	Régulation	VP_Model	Diagnostic
MTC:	Auto-réglage	Zero_Deriv:	Régulation
Min_On_Time:	Configuration	ZN_Stage:	Diagnostic

Tableau 9-17 Classification des paramètres

Paramètres de configuration de la régulation

Pour un fonctionnement optimal, la régulation doit être configurée selon le type de problème de régulation pour lequel elle doit être utilisée. Cette configuration est obtenue en réglant les paramètres d'entrée d'une façon appropriée. L'utilisation de ces paramètres de configuration est décrite ci-après.

Span_High et Span_Low.

Span_High et Span_Low définissent les limites maximale et minimale de la plage utile du bloc fonction. Généralement, celles-ci sont réglées sur des valeurs qui représentent des limites physiques dans le fonctionnement du process, comme la plage d'étalonnage d'un capteur ou les limites de sécurité d'un réservoir sous pression. La bande proportionnelle de l'algorithme PID est définie sous forme d'un pourcentage de la plage utile du bloc fonction, qui s'obtient en soustrayant Span_Low de Span_High. Si Process_Val sort de la plage utile, le bloc fonction se trouve en situation de rupture capteur.

Direct

Direct définit si le bloc fonction est en action directe ou en action inverse. Si le bloc fonction est en action directe, la sortie va tendre à augmenter si Process_Val est supérieur au point de consigne. Si le bloc fonction est en action inverse, la sortie va tendre à augmenter si Process_Val est inférieur au point de consigne.

Deriv_On_PV

Deriv_On_PV définit si l'action dérivée doit répondre aux changements de Process_Val seulement (On_PV (1)) ou aux changements d'écart entre le point de consigne et Process_Val (On_Err (0)).

Update_Time

Update_Time définit le temps d'échantillonnage des étages de sortie du positionneur de vanne. Généralement, celui-ci doit être réglé au 1/10 de Travel_Time. Si Update_Time augmente, l'activité de la vanne diminue, mais cela risque de réduire la précision de la régulation. De même, en diminuant Update_Time, les performances de la régulation sont meilleures mais l'activité de la vanne est augmentée.

Travel_Time

Travel_Time est le temps nécessaire à la vanne pour se déplacer de la position basse de fonctionnement à sa position haute de fonctionnement.

Min_On_Time

Min_On_Time définit le temps minimum pendant lequel la vanne doit maintenir son fonctionnement d'ouverture ou de fermeture ou bien rester immobile.

Break_Output

Break_Output définit le niveau de sortie par défaut du bloc fonction lorsqu'une situation de rupture capteur a été détectée. Il est déclenché soit lorsque Sensor_Break est mis sur Break (1), soit lorsque Process_Val ou le point de consigne sort de la plage utile du bloc fonction.

Paramètre de retour du potentiomètre

Pot_Position

Pot_Position est l'entrée à laquelle est raccordé le potentiomètre de retour de position de la vanne.

Pot_Enable

Si Pot_Enable est mis sur Oui (1), l'algorithme de contre-réaction du potentiomètre est activé. Si Pot_Enable est mis sur Non (0), le bloc fonction fonctionne sans retour de potentiomètre.

Pot_Break

Pot_Break fournit un déclenchement externe pour indiquer que le retour de potentiomètre a été débranché ou est en défaut. Si Pot_Break est mis sur Oui (1), cela signifie qu'une situation de défaut a eu lieu. Si Pot_Break est mis sur Non (0), cela signifie que le capteur fonctionne correctement.

Pot_Limit_Hi

Pot_Limit_Hi définit la position haute du fonctionnement de la vanne.

Pot_Limit_Lo

Pot_Limit_Lo définit la position basse du fonctionnement de la vanne.

Paramètres d'entrée dynamique et de sortie

Setpoint (Point de consigne) et Process_Val (Mesure)

Process_Val est la variable régulée du bloc fonction et Setpoint est la valeur cible par rapport à laquelle est régulé Process_Val. L'algorithme PID agit de façon à réduire à zéro l'écart entre le point de consigne et la mesure.

Output, Raise et Lower

Lorsque le bloc fonctionne en mode Manuel, Output (Sortie) sert d'entrée pour réguler directement les sorties Raise (Monter) ou Lower (Descendre) de la vanne. En mode Auto, Output est mis automatiquement sur Off (0).

Sensor_Break

L'entrée "Sensor-Break" constitue un déclenchement qui peut être utilisé pour placer le bloc fonction dans un état de rupture capteur. Lorsque le bloc fonction est en rupture capteur, la sortie prend la valeur Break_Output et l'algorithme PID est désactivé. Lorsqu'il quitte l'état "Sensor Break", l'algorithme maintient la sortie à Break_Output pendant 16 cycles d'échantillonnage, pour s'assurer que le capteur a bien repris sa tâche d'acquisition.

Debump et Debump_Dis

La fonction Debump est utilisée par le bloc fonction pour éviter que les changements de conditions de fonctionnement ou de paramètres de régulation ne provoquent de brusques déviations du signal de sortie. Dans le PC3000, la suppression d'à-coup (debumping) s'effectue automatiquement lorsque des changements sont apportés aux paramètres Prop_Band, Derivative, Span_High, Span_Low, Ch1_Ch2_D_B, Rel_Ch2_Gain ou lors des commutations entre les modes Manual et Auto. Le paramètre Debump peut être utilisé pour supprimer les à-coups d'autres changements, par exemple ceux du point de consigne. Debump_Dis peut être utilisé pour désactiver la fonctionnalité Debump, permettant ainsi à la sortie de faire un "saut" en réponse à la modification de l'un des paramètres indiqués ci-dessus.

Paramètres de régulation

Le bloc fonction dispose de différents paramètres qui ne sont pas prévus pour changer d'une façon dynamique, mais qui commandent le fonctionnement de l'algorithme. Ils incluent les paramètres de réglage, comme Prop_Band, ainsi que des paramètres booléens servant à modifier le mode de fonctionnement. Ces paramètres sont décrits ci-après.

Manual (Manuel)

Manual est utilisé pour placer le bloc fonction soit en mode manuel, soit en mode automatique. En mode Manuel, l'algorithme PID est désactivé et la valeur Output (Sortie) est prise directement sur l'entrée. En mode Auto, la totalité de la fonction du bloc est active.

Prop_Band

Prop_Band est la bande proportionnelle de l'algorithme de régulation PID, exprimée en % de la plage utile. La bande proportionnelle est définie comme étant le pourcentage de la plage utile de la régulation où un écart de régulation produit un signal de sortie égal à la valeur de sortie maximale de l'instrument. Par exemple, si Span_High = 1000, Span_Low = 0, Setpoint = 500, Process_Val = 490, Prop_Band = 1 %, une régulation à action proportionnelle inverse seulement aura une sortie de +100 %, car l'erreur est de 10 unités, ce qui correspond à la bande proportionnelle.

N.B. : le gain proportionnel est donné par :

$$\text{Gain} = \frac{10{,}000}{(\text{Span\_High} - \text{Span\_Low}) * \text{Prop\_Band}}$$

Integral (Intégrale)

Integral est la constante de temps d'intégrale de l'algorithme de régulation PID. Le temps d'intégrale est défini comme étant la période de temps sur laquelle la partie du signal de sortie due à l'action intégrale augmente d'une quantité égale à la partie du signal due à l'action proportionnelle, pour un état d'écart constant.

Derivative (Dérivée)

Derivative est la constante de temps de dérivéede l'algorithme de régulation PID. Le temps de dérivée est défini comme étant l'intervalle de temps sur lequel la partie du signal de sortie due à l'action de dérivée augmente d'une quantité égale à

la partie du signal due à l'action proportionnelle, lorsque l'écart de régulation change à une vitesse constante.

Feed_Forward (Tendance)

Feed_Forward est ajouté directement en sortie de l'algorithme PID, avant d'effectuer la limitation de sortie et les conversions de sortie double.

Manual_Reset (Intégrale manuelle)

Manual_Reset n'est actif que si Integral est mis à zéro. Il effectue un décalage de la sortie pouvant servir à annuler l'écart de régulation lorsque l'action intégrale n'est pas utilisée.

Cutback_High et Cutback_Low

Le "Cutback" peut être utilisé pour réduire le délai de réponse de Process_Val à des changements importants de point de consigne et pour limiter le dépassement éventuel au cours de la phase transitoire. Cutback_High agit lorsque Process_Val est initialement supérieur au point de consigne recherché et Cutback_Low agit lorsque Process_Val est initialement inférieur au point de consigne recherché. Cutback_High et Cutback_Low sont exprimés en unités physiques. Cutback agit en forçant la sortie à sa limite appropriée, maximale ou minimale, en réponse à un changement de point de consigne supérieur à la bande de cutback. Lorsque Process_Val approche du point de consigne, l'écart de régulation (Setpoint - Process_Val) devient inférieur à la valeur de cutback et le fonctionnement normal reprend.

Integral_Hold

Integral_Hold permet de "geler" la sortie intégrale à sa valeur en cours. Celle-ci reste constante, tant que Integral_Hold est actif.

Algorithmes de réglage de VP_Auto

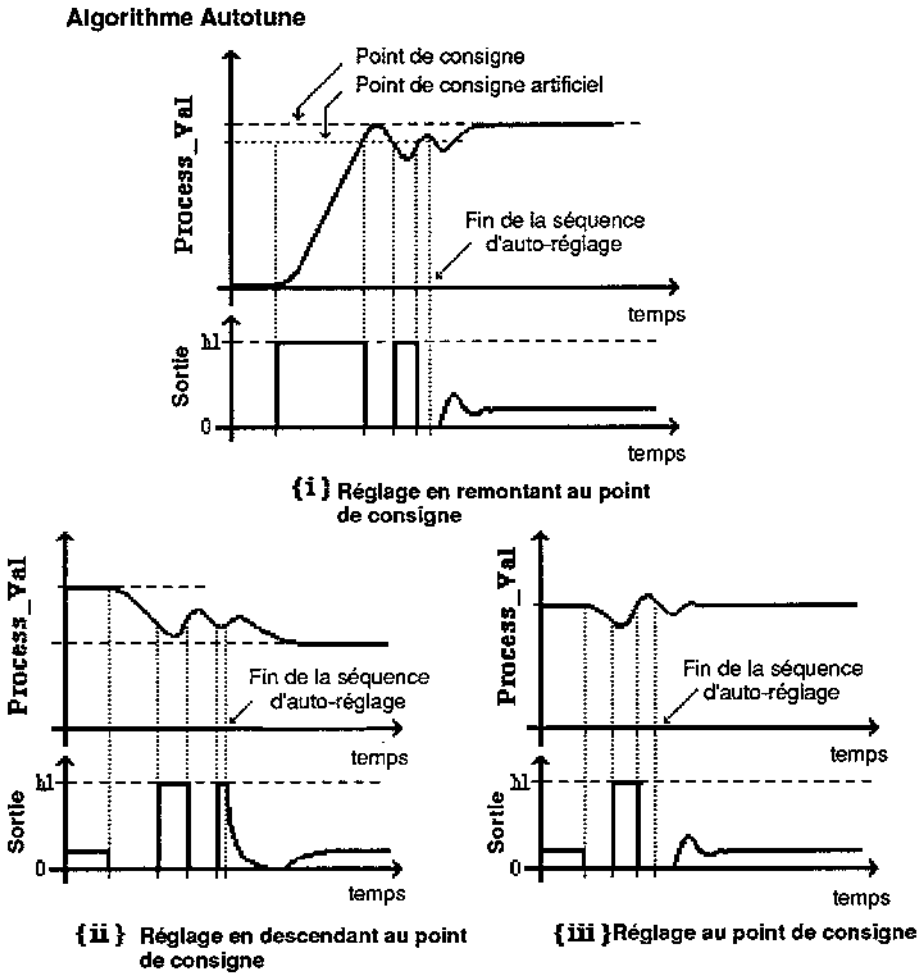


Figure 9-55 Cycles d'auto-réglage, en chauffage seul

L'algorithme d'auto-réglage est un régleur non répétitif prévu initialement pour les instruments Eurotherm 818 et 815. Le principe de fonctionnement a été décrit en détail précédemment dans la Vue d'ensemble du PC3000, dans le présent chapitre.

Lorsque l'algorithme d'auto-réglage est activé, l'algorithme de régulation PID est déconnecté du process et l'auto-réglant pilote directement les sorties de la régulation au moyen d'ordres Marche / Arrêt. Au moment où l'auto-réglage est activé, si la mesure en cours ne s'écarte pas de plus de 1 % du point de consigne, les sorties sont gelées à leur valeur réelle, dans le cas contraire, les sorties sont forcées à zéro. Cette valeur est maintenue pendant une minute, durant laquelle l'auto-réglant surveille le process pour déterminer le niveau de bruit et l'action de la mesure. Pendant cette première minute, l'opérateur peut choisir le point de consigne pour lequel l'auto-réglage doit être effectué. Ce peut être le point de consigne en cours, si nécessaire, ou bien un point de consigne supérieur ou inférieur à la valeur en cours.

Lorsque la phase de surveillance d'une minute est écoulée, le cycle de l'auto-réglant passe à sa phase active. Si le point de consigne en cours est supérieur à la mesure réelle de plus de 1 % de la plage utile, un réglage en montant est effectué. Si le point de consigne en cours est inférieur à la mesure réelle, de plus de 1 % de la plage utile, un réglage en descendant est effectué. Si le point de consigne en cours ne s'écarte pas de la mesure de plus de 1 % de la plage utile, un réglage sur le point de consigne est effectué. Ceci est décrit en détail dans la rubrique citée ci-dessus. La mesure, ainsi que les courbes de sortie des trois types de cycle d'auto-réglage, dans le cas d'un chauffage seul, sont représentées sur la figure ci-dessus.

A la fin du cycle d'auto-réglage, les paramètres suivants sont calculés et chargés dans la régulation :

Bande proportionnelle	
Temps d'intégrale	
Temps de dérivée	
Cutback haut	(réglage en descendant au point de consigne uniquement)
Cutback bas	(réglage en montant au point de consigne uniquement)
Trigger Val	(utilisé par DRA et LSAT)
MTC	(utilisé par LSAT)
Q	(utilisé par LSAT)

Algorithme d'analyse de réponse à une pertubation (DRA)

Le DRA est un régleur adaptatif, prévu initialement pour l'instrument Eurotherm 818. C'est un système expert élémentaire qui agit en identifiant des modèles dans la réponse d'écart (point de consigne - mesure) et ajuste les paramètres de la régulation pour compenser les réponses qui sont trop lentes ou qui oscillent. Contrairement à l'auto-réglant, DRA ne pilote pas directement les sorties de la régulation.

Le DRA est déclenché pour surveiller la réponse de la régulation lorsque l'écart absolu dépasse Trigger_Value, qui peut être réglé manuellement mais qui est fixé automatiquement par l'auto-réglant. Une fois déclenché par une perturbation, comme un changement de point de consigne ou une variation de la charge, le DRA enregistre jusqu'à deux cycles d'oscillation de l'écart avant de décider quelle modification éventuelle des termes de régulation est nécessaire. Le rapport des maxima de l'écart (taux d'amortissement) est le principal critère utilisé pour décider si un nouveau réglage est nécessaire. Si l'amplitude des maxima est faible, l'auto-réglant se désactive lui-même et ne refait pas de réglage des paramètres de la régulation. Dans la version 818 du DRA, si l'algorithme détermine qu'un nouveau réglage est nécessaire, la bande proportionnelle, les temps d'intégrale et de dérivée sont ajustés. Dans la mise en oeuvre du bloc PID_Auto, DRA ajuste également les termes MTC et Q qui référencent le LSAT. MTC est ajusté en série avec les temps d'intégrale et de dérivée. Q est ajusté avec la bande proportionnelle.

Si l'analyse de la réponse montre des oscillations, le temps d'oscillation est utilisé pour déterminer s'il est nécessaire d'ajuster le gain, les temps d'intégrale et de dérivée, ou bien à la fois le gain et les temps. La figure ci-après montre comment l'auto-réglant détermine la stratégie d'adaptation à adopter, qui est calculée conformément aux valeurs en cours des paramètres PID.

Choix de la stratégie de réglage adaptatif DRA

Algorithme du régleur adaptatif des moindres carrés (LSAT).

L'algorithme LSAT est un régleur adaptatif qui, comme l'algorithme de DRA, ne pilote pas directement les sorties de la régulation mais règle les paramètres de régulation PID. Le LSAT doit être considéré comme étant constitué de deux éléments principaux :

{1} Un identifieur de modèle de process

{2} Un configurateur de régulation.

L'algorithme surveille la sortie de la régulation et la mesure. Il envoie en permanence la sortie de la régulation dans un modèle interne du process et compare la mesure réelle à la mesure prédite par ce modèle. L'algorithme récurrent des moindres carrés est utilisé pour ajuster les paramètres du modèle, de telle sorte que la mesure prévue corresponde à la mesure réelle. Ainsi, le modèle est réglé en continu avec précision de telle sorte qu'il corresponde au process réel commandé. Le modèle est ensuite utilisé par le configurateur de régulation pour déterminer et régler les paramètres PID optimisés pour le process régulé. C'est une approche modèle / référence, dans laquelle les performances en boucle fermée du système sont optimisées par rapport à celles du modèle de référence interne en boucle fermée. Le schéma fonctionnel de l'algorithme du LSAT est représenté ci-dessous.

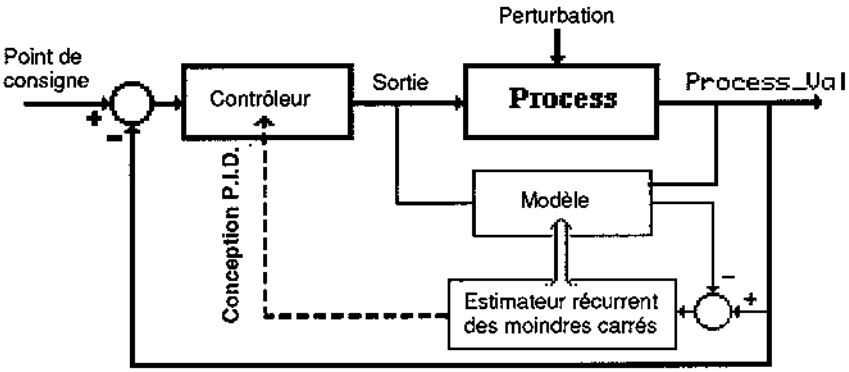


Figure 9-56 Schéma fonctionnel du LSAT

Le LSAT présente l'avantage, contrairement au DRA, de ne pas nécessiter de fortes perturbations de process ou de changements importants du point de consigne pour pouvoir agir. Il peut effectuer un réglage fin de la régulation en cours de régulation, puisqu'il est en mesure de discriminer les composantes du signal de bruit du process et de les utiliser pour l'identification du modèle. L'action de réglage fin du LSAT est une caractéristique importante de la mise en oeuvre du régleur adaptatif PID_Auto. Il permet à la fois de compléter l'action de réglage approché du DRA et de réaliser le réglage fin des paramètres réglés par l'auto-réglant.

La procédure d'identification du modèle par tendance et de configuration par référence au modèle permet au LSAT un réglage en continu des paramètres de la régulation au cours du fonctionnement de la régulation en ligne. Toutefois, l'algorithme du LSAT n'est pas en mesure de déterminer la structure du modèle du process : il fixe les paramètres d'un modèle dont la structure a été définie au préalable. Cette restriction peut limiter sa possibilité de mise en oeuvre en tant que régleur adaptatif autonome d'usage général, c'est pourquoi l'algorithme a été étendu pour adapter son identification et la détermination des paramètres, en réponse à des références externes. Cela se présente sous la forme de deux paramètres : "MTC", mise à l'échelle de la constante de temps process, qui détermine la fréquence d'échantillonnage de LSAT, et "Q", qui fournit une indication des paramètres non modélisés (haute fréquence) du process. Bien que ces informations puissent être entrées manuellement, elles nécessitent un niveau élevé de savoir-faire et de connaissance du process, c'est pourquoi les algorithmes Autotune et DRA ont été étendus pour identifier MTC et Q pour initialiser le LSAT avec ces valeurs. L'interaction résultante de ces trois auto-réglants constitue le système de réglage "Polyvalent".

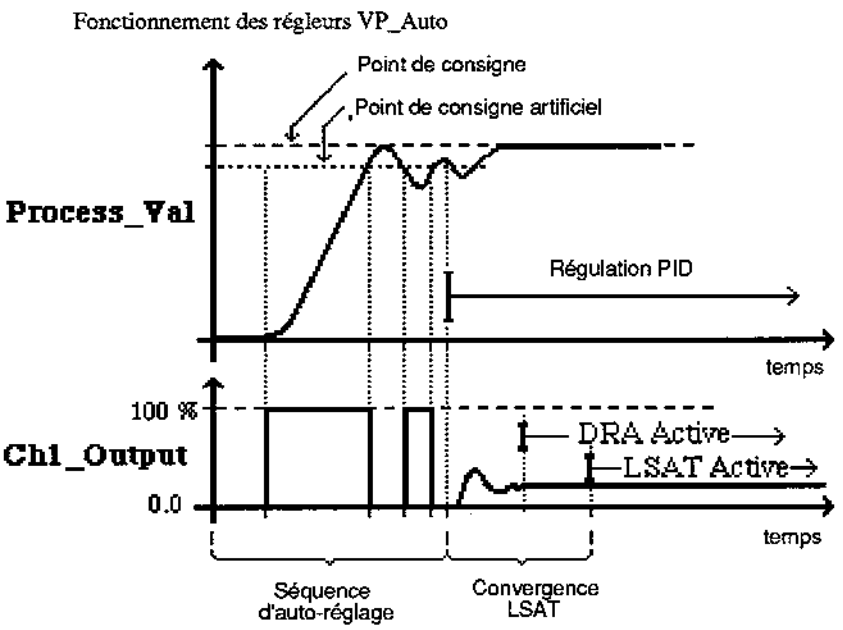


Figure 9-57 Séquence de convergence de réglage Autotune–Adaptive lors d'un réglage en phase de démarrage

Le système de réglage polyvalent associe les fonctions des trois auto-réglants décrits précédemment : Autotune, DRA et LSAT. Il a été conçu pour permettre d'utiliser les possibilités des trois algorithmes pour fournir les performances optimales de réglage, dans les conditions d'utilisation les plus variées de la régulation. Les fonctions assurées par les trois algorithmes de réglage de l'auto-réglant polyvalent peuvent être résumées comme suit :

{i} Autotune : réglage initial non répétitif des paramètres PID, identification approximative du modèle du process, identification des niveaux de bruit du process.

{ii} DRA : adaptation approchée aux changements de condition de fonctionnement du process, ajustement approximatif du modèle de référence du LSAT, ajustement éventuel de la régulation aux perturbations importantes du process.

{iii} LSAT : réglage fin des paramètres PID après la séquence d'auto-réglage et pendant le réglage adaptatif, réglage fin lors des rampes de consigne et en réponse à de faibles modifications du process ou à de faibles perturbations.

La validation de l'auto-réglant a pour effet d'activer la séquence du régleur non répétitif. La validation du réglage adaptatif active à la fois les deux algorithmes de réglage adaptatif DRA et LSAT. Il y a lieu de noter que **les performances optimales de réglage sont obtenues quand AUTO/TUNE précède ADAPT/TUNE.** L'algorithme Autotune a été amélioré pour fournir l'initialisation que nécessite le LSAT. Si Autotune ne précède pas Adaptive, le DRA est alors nécessaire pour initialiser le LSAT, ce qui peut nécessiter plusieurs séquences d'adaption de DRA et peut entraîner en un réglage très lent de la régulation.

Si Autotune suivi par Adaptive est choisi, une séquence Autotune est effectuée, suivie d'une commutation automatique sur le réglage par Adaptive. La séquence de réglage par Adaptive commence par inhiber le LSAT jusqu'à ce que l'estimateur ait convergé sur le modèle du process. Une fois cette convergence sur le LSAT effectuée, la séquence Adaptive Tune se poursuit avec le réglage de la régulation par les algorithmes, à la fois de DRA et de LSAT, sous le contrôle du logiciel du superviseur de réglage.

Paramètres de réglage du bloc fonction

Tune_Type

Tune_Type définit le régleur éventuellement actif. Il peut prendre huit valeurs différentes :

Aucun (0) : Aucun régleur actif

AT (1) : Auto-réglant actif

DRA (2) : Réglage adaptatif (DRA) actif.

LSAT (3) : Réglage adaptatif (DRA et LSAT) actif

DRA_LS (4) : Réglage adaptatif (DRA et LSAT) actif

AT_DRA (5) : Auto-réglant suivi d'un réglage adaptatif(DRA) actif

AT_LSAT (6) : Auto-réglant suivi d'un réglage adaptatif (DRA et LSAT) actif

AT_D_L (7) : Auto-réglant suivi d'un réglage adaptatif (DRA et LSAT) actif

Param_Change

Param_Change est positionné par les systèmes de réglage pour indiquer que l'un des paramètres a été modifié. Une fois mis sur Oui (1) par un des systèmes, il doit être remis sur Non (0) en externe.

Trigger_Val

Trigger_Val fournit au réglage adaptatif une indication sur la quantité de bruit du process présent dans Process_Val. Il est automatiquement indiqué par l'auto-réglage pendant la séquence de réglage, mais l'opérateur peut l'introduire en mode manuel.

Tuning

Tuning indique que les systèmes de réglage sont en fonctionnement. Si des réglages automatiques ou adaptatifs sont actifs, Tuning est actif (1), sinon Tuning est à l'arrêt (0).

Paramètres de diagnostic du bloc fonction

MTC

Constante de temps du modèle de référence de LSAT.

Q

Q est le facteur de déréglage de référence de LSAT.

CS_Pre_Limit

C'est la sortie de la régulation PID après addition de Feed_Forward, mais avant d'effectuer la limitation de Output.

Control_Sig (Signal de régulation)

Control_Sig est la somme des composantes proportionnelle, intégrale, dérivée et de tendance, après limitation haute et basse et limitation de la vitesse de changement, mais avant adjonction de la bande morte et du gain relatif de double canal.

Nerror, Integral_Out et Deriv_Out

Nerror, Integral_Out et Deriv_Out sont respectivement les sorties des composantes proportionnelle, intégrale et dérivée.

Feedback (Contre-réaction)

Feedback est la valeur du signal de retour au PID pour indiquer le signal réel en sortie de la régulation.

Error

Error est la différence entre la valeur de consigne et le point de consigne.

VP_Model

VP_Model est la sortie du modèle interne de la vanne.

AT_State

Indique l'état de l'auto-réglant.

0	Remise à zéro
1	Initialisation
2	Bruit à l'état repos du moniteur
3	Fin du bruit du moniteur
4	Démarrage avec nouveau point de consigne
5	Fin de démarrage avec nouveau point de consigne
6	Démarrage avec mesure au point de consigne
7	Fin de démarrage avec mesure au point de consigne
8	Séquence de Ziegler-Nichols
9	Calcul de nouveaux paramètres
10	Ecriture de la mise à jour de l'état
11	Echec d'Autotune
12	Autotune réussi

Tableau 9-18 Valeurs d'AT_State

ZN_Stage

ZN_Stage indique l'état des étages de réglage Ziegler-Nichols de l'Autotuner.

3	Trouver pic PV & Sortie inverse
4	Trouver intersection PV / PV1 & Test pour retard dominant
5	Trouver pic PV & Sortie inverse ou changement inverse PV
6	Trouver intersection PV / PV1 & Ajuster tendance et sortie
7	Trouver pic PV & Sortie inverse
8	Trouver intersection PV / PV1 & Calculer nouveaux paramètres

Tableau 9-19 Valeurs de Zn_State

DRA_State

DRA_State indique l'état du régleur adaptatif DRA.

0	Autorise un temps de réglage
1	Attente de déclenchement
2	Trouver pic 1
3	Trouver zéro 1
4	Trouver pic 2
5	Trouver zéro 2
6	Trouver pic 3
7	Trouver zéro 3
8	Trouver pic 4
9	Trouver zéro 4
10	Trouver pic 5
11	Echec de fin sur zéro 4
12	Fin sur pic 4 trouvée
13	Echec de fin sur zéro 5
14	Fin sur pic 5 trouvée
15	Préparer mise à jour

Tableau 9-20 Valeurs de DRA_State

DRA_Last

DRA_Last enregistre la dernière stratégie de réglage effectuée par le régleur adaptatif DRA.

0	Pas de changement
1	Réduire l'amortissement
2	Augmenter le gain
3	Temps diminué
4	Temps augmenté
5	Gain diminué

Tableau 9-21 Valeurs de DRA_Last

LSAT_F1

LSAT_F1 indique la forme du modèle de process qui a été identifiée par le LSAT.

Proc_Delay

Proc_Delay est le temps de retard du système en boucle ouverte estimé par l'auto-réglant.

Nerror

Nerror est la sortie de la composante proportionnelle du bloc fonction.

Status

Status fournit une indication du fonctionnement du bloc VP_Auto. Huit états différents sont possibles :

OK (0) : Le bloc fonction fonctionne normalement

SnsrBrk (1) : Détection de rupture d'un capteur externe

PV_High (2) : Process_Val est supérieur à Span_High +10 %

PV_Low (3) : Process_Val est inférieur à Span_Low -10 %

SP_High (4) : Setpoint est supérieur à Span_High

SP_Low (5) : Setpoint est inférieur à Span_Low

GainNeg (6) : Prop_Band a été réglé avec une valeur négative ou Span_High a été réglé plus bas que Span_Low

GainHi (7) : La valeur donnée à Prop_Band est trop faible ou la plage utile du bloc fonction est trop étroite, ce qui donne un gain très important.

Attributs des paramètres du bloc fonction

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
AT_State	**SINT**	0	Config		Lim. haute Lim. basse	255 0
Break_Output	**ENUM**	Lower (1)	Oper	Oper	Détect.	Off (0) Lower (1) Raise (2)
Control_Sig	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
CS_Pre_Limit	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Cutback_High	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Cutback_Low	**REAL**	0	Super	Super	Lim. haute Lim. basse	Span_High 0
Debump	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)
Debump_Dis	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Deriv_On_PV	**BOOL**	On_Err (0)	Config	Config	Détect.	On_Err (0) On_PV (1)
Deriv_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Derivative	**TIME**	50 s	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Direct	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
DRA_Last	**SINT**	0	Config		Lim. haute Lim. basse	255 0
DRA_State	**SINT**	0	Config		Lim. haute Lim. basse	255 0

Tableau 9-22 Attributs des paramètres de VP_Auto

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Error	**REAL**	0	Super		Lim. haute Lim. basse	100 000 -100 000
Feed_Forward	**REAL**	0	Super	Super	Lim. haute Lim. basse	100 -100
Feedback	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Integral	**TIME**	5 m	Oper	Oper	Lim. haute Lim. basse	01d_03 h 0
Integral_Hold	**BOOL**	Non (0)	Config	Config	Détect.	Non (0) Oui (1)
Integral_Out	**REAL**	0	Config		Lim. haute Lim. basse	100 000 -100 000
Lower	**BOOL**	Off (0)	Oper		Détect.	Off (0) Lower (1)
LSAT_F1	**REAL**	0	Config		Lim. haute Lim. basse	100 000 0
Manual	**BOOL**	Manual (1)	Oper	Oper	Détect.	Auto (0) Manual (1)
Manual_Reset	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	100 -100
Min_On_Time	**TIME**	100 ms	Oper	Oper	Lim. haute Lim. basse	5 s 100 ms
MTC	**REAL**	30	Config	Config	Lim. haute Lim. basse	100 000 0,1
Nerror	**REAL**	0	Super		Lim. haute Lim. basse	100 000 -100 000
Output	**ENUM**	Off (0)	Oper	Oper	Détect.	Off (0) Lower (1) Raise (2)
Param_Change	**BOOL**	Non (0)	Super	Config	Détect.	Non (0) Oui (1)

Tableau 9-22 Attributs des paramètres de VP_Auto (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Pot_Break	**BOOL**	Non (0)	Oper	Super	Détect.	Non (0) Oui (1)
Pot_Enable	**BOOL**	Non (0)	Oper	Super	Détect.	Non (0) Oui (1)
Pot_Limit_Hi	**REAL**	100	Oper	Super	Lim. haute Lim. basse	100 0
Pot_Limit_Lo	**REAL**	0	Oper	Super	Lim. haute Lim. basse	100 0
Pot_Position	**REAL**	0	Oper	Super	Lim. haute Lim. basse	100 0
Proc_Delay	**TIME**	0	Config		Lim. haute Lim. basse	1 d 0
Process_Val	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
Prop_Band	**REAL**	5 %	Oper	Oper	Lim. haute Lim. basse	10 000 0.1
Q	**REAL**	0,4	Config	Config	Lim. haute Lim. basse	100 000 0
Raise	**BOOL**	Off (0)	Oper		Détect.	Off (0) Raise (1)
Sensor_Break	**BOOL**	Ok (0)	Oper	Super	Détect.	Ok (0) Break (1)
Setpoint	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	Span_High Span_Low
Span_High	**REAL**	100	Config	Config	Lim. haute Lim. basse	100 000 Span_Low
Span_Low	**REAL**	0	Config	Config	Lim. haute Lim. basse	Span_High -100 000

Tableau 9-22 Attributs des paramètres de VP_Auto (Suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Status	**ENUM**	Ok (0)	Oper		Détect.	Ok (0) SnsrBrk (1) PV_High (2) PV_Low (3) SP_High (4) SP_Low (5) GainNeg (6) GainHi (7)
Travel_Time	**TIME**	20 s	Oper	Oper	Lim. haute Lim. basse	16 m_40 s 5 s
Trigger_Val	**REAL**	0,1	Config	Config	Lim. haute Lim. basse	100 000 0
Tune_Type	**ENUM**	Aucun (0)	Config	Config	Détect.	Aucun (0) AT (1) DRA (2) LSAT (3) DRA_LS (4) AT_DRA (5) AT_LSAT (6) AT_D_L (7)
Tuning	**BOOL**	Off (0)	Super		Détect.	Off (0) Active (1)
Update_Time	**TIME**	1s	Oper	Oper	Lim. haute Lim. basse	20 s 100 ms
VP_Model	**REAL**	50	Config		Lim. haute Lim. basse	100 0
Zn_Stage	**SINT**	0	Config		Lim. haute Lim. basse	255 0

Tableau 9-22 Attributs des paramètres de VP_Auto (Suite)

BLOC FONCTION PIDHEATCOOL

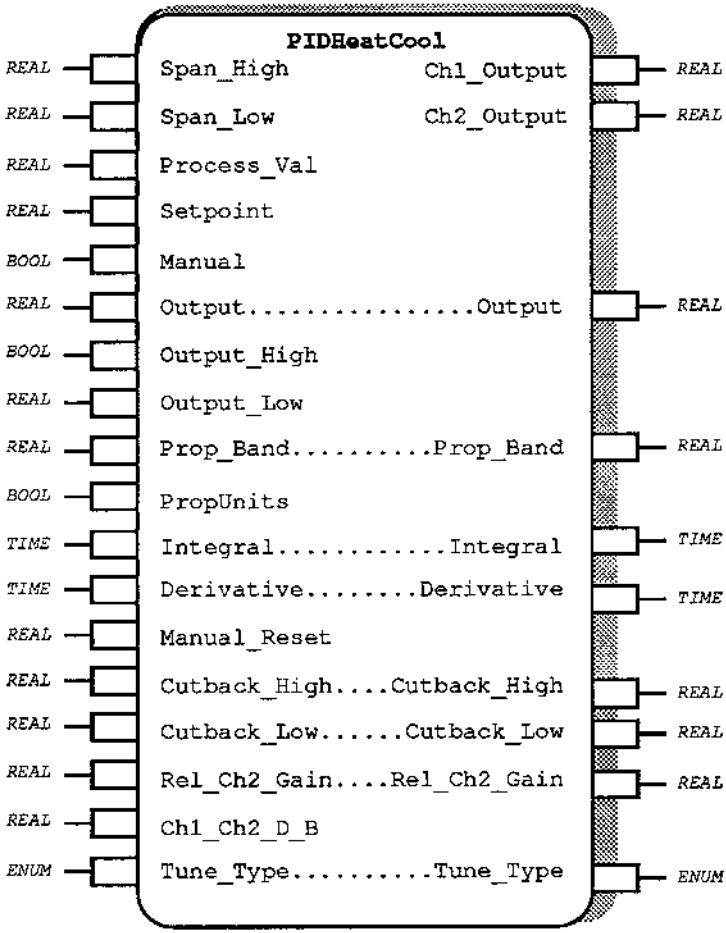


Figure 9-58 Schéma du bloc fonction PIDHeatCool

Description fonctionnelle

Le bloc fonction PIDHeatCool met en oeuvre l'algorithme de régulation proportionnel intégral dérivé et les algorithmes auto-réglant et auto-adaptatif qui sont également utilisés sur les appareils Eurotherm de la série 900 et dans les autres blocs fonctions de régulation PC3000 (PID, VP, PID_Auto et VP_Auto).

Le bloc fonction PIDHeatCool est une version réduite du bloc fonction PID_Auto. Les paramètres d'interface qui ne sont pas couramment utilisés ont été supprimés pour donner un bloc fonction plus simple, utilisable dans un grand nombre d'applications. Ces paramètres d'interface qui ont été supprimés sont réglés en interne sur leurs valeurs par défaut. Pour avoir une description détaillée de ces paramètres, il faut se reporter à la description du bloc fonction PID_Auto.

Attributs du bloc fonction

Type : 20 51

Classe : .....CONTROL

Tâche par défaut : .....Task_2

Liste résumée : .....Setpoint, Process_Val, Manual, Output

Mémoire nécessaire : 2056

Description des paramètres

Span_High et Span_Low.

Span_High et **Span_Low** définissent les limites maximale et minimale de la plage de service du bloc fonction. En règle générale, ces limites sont positionnées sur les valeurs qui représentent les limites physiques de fonctionnement du procédé, comme la plage étalonnée d'un transducteur ou les limites de sécurité d'un réservoir sous pression. La bande proportionnelle de l'algorithme PID peut être définie sous la forme d'un pourcentage de la plage du bloc fonction, calculé par soustraction entre **Span_Low** et **Span_High.**

Setpoint et Process_Val

Process_Val est la variable régulée du bloc fonction et **Setpoint** est la valeur cible par rapport à laquelle **Process_Val** est régulée . L'algorithme PID agit pour ramener à zéro la différence entre **Setpoint** et **Process_Val**.

Manual

Manual sert à sélectionner le mode de fonctionnement du bloc fonction : manuel ou automatique. En mode manuel, l'algorithme PID est désactivé et la valeur d'**Output** provient directement de son entrée. En mode automatique, l'ensemble des fonctions du régulateur sont actives.

Output, Ch1_Output et Ch2_Output

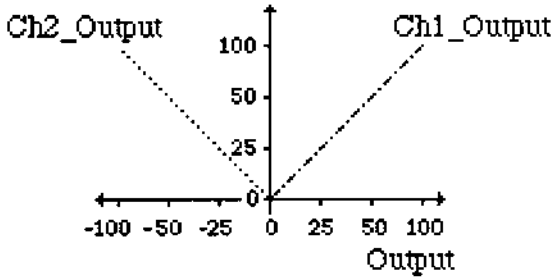


Figure 9-59 Relation de sortie double, avec Rel_Ch2_Gain égal à l'unité et Ch1_Ch2_D_B égal à zéro

Il est possible de configurer la voie du bloc fonction PID pour le fonctionnement en simple voie ou en double voie. En mode simple voie, la régulation est sur **Output**, qui est limité à l'intervalle + 100 % à 0 % par les paramètres **Output_High** et **Output_Low**. Le fonctionnement en double voie est conçu pour les systèmes comme les applications de chauffage-refroidissement dans lesquelles les valeurs négatives d'**Output** doivent être une sortie vers une unité de réfrigération comme valeurs absolues pour augmenter la vitesse de réfrigération. Pour le fonctionnement en double voie, **Output_Low** est fixé à - 100%.

Output_High et Output_Low

Output_High et **Output_Low** définissent les limites supérieure et inférieure d'**Output.** Ces deux limites doivent être comprises entre -100% et +100%, **Output_High** étant supérieur ou égal à **Output_Low.** Lorsque le bloc fonction est utilisé pour le fonctionnement en sortie double, **Ch1_Output** et **Ch2_Output** sont liées à **Output** comme le montre la figure 9-59, **Output** étant limitée par **Output_High** et **Output_Low**.

Rel_Ch2_Gain et Ch1_Ch2_D_B

Lorsque le bloc fonction est utilisé pour la régulation double voie, la relation entre **Ch2_Output** et **Output** est donnée par :

Ch2_Output = (Output + Ch1_Ch2_D_B) * Rel_Ch2_Gain

Rel_Ch2_Gain est destiné aux situations de régulation à double sortie non linéaire, comme les systèmes de chauffage/refroidissement, pour compenser les différences de gain du matériel piloté par les deux voies de sortie. **Ch1_Ch2_D_B** introduit une bande morte entre les deux voies de sortie qui peuvent être positionnées sur une valeur positive pour donner une zone dans laquelle aucune voie n'est active ou sur une valeur négative pour donner une zone de chevauchement dans laquelle les deux sorties sont actives.

Manual_Reset

Manual_Reset est uniquement actif lorsqu'**Integral** est positionné sur zéro. Il donne une compensation pour **Output,** qui peut servir à ramener l'erreur de régulation à zéro lorsque l'intégrale n'est pas utilisée.

Cutback_High et Cutback_Low

Cutback peut servir à réduire le temps nécessaire à **Process_Val** pour réagir aux variations importantes de **Setpoint** et à limiter le dépassement qui peut se produire au cours de la période de transition. **Cutback_High** fonctionne lorsque **Process_Val** est initialement supérieur à la **Setpoint** cible et **Cutback_Low** fonctionne lorsque **Process_Val** est initialement inférieure à la **Setpoint** cible. Les unités de **Cutback_High** et **Cutback_Low** sont les unités techniques. **Cutback** fonctionne par forçage d'**Output** à sa limite maximale ou minimale correcte en réponse à une variation de **Setpoint** supérieure à la bande de cutback. Lorsque **Process_Val** approche de **Setpoint**, l'erreur de régulation (**Setpoint - Process_Val**) revient à un niveau inférieur à la valeur de cutback et la régulation normale reprend.

PropUnits

PropUnits sert à to definir si **Prop_Band** est défini comme un pourcentage de la plage du régulateur (PctSpan (0)) ou en unités techniques (EngUnts (1)).

Prop_Band

Prop_Band est la bande proportionnelle de l'algorithme de régulation PID. Si **PropUnits** est un pourcentage de l'étendue (PctSpan (0)), la bande proportionnelle est définie comme le pourcentage de l'étendue totale du régulateur pour lequel une erreur de régulation produit un signal de sortie équivalent à la sortie maximale de l'appareil. Par exemple, si **Span_High** est égal à 1000, **Span_Low** est égal à 0, Setpoint est égal à 500, **Process_Val** est égal à 490 et **Prop_Band** est égal à 1 %, une régulation proportionnelle seulement, action inverse produit une sortie de +100 % parce que l'erreur de régulation sera de 10 unités, c'est-à-dire une bande proportionnelle.

N.B. : lorsqu'il est défini sous la forme d'un pourcentage de l'étendue, le gain proportionnel est donné par :

$$\text{Gain} = \frac{10{,}000}{(\text{Span\_High} - \text{Span\_Low}) * \text{Prop\_Band}}$$

Si **PropUnits** est exprimé en unités techniques (EngUnts (1)), la bande proportionnelle est définie comme l'amplitude de l'erreur de régulation qui produit un signal de sortie équivalent à la sortie maximale de l'appareil. Par exemple, si **Setpoint** est égal à 500°C, **Process_Val** est égal à 495°C et **Prop_Band** est égal à 5°C, une régulation proportionnelle seulement, action inverse produit une sortie de +100 % parce que l'erreur de régulation est de 5°C, c'est-à-dire une bande proportionnelle.

Lorsqu'il est défini en unités techniques, le gain proportionnel est donné par :

$$\text{Gain} = \frac{100}{\text{Prop\_Band}}$$

Integral

Integral est la constante de temps d'intégrale de l'algorithme de régulation PID. Le temps d'intégrale (**Integral**) est défini comme la période de temps pendant laquelle la partie du signal de sortie due à l'action d'intégration augmente d'une quantité égale à la partie du signal de sortie due à l'action proportionnelle pour un état d'erreur constante.

Derivative

Derivative est la constante de temps de dérivée de l'algorithme de régulation PID. Le temps de dérivée (**Derivative**) est défini comme la période de temps pendant laquelle la partie du signal de sortie due à l'action de dérivation augmente d'une quantité égale à la partie du signal de sortie due à l'action proportionnelle lorsque l'erreur de régulation change à une vitesse constante.

Tune_Type

Tune_Type sert à réguler les différents algorithmes de réglage automatique et adaptatif existant dans le bloc fonction.

Pour avoir une description complète de ces algorithmes, se reporter au bloc fonction **PID_Auto**.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Ch1_Ch2_D_B	REAL	0	Oper	Oper	Limite haute Limite basse	10 -10
Ch1_Output	REAL	0	Oper		Limite haute Limite basse	100 Valeur la plus élevée : 0 ou Output_Low
Ch2_Output	REAL	0	Oper		Limite haute Limite basse	Valeur la plus faible : 0 ou Output_High -100
Cutback_High	REAL	0	Super	Super	Limite haute Limite basse	Span_High 0
Cutback_Low	REAL	0	Super	Super	Limite haute Limite basse	Span_High 0
Derivative	TIME	50s	Oper	Oper	Limite haute Limite basse	01d_03h 0
Integral	TIME	5m	Oper	Oper	Limite haute Limite basse	01d_03h 0
Manual	BOOL	Manual (1)	Oper	Oper	Sens	Auto (0) Manuel (1)
Manual_Reset	REAL	0	Oper	Oper	Limite haute Limite basse	100 -100
Output	REAL	0	Oper	Oper	Limite haute Limite basse	Output_High Output_Low
Output_High	REAL	100	Oper	Oper	Limite haute Limite basse	100 Output_Low
Output_Low	REAL	0	Oper	Oper	Limite haute Limite basse	Output_High -100
Process_Val	REAL	0	Oper	Oper	Limite haute Limite basse	Span_High Span_Low
Prop_Band	REAL	5 %	Oper	Oper	Limite haute Limite basse	10,000 0.1
PropUnits	BOOL	PctSpan(0)	Oper	Oper	Sens	PctSpan (0) EngUnits (1)

Table 9-23 Attributs des paramètres de PIDHeatCool

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Rel_Ch2_Gain	REAL	1	Oper	Oper	Limite haute Limite basse	10 0.1
Setpoint	REAL	0	Oper	Oper	Limite haute Limite basse	Span_High Span_Low
Span_High	REAL	100	Config	Config	Limite haute Limite basse	100,000 Span_Low
Span_Low	REAL	0	Config	Config	Limite haute Limite basse	Span_High - 100,000
Tune_Type	ENUM	None (0)	Config	Config	Sens	Néant (0) AT (1) DRA (2) LSAT (3) DRA_LS (4) AT_DRA (5) AT_LSAT (6) AT_D_L (7)

Table 9-23 Attributs des paramètres de PIDHeatCool (suite)

ANNEXE A - REGLAGES PID

Tableaux d'améliorations de Ziegler et Nichols

La présente annexe traite des différentes améliorations proposées pour le réglage des PID en utilisant la méthode du test de la courbe de contre-réaction en boucle ouverte et de la sensibilité maximale en boucle fermée. Les améliorations de Cohen et Coon pour le réglage des PID par la méthode de la courbe de contre-réaction sont données par le Tableau 1.

Type de régulation	Prop_Band	Intégrale	Dérivée
P	100KpNd/(1 + 0,35Nd)		
PI	100KpNd/(0,9 + 0,083Nd)	(3,3+0,31Nd)Dp/(1+2,2 Nd	
PID	100KpNd/(0,25Nd+1,35)	(2,5+0,46Nd)Dp/(1+0,6 1Nd)	0.37Dp (1+0,19Nd)
PD	100KpNd/(0,16Nd+1,24)		(0,27-0,088Nd)Dp/(1+0,13Nd)

Tableau 1 Réglage PID de Cohen et Coon par la courbe de contre-réaction

Shinskey [8] et Hang, Astrom et Ho [5] indiquent que pour obtenir un réglage manuel raisonnablement sûr, la méthode de la courbe de réaction associée à la méthode de la sensibilité maximale donne les meilleurs résultats. En bref, il apparaît une relation empirique entre le quotient du temps mort en boucle ouverte par la constante de temps, Nd = Dp/Tp, et le quotient du réglage de la bande

proportionnelle pendant le test de la sensibilité maximale par le gain du process en boucle ouverte, K = 100Kp/Pu. Elle est donnée par :

$$K = \frac{1(11N_d + 13)}{(37N_d - 4)}$$

Si K est supérieur à 1,5, le résultat du réglage des régulations PI et PID selon Ziegler et Nichols peut être mauvais. Trois cas sont possibles :

- si K < 1,5 : garder les valeurs de PID ;

- si 1,5 < K < 2,25 : régler le terme intégral Ti = 2KTu/9 avec une pondération du point de consigne de 8(4K/9 + 1)/17 ;

- si 2,25 < K, garder les valeurs de PID et régler le poids du point de consigne à 36/(27+5K).

Pour la régulation PI : si 1,2 < K < 15, régler PB = 6Pu(14K+15)/5(12+K) et Ti = Tu(4K+15)/75. Si nécessaire, pondérer le point de consigne comme dans le cas du PID.

Shinskey [8] a un point de vue très semblable. Il part du quotient de la période maximale sur le temps mort estimé, Tu/Dp. Quatre cas sont possibles :

Tu/Dp < 2 :	Le process est un temps mort total ;
2 < Tu/Dp < 4 :	Le process est à temps mort dominant ;
Tu/Dp=4 :	Capacité dominante unique (système de premier ordre + petit temps mort) ;
Tu/Dp > 4 :	Plus d'une capacité présente (retard).

Le Tableau 2 donne les améliorations des lois de Ziegler et Nichols pour quelques-uns de ces cas.

Si le process est instable en boucle ouverte, le réglage ne peut se faire raisonnablement qu'en mode manuel. Le Tableau 3 donne les recommandations de Shinskey pour un système instable en boucle ouverte avec un gain Kp, un temps mort Dp et une constante de temps Tp.

Régulation PI			Régulation PID		
	Tu/Dp	Ti/Tu	PB/Pu	Ti/Tu	Td/Tu
PB/Pu					
	2	0,25	2,35		No PID
	2,74	0,37	2,17	0,34	0,12
1,66					
	3,09	0,47	2,10	0,38	0,12
1,51					
	3,41	0,66	1,85	0,42	0,12
1,43					
	3,71	0,81	1,71	0,48	0,11
1,30					
	4,00	1,00	1,65	0,48	0,12
1,70					

Tableau 2 Réglages PI/PID à partir des tests en boucle ouverte et fermée : améliorations de Shinskey

Dp/Tp	TI/Dp	Td/Dp	PB
-0,10	1,70	0,6	11Kp
-0,20	1,90	0,60	20Kp
-0,50	2,00	0,80	56Kp
-0,67	2,25	0,90	77Kp
-0,80	2,40	1,00	96Kp

Tableau 3 Réglages PID pour systèmes de premier ordre INSTABLES avec retard

Bibliographie

[1] K.J. Astrom and B. Wittenmark, 1984, Computer Controlled Systems: Theory and Design, Prentice-Hall International.

[2] K.J. Astrom and B. Wittenmark, 1989, Adaptive Control, Addison Wesley Publishing Company.

[3] P.J. Gawthrop, P.E. Nomikos and L.S.P.S. Smith, 1990, Adaptive Temperature

[4] T. Hugglund, 1991, Process Control in Practice, Chartwell Bratt Ltd.

[5] C.C. Hang, K.J. Astrom and W.K. Ho, 1991, Refinements of Ziegler-Nichols tuning formula, IEE Proceedings Part D, Vol 138, No. 2.

[6] J.M. Maceijowski, 1989, Multivariable Feedback Design, Adison-Wesley Publication Company.

[7] M.Morari and E. Zafiriou, 1989, Robust Process Control, Prentice Hall Internaional.

[8] F.G. Shinskey, 1988, Process Control Systems, McGraw-Hill Book Co.

[9] J.G. Ziegler and N.B. Nichols, 1942, Optimum settings for automatic controllers, Transctions ASME, Vol 65, pp. 433-444.

Chapitre 10

TIMERS

Edition 1

Vue d'ensemble

Cette classe de blocs fonctions comporte une variété de Timers d'usage général qui fournissent des impulsions de sortie, des temporisations et des fonctions d'horloge.

BLOC FONCTION PULSE_TIMER

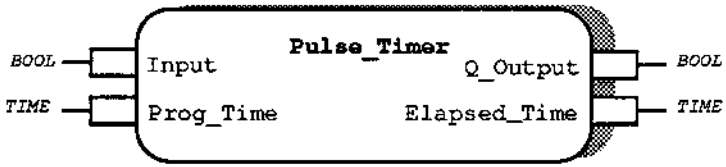
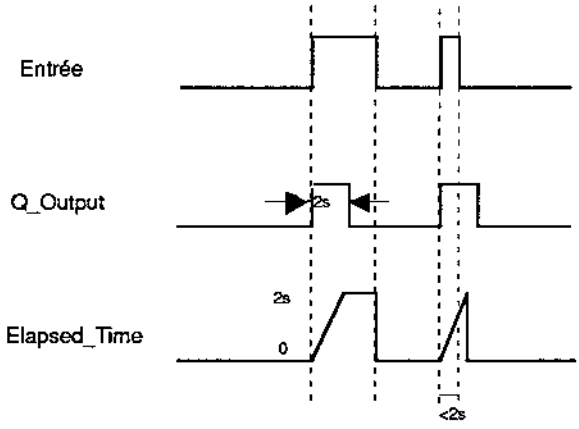


Figure 10-1 Schéma du bloc fonction Pulse_Time

Description fonctionnelle

Le bloc fonction Pulse_Timer (minuterie d'impulsion) fournit un impulsion de durée fixe sur réception d'un front d'entrée montant.

Sur le front montant du passage de Input (entrée) de "faux" (0) à "vrai" (1), la sortie Q_Output passe de "faux" (0) à "vrai" (1), et Elapsed_Time (temps écoulé) commence à s'incrémenter en temps réel. Lorsque Elapsed_Time a atteint Prog_Time (temps programmé), Q_Output repasse de "vrai" (1) à "faux" (0), et Elapsed_Time reste à sa valeur finale jusqu'à ce que Input redevienne "faux" (0). La remise à "faux" (0) de Input pendant la rampe de Elapsed_Time, n'a pas d'effet sur la durée de l'impulsion, mais lorsque la rampe est terminée, Elapsed_Time est remis à zéro.



Attributs du bloc fonction

Type : 28 10
Classe : ..... TIMERS
Tâche par défaut : ..... Task_1
Liste récapitulative : ..... Input, Prog_Time, Q_Output, Elapsed_Time
Besoins de capacité mémoire : 18 octets
Durée d'exécution : ..... 20,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	BOOL	Faux(0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Prog_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23 jours 0
Q_Output	BOOL	Faux(0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Elapsed_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23 jours 0

Tableau 10-1 Attributs des paramètres de Pulse_Timer

BLOC FONCTION ON_DELAY

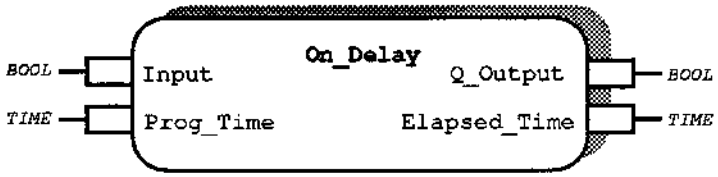
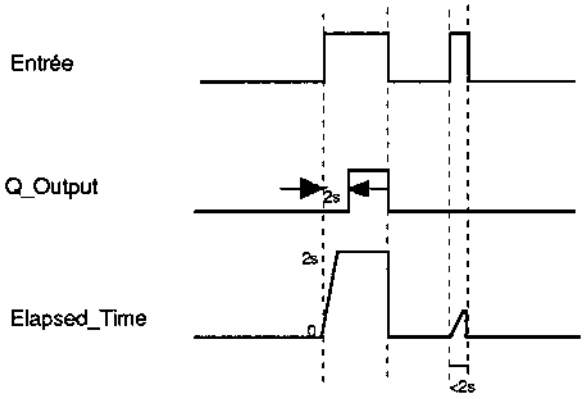


Figure 10-2 Schéma du bloc fonction On_Delay

Description fonctionnelle

Le bloc fonction On_Delay (retard à l'activation) fournit un retard fixe entre le passage de Input (entrée) à "vrai" (1) et le passage de Q_Output sortie à "vrai" (1). Quand Input repasse à "faux" (0), Q_Output repasse à "faux" (0) immédiatement.

Sur le passage de Input (entrée) de "faux" (0) à "vrai" (1), Elapsed_Time (temps écoulé) commence à s'incrémenter en temps réel. Lorsque Elapsed_Time est égal à Prog_Time (temps programmé), Q_Output passe à "vrai" (1), et Elapsed_Time est maintenu égal à Prog_Time. Si Input repasse à "faux" (0) à un moment quelconque, Q_Output repasse immédiatement à "faux" (0) et Elapsed_Time est remis à zéro.



Prog_Time = 2s

Attributs du bloc fonction

Type : 28 20
Classe : ..... TIMERS
Tâche par défaut : ..... Task_1
Liste récapitulative : ..... Input, Prog_Time, Q_Output, Elapsed_Time
Besoins de capacité mémoire : 18 octets
Durée d'exécution : ..... 17,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Elapsed_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23 jours 0
Input	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Prog_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23 jours 0
Q_Output	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)

Tableau 10-2 Attributs des paramètres de On_Delay

BLOC FONCTION OFF_DELAY

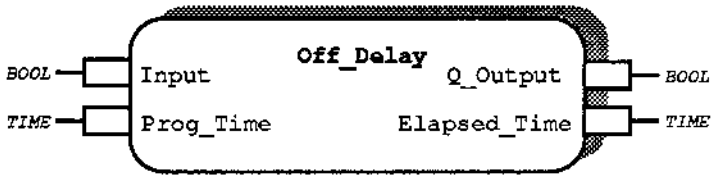
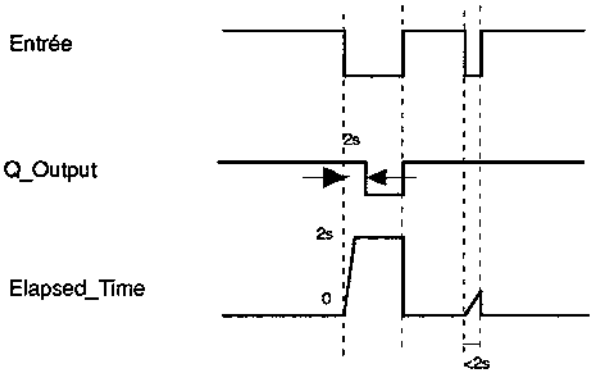


Figure 10-3 Schéma du bloc fonction Off_Delay

Description fonctionnelle

Le bloc fonction Off_Delay (retard à la désactivation) fournit un retard fixe entre le passage de Input (entrée) à "faux" (0) et le passage de Q_Output sortie à "faux" (0). Quand Input repasse à "vrai" (1), Q_Output repasse à "vrai" (1) immédiatement.

Sur le passage de Input (entrée) de "vrai" (1) à "faux" (0), Elapsed_Time (temps écoulé) commence à s'incrémenter en temps réel. Lorsque Elapsed_Time est égal à Prog_Time (temps programmé), Q_Output passe à "faux" (0), et Elapsed_Time est maintenu égal à Prog_Time. Si Input repasse à "vrai" (1) à un moment quelconque, Q_Output repasse immédiatement à "vrai" (1) et Elapsed_Time est remis à zéro.



Prog_Time = 2s

Attributs du bloc fonction

Type : 28 30
Classe : ..... TIMERS
Tâche par défaut : ..... Task_1
Liste récapitulative : ..... Input, Prog_Time, Q_Output, Elapsed_Time
Besoins de capacité mémoire : 18 octets
Durée d'exécution : ..... 17,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Prog_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23 jours 0
Q_Output	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Elapsed_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23 jours 0

Tableau 10-3 Attributs des paramètres de Off_Delay

BLOC FONCTION STOPWATCH

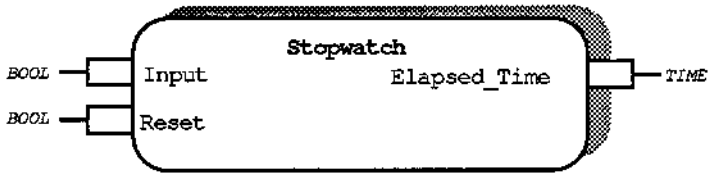


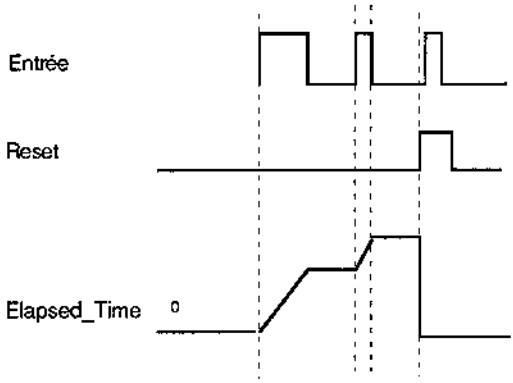
Figure 10-4 Schéma du bloc fonction Stopwatch

Description fonctionnelle

Le bloc fonction Stopwatch (chronomètre) dispose de deux entrées booléennes (Input et Reset) et d'une sortie temps (Elapsed_Time). Le bloc fonctionne comme un chronomètre classique. Lorsque Reset (réinitialisation) est Off (0), si Input est mis On (1), le chronomètre démarre, et si Input est mis Off (0) le chronomètre s'arrête. Si Reset est mis On (1), le chronomètre est remis à zéro et bloqué. Les modes de fonctionnement de Stopwatch sont résumés dans le tableau ci-dessous.

Input	Reset	Elapsed_Time
Off (0)	Off (0)	Maintenu à une valeur constante
On (1)	Off (0)	Incrémente en temps réel
Off (0)	On (1)	Remis à zéro et bloqué à zéro
On (1)	On (1)	Réinitialisation et maintien à zéro

Tableau 10-4 Modes de fonctionnement de Stopwatch



Attributs du bloc fonction

Type : 64 40
Classe : ..... TIMERS
Tâche par défaut : ..... Task_1
Liste récapitulative : ..... Input, Reset, Elapsed_Time
Besoins de capacité mémoire : 6 octets
Durée d'exécution : ..... 13,6 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	BOOL	Off(0)	Oper	Oper	Détect.	Off (0) On (1)
Reset	BOOL	Off(0)	Oper	Oper	Détect.	Off (0) On (1)
Elapsed_Time	TIME	0 ms	Oper	Block	Lim. haute Lim. basse	24d_21h_31m_23s 0

Tableau 10-5 Attributs des paramètres de Stopwatch

Chapitre 11

VARIABLE INTERNE

Edition 1

VUE D'ENSEMBLE

Sommaire (suite)

Vue d'ensemble

Cette classe comporte une gamme de blocs fonctions qui fournissent des variables internes pour une manipulation ultérieure ou pour le stockage intermédiaire de données. Une variété de données est compatible :

Bool (variables booléennes), Real (variables réelles), Integer (variables entières), Time (temps), Time_of_day (heure du jour), Date (date), Date_and_time (date et heure), et String (chaîne de caractères).

Long_String est un cas particulier de String (chaîne) dans lequel la longueur maximale est passée de 80 à 255 caractères.

BLOC FONCTION BOOL

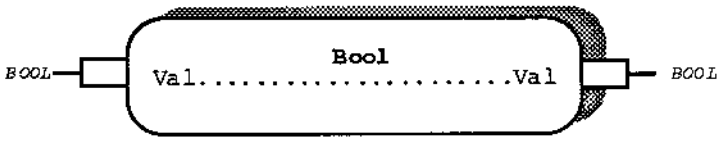


Figure 11-1 Schéma du bloc fonction Bool

Description fonctionnelle

Le bloc fonction Bool (booléen) est une variable interne, qui peut servir pour le stockage de valeurs intermédiaires du programme utilisateur. La valeur booléenne est maintenue dans le paramètre d'entrée / sortie Val.

Attributs du bloc fonction

Type : 56 16

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Val

Besoins de capacité mémoire : 2 octets

Durée d'exécution : ..... 17,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)

Tableau 11-1 Attributs des paramètres de Bool

BLOC FONCTION REAL

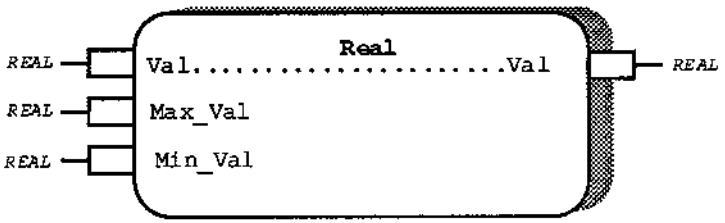


Figure 11-2 Schéma du bloc fonction Real

Description fonctionnelle

Le bloc fonction Real (réel) est une variable interne qui peut servir pour la définition de valeurs intermédiaires dans le programme utilisateur. La valeur réelle est maintenue dans le paramètre d'entrée / sortie Val.

Les paramètres de limite Max_Val et Min_Val sont utilisés pour fixer des limites aux valeurs maximale et minimale qui peuvent être données au bloc fonction lorsqu'il fait l'objet d'un accès par communication série externe.

Les valeurs entrées dans le bloc en Grafcet ou venant d'autres blocs par câblage utilisateur ne sont pas limitées par les paramètres Max_Val et Min_Val.

Attributs du bloc fonction

Type : 56 32

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val, Max_Val, Min_Val

Besoins de capacité mémoire : 12 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	**REAL**	0.0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val Défini par Min_Val
Max_Val	**REAL**	1000,0	Oper	Oper	Lim. haute Lim. basse	1 000 000 Défini par Min_Val
Min_Val	**REAL**	-1000,0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val -1 000 000

Tableau 11-2 Attributs des paramètres de Real

BLOC FONCTION INTEGER

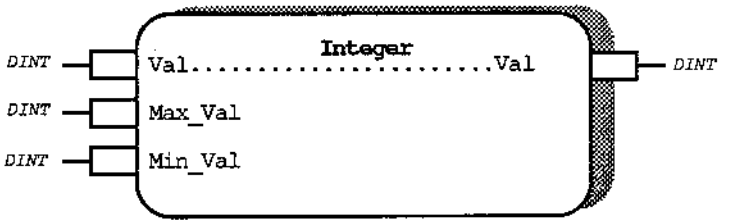


Figure 11-3 Schéma du bloc fonction Integer

Description fonctionnelle

Le bloc fonction Integer (entier) est une variable interne, qui peut servir pour la définition de valeurs intermédiaires du programme utilisateur. La valeur entière est maintenue dans le paramètre d'entrée / sortie Val.

Les paramètres de limite Max_Val et Min_Val sont utilisés pour les valeurs maximale et minimale qui peuvent être données au bloc fonction lorsqu'il fait l'objet d'un accès par communication série externe.

Les valeurs entrées dans le bloc en Grafcet ou venant d'autres blocs par câblage utilisateur ne sont pas limitées par les paramètres Max_Val et Min_Val.

Attributs du bloc fonction

Type : 38 30

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val, Max_Val, Min_Val

Besoins de capacité mémoire : 12 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	DINT	0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val Défini par Min_Val
Max_Val	DINT	1000	Oper	Oper	Lim. haute Lim. basse	1 000 000 Défini par Min_Val
Min_Val	DINT	-1000	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val -1 000 000

Tableau 11-3 Attributs des paramètres de Integer

BLOC FONCTION TIME

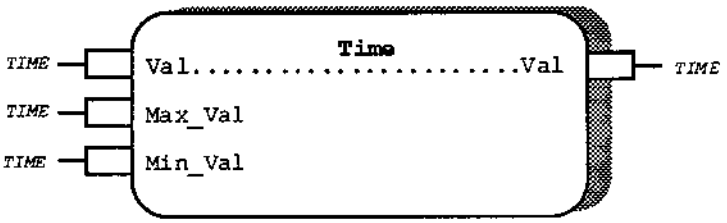


Figure 11-4 Schéma du bloc fonction Time

Description fonctionnelle

Le bloc fonction Time (temps) est une variable interne, qui peut servir pour la définition de valeurs intermédiaires du programme utilisateur. La valeur du temps est maintenue dans le paramètre d'entrée / sortie Val.

Les paramètres de limite Max_Val et Min_Val sont utilisés pour les valeurs maximale et minimale qui peuvent être données au bloc fonction Time lorsqu'il fait l'objet d'un accès par communication série externe.

Les valeurs entrées dans le bloc en Grafcet ou venant d'autres blocs par câblage utilisateur ne sont pas limitées par les paramètres Max_Val et Min_Val.

Attributs du bloc fonction

Type : 38 40

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val, Max_Val, Min_Val

Besoins de capacité mémoire : 12 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	TIME	0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val Défini par Min_Val
Max_Val	TIME	1 d	Oper	Oper	Lim. haute Lim. basse	23d 23h 59m 59s Défini par Min_Val
Min_Val	TIME	0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val 0

Tableau 11-4 Attributs des paramètres de Time

BLOC FONCTION TIME_OF_DAY

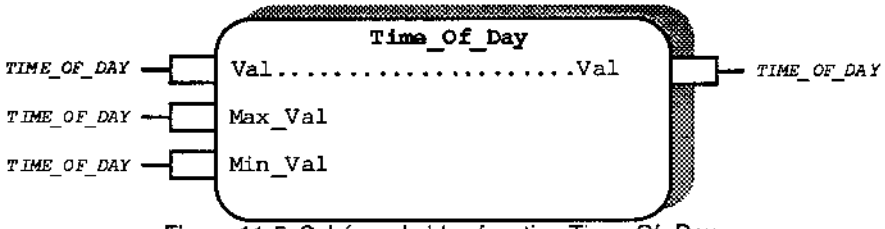


Figure 11-5 Schéma du bloc fonction Time_Of_Day

Description fonctionnelle

Le bloc fonction Time_Of_Day (heure du jour) est une variable interne, qui peut s'utiliser pour la définition de valeurs intermédiaires du programme utilisateur. L'heure du jour est maintenue dans le paramètre d'entrée / sortie Val. La limite supérieure de Val est fixée par Max_Val, et la limite inférieure de Val est fixée par Min_Val.

Attributs du bloc fonction

Type : 38 50

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val, Max_Val, Min_Val

Besoins de capacité mémoire : 12 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	TIME_OF_DAY	0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val Défini par Min_Val
Max_Val	TIME_OF_DAY	23:59:59	Oper	Oper	Lim. haute Lim. basse	23:59:59 Défini par Min_Val
Min_Val	TIME_OF_DAY	0	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val 0

Tableau 11-5 Attributs des paramètres de Time_Of_Day

BLOC FONCTION DATE

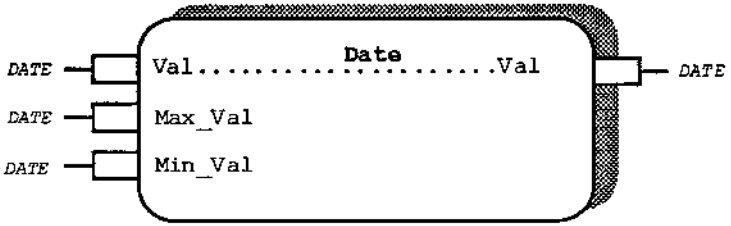


Figure 11-6 Schéma du bloc fonction Date

Description fonctionnelle

Le bloc fonction Date (date) est une variable interne, qui peut s'utiliser pour la définition de valeurs intermédiaires du programme utilisateur. La date est maintenue dans le paramètre d'entrée / sortie Val.

Les paramètres de limite Max_Val et Min_Val sont utilisés pour les valeurs maximale et minimale qui peuvent être données au bloc fonction lorsqu'il fait l'objet d'un accès par communication série externe.

Les valeurs entrées dans le bloc en Grafcet ou venant d'autres blocs par câblage utilisateur ne sont pas limitées par les paramètres Max_Val et Min_Val.

Attributs du bloc fonction

Type : 38 60

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val, Max_Val, Min_Val

Besoins de capacité mémoire : 12 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	DATE	01-Jan-1970	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val Défini par Min_Val
Max_Val	DATE	31-Déc-2032	Oper	Oper	Lim. haute Lim. basse	31-Déc-2032 Défini par Min_Val
Min_Val	DATE	01-Jan-1970	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val 01-Jan-1970

Tableau 11-6 Attributs des paramètres de Date

BLOC FONCTION DATEANDTIME

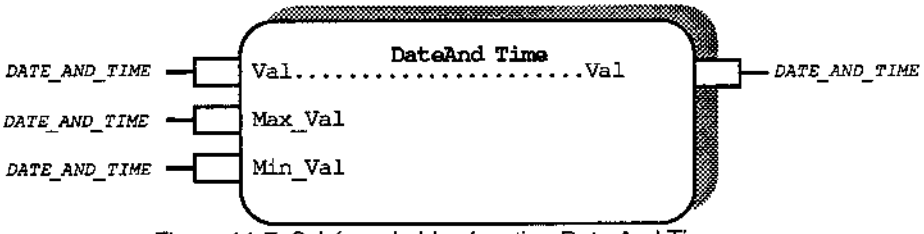


Figure 11-7 Schéma du bloc fonction Date And Time

Description fonctionnelle

Le bloc fonction Date And Time (date et heure) est une variable interne, qui peut s'utiliser pour le stockage de valeurs intermédiaires du programme utilisateur. La date et l'heure sont maintenues dans le paramètre d'entrée / sortie Val.

Les paramètres de limite Max_Val et Min_Val sont utilisés pour les valeurs maximale et minimale qui peuvent être données au bloc fonction Date And Time lorsqu'il fait l'objet d'un accès par communication série externe.

Les valeurs entrées dans le bloc en Grafcet ou venant d'autres blocs par câblage utilisateur ne sont pas limitées par les paramètres Max_Val et Min_Val.

Attributs du bloc fonction

Type : 38 70

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val, Max_Val, Min_Val

Besoins de capacité mémoire : 12 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	DATE_AND_TIME	01-Jan-1970-00:00:00	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val Défini par Min_Val
Max_Val	DATE_AND_TIME	31-Déc-2032-23:59:59	Oper	Oper	Lim. haute Lim. basse	31-Déc-2032 23:59:59 Défini par Min_Val
Min_Val	DATE_AND_TIME	01-Jan-1970-00:00:00	Oper	Oper	Lim. haute Lim. basse	Défini par Max_Val 01-Jan-1970-00:00:00

Tableau 11-7 Attributs des paramètres de Date and Time

BLOC FONCTION STRING

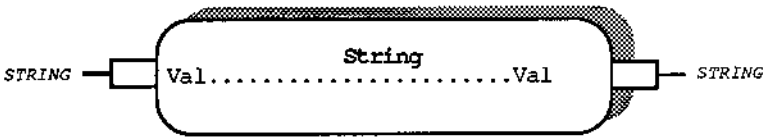


Figure 11-8 Schéma du bloc fonction String

Description fonctionnelle

Le bloc fonction String (chaîne de caractères) est une variable interne, qui peut servir pour la définition de valeurs intermédiaires du programme utilisateur. La valeur de chaîne, d'une longueur de 80 caractères maximum, est maintenue dans le paramètre d'entrée / sortie Val.

Attributs du bloc fonction

Type : 38 80

Classe : ..... VARIABLE INTERNE

Tâche par défaut :..... Task_2

Liste récapitulative : ..... Val

Besoins de capacité mémoire : 82 octets

Durée d'exécution : ..... 68,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	STRING	' '	Oper	Oper	S/O	S/O

Tableau 11-8 Attributs des paramètres de String

BLOC FONCTION LONG_STRING

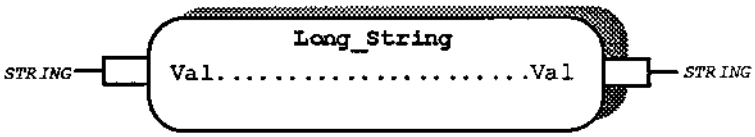


Figure 11-9 Schéma du bloc fonction Long_String

Description fonctionnelle

Le bloc fonction Long String (longue chaîne de caractères) est une variable interne, qui peut servir pour la définition de valeurs intermédiaires du programme utilisateur. La valeur de chaîne, d'une longueur de 255 caractères maximum, est maintenue dans le paramètre d'entrée / sortie Val.

Attributs du bloc fonction

Type : 38 81

Classe : ..... VARIABLE INTERNE

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Val

Besoins de capacité mémoire : 258 octets

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Val	STRING	''	Oper	Oper	S/O	S/O

Tableau 11-9 Attributs des paramètres de Long_String

Chapitre 12

VARIABLES D'ELEMENTS DEPORTES

Edition 1

Sommaire (suite)

VUE D'ENSEMBLE

Les paramètres REMOTE (déportés) sont utilisés en association avec un pilote de communication MAITRE approprié. Ils sont utilisés lorsque le PC3000 est l'équipement MAITRE. Les blocs sont nommés ainsi parce que les données appartenant au dispositif de communication esclave apparaissent comme étant déportées par rapport au PC3000. Les blocs peuvent être considérés comme étant des adresses fixes, ou "boîtes aux lettres" dans lesquelles des données peuvent être écrites ou lues par le PC3000. Les blocs permettent le diagnostic, la commande lecture / écriture, l'horodatage, etc.

Les paramètres REMOTE à éléments multiples permettent une utilisation plus efficace de la mémoire, étant donné qu'un bloc peut être utilisé pour commander des transactions associées à différentes valeurs. De plus, elles gagnent en vitesse d'initialisation, étant donné qu'une seule adresse a besoin d'être indiquée pour son démarrage à froid, alors qu'il en faut plusieurs lorsque l'on utilise des paramètres individuels tels que :

Boolean (booléen), Real (réel), Integer (entier), Time (temps), String (chaîne) and Status_Word (mot d'état).

Paramètres déportés
Rmt_Bool
Rmt_Real
Rmt_Dint
Rmt_Time
Rmt_Str
Rmt_Bool_8
Rmt_Real_8
Rmt_Dint_64
Rmt_Bool_64
Rmt_Real_64
Rmt_Dint_64
Rmt_SW

Cette classe est utilisée en association avec les blocs fonctions de la classe COMMS (communication).

Pour plus de détail, se reporter à la Vue d'ensemble de la Communication du PC3000, au chapitre 3.

Nota 1: Avant la diffusion de la version 2.27 du progiciel système, ces blocs étaient appelés Remote_****. La version 2.27 a ajouté un certain nombre de paramètre multi-éléments nouveaux, et tous les blocs de cette classe ont été renommés. Tous les blocs sont maintenant identifiés en tant que Rmt_**** , dans lesquels **** peut être BOOL, REAL, DINT ou STR.

Ceci n'a aucune influence sur les programmes écrits avec les versions précédentes.

Nota 2: Tous les protocoles ne sont pas forcément compatibles avec les paramètres déportés multi-éléments. Pour la version 2.27 l'utilisation est la suivante :

Commande	Support de paramètre multi-élément
EI_Bisync_M	3
Raw_Comms	S/O
JBus_M	8
Siemens_M_S	8
Toshiba_M	8

BLOC FONCTION RMT_BOOL

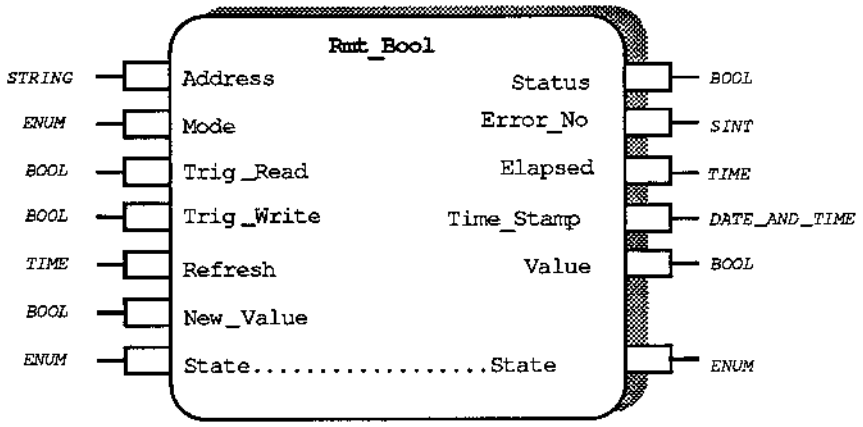


Figure 12-1 Schéma du bloc fonction de la variable Rmt_Bool

Description fonctionnelle

Le bloc fonction de la variable Rmt_Bool permet l'écriture / lecture de la valeur d'un paramètre booléen via un port de communication. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'avec des ports qui sont compatibles avec des protocoles fonctionnant en mode Maître.

La description détaillée de l'utilisation des blocs fonctions Rmt_Vars est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000.

Attributs du bloc fonction

Type : 3A 10

Classe : ..... VARIABLE D'ELEMENTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, New_Value, Mode, State

Besoins de capacité mémoire : 88 octets

Durée d'exécution : ..... 128 µs

Description des paramètres

Address (A)

Ce paramètre est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Le premier caractère de Address désigne le module PC3000 sur lequel le port de communications est situé. Le second caractère désigne le port du module. Le reste de l'adresse est spécifique au protocole, et sert à identifier l'instrument et le paramètre qui est en communication avec le bloc fonction. Pour plus de détails concernant le format d'adresse, se reporter au bloc fonction de communications maître associé.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Quatre valeurs sont possibles :

Demand (0) :	En mode Demand, une lecture / écriture unique peut être déclenchée en passant le paramètre d'état de OK (0) à Read (4) ou Write (3).
R_Cont (1) :	En mode R_Cont, la variable déportée est constamment invitée à émettre à intervalles réguliers définis par Refresh (actualisation).
W_Cont (2) :	En mode W_Cont, la variable déportée est constamment écrite, à intervalles réguliers définis par Refresh (actualisation).
Change (3) :	En mode Change, la variable déportée est écrite lorsque la valeur de l'entrée New_Value (nouvelle valeur) ne coïncide plus avec Value. La période minimale entre écritures de changements dans le dispositif déporté est défini par Refresh (actualisation).

Trig_Read (TR)

Lorsque le bloc fonction est en mode Demand, si Trig_Read passe de Off (0) à On (1), une lecture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une lecture à partir du programme.

Trig_Write (TW)

Lorsque le bloc fonction est en mode Demand, si Trig_Write passe de Off (0) à On (1), une écriture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une écriture à partir du programme.

Refresh (R)

Refresh (actualisation) définit le délai de mise à jour du paramètre, lorsque le bloc fonction est en mode R_Cont ou W_Cont.

New_Value (NV)

New_Value est la valeur à écrire dans la variable déportée lorsque le bloc fonction est en mode écriture.

State (S)

Le paramètre State indique l'état actuel du bloc fonction. Cinq valeurs sont possibles :

Ok (0) :	Aucune action de communication n'est en cours
Pending (1) :	Une transaction a démarré et n'est pas encore terminée
Error (2) :	Une transaction de communication a échoué. Le paramètre Error_No fournit un indicateur de diagnostic sur le type d'erreur
Write (3) :	La mise à Write (3) de State déclenche un écriture unique de New_Value pour l'équipement déporté
Read (4) :	La mise à Read (4) de State déclenche une lecture unique du paramètre déporté

Quand State est mis à Write (3) ou Read (4), il repasse automatiquement à OK (0) si la transmission a été réussie, ou reste à l'état Error (2).

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au driver de communication Maître approprié, au chapitre 3, en ce qui concerne les codes d'erreur.

Elapsed (E)

Elapsed indique le temps déjà écoulé depuis la dernière transaction de lecture ou d'écriture. Elapsed n'est valide que lorsque le bloc fonction est en mode R_Cont ou W_Cont.

Time_Stamp (TS)

Time_Stamp définit l'heure et la date de la dernière transaction de lecture ou d'écriture effectuée avec succès. Son activation coïncide avec le passage du paramètre State à Ok (0).

Value (VAL)

Le paramètre Value contient la dernière valeur qui a été écrite ou lue avec succès dans la variable déportée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Elapsed	TIME	0	Oper			
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Demand (0)	Oper	Super	Détect.	Demand (0) R_Cont (1) W_Cont (2) Change (3)
New_Value	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Refresh	TIME	10 s	Oper	Super	Lim. haute Lim. basse	19d_59m_59s 100 ms
State	ENUM	Ok (0)	Oper	Oper	Détect.	Ok (0) Pending (1) Error (2) Write (3) Read (4)
Status	BOOL	Go (1)	Oper		Détect.	NOGO (0) Go (1)
Time_Stamp	DATE AND TIME	0	Oper			
Trig_Read	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Trig_Write	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	BOOL	Off (0)	Oper		Détect.	Off (0) On (1)

Tableau 12-1 Attributs des paramètres de Remote_Bool

BLOC FONCTION RMT_REAL

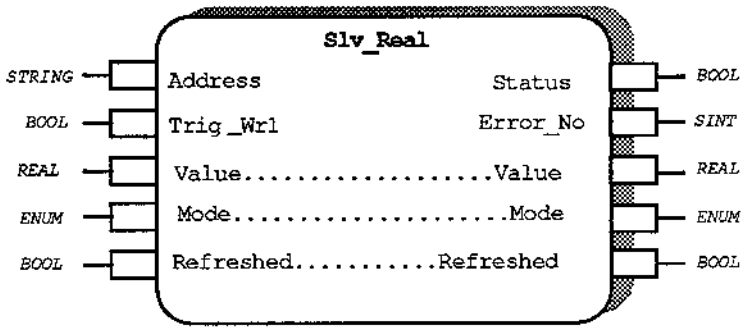


Figure 12-2 Schéma du bloc fonction de la variable Rmt_Real

Description fonctionnelle

Le bloc fonction de la variable Rmt_Real permet la lecture et/ou l'écriture de la valeur d'un paramètre réel, via un port de communication. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'avec des ports qui sont compatibles avec des protocoles fonctionnant en mode Maître.

Une description complète de l'utilisation des blocs fonctions Rmt_Vars est donnée au chapitre 3, Vue d'ensemble des communications PC3000.

Attributs du bloc fonction

Type : 3A 20

Classe : ..... VARIABLES D'ELEMENTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, New_Value, Mode, State

Besoins de capacité mémoire : . 98 octets

Durée d'exécution : ..... 129 µs

Description des paramètres

Address (A)

Ce paramètre est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Le premier caractère de Address désigne le module PC3000 sur lequel le port de communications est situé. Le second caractère désigne le port du module. Le reste de l'adresse est spécifique au protocole, et sert à identifier l'instrument et le paramètre qui est en communication avec le bloc fonction.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Quatre valeurs sont possibles :

Demand (0) :	En mode Demand, une lecture / écriture unique peut être déclenchée en passant le paramètre d'état de OK (0) à Read (4) ou Write (3).
R_Cont (1) :	En mode R_Cont, la variable déportée est constamment invitée à émettre à intervalles réguliers définis par Refresh (actualisation).
W_Cont (2) :	En mode W_Cont, la variable déportée est constamment écrite, à intervalles réguliers définis par Refresh (actualisation).
Change (3) :	En mode Change, la variable déportée est écrite lorsque la valeur de l'entrée New_Value (nouvelle valeur) ne coïncide plus avec Value. La période minimale entre écritures de changements dans le dispositif déporté est défini par Refresh (actualisation).

Trig_Read (TR)

Lorsque le bloc fonction est en mode Demand, si Trig_Read passe de Off (0) à On (1), une lecture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une lecture à partir du programme.

Trig_Write (TW)

Lorsque le bloc fonction est en mode Demand, si Trig_Write passe de Off (0) à On (1), une écriture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une écriture à partir du programme.

Refresh (R)

Refresh (actualisation) définit le délai de mise à jour du paramètre, lorsque le bloc fonction est en mode R_Cont ou W_Cont.

New_Value (NV)

New_Value est la valeur à écrire dans la variable déportée lorsque le bloc fonction est en mode écriture.

State (S)

Le paramètre State indique l'état actuel du bloc fonction. Cinq valeurs sont possibles :

Ok (0) :	Aucune action de communication n'est en cours
Pending (1) :	Une transaction a démarré et n'est pas encore terminée
Error (2) :	Une transaction de communication a échoué. Le paramètre Error_No fournit un indicateur de diagnostic sur le type d'erreur
Write (3) :	La mise à Write (3) de State déclenche un écriture unique de New_Value pour l'équipement déporté
Read (4) :	La mise à Read (4) de State déclenche une lecture unique du paramètre déporté

Quand State est mis à Write (3) ou Read (4), il repasse automatiquement à OK (0) si la transmission a été réussie, ou reste à l'état Error (2).

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication Maître approprié, au chapitre 3, en ce qui concerne les codes d'erreur.

Elapsed (E)

Elapsed indique le temps déjà écoulé depuis la dernière transaction de lecture ou d'écriture. Elapsed n'est valide que lorsque le bloc fonction est en mode R_Cont ou W_Cont.

Time_Stamp (TS)

Time_Stamp définit l'heure et la date de la dernière transaction de lecture ou d'écriture effectuée avec succès. Son activation coïncide avec le passage du paramètre State à Ok (0).

Value (VAL)

Le paramètre Value contient la dernière valeur qui a été écrite ou lue avec succès dans la variable déportée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Elapsed	TIME	0	Oper			
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Demand (0)	Oper	Super	Détect.	Demand (0) R_Cont (1) W_Cont (2) Change (3)
New_Value	REAL	0	Oper	Oper	Lim. haute Lim. basse	10 000 000 -10 000 000
Refresh	TIME	10s	Oper	Super	Lim. haute Lim. basse	19d_59m_59s 100 ms
State	ENUM	Ok (0)	Oper	Oper	Détect.	Ok (0) Pending (1) Error (2) Write (3) Read (4)
Status	BOOL	Go (1)	Oper		Détect.	NOGO (0) Go (1)
Time_Stamp	DATE AND TIME	0	Oper			
Trig_Read	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Trig_Write	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	REAL	0	Oper		Lim. haute Lim. basse	10 000 000 -10 000 000

Tableau 12-2 Attributs des paramètres de la variable Remote_Real

BLOC FONCTION RMT_INT

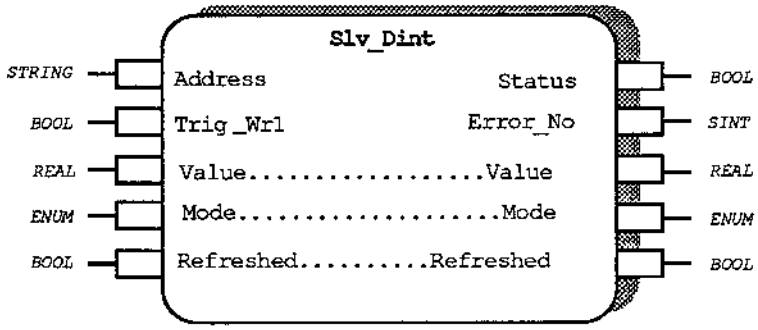


Figure 12-3 Schéma du bloc fonction de la variable Rmt_Int

Description fonctionnelle

Le bloc fonction de la variable Rmt_Int permet la lecture et/ou l'écriture de la valeur d'une variable entière sur 32 bits, via un port de communication. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'avec des ports qui sont compatibles avec des protocoles fonctionnant en mode Maître.

Une description complète de l'utilisation des blocs fonctions Rmt_Vars est donnée au chapitre 3, Vue d'ensemble des communications PC3000.

Attributs du bloc fonction

Type : 3A 30

Classe : ..... VARIABLE D'ELEMENTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, New_Value, Mode, State

Besoins de capacité mémoire : .98 octets

Durée d'exécution : ..... 124 µs

Description des paramètres

Address (A)

Ce paramètre est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Le premier caractère de Address désigne le module PC3000 sur lequel le port de communications est situé. Le second caractère désigne le port du module. Le reste de l'adresse est spécifique au protocole, et sert à identifier l'instrument et le paramètre qui est en communication avec le bloc fonction.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Quatre valeurs sont possibles :

Demand (0) : En mode Demand, une lecture / écriture unique peut être déclenchée en passant le paramètre d'état de OK (0) à Read (4) ou Write (3).

R_Cont (1) : En mode R_Cont, la variable déportée est constamment invitée à émettre à intervalles réguliers définis par Refresh (actualisation).

W_Cont (2) : En mode W_Cont, la variable déportée est constamment écrite, à intervalles réguliers définis par Refresh (actualisation).

Change (3) : En mode Change, la variable déportée est écrite lorsque la valeur de l'entrée New_Value (nouvelle valeur) ne coïncide plus avec Value. La période minimale entre écritures de changements dans le dispositif déporté est défini par Refresh (actualisation).

Trig_Read (TR)

Lorsque le bloc fonction est en mode Demand, si Trig_Read passe de Off (0) à On (1), une lecture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une lecture à partir du programme.

Trig_Write (TW)

Lorsque le bloc fonction est en mode Demand, si Trig_Write passe de Off (0) à On (1), une écriture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une écriture à partir du programme.

Refresh (R)

Refresh (actualisation) définit le délai de mise à jour du paramètre, lorsque le bloc fonction est en mode R_Cont ou W_Cont.

New_Value (NV)

New_Value est la valeur à écrire dans la variable déportée lorsque le bloc fonction est en mode écriture.

State (S)

Le paramètre State indique l'état actuel du bloc fonction. Cinq valeurs sont possibles :

Ok (0) :	Aucune action de communication n'est en cours
Pending (1) :	Une transaction a démarré et n'est pas encore terminée
Error (2) :	Une transaction de communication a échoué. Le paramètre Error_No fournit un indicateur de diagnostic sur le type d'erreur
Write (3) :	La mise à Write (3) de State déclenche un écriture unique de New_Value pour l'équipement déporté
Read (4) :	La mise à Read (4) de State déclenche une lecture unique du paramètre déporté

Quand State est mis à Write (3) ou Read (4), il repasse automatiquement à OK (0) si la transmission a été réussie, ou reste à l'état Error (2).

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication Maître approprié, au chapitre 3, en ce qui concerne les codes d'erreur.

Elapsed (E)

Elapsed indique le temps déjà écoulé depuis la dernière transaction de lecture ou d'écriture. Elapsed n'est valide que lorsque le bloc fonction est en mode R_Cont ou W_Cont.

Time_Stamp (TS)

Time_Stamp définit l'heure et la date de la dernière transaction de lecture ou d'écriture effectuée avec succès. Son activation coïncide avec le passage du paramètre State à Ok (0).

Value (VAL)

Le paramètre Value contient la dernière valeur qui a été écrite ou lue avec succès dans la variable déportée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Elapsed	TIME	0	Oper			
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Demand (0)	Oper	Super	Détect.	Demand (0) R_Cont (1) W_Cont (2) Change (3)
New_Value	DINT	0	Oper	Oper	Lim. haute Lim. basse	10 000 000 -10 000 000
Refresh	TIME	10 s	Oper	Super	Lim. haute Lim. basse	19d_59m_59s 100 ms
State	ENUM	Ok (0)	Oper	Oper	Détect.	Ok (0) Pending (1) Error (2) Write (3) Read (4)
Status	BOOL	Go (1)	Oper		Détect.	NOGO (0) Go (1)
Time_Stamp	DATE AND TIME	0	Oper			
Trig_Read	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Trig_Write	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	DINT	0	Oper		Lim. haute Lim. basse	10 000 000 -10 000 000

Tableau 12-3 Attributs des paramètres de la variable Remote_Int

BLOC FONCTION RMT_STR

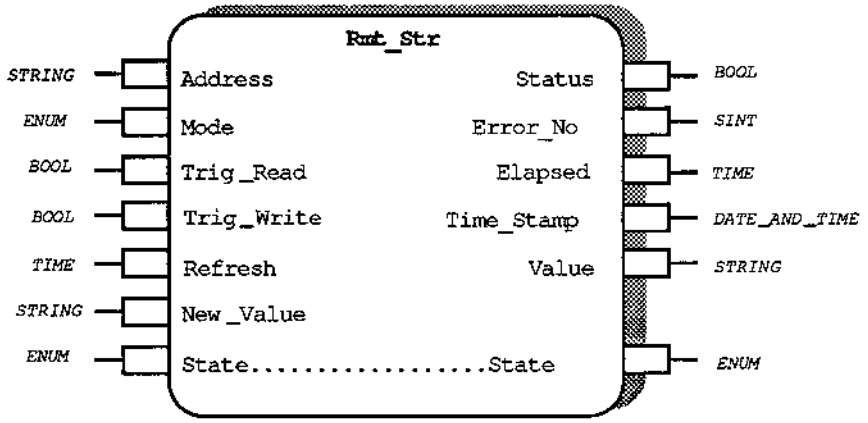


Figure 12-4 Schéma du bloc fonction de la variable Rmt_Str

Description fonctionnelle

Le bloc fonction de la variable Rmt_Str permet la lecture et/ou l'écriture de la valeur d'une variable de chaîne de caractères, via un port de communication. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'avec des ports qui sont compatibles avec des protocoles fonctionnant en mode Maître.

Une description complète de l'utilisation des blocs fonctions Rmt_Vars est donnée au chapitre 3, Vue d'ensemble des communications PC3000.

Attributs du bloc fonction

Type : 3A 80

Classe : .....VARIABLES D'ELEMENTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, New_Value, Mode, State

Besoins de capacité mémoire : .474 octets

Durée d'exécution : .....316 µs

Description des paramètres

Address (A)

Ce paramètre est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Le premier caractère de Address désigne le module PC3000 sur lequel le port de communications est situé. Le second caractère désigne le port du module. Le reste de l'adresse est spécifique au protocole, et sert à identifier l'instrument et le paramètre qui est en communication avec le bloc fonction.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Quatre valeurs sont possibles :

Demand (0) :	En mode Demand, une lecture / écriture unique peut être déclenchée en passant le paramètre d'état de OK (0) à Read (4) ou Write (3).
R_Cont (1) :	En mode R_Cont, la variable déportée est constamment invitée à émettre à intervalles réguliers définis par Refresh (actualisation).
W_Cont (2) :	En mode W_Cont, la variable déportée est constamment écrite, à intervalles réguliers définis par Refresh (actualisation).
Change (3) :	En mode Change, la variable déportée est écrite lorsque la valeur de l'entrée New_Value (nouvelle valeur) ne coïncide plus avec Value. La période minimale entre écritures de changements dans le dispositif déporté est défini par Refresh (actualisation).

Trig_Read (TR)

Lorsque le bloc fonction est en mode Demand, si Trig_Read passe de Off (0) à On (1), une lecture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une lecture à partir du programme.

Trig_Write (TW)

Lorsque le bloc fonction est en mode Demand, si Trig_Write passe de Off (0) à On (1), une écriture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une écriture à partir du programme.

Refresh (R)

Refresh (actualisation) définit le délai de mise à jour du paramètre, lorsque le bloc fonction est en mode R_Cont ou W_Cont.

New_Value (NV)

New_Value est la valeur à écrire dans la variable déportée lorsque le bloc fonction est en mode écriture.

State (S)

Le paramètre State indique l'état actuel du bloc fonction. Cinq valeurs sont possibles :

Ok (0) :	Aucune action de communication n'est en cours
Pending (1) :	Une transaction a démarré et n'est pas encore terminée
Error (2) :	Une transaction de communication a échoué. Le paramètre Error_No fournit un indicateur de diagnostic sur le type d'erreur
Write (3) :	La mise à Write (3) de State déclenche un écriture unique de New_Value pour l'équipement déporté
Read (4) :	La mise à Read (4) de State déclenche une lecture unique du paramètre déporté

Quand State est mis à Write (3) ou Read (4), il repasse automatiquement à OK (0) si la transmission a été réussie, ou reste à l'état Error (2).

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication Maître approprié, au chapitre 3, en ce qui concerne les codes d'erreur.

Elapsed (E)

Elapsed indique le temps déjà écoulé depuis la dernière transaction de lecture ou d'écriture. Elapsed n'est valide que lorsque le bloc fonction est en mode R_Cont ou W_Cont.

Time_Stamp (TS)

Time_Stamp définit l'heure et la date de la dernière transaction de lecture ou d'écriture effectuée avec succès. Son activation coïncide avec le passage du paramètre State à Ok (0).

Value (VAL)

Le paramètre Value contient la dernière valeur qui a été écrite ou lue avec succès dans la variable déportée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Elapsed	TIME	0	Oper			
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Demand (0)	Oper	Super	Détect.	Demand (0) R_Cont (1) W_Cont (2) Change (3)
New_Value	STRING		Oper	Oper		
Refresh	TIME	10 s	Oper	Super	Lim. haute Lim. basse	19d_59m_59s 100 ms
State	ENUM	Ok (0)	Oper	Oper	Détect.	Ok (0) Pending (1) Error (2) Write (3) Read (4)
Status	BOOL	Go (1)	Oper		Détect.	NOGO (0) Go (1)
Time_Stamp	DATE AND TIME	0	Oper			
Trig_Read	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Trig_Write	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	STRING		Oper			

Tableau 12-4 Attributs des paramètres de la variable Rmt_Str

BLOC FONCTION RMT_SW

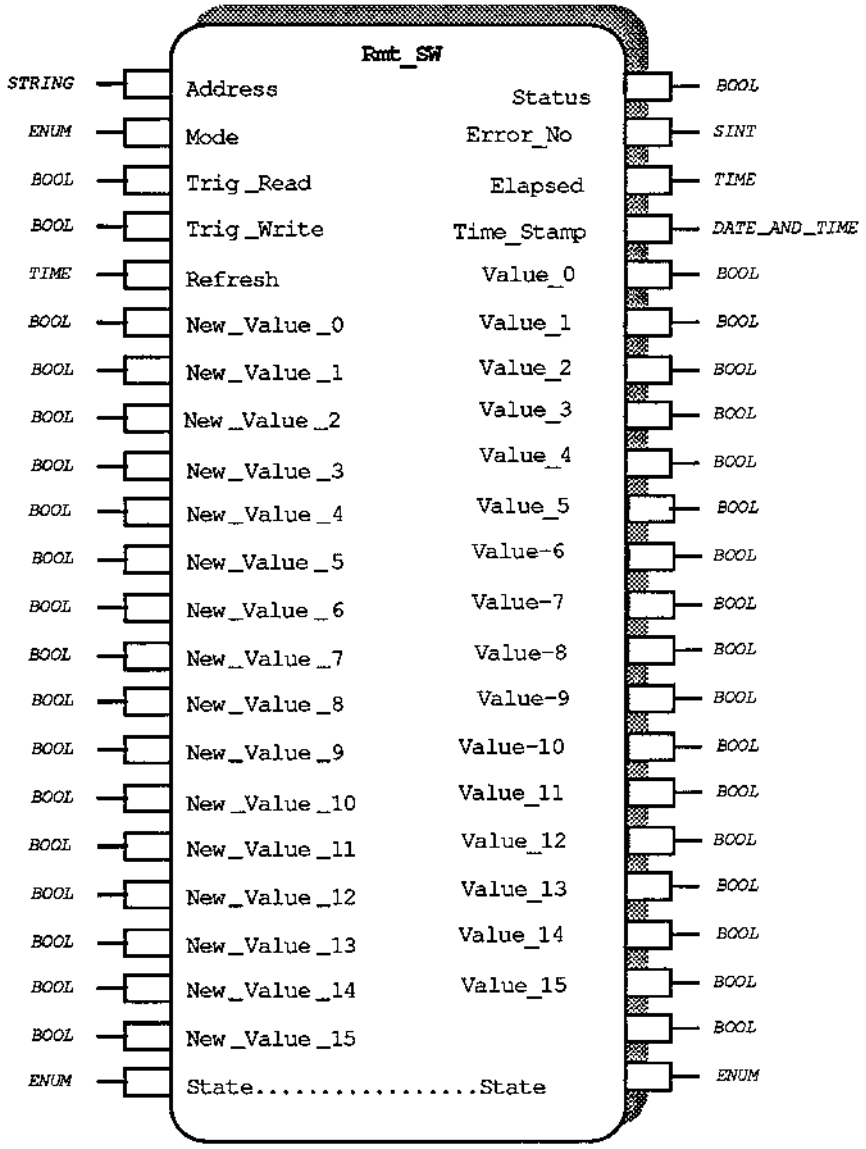


Figure 12-5 Schéma du bloc fonction de la variable Rmt_SW

Description fonctionnelle

Le bloc fonction de la variable Rmt_SW permet la lecture et/ou l'écriture de la valeur d'une variable de 16 bits via un port de communication. Les 16 bits peuvent être adressés individuellement en tant que variables booléennes. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'avec des ports qui sont compatibles avec des protocoles fonctionnant en mode Maître.

Une description complète de l'utilisation des blocs fonctions Rmt_Vars est donnée au chapitre 3, Vue d'ensemble des communications PC3000.

Attributs du bloc fonction

Type : ..... 3A FO

Classe : ..... VARIABLES D'ELEMENTS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Mode, State

Besoins de capacité mémoire : . 122 octets

Durée d'exécution : ..... 132 µs

Description des paramètres

Address (A)

Ce paramètre est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Le premier caractère de Address désigne le module PC3000 sur lequel le port de communications est situé. Le second caractère désigne le port du module. Le reste de l'adresse est spécifique au protocole, et sert à identifier l'instrument et le paramètre qui est en communication avec le bloc fonction.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Quatre valeurs sont possibles :

Demand (0) : En mode Demand, une lecture / écriture unique peut être déclenchée en passant le paramètre d'état de OK (0) à Read (4) ou Write (3).

R_Cont (1) : En mode R_Cont, la variable déportée est constamment invitée à émettre à intervalles réguliers définis par Refresh (actualisation).

W_Cont (2) : En mode W_Cont, la variable déportée est constamment écrite, à intervalles réguliers définis par Refresh (actualisation).

Change (3) : En mode Change, la variable déportée est écrite lorsque la valeur de l'entrée New_Value (nouvelle valeur) ne coïncide plus avec Value. La période minimale entre écritures de changements dans le dispositif déporté est défini par Refresh (actualisation).

Trig_Read (TR)

Lorsque le bloc fonction est en mode Demand, si Trig_Read passe de Off (0) à On (1), une lecture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une lecture à partir du programme.

Trig_Write (TW)

Lorsque le bloc fonction est en mode Demand, si Trig_Write passe de Off (0) à On (1), une écriture de la variable déportée est effectuée. Cette entrée est prévue pour permettre de déclencher une écriture à partir du programme.

Refresh (R)

Refresh (actualisation) définit le délai de mise à jour du paramètre, lorsque le bloc fonction est en mode R_Cont ou W_Cont.

New_Value_0 à New_Value_15 (N0 à N15)

New_Value_0 à New_Value_15 sont les valeurs à écrire dans les bits respectifs 0 à 15, de la variable déportée, lorsque le bloc fonction est en mode écriture.

State (S)

Le paramètre State indique l'état actuel du bloc fonction. Cinq valeurs sont possibles :

Ok (0) : Aucune action de communication n'est en cours

Pending (1) : Une transaction a démarré et n'est pas encore terminée

Error (2) : Une transaction de communication a échoué. Le paramètre Error_No fournit un indicateur de diagnostic sur le type d'erreur

Write (3) : La mise à Write (3) de State déclenche un écriture unique de New_Value pour l'équipement déporté

Read (4) : La mise à Read (4) de State déclenche une lecture unique du paramètre déporté

Quand State est mis à Write (3) ou Read (4), il repasse automatiquement à OK (0) si la transmission a été réussie, ou reste à l'état Error (2).

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au driver de communication Maître approprié, au chapitre 3, en ce qui concerne les codes d'erreur.

Elapsed (E)

Elapsed indique le temps déjà écoulé depuis la dernière transaction de lecture ou d'écriture. Elapsed n'est valide que lorsque le bloc fonction est en mode R_Cont ou W_Cont.

Time_Stamp (TS)

Time_Stamp définit l'heure et la date de la dernière transaction de lecture ou d'écriture effectuée avec succès. Son activation coïncide avec le passage du paramètre State à Ok (0).

Value_0 à Value_15 (V0 à V15)

Les paramètres Value_0 à Value_15 contiennent les dernières valeurs qui ont été écrites ou lues dans les bits respectifs 0 à 15 de la variable déportée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Elapsed	TIME	0	Oper			
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Demand (0)	Oper	Super	Détect.	Demand (0) R_Cont (1) W_Cont (2) Change (3)
New_Value_0 to New_Value_15	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Refresh	TIME	10s	Oper	Super	Lim. haute Lim. basse	19d_59m_59s 100 ms
State	ENUM	Ok (0)	Oper	Oper	Détect.	Ok (0) Pending (1) Error (2) Write (3) Read (4)
Status	BOOL	Go (1)	Oper		Détect.	NOGO (0) Go (1)
Time_Stamp	DATE AND TIME	0	Oper			
Trig_Read	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Trig_Write	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value_0 to Value_15	BOOL	Off (0)	Oper		Détect.	Off (0) On (1)

Tableau 12-5 Attributs des paramètres de la variable Rmt_SW

BLOC FONCTION RMT_****_8

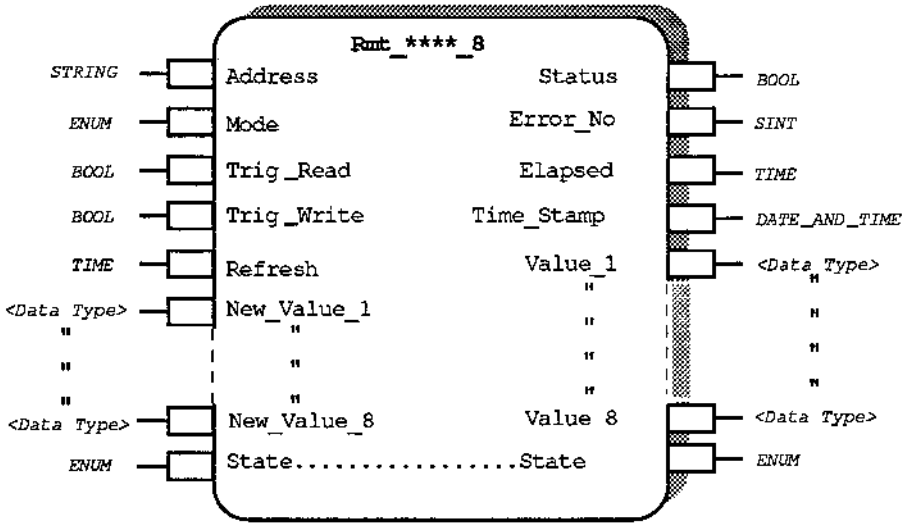


Figure 12-6 Schéma du bloc fonction Rmt_****_8

Description fonctionnelle

Tous les blocs de cette classe sont désignés par Rmt_****_8, dans lequel **** remplace BOOL, REAL, DINT ou STR.

L'utilisation et les fonctionnalités sont identiques à celles des blocs fonctions Rmt_*** décrits précédemment dans ce chapitre. La différence essentielle est le fait qu'ils disposent d'entrées multiples New_Value (New_Value_1 à New_Value_8 [mnémoniques N1 - N8]) et de sorties multiples Value (Value_1 à Value_8 [mnémoniques V1 - V8]).

Ces blocs fonctions représentent un moyen efficace pour accéder à des paramètres multiples avec une seule transaction de lecture ou d'écriture. Les blocs peuvent accéder à 8 emplacements contigus. L'accès aux données à lire ou à écrire se fait au moyen d'un simple paramètre composé. L'ordre des données à l'intérieur du paramètre composé est le même que l'ordre affiché, c'est-à-dire Value_1 à Value_8.

Ces blocs permettent une utilisation plus efficace de la mémoire, étant donné qu'une seule adresse est définie et qu'il n'y a qu'un seul jeu de paramètres de commande et d'état pour 8 emplacements de données.

Ces blocs peuvent être utilisés en tant qu'alternative au compactage et au décompactage de données en chaînes de caractères , en utilisant les fonctions COMPACT dont dispose le langage en texte structuré.

Nota: Ce type de bloc ne peut être utilisé, actuellement, qu'avec le bloc fonction EI_Bisync_M. D'autres protocoles pourront être pris en charge ultérieurement.

Exemple d'adressage :

Pour Eurotherm Bisync,

Address = 0C 7 0 0 0 1

a pour résultat la lecture / écriture, en une seule transaction, des 8 valeurs qui sont mémorisées aux adresses 001 à 008.

BLOC FONCTION RMT_****_64

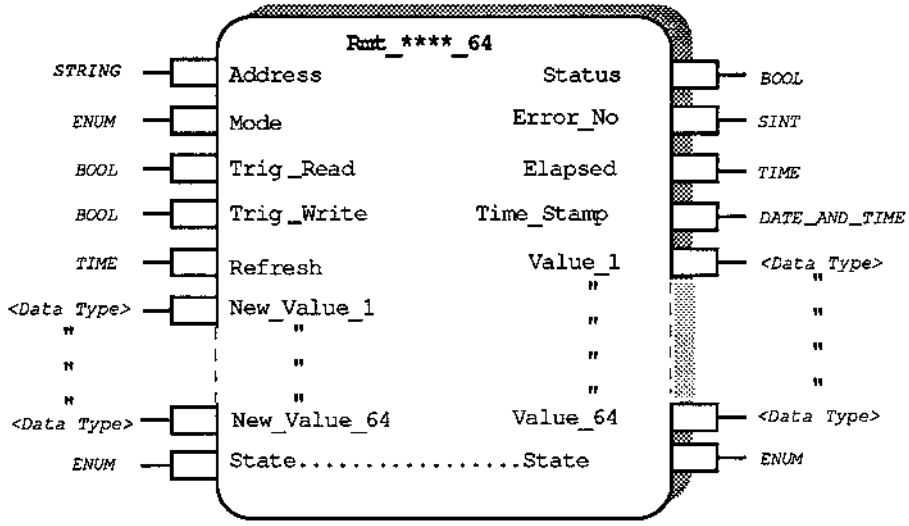


Figure 12-7 Schéma du bloc fonction Rmt_****_64

Description fonctionnelle

Tous les blocs de cette classe sont désignés par Rmt_****_64, dans lequel **** remplace BOOL, REAL, DINT ou STR.

L'utilisation et les fonctionnalités sont identiques à celles des blocs fonctions Rmt_**** décrits précédemment dans ce chapitre. La différence essentielle est le fait qu'ils disposent d'entrées multiples New_Value (New_Value_1 à New_Value_64 [mnémoniques N1 - N64]) et de sorties multiples Value (Value_1 à Value_64 [mnémoniques V1 - V64]).

Ces blocs fonctions représentent un moyen efficace pour accéder à des paramètres multiples avec une seule transaction de lecture ou d'écriture. Les blocs peuvent accéder à 64 emplacements contigus. L'accès aux données à lire ou à écrire se fait au moyen d'un simple paramètre composé. L'ordre des données à l'intérieur du paramètre composé est le même que l'ordre affiché, c'est-à-dire Value_1 à Value_64.

Ces blocs permettent une utilisation plus efficace de la mémoire, étant donné qu'une seule adresse est définie et qu'il n'y a qu'un seul jeu de paramètres de commande et d'état pour 64 emplacements de données.

Ces blocs peuvent être utilisés en tant qu'alternative au compactage et au décompactage de données en chaînes de caractères, en utilisant les fonctions COMPACT dont dispose le langage en texte structuré.

Nota: Ce type de bloc ne peut être utilisé, actuellement, qu'avec le bloc fonction EI_Bisync_M. D'autres protocoles pourront être pris en charge ultérieurement.

Exemple d'adressage :

Pour Eurotherm Bisync,

Address = 0C 7 0 0 0 1

a pour résultat la lecture / écriture, en une seule transaction, des 64 valeurs qui sont mémorisées aux adresses 001 à 064.

Chapitre 13

VARIABLES ESCLAVES

Edition 1

Sommaire (suite)

VUE D'ENSEMBLE

Ce chapitre décrit la classe des blocs fonctions SLAVE-VARS (variables esclaves).

Les paramètres ESCLAVES sont utilisés en association avec un pilote de communication ESCLAVE approprié. Ils sont utilisés lorsque le PC3000 est l'équipement ESCLAVE. Les blocs peuvent être considérés commes des adresses fixes ou "boîte aux lettres" dans lesquelles des données peuvent être écrites ou lues par un équipement maître approprié, par exemple un superviseur à base de PC. Les blocs disposent de diagnostics, de commande d'accès lecture / écriture, etc.

Les paramètres ESCLAVES à éléments multiples permettent une utilisation plus efficace de la mémoire, étant donné qu'un bloc peut être utilisé pour commander des transactions associées à différentes valeurs. De plus, elles gagnent en vitesse d'initialisation, étant donné qu'une seule adresse a besoin d'être indiquée pour son démarrage à froid, alors qu'il en faut plusieurs lorsque l'on utilise des paramètres individuels.

Paramètres esclaves
Slv_Bool
Slv_Real
Slv_Dint
Slv_Time
Slv_Str
Slv_Bool_8
Slv_Real_8
Slv_Dint_8
Slv_Bool_64
Slv_Real_64
Slv_Dint_64
Slv_SW

Le type de données du mot d'état est un type de données spécial qui regroupe 16 variables booléennes dans un seul bloc. Il est utilisé pour les informations de type état.

Cette classe est utilisée en association avec les blocs fonctions de la classe COMMS (communication).

BLOC FONCTION SLV_BOOL

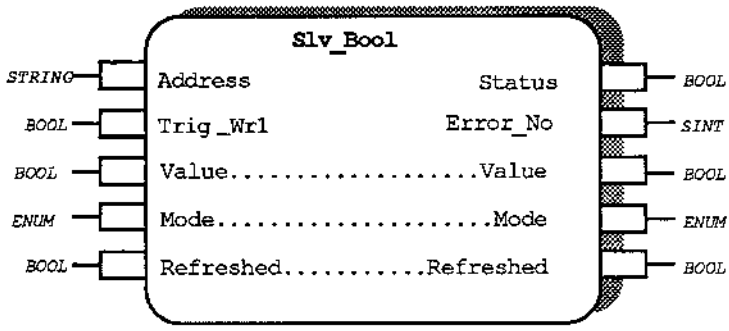


Figure 13-1 Schéma du bloc fonction Slv_Bool

Description fonctionnelle

Le bloc fonction Slv_Bool permet l'écriture et/ou la lecture de la valeur d'un paramètre booléen dans le PC3000 par un équipement externe, via un port de communication du PC3000. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'en association avec un bloc fonction de protocole esclave, par exemple EI_Bisync_S.

La description détaillée de l'utilisation des blocs fonctions Variables esclaves est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000.

Attributs du bloc fonction

Type : 3c 10

Classe : ..... VARIABLES ESCLAVES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, Mode, Refreshed, Status

Besoins de capacité mémoire : 100 octets

Durée d'exécution : ..... 122 µs

Description des paramètres

Address (A)

Le paramètre Address est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Les deux premiers caractères de Address définissent le protocole qui peut être utilisé pour accéder au bloc fonction. Le reste de l'adresse est spécifique au protocole et sert à l'équipement déporté pour identifier la variable. Pour plus de détails sur les adresses de variable esclave, se reporter au pilote de communication esclave associé, au chapitre 3.

Trig_Wr1 (TRW)

Sur le front avant du passage de Trig_Wr1 de Off (0) à On (1), le Mode passe à Wr_Once (2). Ceci peut être utilisé comme câblage logiciel pour créer une condition de verrouillage mutuel. Sur réception d'une nouvelle valeur, les données ne peuvent pas être écrasées, tant que Trig_Wr1 n'est pas revenu à l'état Off (0).

Value (VAL)

Value est le paramètre booléen qui doit être lu ou écrit, via le port de communication associé au protocole sélectionné par le paramètre Address.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Trois valeurs sont possibles :

Rd_Wr (0) :	Accès autorisé pour toute lecture ou écriture déportée
R_Cont (1) :	Accès autorisé pour toute lecture déportée, mais blocage de toute écriture déportée
Wr_once (2) :	Accès toujours autorisé pour lecture déportée, mais à la suite d'une écriture déportée, le mode bascule sur Rd_Only (1), afin d'empêcher l'écrasement des valeurs qui viennent d'être recues.

Refreshed (R)

Il est mis à Yes (1), à chaque fois qu'une valeur est écrite via la voie de communication. Il doit être remis à zéro par le programme utilisateur pour détecter un changement de valeur.

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication esclave approprié, au chapitre 3, en ce qui concerne l'explication des codes d'erreur.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Rd_Wr (0)	Oper	Oper	Détect.	Rd_Wr (0) Rd_Only (1) Wr_Once(2)
Refreshed	BOOL	No (0)	Oper	Super	Détect.	No (0) Yes (1)
Status	BOOL	NOGO (0)	Oper		Détect.	NOGO (0) Go (1)
Trig_Wr1	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	BOOL	Value (0)	Oper	Super	Détect.	Value (0) Value (1)

Tableau 13-1 Attributs des paramètres de Slv_Bool

BLOC FONCTION SLV_REAL

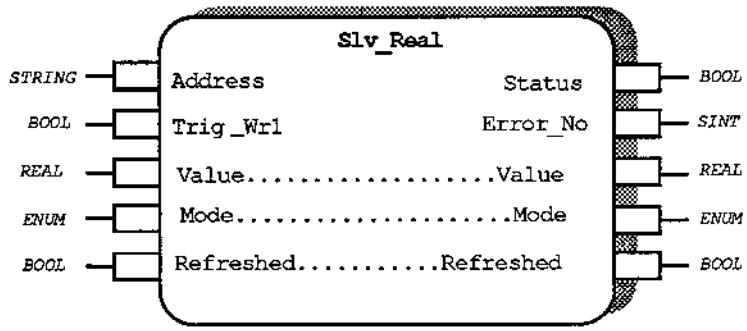


Figure 13-2 Schéma du bloc fonction Slv_Real

Description fonctionnelle

Le bloc fonction Slv_Real permet la lecture et/ou l'écriture de la valeur d'un paramètre à virgule flottante dans le PC3000, via un port de communication du PC3000. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'en association avec un bloc fonction de protocole esclave, par exemple EI_Bisync_S.

La description détaillée de l'utilisation des blocs fonctions Variables esclaves est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000.

Attributs du bloc fonction

Type : 3c 20

Classe : ..... VARIABLES ESCLAVES

Tâche par défaut : .....Task_2

Liste récapitulative : .....Value, Mode, Refreshed, Status

Besoins de capacité mémoire : .106 octets

Durée d'exécution : ..... 128 µs

Description des paramètres

Address (A)

Le paramètre Address est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Les deux premiers caractères de Address définissent le protocole qui peut être utilisé pour accéder au bloc fonction. Le reste de l'adresse est spécifique au protocole et sert à l'équipement déporté pour identifier la variable, etc. Pour plus de détails sur les adresses de variable esclave, se reporter au pilote de communication esclave associé, au chapitre 3.

Trig_Wr1 (TRW)

Sur le front avant du passage de Trig_Wr1 de Off (0) à On (1), le Mode passe à Wr_Once (2). Ceci peut être utilisé comme câblage logiciel pour créer une condition de verrouillage mutuel. Sur réception d'une nouvelle valeur, les données ne peuvent pas être écrasées, tant que Trig_Wr1 n'est pas revenu à l'état Off (0).

Value (VAL)

Value est le paramètre à virgule flottante (Real) qui doit être lu ou écrit, via le port de communication associé, etc.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Trois valeurs sont possibles :

Rd_Wr (0) : Accès autorisé pour toute lecture ou écriture déportée

R_Cont (1) : Accès autorisé pour toute lecture déportée, mais blocage de toute écriture déportée

Wr_Once (2) : Accès toujours autorisé pour lecture déportée, mais à la suite d'une écriture déportée, le mode bascule sur Rd_Only (1), afin d'empêcher l'écrasement des valeurs qui viennent d'être recues.

Refreshed (R)

Il est mis à Yes (1), à chaque fois qu'une valeur est écrite via la voie de communication.

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication esclave approprié, au chapitre 3, en ce qui concerne l'explication des codes d'erreur.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Rd_Wr (0)	Oper	Oper	Détect.	Rd_Wr (0) Rd_Only (1) Wr_Once (2)
Refreshed	BOOL	No (0)	Oper	Super	Détect.	No (0) Yes (1)
Status	BOOL	NOGO (0)	Oper		Détect.	NOGO (0) Go (1)
Trig_Wr1	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	REAL	0	Oper	Super	Lim. haute Lim. basse	1 000 000 -1 000 000

Tableau 13-2 Attributs des paramètres de Slv_Real

BLOC FONCTION SLV_DINT

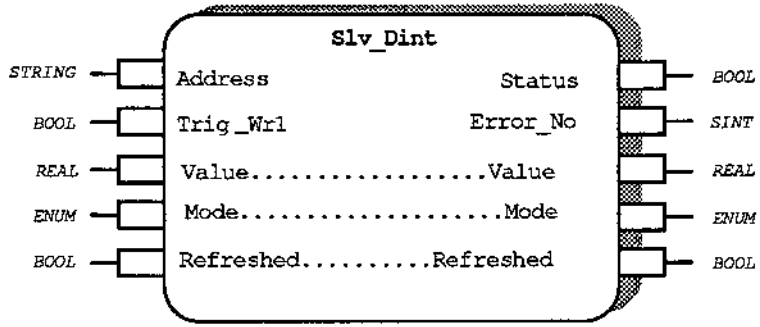


Figure 13-3 Schéma du bloc fonction Slv_Dint

Description fonctionnelle

Le bloc fonction Slave_Dint permet la lecture et/ou l'écriture de la valeur d'un paramètre entier de 32 bits plus signe dans le PC3000, par un équipement externe, via un port de communication du PC3000. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'en association avec un bloc fonction de protocole esclave, par exemple EI_Bisync_S.

La description détaillée de l'utilisation des blocs fonctions Variables esclaves est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000.

Attributs du bloc fonction

Type : 3c 30

Classe : ..... VARIABLES ESCLAVES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, Mode, Refreshed, Status

Besoins de capacité mémoire : 110 octets

Durée d'exécution : ..... 123 µs

Description des paramètres

Address (A)

Le paramètre Address est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Les deux premiers caractères de Address définissent le protocole qui peut être utilisé pour accéder au bloc fonction. Le reste de l'adresse est spécifique au protocole et sert à l'équipement déporté pour identifier la variable, etc. Pour plus de détails sur les adresses de variable esclave, se reporter au chapitre 3.

Trig_Wr1 (TRW)

Sur le front avant du passage de Trig_Wr1 de Off (0) à On (1), le Mode passe à Wr_Once (2). Ceci peut être utilisé comme câblage logiciel pour créer une condition de verrouillage mutuel. Sur réception d'une nouvelle valeur, les données ne peuvent pas être écrasées, tant que Trig_Wr1 n'est pas revenu à l'état Off (0).

Value (VAL)

Value est le paramètre entier (Integer) qui doit être lu ou écrit, via le port de communication associé, etc.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Trois valeurs sont possibles :

Rd_Wr (0) :	Accès autorisé pour toute lecture ou écriture déportée
R_Cont (1) :	Accès autorisé pour toute lecture déportée, mais blocage de toute écriture déportée
Wr_Once (2) :	Accès toujours autorisé pour lecture déportée, mais à la suite d'une écriture déportée, le mode bascule sur Rd_Only (1), afin d'empêcher l'écrasement des valeurs qui viennent d'être recues.

Refreshed (R)

Il est mis à Yes (1), à chaque fois qu'une valeur est écrite via la voie de communication.

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication esclave approprié, au chapitre 3, en ce qui concerne l'explication des codes d'erreur.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Rd_Wr (0)	Oper	Oper	Détect.	Rd_Wr (0) Rd_Only (1) Wr_Once (2)
Refreshed	BOOL	No (0)	Oper	Super	Détect.	No (0) Yes (1)
Status	BOOL	NOGO (0)	Oper		Détect.	NOGO (0) Go (1)
Trig_Wr1	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	DINT	0	Oper	Super	Lim. haute Lim. basse	1 000 000 -1 000 000

Tableau 13-3 Attributs des paramètres de Slv_Dint

BLOC FONCTION SLV_TIME

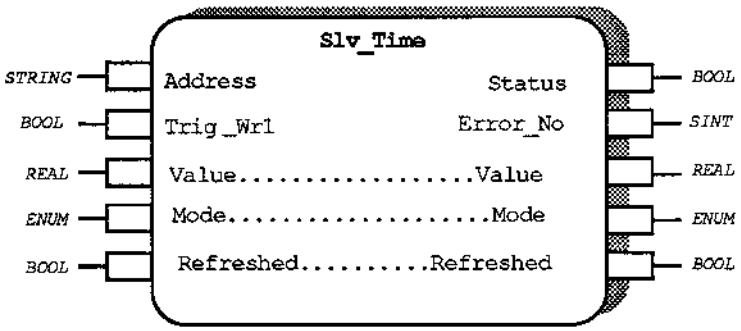


Figure 13-4 Schéma du bloc fonction Slv_Time

Description fonctionnelle

Le bloc fonction Slv_Time permet la lecture et/ou l'écriture de la valeur d'un paramètre de durée dans le PC3000, par un équipement externe, via un port de communication du PC3000. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'en association avec un bloc fonction de protocole esclave, par exemple EI_Bisync_S.

La description détaillée de l'utilisation des blocs fonctions Variables esclaves est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000.

Attributs du bloc fonction

Type : 3c 40

Classe : .....VARIABLES ESCLAVES

Tâche par défaut : .....Task_2

Liste récapitulative : .....Value, Mode, Refreshed, Status

Besoins de capacité mémoire : .106 octets

Durée d'exécution : .....123 µs

Description des paramètres

Address (A)

Le paramètre Address est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Les deux premiers caractères de Address définissent le protocole qui peut être utilisé pour accéder au bloc fonction. Le reste de l'adresse est spécifique au protocole et sert à l'équipement déporté pour identifier la variable, etc. Pour plus de détails sur les adresses de variable esclave, se reporter au pilote de communication esclave associé, au chapitre 3.

Trig_Wr1 (TRW)

Sur le front avant du passage de Trig_Wr1 de Off (0) à On (1), le Mode passe à Wr_Once (2). Ceci peut être utilisé comme câblage logiciel pour créer une condition de verrouillage mutuel. Sur réception d'une nouvelle valeur, les données ne peuvent pas être écrasées, tant que Trig_Wr1 n'est pas revenu à l'état Off (0).

Value (VAL)

Value est le paramètre durée (TIME) qui doit être lu ou écrit, via le port de communication associé, etc.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Trois valeurs sont possibles :

Rd_Wr (0) :	Accès autorisé pour toute lecture ou écriture déportée
R_Cont (1) :	Accès autorisé pour toute lecture déportée, mais blocage de toute écriture déportée
Wr_Once (2) :	Accès toujours autorisé pour lecture déportée, mais à la suite d'une écriture déportée, le mode bascule sur Rd_Only (1), afin d'empêcher l'écrasement des valeurs qui viennent d'être recues.

Refreshed (R)

Il est mis à Yes (1), à chaque fois qu'une valeur est écrite via la voie de communication.

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication esclave approprié, au chapitre 3, en ce qui concerne l'explication des codes d'erreur.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Rd_Wr (0)	Oper	Oper	Détect.	Rd_Wr (0) Rd_Only (1) Wr_Once (2)
Refreshed	BOOL	No (0)	Oper	Super	Détect.	No (0) Yes (1)
Status	BOOL	NOGO (0)	Oper		Détect.	NOGO (0) Go (1)
Trig_Wr1	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	TIME	0	Oper	Super		

Tableau 13-4 Attributs des paramètres de Slv_Time

BLOC FONCTION SLV_STR

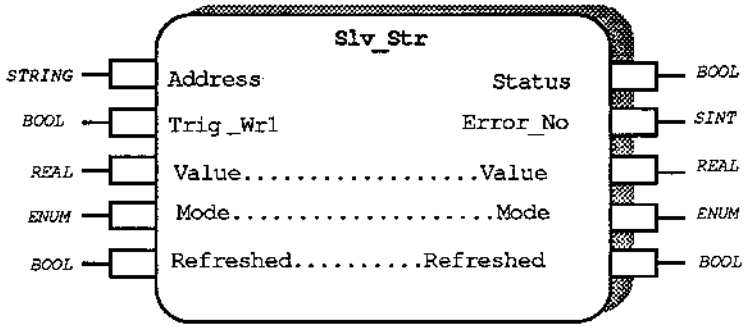


Figure 13-5 Schéma du bloc fonction Slv_Str

Description fonctionnelle

Le bloc fonction Slv_Str permet la lecture et/ou l'écriture de la valeur d'un paramètre de chaîne de caractères dans le PC3000, par un équipement externe, via un port de communication du PC3000. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'en association avec un bloc fonction de protocole esclave, par exemple EI_Bisync_S.

La description détaillée de l'utilisation des blocs fonctions Variables esclaves est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000.

Attributs du bloc fonction

Type : 3c 80

Classe : ..... VARIABLES ESCLAVES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value, Mode, Refreshed, Status

Besoins de capacité mémoire : . 486 octets

Durée d'exécution : ..... 514 µs

Description des paramètres

Address (A)

Le paramètre Address est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Les deux premiers caractères de Address définissent le protocole qui peut être utilisé pour accéder au bloc fonction. Le reste de l'adresse est spécifique au protocole et sert à l'équipement déporté pour identifier la variable, etc. Pour plus de détails sur les adresses de variable esclave, se reporter au pilote de communication esclave associé, au chapitre 3.

Trig_Wr1 (TRW)

Sur le front avant du passage de Trig_Wr1 de Off (0) à On (1), le Mode passe à Wr_Once (2). Ceci peut être utilisé comme câblage logiciel pour créer une condition de verrouillage mutuel. Sur réception d'une nouvelle valeur, les données ne peuvent pas être écrasées, tant que Trig_Wr1 n'est pas revenu à l'état Off (0).

Value (VAL)

Value est le paramètre de chaîne de caractères (STRING) qui doit être lu ou écrit, via le port de communication associé, etc.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Trois valeurs sont possibles :

Rd_Wr (0) : Accès autorisé pour toute lecture ou écriture déportée

R_Cont (1) : Accès autorisé pour toute lecture déportée, mais blocage de toute écriture déportée

Wr_Once (2) : Accès toujours autorisé pour lecture déportée, mais à la suite d'une écriture déportée, le mode bascule sur Rd_Only (1), afin d'empêcher l'écrasement des valeurs qui viennent d'être recues.

Refreshed (R)

Il est mis à Yes (1), à chaque fois qu'une valeur est écrite via la voie de communication.

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication esclave approprié, au chapitre 3, en ce qui concerne l'explication des codes d'erreur.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	STRING		Oper	Config		
Error_No	SINT	0	Oper		Lim. haute Lim. basse	255 0
Mode	ENUM	Rd_Wr (0)	Oper	Oper	Détect.	Rd_Wr (0) Rd_Only (1) Wr_Once (2)
Refreshed	BOOL	No (0)	Oper	Super	Détect.	No (0) Yes (1)
Status	BOOL	NOGO (0)	Oper		Détect.	NOGO (0) Go (1)
Trig_Wr1	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value	STRING	' '	Oper	Super		

Tableau 13-5 Attributs des paramètres de Slv_Str

BLOC FONCTION SLV_SW

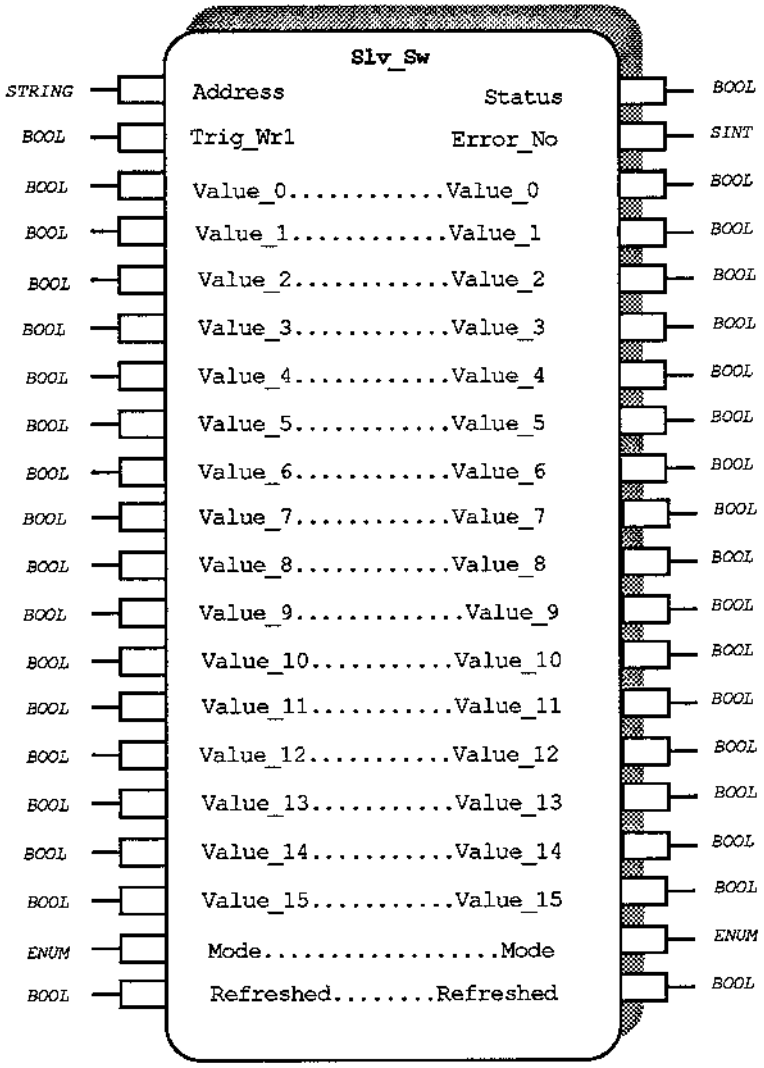


Figure 13-6 Schéma du bloc fonction Slv_SW

Description fonctionnelle

Le bloc fonction de mot Slave Status permet la lecture et/ou l'écriture de la valeur d'un paramètre de 16 bits dans le PC3000, par un équipement externe, via un port de communication du PC3000. Le bloc fonction n'est pas spécifique à un protocole de communication, mais il ne peut être utilisé qu'en association avec un bloc fonction de protocole esclave, par exemple EI_Bisync_S.

La description détaillée de l'utilisation des blocs fonctions Variables esclaves est donnée au chapitre 3, Vue d'ensemble de la Communication du PC3000

Attributs du bloc fonction

Type : 3c f0

Classe : ..... VARIABLES ESCLAVES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Value_1, Mode, Refreshed, Status

Besoins de capacité mémoire : . 126 octets

Durée d'exécution : ..... 136 µs

Description des paramètres

Address (A)

Le paramètre Address est une chaîne de caractères qui est utilisée pour associer le bloc fonction à un paramètre du dispositif déporté auquel le PC3000 est relié. Les deux premiers caractères de Address définissent le protocole qui peut être utilisé pour accéder au bloc fonction. Le reste de l'adresse est spécifique au protocole et sert à l'équipement déporté pour identifier la variable, etc. Pour plus de détails sur les adresses de variable esclave, se reporter au pilote de communication esclave associé, au chapitre 3.

Trig_Wr1 (TRW)

Sur le front avant du passage de Trig_Wr1 de Off (0) à On (1), le Mode passe à Wr_Once (2). Ceci peut être utilisé comme câblage logiciel pour créer une condition de verrouillage mutuel. Sur réception d'une nouvelle valeur, les données ne peuvent pas être écrasées, tant que Trig_Wr1 n'est pas revenu à l'état Off (0).

Value_0 à Value_15 (V0 à V15)

Value_0 à Value_15 sont les paramètres qui doivent être lus ou écrits via le port de communication, et représentent respectivement les bits 0 à 15.

Mode (M)

Le paramètre Mode spécifie le mode de fonctionnement du bloc fonction. Trois valeurs sont possibles :

Rd_Wr (0) : Accès autorisé pour toute lecture ou écriture déportée

R_Cont (1) : Accès autorisé pour toute lecture déportée, mais blocage de toute écriture déportée

Wr_Once (2) : Accès toujours autorisé pour lecture déportée, mais à la suite d'une écriture déportée, le mode bascule sur Rd_Only (1), afin d'empêcher l'écrasement des valeurs qui viennent d'être recues.

Refreshed (R)

Il est mis à Yes (1), à chaque fois qu'une valeur est écrite via la voie de communication.

Etat (ST)

Si Etat est mis à Go (1), la demande précédente de communication s'est terminée avec succès. Si Status est mis à NOGO (0), une erreur a eu lieu.

Error_No (ERR)

Error_No est un paramètre de diagnostic qui fournit une indication sur le type d'erreur de communication qui a eu lieu. Se reporter au pilote de communication esclave approprié, au chapitre 3, en ce qui concerne l'explication des codes d'erreur.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Address	**STRING**		Oper	Config		
Error_No	**SINT**	0	Oper		Lim. haute Lim. basse	255 0
Mode	**ENUM**	Rd_Wr (0)	Oper	Oper	Détect.	Rd_Wr (0) Rd_Only (1) Wr_Once (2)
Refreshed	**BOOL**	No (0)	Oper	Super	Détect.	No (0) Yes (1)
Status	**BOOL**	NOGO (0)	Oper		Détect.	NOGO (0) Go (1)
Trig_Wr1	**BOOL**	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Value_0 to Value_15	**BOOL**	Value (0)	Oper	Super	Détect.	Value (0) Value (1)

Tableau 13_6 Attributs des paramètres de Slv_SW

BLOC FONCTION SLV_****_8

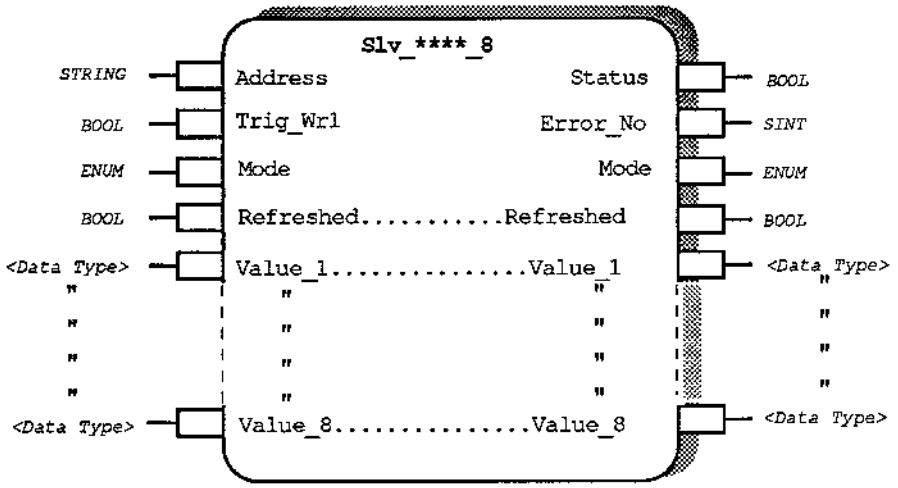


Figure 13-7 Schéma du bloc fonction Slv_****_8

Description fonctionnelle

Tous les blocs de cette classe sont désignés par Slv_****_8 dans lequel **** remplace BOOL, REAL, DINT ou STR.

L'utilisation et les fonctionnalités sont identiques à celles des blocs fonctions Slv_*** décrits précédemment dans ce chapitre. La différence essentielle est le fait qu'ils disposent d'entrées multiples Value (Value_1 à Value_8 [mnémoniques V1 - V8]).

Ces blocs fonctions représentent un moyen efficace pour accéder à des paramètres multiples avec une seule transaction de lecture ou d'écriture. Les blocs peuvent accéder à 8 emplacements contigus. L'accès aux données à lire ou a écrire se fait au moyen d'un simple paramètre composé. L'ordre des données à l'intérieur du paramètre composé est le même que l'ordre affiché, c'est-à-dire Value_1 à Value_8.

Ces blocs permettent une utilisation plus efficace de la mémoire, étant donné qu'une seule adresse est définie et qu'il n'y a qu'un seul jeu de paramètres de commande et d'état pour 8 emplacements de données.

Ces blocs peuvent être utilisés en tant qu'alternative au compactage et au décompactage de données en chaînes de caractères, en utilisant les fonctions COMPACT dont dispose le langage en texte structuré.

BLOC FONCTION SLV_****_64

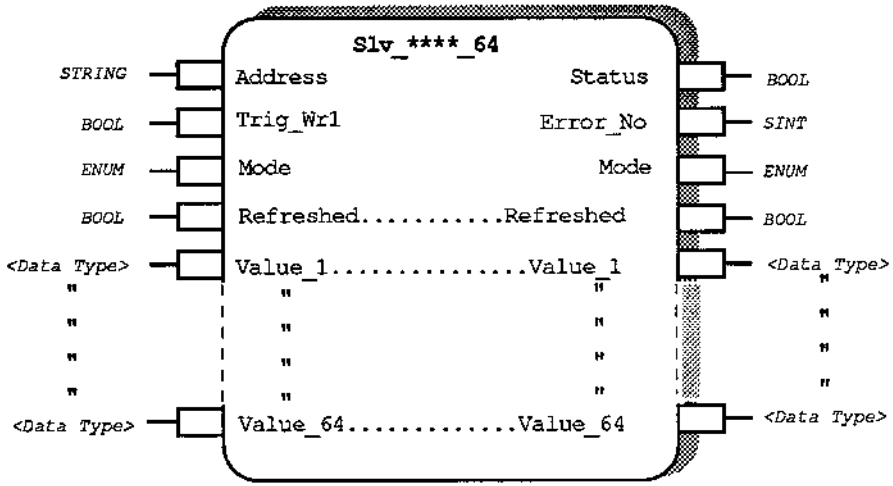


Figure 13-8 Schéma du bloc fonction Slv_****_64

Description fonctionnelle

Tous les blocs de cette classe sont désignés par Slv_****_64, dans lequel **** remplace BOOL, REAL, DINT ou STR.

L'utilisation et les fonctionnalités sont identiques à celles des blocs fonctions Slv_*** décrits précédemment dans ce chapitre. La différence essentielle est le fait qu'ils disposent d'entrées multiples Value (Value_1 à Value_64 [mnémoniques V1 - V64]).

Ces blocs fonctions représentent un moyen efficace pour accéder à des paramètres multiples avec une seule transaction de lecture ou d'écriture. Les blocs peuvent accéder à 64 emplacements contigus. L'accès aux données à lire ou à écrire se fait au moyen d'un simple paramètre composé. L'ordre des données à l'intérieur du paramètre composé est le même que l'ordre affiché, c'est-à-dire Value_1 à Value_64.

Ces blocs permettent une utilisation plus efficace de la mémoire, étant donné qu'une seule adresse est définie et qu'il n'y a qu'un seul jeu de paramètres de commande et d'état pour 64 emplacements de données.

Ces blocs peuvent être utilisés en tant qu'alternative au compactage et au décompactage de données en chaînes de caractères, en utilisant les fonctions COMPACT dont dispose le langage en texte structuré.

BLOC FONCTION RECIPEMAN

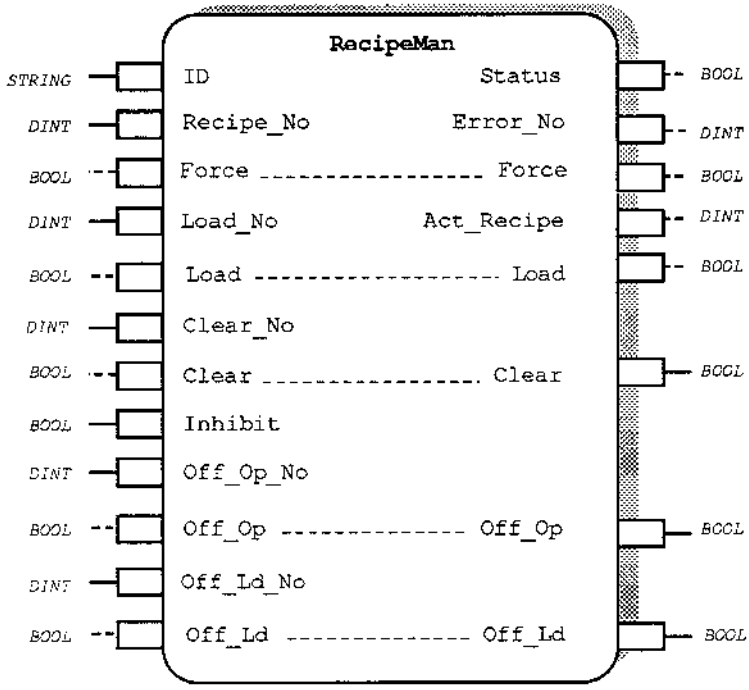


Figure 14-7 Schéma du bloc fonction RecipeMan

Description fonctionnelle

Le bloc fonction Recipe Manager sert à coordonner un certain nombre de blocs fonctions d'esclaves de recettes. Il existe des fonctions permettant de sélectionner les recettes, de les modifier, de les supprimer, etc.

Attributs du bloc fonction

Type : 3E00

Classe : ..... RECIPE

Tâche par défaut : ..... Task_2

Liste résumée : ..... ID, Recipe No, Status

Mémoire nécessaire : ..... 66 octets

Description des paramètres

ID (ID)

Chaîne à un caractère qui associe le bloc fonction Gestionnaire de recettes à un certain nombre d'esclaves de recettes. L'ensemble des blocs fonctions Esclaves de recettes portant le même ID seront commandés par ce gestionnaire de recettes.

Recipe_No(RN)

Sert à fixer le numéro de recette souhaité. Une nouvelle recette sera sélectionnée à chaque changement de Recipe_No. Pour sélectionner à nouveau la même recette, il ne faut utiliser aucun paramètre Force.

Force (F)

Sert à forcer une nouvelle sélection de la recette actuelle. Utilisable à la suite d'une modification de recette pour forcer l'utilisation des dernières valeurs.

Load_No (LN)

Définit l'emplacement de recette dans lequel seront chargées les sorties actuelles lorsque Load est sur Load.

Load (L)

Lorsque ce paramètre est sur Load, les sorties des recettes actuelles seront chargées dans l'emplacement de recettes défini par Load_No.

Clear_No (CN)

Définit l'emplacement de recette qui sera effacé (supprimé) lorsque Clear est sur Clear.

Clear (C)

Lorsque ce paramètre est sur Clear, le contenu de l'emplacement de recettes défini par Clear_No sera effacé (supprimé).

Inhibit (I)

Lorsqu'il est sur Inhibit, le système de recettes est protégé en écriture et aucune donnée supplémentaire ne sera écrite dans les emplacements de recettes.

Off_Op_No (OON)

Définit l'emplacement de recettes dont le contenu sera transféré aux broches de modification hors ligne lorsque Off_Op est sur Output.

Off_Op (OO)

Lorsque ce paramètre est sur Output, le contenu de l'emplacement de recettes défini par Off_Op_No sera transféré vers les broches de modification en ligne.

Off_Ld_No (OLN)

Définit l'emplacement de recettes vers lequel les valeurs des broches de modification hors ligne seront écrites lorsqu'Off_Ld est sur Load.

Off_Ld (OL)

Lorsque ce paramètre est sur Load, les valeurs des broches de modification hors ligne seront écrites dans l'emplacement de recettes défini par Off_Ld_No.

Status (S)

Positionné sur Go si le système de recettes fonctionne normalement. S'il est positionné sur NOGO, Error_No indiquera la cause de l'erreur.

Error_No (EN)

Définit la condition d'erreur si Status est sur NOGO

100 = recette non définie

101 = Load_No hors plage

102 = Clear_No hors plage

Act_Recipe (AR)

Numéro de recette actuellement utilisé. Peut être différent de Recipe_No si un numéro de recette incorrect a été demandé.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
ID	STRING		Oper	Oper		
Recipe_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Force	BOOL	Idle (0)	Oper	Oper	Sens	Force (1) Idle (0)
Load_No	DINT	1	Oper	Oper	Limite haute Limite basse	128 1
Load	BOOL	Idle (0)	Oper	Oper	Sens	Load (1) Idle (0)
Clear_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Clear	BOOL	Idle (0)	Oper	Oper	Sens	Clear (1) Idle (0)
Inhibit	BOOL	Enable (0)	Oper	Oper	Sens	Inhibit (1) Enable (0)
Off_Op_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Off_Op	BOOL	Idle (0)	Oper	Oper	Sens	Output 1) Idle (0)
Off_Ld_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Off_Ld	BOOL	Idle (0)	Oper	Oper	Sens	Load (1) Idle (0)
Status	BOOL	NOGO (0)	Oper	Block	Sens	NOGO (0) Go (1)
Error_No	DINT	0	Oper	Block	Limite haute Limite basse	255 0
Act_Recipe	DINT	0	Oper	Block	Limite haute Limite basse	128 0

Tableau 14-3 Attributs des paramètres RecipeMan

BLOC FONCTION STAGEMAN

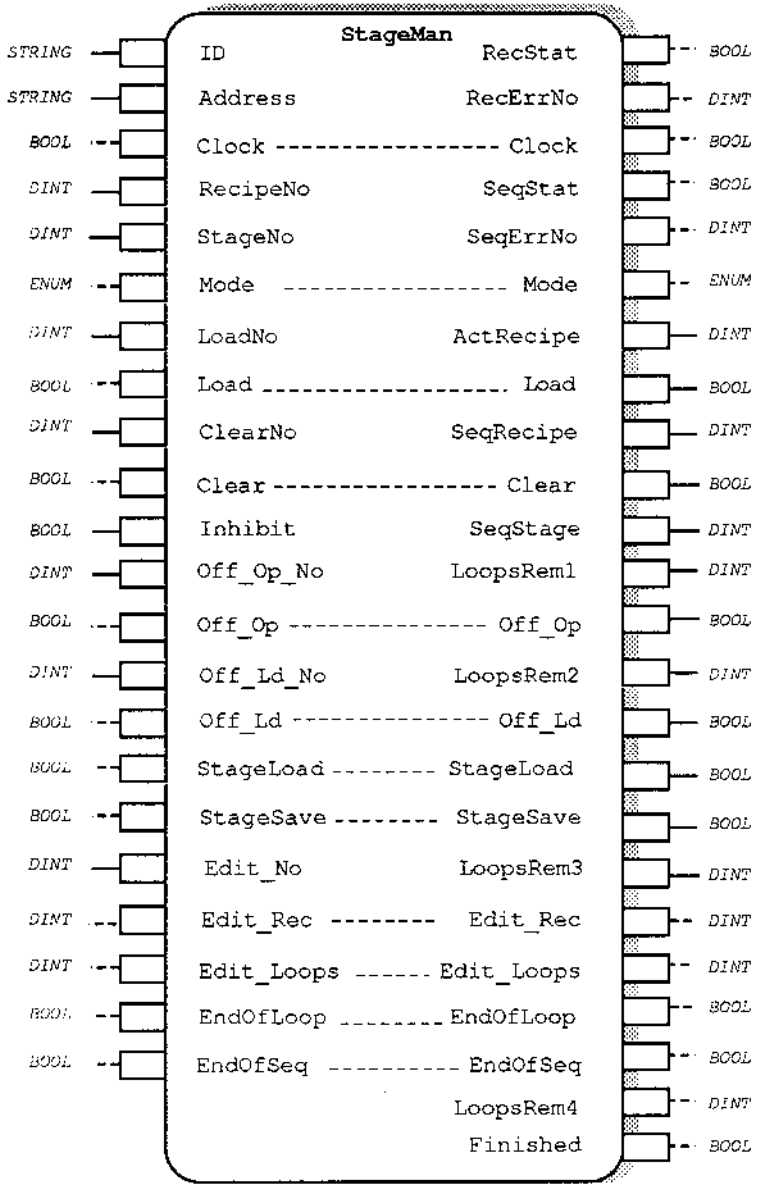


Figure 14-8 Bloc fonction StageMan

Description fonctionnelle

Le bloc fonction Stage Manager est une extension des blocs fonctions Recipe Manager conçue pour prendre en charge le séquencement et le bouclage des recettes.

La séquence de recettes est préchargée dans le bloc fonction Stage Manager à l'aide de l'interface de communications ou du programme utilisateur.

Attributs du bloc fonction

Type : 3E08

Classe : .....RECIPE

Tâche par défaut : .....Task_2

Liste résumée : .....ID, SeqStage, Act Recipe

Mémoire nécessaire : .....3408 octets

Description des paramètres

ID (ID)

Chaîne à un seul caractère servant à associer les blocs fonctions esclaves de recettes au bloc fonction Stage Manager. Par exemple, un Stage Manager dont l'ID est 'A' régule tous les blocs fonctions esclaves de recettes dont l'ID est également 'A'.

Address (A)

L'adresse définit l'adresse de communications permettant d'accéder aux matrices internes de données servant à stocker les informations mises en séquence. Par exemple, si elle est positionnée sur 'EBx00', l'adresse du paramètre multi-éléments est x00, le premier paramètre x01, etc.

Clock (C)

Si Mode est Auto, le fait de positionner Clock sur Tick provoque la sortie de la recette suivante de la séquence. Une fois que la nouvelle recette a été sortie, Clock revient à Tock.

RecipeNo (RN)

Si Mode est égal à Manual, cette entrée spécifie le numéro de recette qui sera sélectionné. Une nouvelle recette est sélectionnée à chaque changement de RecipeNo. Stage Manager indique une erreur si un numéro de recette incorrect est sélectionné.

Stage No(SN)

Lorsque Mode est sur Jump, Stage Manager passe à l'étape de la séquence définie par StageNo.

Mode (M)

Le mode normal de fonctionnement est Auto. Dans ce cas, les recettes sont commandées par la séquence et l'entrée Clock.

Si Mode est positionné sur Manuel, Stage Manager fait fonctionner un Recipe Manager. RecipeNo sert à sélectionner une recette donnée.

Si Mode est positionné sur Reset, la séquence est ramenée à la première étape de la séquence.

Si Mode est sur Jump, la séquence passe à l'étape de la séquence spécifiée par StageNo.

LoadNo (LN)

Définit la destination pour le chargement de la recette actuelle lorsque Load est positionné sur Load.

ClearNo (CN)

Définit l'emplacement de recette à effacer lorsque Clear est sur Clear.

Clear(C)

Sur la position Clear, le contenu de l'emplacement de recette défini par ClearNo est effacé.

Inhibit (I)

Sur la position Inhibit, le système de recettes est protégé en écriture et aucune modification n'est autorisée sur les données de recettes enregistrées.

Off_Op_No (OON)

Définit l'emplacement de recette dont le contenu est transféré vers les broches de modification hors ligne lorsque Off_Op est positionné sur Output.

Off_Op (OO)

Sur la position Output, le contenu de l'emplacement de recette défini par Off_Op_No est transféré vers les broches de modification hors ligne.

Off_Ld_No (OLN)

Définit l'emplacement de recette dans lequel les valeurs des broches de modification hors ligne sont écrites lorsque Off_Ld est positionné sur Load.

Off_Ld (OL)

Sur la position Load, les valeurs des broches de modification hors ligne sont écrites dans l'emplacement de recette défini par Off_Ld_No.

RecStat (RS)

Sur la position Go, le système de recettes fonctionne normalement. Sur la position NOGO, une erreur s'est produite au cours de la dernière transaction de recette, par exemple output, load clear, etc. RecErrNo indique la cause de l'erreur.

RecErrNo (REN)

Définit l'état d'erreur si RecStat est sur NOGO.

100 = recette non définie

101 = Load_No hors plage

102 = Clear_No hors plage

SeqStat (SS)

Sur la position Go, la dernière transaction de communication avec Stage Manager a réussi. Sur la position NOGO, une erreur s'est produite au cours de la dernière transaction de communications. SeqErrNo indique la cause de l'erreur.

SeqErrNo (SEN)

Définit l'état d'erreur si SeqStat est NOGO. Se reporter au driver d'esclave de communications du chapitre 3 pour avoir une explication du numéro d'erreur.

ActRecipe (AR)

Numéro de la recette actuellement sortie par le bloc fonction esclave de recettes.

SeqRecipe (SR)

Numéro de la recette demandée par la séquence Stage Manager. Peut être différent d'ActRecipe si une recette incorrecte est sélectionnée.

SeqStage (SS)

Cette sortie indique l'étape en cours dans la séquence de recettes Stage Manager.

LoopsRem1 (LR)

Cette sortie indique le nombre de boucles restant pour la boucle 1.

LoopsRem2 (LR)

Cette sortie indique le nombre de boucles restant pour la boucle 2.

LoopsRem3 (LR)

Cette sortie indique le nombre de boucles restant pour la boucle 3.

LoopsRem4 (LR)

Cette sortie indique le nombre de boucles restant pour la boucle 4.

Finished (F)

Positionné sur Done lorsque la fin de la séquence de recettes est atteinte.

StageLoad (SL)

Sur la position Load, le contenu de l'étape de séquence défini par Edit_No est écrit dans les broches de modification Edit_Rec, Edit_Loops, EndOfLoop et EndOfSeq.

StageSave (SS)

Sur la position Save, les valeurs des broches de modification Edit_Rec, Edit_Loops, EndOfLoop et EndOfSeq sont écrites dans l'étape de séquence définie par by Edit_No.

Edit_No (EN)

Spécifie le numéro d'étape de Stage Manager dans laquelle est effectuée une lecture ou une écriture lorsque StageLoad et StageSave sont positionnés sur 1.

Edit_Rec (ER)

Numéro de recette sélectionné pour cette étape.

Edit_Loops (EL)

Définit le nombre de boucles pour cette étape. Le nombre de boucles est défini pour la première étape d'une boucle. Quatre boucles emboîtées au maximum sont prises en charge.

EndOfLoop (EOL)

Positionné sur End si cette étape est la dernière d'une boucle.

EndOfSeq (EOS)

Positionné sur End si cette étape est la dernière de la séquence.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
D	STRING	Null	Oper	Oper	Caractère unique	
Address	STRING	Null	Oper	Oper		
Clock	BOOL	Tock (0)	Oper	Oper	Sens	Tick (1) Tock (0)
Recipe_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
StageNo	DINT	1	Oper	Super	Limite haute Limite basse	512 1
Mode	ENUM	Auto (0)	Oper	Oper	Sens	Auto (0) Manual (1) Reset (2) Jump (3)
Load_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Load	BOOL	Idle (0)	Oper	Oper	Sens	Load (1) Idle (0)
ClearNo	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Clear	BOOL	Idle (0)	Oper	Oper	Sens	Clear(1) Idle (0)
Inhibit	BOOL	Enable (0)	Oper	Oper	Sens	Inhibit (1) Enable (0)
Off_Op_No	DINT	0	Oper	Super	Limite haute Limite basse	128 0
Off_Op	BOOL	Idle (0)	Oper	Oper	Sens	Output (1) Idle (0)
Off_Ld_No	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Off_Ld	BOOL	Idle (0)	Oper	Oper	Sens	Load (1) Idle (0)
StageLoad	BOOL	Idle (0)	Oper	Oper	Sens	Load (1) Idle (0)

Tableau 14-4 Attributs des paramètres StageMan

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
StgeSave	BOOL	Idle (0)	Oper	Oper	Sens	Save (1) Idle (0)
Edit_No	DINT	1	Oper	Super	Limite haute Limite basse	512 1
Edit_Rec	DINT	1	Oper	Super	Limite haute Limite basse	128 1
Edit_Loops	DINT	0	Oper	Super	Limite haute Limite basse	65535 0
EndOfLoop	BOOL	Off (0)	Oper	Oper	Sens	End (1) Off (0)
EndOfSeq	BOOL	Off (0)	Oper	Oper	Sens	End (1) Off (0)
RecStat	BOOL	NOGO (0)	Oper	Block	Sens	GO (1) NOGO (0)
RecErrNo	DINT	0	Oper	Block	Limite haute Limite basse	255 0
SeqStat	BOOL	NOGO (0)	Oper	Block	Sens	GO (1) NOGO (0)
SeqErrNo	DINT	0	Oper	Block	Limite haute Limite basse	255 0
ActRecipe	DINT	0	Oper	Block	Limite haute Limite basse	128 0
SeqRecipe	DINT	0	Oper	Block	Limite haute Limite basse	128 0
SeqStage	DINT	0	Oper	Block	Limite haute Limite basse	512 0
LoopsRem1-4	DINT	0	Oper	Block	Limite haute Limite basse	255 0
Finished	BOOL	Running (0)	Oper	Block	Sens	Done (1) Running (0)

Table 14-4 Attributs des paramètres StageMan (suite)

BLOC FONCTION BOOL_16X128

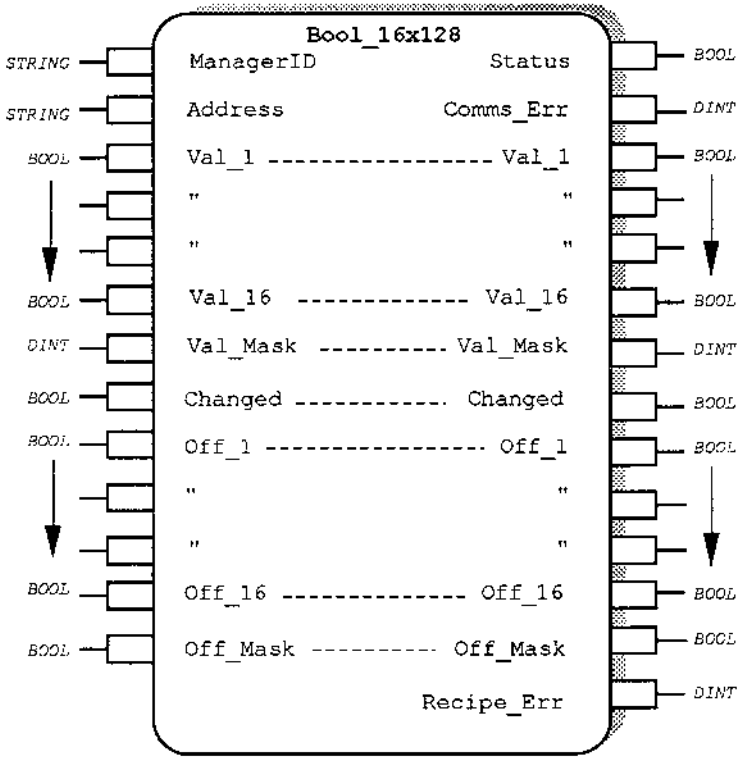


Figure 14-9 Schéma du bloc fonction Bool_16x128

Description fonctionnelle

Le bloc fonction Bool _16x128 est utilisé pour les données de recettes booléennes. Chaque bloc peut accepter un maximum de 16 variables de recettes et de 128 recettes. Il faut aussi créer un bloc fonction Recipe Manager ou Stage Manager pour réguler le système de recettes.

Attributs du bloc fonction

Type : 3E90

Classe : ..... RECIPE

Tâche par défaut : ..... Task_2

Liste résumée : ..... Manager ID, Status

Mémoire nécessaire : ..... 3088 octets

Description des paramètres

ManagerID (ID)

ID est un caractère unique servant à associer le bloc fonction Recipe Slave à un bloc fonction Recipe Manager ou Stage Manager. Ainsi, un bloc fonction Recipe Slave dont le ManagerID est 'A' est associé au bloc fonction Recipe ou Stage Manager dont l'ID est également 'A'.

Address (A)

Address définit l'adresse de communication permettant d'accéder aux matrices de données internes pour la lecture et l'écriture directes des données de recettes.

Status (S)

Positionné sur Go si le système de recettes fonctionne normalement. Si une erreur s'est produite en raison d'une transaction de recette ou de communication, Status est positionné sur NOGO et la cause de l'erreur est indiquée par Comms_Err ou Recipe_Err.

Comms_Err (CE)

Donne le numéro d'erreur pour une erreur de communication lorsque Status est NOGO. Se reporter au driver esclave de communication correspondant dans le chapitre 3 pour avoir une explication des numéros d'erreur.

Recipe_Err (RE)

Donne le numéro d'erreur pour une recette lorsque Status est NOGO.

1 = recette hors plage

2 = recette non définie

3 = recette par défaut non définie

4 = chargement de recette hors plage

5 = suppression de recette hors plage

Val_1(V1) à Val_16(V16)

Valeur de recette effective pour les paramètres 1 à 16. On peut écrire dedans pour modifier la valeur de la sortie mais cette modification est permanente, sauf si l'on recharge la recette à l'aide de la broche Load sur le gestionnaire de recettes ou d'étapes.

Val_Mask (VM)

Le masque spécifie les valeurs à modifier lorsque la recette est sortie. Chaque paramètre est représenté par un bit dans le masque. Par exemple, un emplacement de recette dont le masque est 15 (0000000000001111 en binaire) agit uniquement sur les valeurs 1 à 4.

Changed (C)

Positionné sur Changed lorsqu'une nouvelle recette a été sortie. Ce paramètre doit être réinitialisé par le programme.

Off_1(O1) to Off_16(O16)

Les broches de modification hors ligne servent à modifier une recette. L'écriture et la lecture des paramètres dans les broches hors ligne sont commandées par les broches Off-Op et Off_Ld sur le bloc fonction Recipe Manager ou Stage Manager.

OffMask(OM)

Valeur de masque pour la recette hors ligne.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
ManagerID	STRING	Null	Oper	Oper		
Address	STRING	Null	Super	Super		
Val_1 to Val_16	BOOL	Off (0)	Oper	Oper	Sens	On (1) Off (0)
Val_Mask	DINT	0	Oper	Super	Limite haute Limite basse	65536 0
Changed	BOOL	Off (0)	Oper	Oper	Sens	Changed (1) Off (0)
Off_1 to Off_16	BOOL	Off (0)	Oper	Oper	Sens	On (1) Off (0)
OffMask	DINT	0	Oper	Super	Limite haute Limite basse	65536 0
Status	BOOL	NOGO (0)	Oper	Block	Sens	NOGO (1) Go (0)
Comms_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0
Recipe_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0

Tableau 14-5 Attributs des paramètres Bool_16x128

BLOC FONCTION REAL_16X128

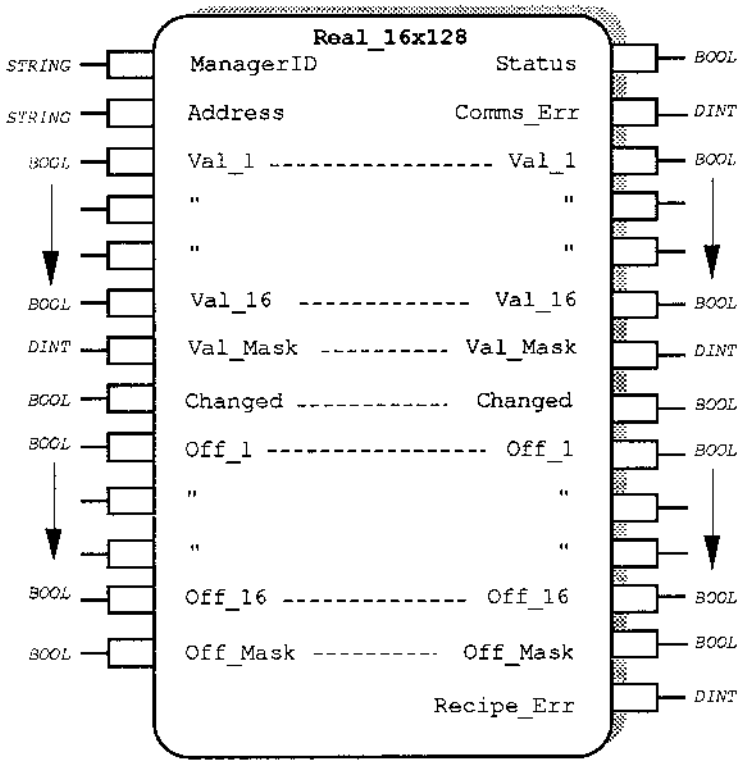


Figure 14-10 Schéma du bloc fonction Real_16x128

Description fonctionnelle

Le bloc fonction Real_16x128 est utilisé pour les données de recettes réelles. Chaque bloc peut accepter un maximum de 16 variables de recettes et de 128 recettes. Il faut aussi créer un bloc fonction Recipe Manager ou Stage Manager pour réguler le système de recettes.

Attributs du bloc fonction

Type : 3EA0

Classe : ..... RECIPE

Tâche par défaut : ..... Task_2

Liste résumée : ..... Manager ID, Status

Mémoire nécessaire : ..... 9376 octets

Description des paramètres

Cf. Description des paramètres de Bool_16x128

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
ManagerID	STRING	Null	Oper	Oper		
Address	STRING	Null	Super	Config		
Val_1 to Val_16	REAL		Oper	Oper	Limite haute Limite basse	1000000 -1000000
Val_Mask	DINT	0	Oper	Super	Limite haute Limite basse	65536 0
Changed	BOOL	Off (0)	Oper	Oper	Sens	Changed (1) Off (0)
Off_1 to Off_16	REAL		Oper	Oper	Limite haute Limite basse	1000000 -1000000
OffMask	DINT	0	Oper	Super	Limite haute Limite basse	65536 0
Status	BOOL	NOGO (0)	Oper	Block	Sens	NOGO (0) Go (1)
Comms_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0
Recipe_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0

Tableau 14-6 Attributs des paramètres Real_16x128

BLOC FONCTION DINT_16X128

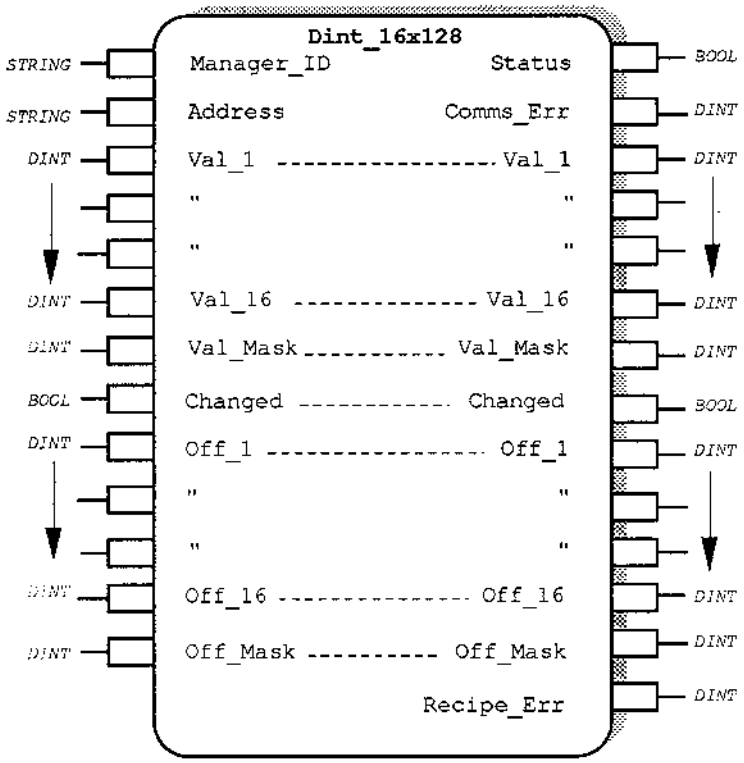


Figure 14-11 Schéma du bloc fonction Dint _16x128

Description fonctionnelle

Le bloc fonction Dint_16x128 est utilisé pour les données de recettes entières. Chaque bloc peut accepter un maximum de 16 variables de recettes et de 128 recettes. Il faut aussi créer un bloc fonction Recipe Manager ou Stage Manager pour réguler le système de recettes.

Attributs du bloc fonction

Type : 3EB0

Classe : ..... RECIPE

Tâche par défaut : ..... Task_2

Liste résumée : ..... Manager ID, Status

Mémoire nécessaire : ..... 9424 octets

Description des paramètres

Cf. Description des paramètres Bool_16x128

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
ManagerID	STRING	Null	Oper	Oper		
Address	STRING	Null	Super	Config		
Val_1 to Val_16	DINT	0	Oper	Oper	Limite haute Limite basse	1000000 -1000000
Val_Mask	DINT	0	Oper	Super	Limite haute Limite basse	65536 0
Changed	BOOL	Off (0)	Oper	Oper	Sens	Changed (1) Off (0)
Off_1 to Off_16	DINT	0	Oper	Oper	Limite haute Limite basse	1000000 -1000000
OffMask	DINT	0	Oper	Super	Limite haute Limite basse	65536 0
Status	BOOL	NOGO (0)	Oper	Block	Sens	NOGO (0) Go (1)
Comms_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0
Recipe_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0

Table 14-7 Attributs des paramètres Dint _16x128

BLOC FONCTION STR_1X128

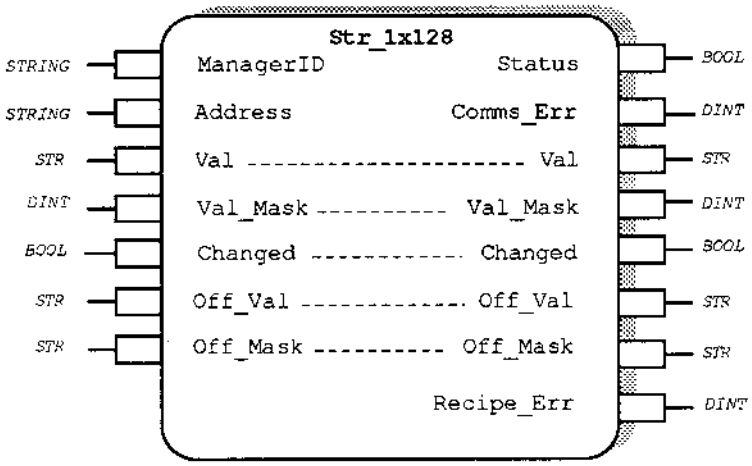


Figure 14-12 Schéma du bloc fonction Str_1x128

Description fonctionnelle

Le bloc fonction Str_1x128 est utilisé pour les données de recettes de type chaîne de caractères. Chaque bloc peut accepter une variable de recettes et un maximum de 128 recettes. Il faut aussi créer un bloc fonction Recipe Manager ou Stage Manager pour réguler le système de recettes.

Attributs du bloc fonction

Type : 3EC0

Classe : ..... RECIPE

Tâche par défaut : ..... Task_2

Liste résumée : ..... Manager ID, Status, Value

Mémoire nécessaire : ..... 4704 octets

Description des paramètres

ManagerID (ID)

ID est un caractère unique qui sert à associer le bloc fonction Recipe Slave à un bloc fonction Recipe Manager ou Stage Manager. Ainsi, un bloc fonction Recipe Slave dont le ManagerID est 'A' est associé au bloc fonction Recipe ou Stage Manager dont l'ID est également 'A'.

Address (A)

Address définit l'adresse de communication permettant d'accéder aux matrices de données internes pour la lecture et l'écriture directes des données de recettes.

Status (S)

Positionné sur Go si le système de recettes fonctionne normalement. Si une erreur s'est produite en raison d'une transaction de recette ou de communication, Status est positionné sur NOGO et la cause de l'erreur est indiquée par Comms_Err ou Recipe_Err.

Comms_Err (CE)

Donne le numéro d'erreur pour une erreur de communication lorsque Status est NOGO. Se reporter au driver esclave de communication correspondant dans le chapitre 3 pour avoir une explication des numéros d'erreur.

Recipe_Err (RE)

Donne le numéro d'erreur pour une recette lorsque Status est NOGO.

1 = recette hors plage

2 = recette non définie

3 = recette par défaut non définie

4 = chargement de recette hors plage

5 = suppression de recette hors plage

Val(V)

Valeur de recette effective. On peut écrire dedans pour modifier la valeur de la sortie mais cette modification est permanente, sauf si l'on recharge la recette à l'aide de la broche Load sur le gestionnaire de recettes ou d'étapes.

Val_Mask (VM)

Le masque spécifie si la sortie doit changer lors de la sortie de la recette.

Changed (C)

Positionné sur Changed lorsqu'une nouvelle recette a été sortie. Ce paramètre doit être réinitialisé par le programme.

Off_Val(OV)

La broche de modification hors ligne sert à modifier une recette. L'écriture et la lecture des paramètres dans la broche hors ligne sont commandées par les broches Off-Op et Off_Ld sur le bloc fonction Recipe Manager ou Stage Manager.

OffMask(OM)

Valeur de masque pour la recette hors ligne.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
ManagerID	STRING	Null	Oper	Oper		
Address	STRING	Null	Super	Oper		
Val	STRING	Null	Oper	Oper		
Mask	BOOL	Off (0)	Oper	Oper	Sens	On (1) Off (0)
Changed	BOOL	Off (0)	Oper	Oper	Sens	Changed (1) Off (0)
Off_Val	STRING	Null	Oper	Oper		
Off_Mask	BOOL	Off (0)	Oper	Oper	Sens	On (1) Off (0)
Status	BOOL	NOGO (0)	Oper	Block	Sens	NOGO (0) Go (1)
Comms_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0
Recipe_Err	DINT	0	Oper	Block	Limite haute Limite basse	255 0

Tableau 14-8 Attributs des paramètres Str_1x128

Chapitre 14

RECETTES

Edition 2

Présentation des recettes

Ces blocs offrent un moyen d'utiliser les systèmes de recettes dans le PC3000. Les recettes peuvent être créées, modifiées, stockées et chargées localement ou depuis un système de surveillance. La bibliothèque comprend un bloc fonction de commande, le 'gestionnaire de recettes', qui est responsable d'un certain nombre d'esclaves de recettes' qui lui sont affectés.

Le gestionnaire de recettes fournit le mécanisme de sélection des recettes et l'interface de commande.

Les esclaves de recettes fournissent le stockage des données des recettes, ils sont identiques aux matrices bidimensionnelles mais avec des dimensions fixes : 16 valeurs en largeur par 128 valeurs en longueur. Différents types de données sont acceptés :

booléen, réel, entier et chaîne de caractères

Il existe également un mécanisme de séquencement du temps pour une liste prédéfinie de recettes. Ce mécanisme est appelé gestionnaire d'étapes.

Utilisation

Ce système de recettes PC3000 repose sur deux principes.

Le premier consiste à fournir un stockage de recettes sans qu'il soit nécessaire d'avoir un système de surveillance. Dans ce cas, les recettes sont stockées dans la RAM de l'unité centrale et sont utilisées lorsque cela est nécessaire. Les blocs permettent la modification hors ligne de n'importe quelle recette, même lorsqu'une autre recette est sélectionnée.

La deuxième utilisation fait appel à un système de surveillance, où le problème réside dans le temps de chargement d'une recette à l'autre. L'ensemble des recettes sont chargées depuis le système de surveillance avant que le procédé puisse démarrer. Au cours du procédé, Stage_Manager sert à classer les recettes demandées dans le bon ordre sans qu'il soit nécessaire d'utiliser d'autres communications.

Dans ces deux cas, les données des recettes sont stockées dans la RAM du LCM et sont par conséquent perdues à chaque réinitialisation de l'unité centrale, par exemple après une reconstitution et un chargement. Tant que les données de la recette ne sont pas saisies, il n'y a aucune recette valable.

Pour stocker les recettes en cas de réinitialisation ou de démarrage à froid, il faut stocker les données de ces recettes quelque part. Cela ne pose aucun problème dans le PC3000 avec le système de surveillance qui contient un disque dur, mais cela pourrait être gênant avec le PC3000 autonome. Dans ce cas, le système d'enregistrement fichier est la seule solution possible.

Les blocs fonctions Prog8 possèdent leur propre système de recettes intégré qui comprend l'enregistrement fichier, à utiliser si besoin est.

La seule autre possibilité consiste à produire une macro dans le programme d'application qui stocke régulièrement la recette à un autre endroit en fonctionnement normal et qui, au démarrage à froid, récupère les données dans le système de fichiers avant de reprendre ses fonctions normales.

Dans ce cas, stocker les données dans des chaînes de caractères à l'aide des fonctions COMPACTES REP_REAL_IN_STR et EXT-REAL-FROM-STR, etc.)

Définitions

Recette :	une recette est l'état défini par l'ensemble des sorties associées à un gestionnaire de recettes ou d'étapes.
Gestionnaire :	le terme de gestionnaire seul est utilisé dans ce document pour décrire un bloc fonction gestionnaire d'étapes ou gestionnaire de recettes.
Séquence :	la séquence est la liste de recettes stockées dans le gestionnaire de recettes, boucles et points finaux compris.
Etape :	une étape est un élément unique d'une séquence, composé d'un numéro de recette et d'informations sur la boucle.
Recette en ligne :	la recette en ligne est l'ensemble des valeurs qui ont un effet sur le procédé.
Recette hors ligne :	la recette hors ligne est l'ensemble des valeurs utilisées pour modifier le contenu de la recette sans agir sur le procédé.
Chargement sélectif :	"chargement sélectif" est le terme qui désigne le fait de recharger les valeurs de recettes en ligne dans une recette.

Exemple d'application

Une stratégie de régulation type est régie par un certain nombre de paramètres de types différents. Nous prendrons comme exemple de système une boucle de régulation analogique couplée à une sortie numérique. La boucle analogique contient un bloc fonction rampe et la sortie est déclenchée par un timer. Dans une application type, cette configuration se répète un certain nombre de fois (zones d'une extrudeuse par exemple). La figure 14-1 montre une boucle simple.

Dans cet exemple, les paramètres qui régulent un lot donné sont les suivants :

- consigne de rampe
- vitesse de rampe
- état de sortie numérique
- temps de lot

La fonction d'un système de recettes consiste à configurer ces paramètres et à synchroniser le début et la fin d'un lot. Si l'on utilisait un système de surveillance, avant le début de chaque lot, il faudrait charger ces paramètres pour chaque boucle à l'aide d'une liaison de communications avec le PC3000. Une fois dans le PC3000, le système de surveillance signalerait le début du lot.

Pour mettre en oeuvre le système de recettes représenté sur la figure 14-1, il faut avoir deux types de blocs fonctions : les gestionnaires de recettes et les esclaves de recettes.

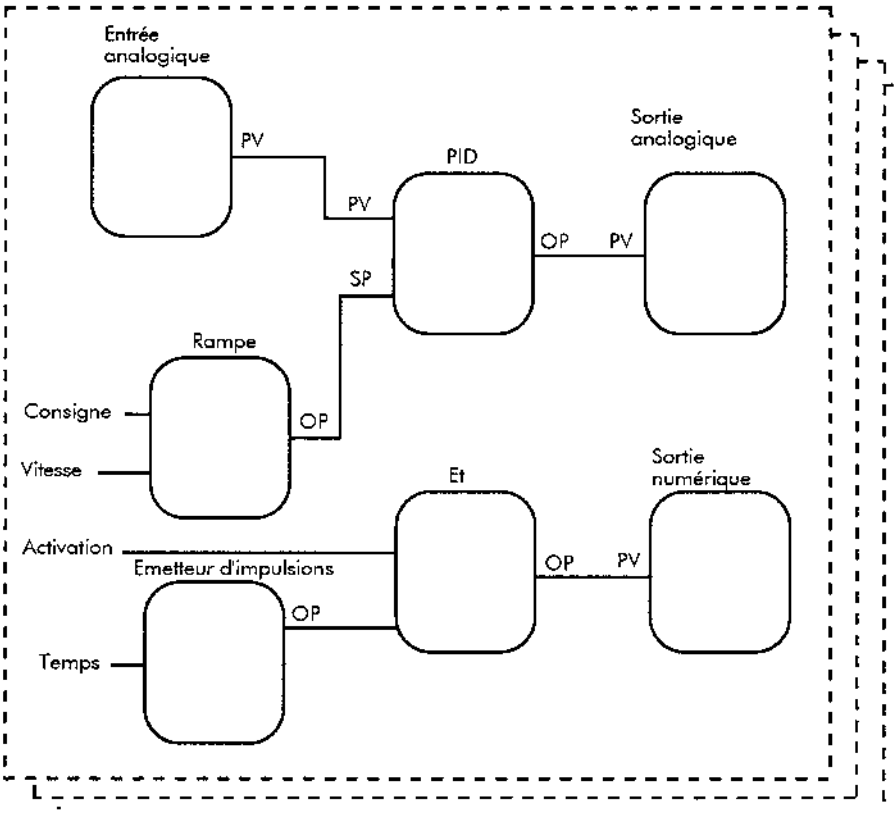


Figure 14-1 Exemple de système de recettes

Esclaves de recettes

Les blocs fonctions esclaves de recettes constituent l'enregistrement des données du système de recettes. Les esclaves de recettes sont définis par leur type de données (par exemple entier double, réel), le nombre de sorties par bloc et le nombre de recettes qui peuvent être enregistrées. Par exemple, un bloc de recettes de type réel qui accepte 16 paramètres et peut enregistrer 128 recettes est appelé Real _16 x 128

Les blocs fonctions esclaves de recettes fournissent aussi l'interface de communications dans le système de recettes en agissant comme variables esclaves de communications.

Gestionnaires de recettes

Le gestionnaire de recettes gère les esclaves de recettes. Il commande la recette qu'ils sortent. Il agit aussi sur des utilitaires de modification des recettes.

Dans certaines applications, il est nécessaire de faire fonctionner simultanément plusieurs systèmes de recettes. Par exemple, pour combiner trois liquides dans des quantités différentes, il peut être nécessaire d'avoir des consignes de température pour les trois liquides (qui changent rarement) et des réglages de vannes pour les trois liquides (qui changent à chaque lot). Au lieu de répéter les consignes de température dans chaque recette, il est possible de scinder l'ensemble en deux recettes : une pour les températures et une pour les débits.

Pour faciliter cette opération, il y a un caractère d'identification (ID) pour chaque système de recettes présent dans le PC3000. Dans chaque système de recettes, il y a un gestionnaire de recettes et un certain nombre d'esclaves de recettes portant tous le même ID.

Par conséquent, chaque gestionnaire de recettes doit avoir un ID unique pour permettre de faire la distinction entre les recettes.

Dans la plupart des applications, il y a un seul gestionnaire. Tous les esclaves de recettes ont le même ID que ce gestionnaire.

Gestionnaires de recettes

Le gestionnaire de recettes effectue les tâches associées à l'ensemble des esclaves de recettes qu'il commande.

Il s'agit par exemple du chargement sélectif des valeurs de la recette actuelle (c'est-à-dire le chargement des valeurs en ligne du système dans une recette), de la suppression de recettes isolées ou de l'ensemble de l'enregistrement de recettes, de la modification de la recette actuellement sortie ou de l'inhibition des modifications de la recette. Le gestionnaire possède un ID servant à associer des blocs fonctions d'esclaves de recettes à des gestionnaires donnés et un numéro de recette.

L'entrée du numéro de recette positionne la recette sur les sorties des blocs fonctions de l'esclave de recettes. Lors de la première exécution, les blocs

fonctions de l'esclave de recettes se déclarent dans les gestionnaires de recettes. Lorsque l'entrée du numéro de recette du gestionnaire de recettes change, le gestionnaire déclare cette nouvelle recette à l'ensemble de ses esclaves associés.

Le gestionnaire de recettes peut effectuer les tâches suivantes :

- il peut provoquer un changement de la recette en ligne actuelle. Pour cela, il faut mettre sur 1 l'entrée/sortie Force ou changer le numéro de recette. Le fait de mettre sur 1 le paramètre booléen Force provoque une nouvelle sortie de la recette actuelle.
- il peut provoquer un chargement de la recette en ligne actuelle vers l'enregistrement de recettes. De cette manière, l'utilisateur peut apporter des modifications à la recette actuelle et les enregistrer dans une recette sur le PC3000. On appelle cette opération le 'chargement sélectif'.
- il peut déclarer une recette donnée comme non valable. Une recette est uniquement considérée comme valable si une recette a été configurée à l'intérieur. Une tentative de changement d'une recette non valable ne peut pas réussir. Le gestionnaire de recettes demande à chaque esclave de recettes la nouvelle recette. Lors de l'exécution suivante de l'esclave, celui-ci vérifie si la recette est valable et le signale au gestionnaire. Lors de l'exécution suivante du gestionnaire, si tous les esclaves de recettes associés ont des données valables pour cette recette, le gestionnaire indique à l'ensemble des esclaves d'effectuer le changement. Dans le cas contraire, aucun des esclaves de recettes n'est invité à effectuer le changement de recette.
- il peut sortir une recette vers les valeurs hors ligne. Ces valeurs servent à la modification des recettes depuis le PC3000. Elles offrent un mécanisme permettant de sortir, de visualiser et de modifier les recettes sans agir sur la recette en ligne.
- il peut charger les valeurs hors ligne dans une recette.
- il peut inhiber les changements des recettes enregistrées.

Esclaves de recettes

Les esclaves de recettes effectuent la manipulation des données pour le système de recettes. Ils sont composés en interne d'une mémoire tampon de communications et d'un ensemble d'enregistrement des données.

Les paramètres des recettes sont sortis vers le programme utilisateur par les blocs fonctions d'esclaves de recettes. Chacun de ces blocs fonctions possède un ensemble de données internes relatives aux différentes recettes. Chaque ensemble de données est associé à un gestionnaire de recettes et possède une adresse de communications pour la configuration et la surveillance.

Des fonctions supplémentaires permettent de faire des copies d'une recette à une autre et de spécifier les valeurs qui changent dans une recette donnée. Il existe aussi un ensemble de valeurs relatives à la recette ``hors-ligne" qui peut servir à modifier le contenu de l'ensemble de données sans avoir d'effet sur le procédé.

Valeurs des recettes en ligne

Les valeurs des recettes en ligne sont les paramètres qui doivent être liés au système.

Valeurs des recettes hors ligne

Les valeurs des recettes hors ligne sont les paramètres qui peuvent être modifiés sans avoir d'effet sur le procédé en cours d'exécution. Elles offrent une possibilité simple de modifier les recettes à l'aide du PC3000 proprement dit.

Masque de recettes

Il existe des applications dans lesquelles les valeurs sont configurées au début de l'exécution d'un procédé mais sont modifiées manuellement au cours de l'exécution du procédé, soit pour régler finement le procédé avant de charger sélectivement les valeurs en ligne soit parce qu'une commande manuelle est nécessaire. Dans cette situation, il est nécessaire de sortir une valeur de recette une fois, au début de l'exécution, mais sans écraser la valeur lorsque la recette changera ultérieurement.

Pour cette fonction, il existe un système de masques de recettes.

La valeur d'une recette change UNIQUEMENT si le bit du masque est POSITIONNE SUR 1 pour cette valeur.

Par exemple, un bloc fonction Réel a un masque de recette 15 (0000000000001111 en binaire) pour la recette numéro 20. Cela signifie que, lorsqu'on effectue des changements sur la recette 20, les valeurs des sorties 5 à 16 vont rester inchangées. Seules les valeurs des sorties 1 à 4 changeront.

Recettes valables et non valables

Au démarrage à froid, aucune recette de l'esclave de recettes n'est valable : aucune ne contient de données et il est risqué de sortir leurs valeurs comme recette. Une fois que l'on a écrit dans une recette, soit à l'aide des communications lors du chargement de la recette en ligne en cours, soit par le chargement de la recette hors ligne actuelle, cette recette est considérée comme valable dans cet esclave. Si l'ensemble des esclaves associés à un gestionnaire ont une recette valable au même numéro, cette recette est considérée comme valable dans sa totalité. Pour transformer une recette donnée en recette non valable, il faut que le gestionnaire associé active Clear et le numéro de recette nécessaire.

Tampon de communications

En interne, l'esclave de recettes contient une variable esclave multi-éléments à 19 paramètres. Cette variable est interprétée par le bloc comme un pointeur de charge, un pointeur de destination, un masque et 16 valeurs.

Pointeur de charge

L'écriture d'un numéro de recette dans le pointeur de charge provoque la copie, par l'esclave de recettes, de cette recette (c'est-à-dire 16 valeurs et un masque) dans la recette indiquée par le pointeur de destination pour les valeurs de pointeur de destination comprises entre 1 et 128. Toute valeur supérieure à 128 ou inférieure à 0 est non-valable. 0 sert de numéro spécial indiquant que cette transaction est une charge provenant des entrées/sorties de valeurs.

Pointeur de destination

Le pointeur de destination sélectionne la recette qui fait l'objet des lectures et des écritures. La recette 0 est une recette spéciale indiquant les entrées/sorties de valeurs. Pour la lecture des valeurs d'une recette donnée, il faut pour commencer écrire le numéro de cette recette dans le pointeur de destination.

Masque

Définit le masque de changement de sortie pour cette recette. Le bit de poids le plus faible se rapporte à la valeur 1, le bit de poids le plus fort à la valeur 16 (par exemple, une valeur de masque de 1000000000000001 en binaire permet uniquement la mise à jour des valeurs 16 et 1 pour cette recette).

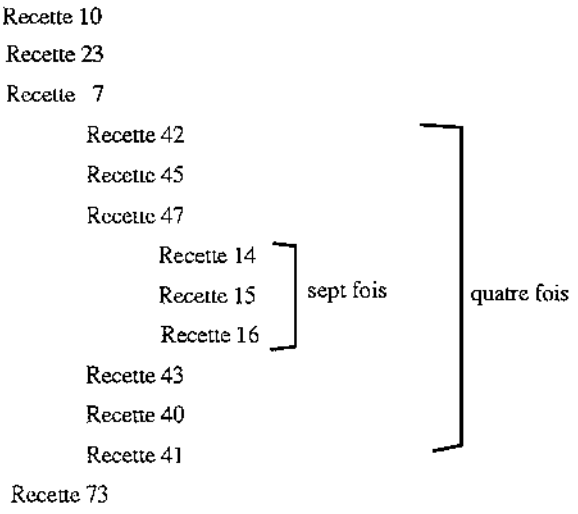
Valeurs

Le masque est suivi des 16 valeurs de la recette actuelle. Il est possible de les laisser comme valeurs par défaut ou de les écraser. Les valeurs par défaut sont une chaîne nulle pour le type chaîne de caractères et zéro pour les types réel, booléen et entier.

Gestionnaires d'étapes

Le gestionnaire d'étapes possède une liste de séquences internes accessible par les communications ou le programme utilisateur. En mode automatique, un flanc ascendant sur l'entrée de l'horloge fait passer le gestionnaire interne de recettes sur la recette suivante de la liste de séquences. Cette liste peut également prendre en charge un maximum de quatre niveaux imbriqués de bouclage.

Dans certains systèmes, il faut passer successivement par un certain nombre de recettes. Prenons par exemple la liste de recettes ci-après :



Avec le bloc fonction gestionnaire d'étapes, il est possible de spécifier de cette manière une séquence de recettes.

La figure 14-7 montre le schéma du bloc fonction RecipeMan.

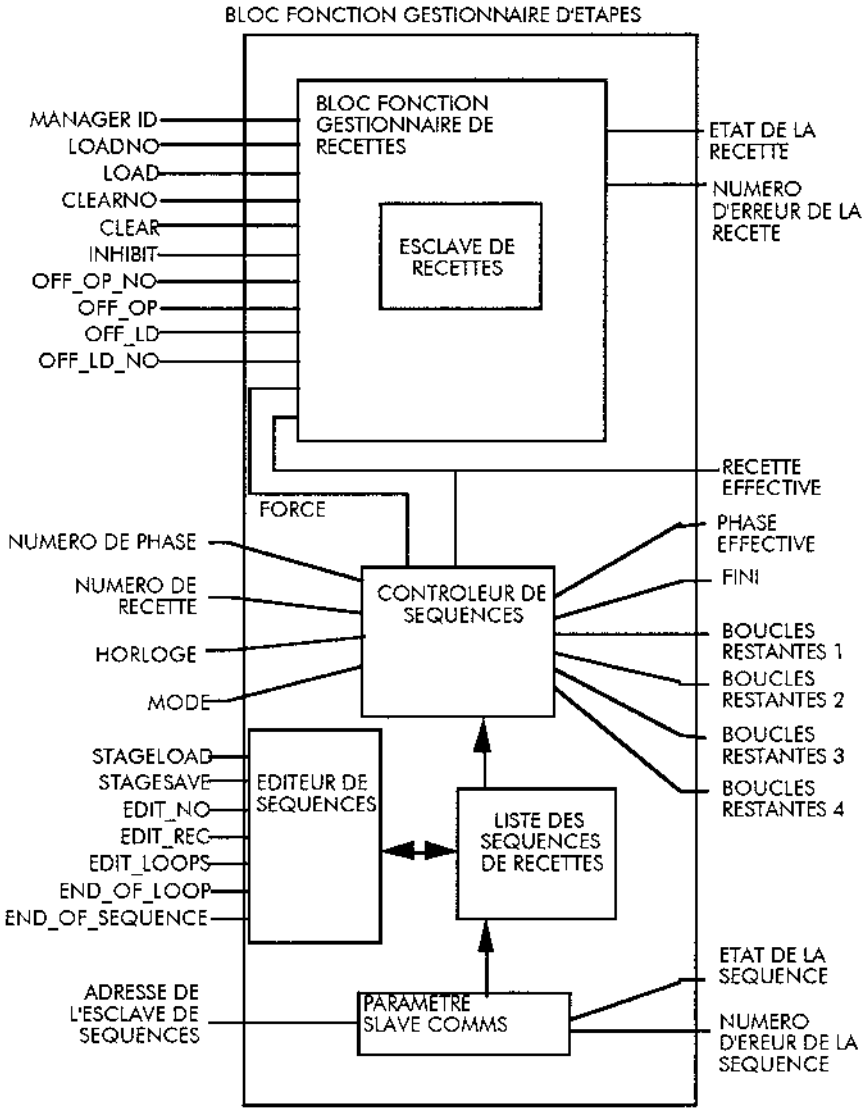


Figure 14-2 Bloc fonction Gestionnaire d'étapes

En interne, le gestionnaire d'étapes comporte deux parties fonctionnelles : un bloc fonction Gestionnaire de recettes situé dans le gestionnaire d'étapes et le gestionnaire de séquences. Ce gestionnaire de séquences doit choisir la recette suivante à sortir en fonction de la liste de séquences, qui est accessible par les communications (cf. chapitre 3 annexe C).

Chaque étape de la liste contient les informations suivantes :

Numéro d'étape : le numéro d'étape est l'emplacement de cette étape dans la liste d'étapes. Le gestionnaire de séquences du gestionnaire d'étapes travaille à partir de l'étape 1 par ordre numérique croissant, sauf lorsqu'il travaille avec des boucles.

Numéro de recette : le numéro de recette est la recette à sortir pour cette étape.

Comptage de boucle : si aucune boucle ne commence sur cette étape, le comptage de boucle est fixé à 0. Si une boucle commence sur cette étape, le nombre de boucles est fixé dans ce champ.

Fin de la boucle : si cette étape est la fin de la boucle actuelle, l'indicateur de fin de boucle est positionné sur 1.

Fin de la séquence : si cette étape est la dernière de la séquence, cet indicateur est positionné sur 1.

Par exemple, pour exécuter la séquence présentée antérieurement :

Étape	Recette	Boucles	Début de la boucle	Fin de la séquence
1	10	0	Off	Off
2	23	0	Off	Off
3	7	0	Off	Off
4	42	4	Off	Off
5	45	0	Off	Off
6	47	0	Off	Off
7	14	7	Off	Off
8	15	0	Off	Off
9	16	0	On	Off
10	43	0	Off	Off
11	40	0	Off	Off
12	41	0	On	Off
13	73	0	Off	On

Tableau 14-1 Informations des étapes

Il est possible de passer à une étape de la séquence en positionnant sur 1 le paramètre StageNo et en positionnant le mode sur Jump(3). Si le passage a lieu au milieu d'une boucle, la fin des boucles ne sera pas prise en compte. Par exemple, si l'on passe à l'étape 8 de la séquence ci-dessus, le gestionnaire d'étapes fait défiler dans l'ordre les étapes 8 à 13 en ne tenant pas compte de la fin de la boucle aux étapes 9 et 12.

Exemple d'utilisation du système de recettes du PC3000

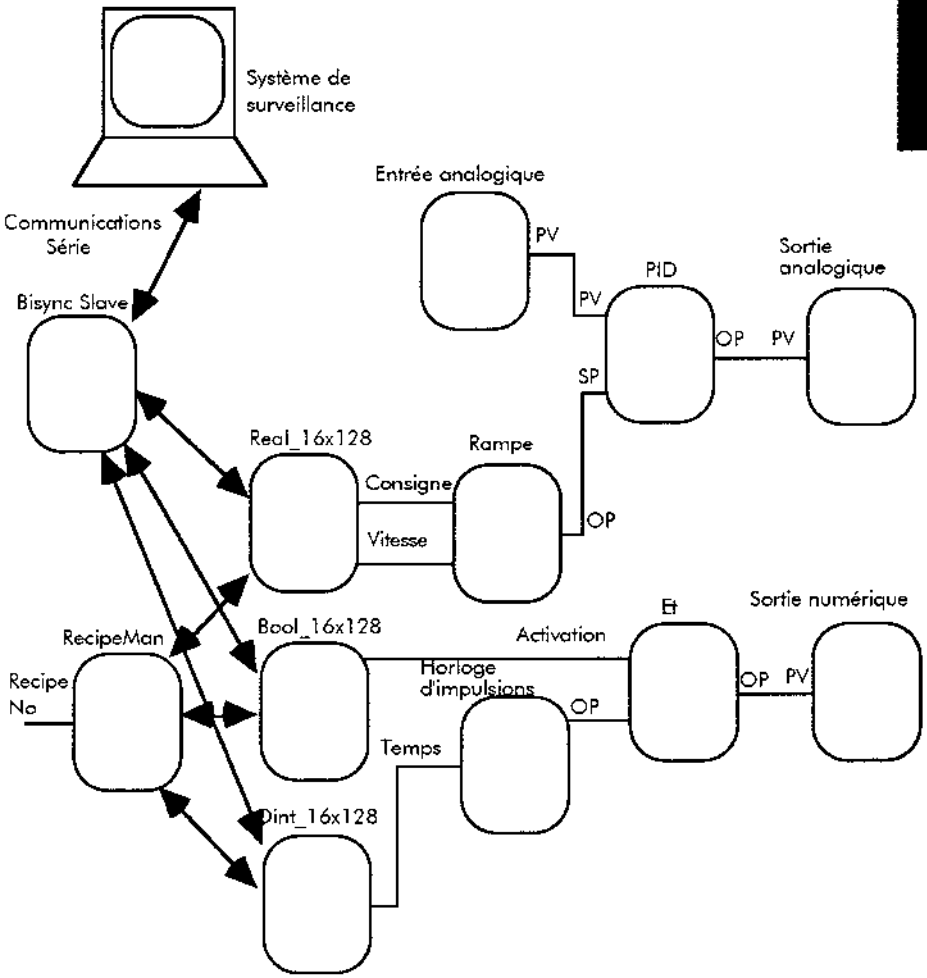


Figure 14-3 Exemple de système de recettes PC3000

En prenant comme exemple le système de la page 14-iv, il faut un seul gestionnaire de recettes et des esclaves de recettes de type Entier double, Réel et Booléen. Les esclaves de recettes ont tous le même ID de gestionnaire et, par conséquent, lorsque le numéro de recette change sur le gestionnaire, la recette change sur tous les blocs fonctions esclaves.

Pour faciliter les communications avec un système de surveillance, il faut une déclaration d'un driver de communications, le bloc fonction EI par exemple.

Le bloc fonction RecipeMan reçoit un ID de '1' et tous les blocs fonctions esclaves un ManagerID de '1'.

Pour les adresses des blocs esclaves, nous utilisons les adresses EI Bisync. Réel reçoit une adresse 'EBd00', Booléen une adresse 'EBb00' et Entier double une adresse 'EBd00'

La figure 14-3 montre le câblage des blocs fonctions. Dans un système à quatre boucles, on peut utiliser Val_1 et Val_2 respectivement pour la consigne et la vitesse de rampe de la première boucle, Val_3 et Val_4 pour la deuxième boucle, etc.

Sans raccordement à un système de surveillance, ce système a immédiatement un certain nombre de fonctions. Par exemple, il serait possible de configurer manuellement les quatre consignes, les quatre vitesses de rampes, les paramètres booléens et le temps de procédé. Ensuite, si l'on positionne sur 20 l'entrée du bloc fonction RecipeMan et sur "charge" l'entrée/sortie Load, les valeurs des sorties seront chargées dans la recette numéro 20. Si l'on modifie ensuite les valeurs, il est possible de récupérer les valeurs précédentes en positionnant RecipeNo sur 20. Si RecipeNo est déjà égal à 20, le bloc ne peut détecter aucun changement et il est par conséquent nécessaire de positionner sur 1 l'entrée/sortie Force qui force le bloc de recettes à ressortir la recette.

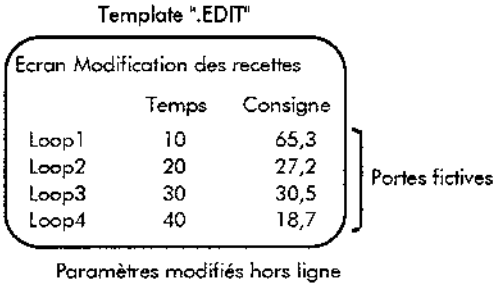


Figure 14-4 Ecran Modification des recettes

Intégration des systèmes de recettes ESP et PC3000

Il est possible d'intégrer le système de recettes PC3000 au système de recettes Eurotherm Supervisory Package (ESP) et d'utiliser la modification, la sauvegarde et le chargement des recettes ESP pour transférer les recettes au PC3000.

L'exemple ci-dessous reprend l'exemple précédent et montre la manière d'y ajouter une interface de recettes ESP.

Introduction

L'interface à présenter est composée de deux écrans principaux : un écran de modification des recettes et un écran de chargement des recettes. L'écran de modification des recettes permet à l'utilisateur de modifier les valeurs des recettes hors ligne et de les sauvegarder sous la forme de fichiers de recettes ESP. L'écran en ligne montre simultanément une recette hors ligne et une recette PC3000 et permet le chargement depuis ESP et la modification en ligne de la recette PC3000.

Configuration des paramètres de portes

Les paramètres de portes sont configurés avec les adresses présentées dans le chapitre 3 annexe B. Les portes qui doivent être présentes sont le numéro de recette pour chaque bloc, les valeurs des recettes et le masque de recettes.

Si l'on prend un bloc de recettes de type entier, portant l'adresse 'EBA90', les portes nécessaires sont celles indiquées dans le tableau du chapitre 3 annexe B.

En plus des paramètres des esclaves de recettes, il faut avoir les paramètres des blocs fonctions du gestionnaire de recettes. Il est possible de les produire en marquant les portes dans la station de programmation PC3000 puis en sauvegardant ces portes.

Ecrans de modification des recettes

L'écran de modification (cf. la figure Modification) est uniquement composé de portes fictives. Il faut créer une porte fictive pour représenter chaque paramètre des recettes. Il est ensuite possible de modifier ces paramètres et de les sauvegarder sous la forme d'une recette ESP.

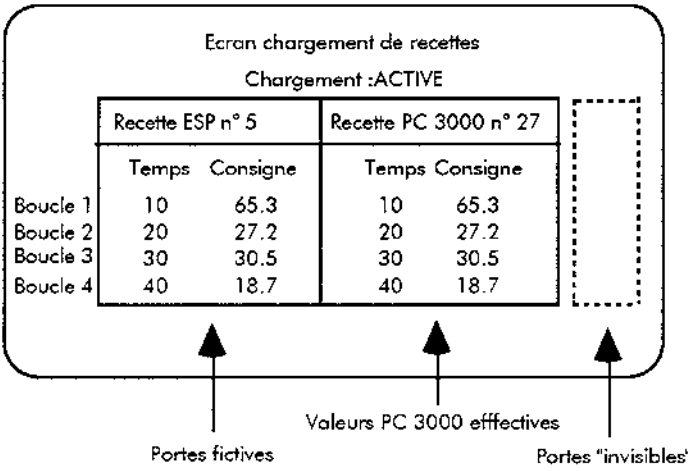


Fig 14-5 Ecran Chargement des recettes

Ecrans de chargement des recettes

L'écran Chargement des recettes (cf. figure Chargement) utilise les mêmes fichiers de recettes que l'écran Modification des recettes.

Il se compose de trois parties. Il y a les portes fictives qui apparaissent dans le gabarit de modification : elles servent à charger la recette ESP. Il y a pour cela détection d'un changement de la porte du numéro de recette ESP et chargement de la nouvelle recette à l'aide du langage Wizcon, qui permet de charger un fichier de recette portant un nom différent. Les portes de recettes PC3000 montrent la recette telle qu'elle est enregistrée dans le PC3000.

Il y a un ensemble de portes 'invisibles' (couleur du premier plan identique à la couleur du fond). Ce sont le pointeur de destination et le masque de recette pour les différents esclaves de recettes. Les pointeurs de destination sont mis à jour à l'aide du langage Wizcon de manière à être identiques à la porte du numéro de recette visible. Le masque de recette doit être positionné sur la valeur binaire 1111111111111111 (c'est-à-dire mise à jour systématique de l'ensemble des 16 valeurs).

Le mécanisme de chargement des recettes fonctionne en langage Wizcon. Il détecte le changement de porte ``Chargement'' puis copie le contenu des portes fictives dans les portes de recettes PC3000.

Utilisation d'EI Bisync pour communiquer avec un esclave de recettes

Cet exemple montre les communications avec un bloc fonction entier d'esclave de recettes

Imaginons qu'un bloc fonction de type entier double ait l'adresse 'EBA90'.

Imaginons que le bloc fonction EI ait un Gid égal à '0'.

Le bloc est configuré par les communications soit à l'aide d'un paramètre composite unique soit à l'aide des 19 paramètres de base suivants :

Adresse	
00 A90	Paramètre composite
00 A91	Destination
00 A92	Origine du chargement
00 A93	Masque
00 A94	Valeur 1
00 A95	Valeur 2
00 A96	Valeur 3
00 A97	Valeur 4
00 A98	Valeur 5
00 A99	Valeur 6
00 A9:	Valeur 7
00 A9;	Valeur 8
00 A9 <	Valeur 9
00 A9	Valeur 10
00 A9 >	Valeur 11
00 A9?	Valeur 12
00 A9@	Valeur 13
00 A9A	Valeur 14
00 A9B	Valeur 15
00 A9C	Valeur 16

Tableau 14-2 Paramètres de base

Les champs à l'intérieur du paramètre composite sont dans le même ordre que les différents paramètres du tableau ci-dessus.

Exécution, chronométrage et synchronisation

Exécution

Lorsqu'une commande est émise vers le gestionnaire de recettes, un certain nombre d'actions se produisent. Pour commencer, la commande (par exemple une demande de changement dans une nouvelle recette) est envoyée à l'ensemble des esclaves associés au gestionnaire. Lorsque chaque esclave est exécuté, il décide si la commande est valable (il vérifie par exemple si la nouvelle recette est valable). Lorsque le gestionnaire sera exécuté la prochaine fois, il vérifiera les réponses de l'ensemble de ses esclaves associés. Si l'un des esclaves ne peut pas effectuer l'action demandée, la tentative est arrêtée.

Si l'ensemble des esclaves peuvent effectuer l'action, le gestionnaire envoie une commande à chacun de ses esclaves associés pour leur demander d'effectuer l'action. Ensuite, lorsque chaque esclave s'exécute la fois suivante, il met ses sorties à jour en fonction des besoins.

Chronométrage

Le temps d'attente entre un changement de numéro de recette et un changement des sorties d'esclaves de recettes est de trois tops d'horloge au maximum. Ce cas le plus défavorable se produit lorsque l'exécution de l'esclave a lieu immédiatement avant l'exécution du gestionnaire et lorsque le changement du numéro de recette a lieu immédiatement après l'exécution du gestionnaire, comme le montre la figure ci-dessous.

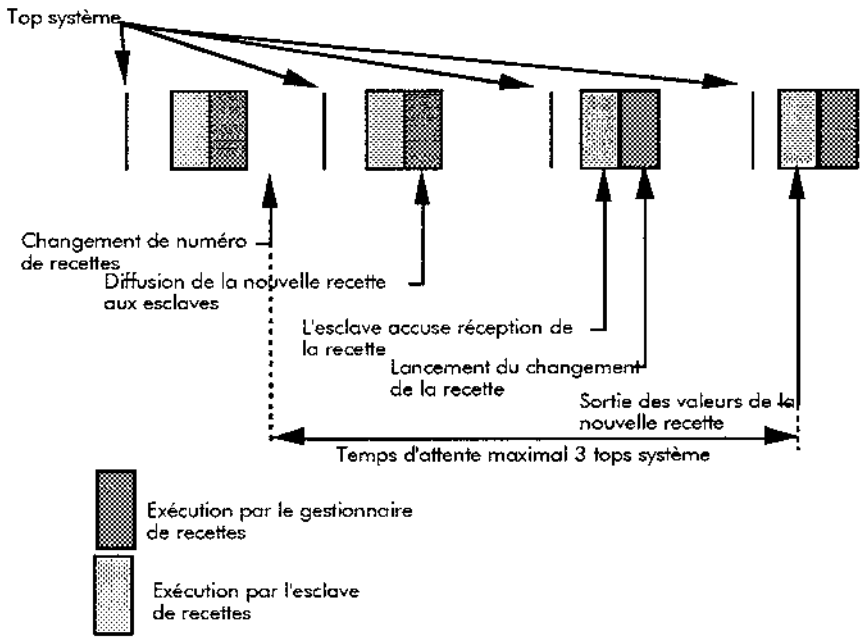


Fig 14-6 Chronométrage du gestionnnaire et des esclaves de recettes

Synchronisation

Rien ne garantit l'ordre d'exécution des blocs fonctions exécutés sur la même tâche: la station de programmation choisit l'ordre d'exécution en fonction du câblage entre les blocs. Pour cette raison, le changement de valeurs dans une recette n'a pas forcément lieu sur le même top d'horloge. Toutefois, des informations sont disponibles sous la forme de l'indicateur Modifié sur les esclaves de recettes. Cet indicateur est réglé par le bloc et uniquement remis à zéro par le programme utilisateur.

Chronométrage des communications

Lorsqu'EI Bisync a accès aux esclaves de recettes comme paramètres d'éléments simples, les écritures multiples qui ont lieu sur le même paramètre peuvent provoquer des erreurs de communications. Cela est dû au fait que l'esclave doit traiter séparément chaque transaction et est protégé en écriture jusqu'à ce qu'il ait traité la dernière transaction.

Il y a deux manières de faire face à cette situation : déplacer les esclaves de recettes vers une tâche plus rapide ou augmenter la durée entre chaque transaction vers un esclave donné. On peut pour cela imbriquer les écritures dans un certain nombre d'esclaves de recettes différents.

Chapitre 15

PROGRAMMATEUR

Edition 2

Présentation

Cette classe de blocs fonctions comprend à la fois des blocs fonctions simples en rampe servant à faire progresser la valeur d'une variable de façon linéaire en fonction du temps et des blocs fonctions de programmation plus complexes permettant de créer des profils à plusieurs segments rampe / palier. Ces deux types de blocs fonctions sont disponibles en version "temps de rampe" et "temps et niveau final".

BLOC FONCTION RAMP_TARGET

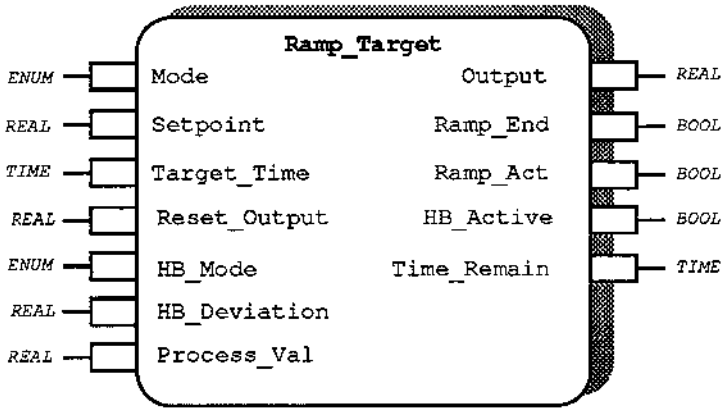


Figure 15-1 Schéma du bloc fonction Ramp_Target

Description fonctionnelle

Le bloc fonction Ramp_Target fait progresser **Output** vers une cible **Setpoint** de telle sorte que cette consigne soit atteinte dans le temps fixé par **Target_Time.** Le bloc a trois modes de fonctionnement définis par **Mode.** La fonction **Holdback** peut être activée pour limiter **Rate** dans le cas d'un paramètre **Process_Val** lent.

Holdback sert à maintenir **Output** à une valeur constante si l'écart entre **Output** et **Process_Val** dépasse la limite définie par **HB_Deviation.** La figure ci-après montre l'activation du maintien sur écart par suite du retard de **Process_Val** sur la rampe d'accroissement **Output.** Si l'entrée **Process_Val** s'écarte de la rampe **Output** de plus de **HB_Deviation, Output** est maintenu à une valeur constante jusqu'à ce que l'écart repasse en dessous de **HB_Deviation.** Dans le cas représenté, ceci a pour effet d'augmenter le temps nécessaire pour atteindre la consigne au-dessus de celle indiqué epar **Target_Time.**

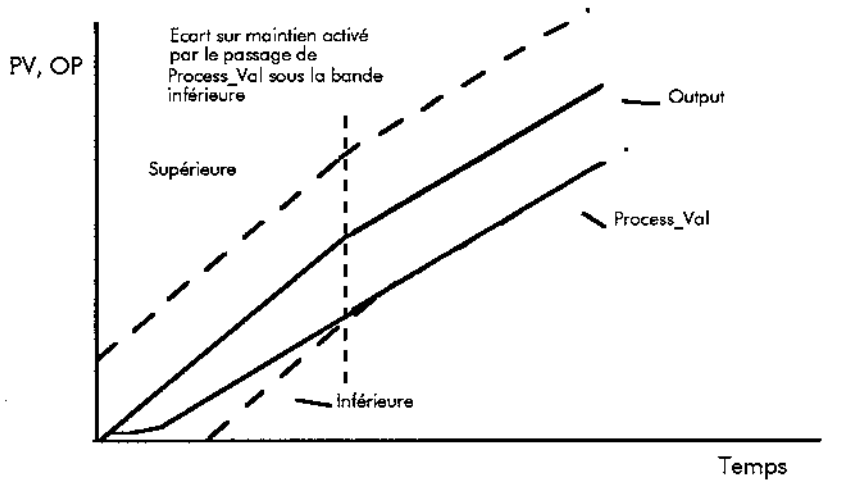


Figure 15-2 Fonctionnement de Holdback avec Process_Val en retard

Attributs du bloc fonction

Type : 3F03

Classe : .....**PROGRAMMATEUR**

Tâche par défaut : .....Task_2

Liste résumée : .....Mode, Setpoint, Target_Time output

Mémoire nécessaire : .....60 octets

Description des paramètres

Mode (M)

Détermine si **Output** est actualisé ou non.

Reset(0) : en mode Reset ,Output suit la valeur de **Reset_Output** et **Ramp_Act** sont mis à No.

Run(1) : en mode Run,**Output** progresse linéairement vers **Setpoint** à vitesse constante pour atteindre Setpoint dans le temps **Target_Time**, à condition que le bloc reste en mode Run.

Hold(2) : en mode Hold ,**Output** reste bloqué à la valeur atteinte avant l'entrée en mode Hold.

Setpoint (SP)

Setpoint est la cible vers laquelle **Output** progresse en rampe.

Target_Time (TT)

Temps nécessaire pour qu'**Output** pour passerde sa valeur actuelle à **Setpoint**. Si le bloc est mis en mode Run en partant de Reset, la vitesse nécessaire pour obtenir ce changement en sortie est calculée et maintenue, à moins d'un changement sur l'une des entrées du bloc fonction.

Si le maintien sur écart est activé ou si le mode passe à Hold **Output** est gelé. Si le maintien sur écart est désactivé ou si le mode repasse à Run, la sortie continue à progresser à la vitesse calculée, à condition que ni **Target_Time** ni **Setpoint** n'aient été modifés pendant la durée du maintien sur écart.

Si **Target_Time** ou **Setpoint** sont modifiés alors que le bloc fonction est en mode Run, la vitesse de variation est immédiatement recalculée et **Output** se met à progresser à cette nouvelle vitesse.

Reset_Output (ROP)

La valeur de cette entrée pilote **Output** quant le bloc fonction est en mode Reset.

HB_Mode (HBM)

Détermine les conditions de fonctionnement du maintien sur écart.

Off(0) : le maintien sur écart est désactivé.

Lower(1) : le maintien sur écart est activé si **Output** moins **Process_Val** est supérieur à **HB_Deviation.**

Upper(2) : le maintien sur écart est activé si **Process_Val** moins **Output** est supérieur à **HB_Deviation.**

Band(3) : le maintien sur écart est activé si la valeur absolue de **Output** moins **Process_Val** est supérieure à **HB_Deviation**.

HB_Deviation (HBD)

Le paramètre **HB_Deviation** définit la valeur de la différence permise entre **Output** et **Process_Val** avant que le maintien sur écart ne soit activé, compte tenu de **HB_Mode.**

Process_Val (PV)

Le paramètre **Process_Val** opère en liaison avec **HB_Deviation** pour déterminer si le maintien sur écart est actif. Ce paramètre n'est pas utilisé si **HB_Mode** est mis sur Off (0).

Output (OP)

Le paramètre **Output** est la sortie réelle du bloc fonction **Ramp_Target** et progresse vers **Setpoint** quand **Mode** est mis sur Run et si le bloc fonction n'est pas en maintien sur écart. **Output** est égal à **Reset_Output** si **Mode** est mis sur Reset

Ramp_End (RE)

Le paramètre **Ramp_End** indique quand **Output** a terminé sa progression en rampe vers **Setpoint.** Si **Mode** est égal à Reset, **Ramp_End** est mis à Faux sauf si **Setpoint** est égal à **Output.** Quand **Mode** est égal à Run, **Ramp_End** n'est mis à Vrai que si **Output** est égal à **Setpoint.** Si **Setpoint** est modifié après que **Ramp_End** soit devenu Vrai, **Ramp_End** devient alors Faux, jusqu'à ce que **Output** soit à nouveau égal à **Setpoint**. A l'état bloqué, le fonctionnement de **Ramp_End** est inchangé, **Ramp_End** étant mis à Faux, sauf si **Output** est égal à **Setpoint.**

Ramp_Act (RAC)

Ramp_Act définit si **Output** est en rampe vers **Setpoint.** Si Mode est mis sur Reset, **Ramp_Act** est mis à No. Si **Mode** est mis sur Run ou Hold, **Ramp_Act** est mis sur Yes si **Output** n'est pas égal à **Setpoint** et sur No si **Output** est égal à **Setpoint.**

HB_Active (HBA)

Le paramètre **HB_Active** est un indicateur pour la fonction de maintien sur écart. Si le paramètre **Mode** est mis sur Reset (0) ou Hold (2), **HB_Active** est égal à No (0). Si le paramètre **Mode** est mis sur Run (1) et **HB_Mode** est mis sur Lower (1), Upper (2) ou Band (3), **HB_Active** est égal à Yes (1) si le bloc est en maintien sur écart.

Time_Remain (TR)

Indique le temps actuellement nécessaire pour que **Output** atteigne **Setpoint.** Ce paramètre est remis à jour à chaque cycle d'exécution du bloc fonction.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Mode	**ENUM**	Reset (0)	Oper	Oper	Cf. description des paramètres	
Setpoint	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	+3·402823E+38 -3·402823E+38
Target_Time	**TIME**	0	Oper	Oper	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
Reset_Output	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	+3·402823E+38 -3·402823E+38
HB_Mode	**ENUM**	Off (0)	Oper	Oper	Cf. description des aramètres	
HB_Deviation	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	+3·402823E+38 -3·402823E+38
Process_Val	**REAL**	0	Oper	Bloc	Lim. haute Lim. basse	+3·402823E+38 -3·402823E+38
Output	**REAL**	0	Oper	Bloc	Lim. haute Lim. basse	+3·402823E+38 -3·402823E+38
Ramp_End	**BOOL**	False (0)	Oper	Bloc	Sens	False(0) True(1)
Ramo_Act	**BOOL**	No (0)	Oper	Bloc	Sens	No(0) Yes(1)
HB_Active	**BOOL**	No(0)	Oper	Bloc	Sens	No(0) Yes(1)
Time_Remain	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms

Tableau 15-1 Attibuts des paramètres Ramp_Target

BLOC FONCTION RAMP_RATE

Dans les versions antérieures à 3.00, ce bloc fonction faisait partie de la classe DIVERS et s'appelait Ramp.

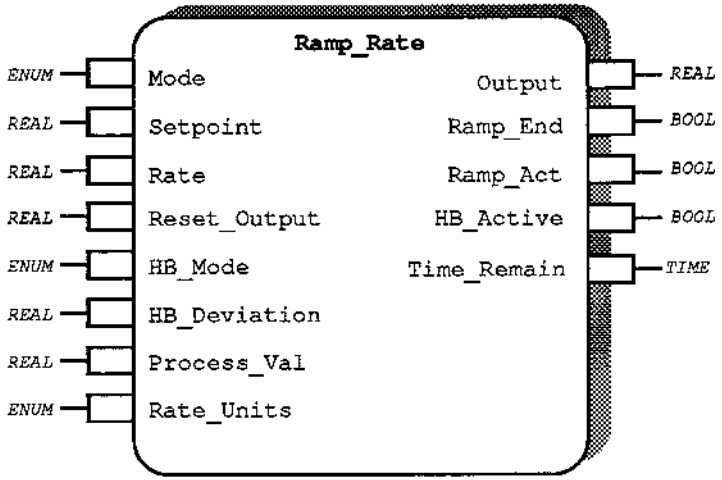


Figure 15-3 Bloc fonction Ramp_Rate

Description fonctionnelle

Le bloc fonction **Ramp_Rate** fait progresser **Output** en rampe à une vitesse **Rate** constante vers une cible **Setpoint.** Le bloc a trois modes de fonctionnement qui sont définis par **Mode.** La fonction **Holdback** peut être activée pour limiter **Rate** dans le cas d'un paramètre **Process_Val** lente

Le temps restant pour finir la rampe actuelle est donné en sortie.

Modes de fonctionnement :

Reset (0) : en mode Reset, Output suit la valeur de **Reset_Output** et **Ramp_Act** est mis à No (0).

Hold (2) : en mode Hold, **Output** conserve la valeur atteinte avant d'entrer en mode Hold.

Run (1) : en mode Run, **Output** progresse en rampe vers **Setpoint** à la vitesse définie par le paramètre Rate et **Ramp_Act** est mis à Yes (1). Quand **Output** atteint **Setpoint, Ramp_End** est mis à True (1).

Fonctionnement de Holdback

Holdback sert à maintenir **Output** à une valeur constante si l'écart entre **Output** et **Process_Val** dépasse la limite définie par **HB_Deviation.** La Fig. 15-4 ci-après montre l'activation du maintien sur écart par suite du retard de **Process_Val** par rapport à la rampe d'accroissement **Output.** Si l'entrée **Process_Val** s'écarte de la rampe **Output** de plus de **HB_Deviation, Output** est maintenu à une valeur constante jusqu'à ce que l'écart repasse en dessous de **HB_Deviation.** Dans le cas représenté, ceci a pour effet de limiter Rate à la vitesse maximale de progression de **Process_Val.**

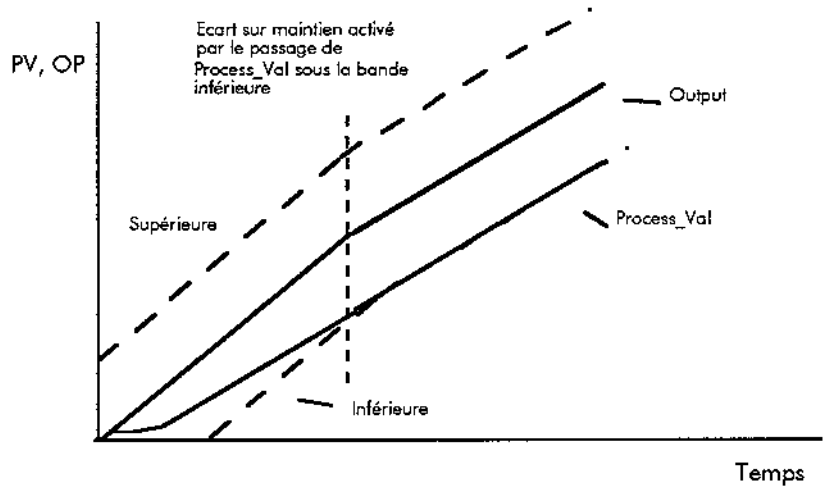


Figure 15-4 Fonctionnement de Holdback avec Process_Val en retard

Modes de fonctionnement de Holdback :

OFF(0) : en mode OFF, la fonction de maintien sur écart est désactivée.

LOWER (1): en mode **LOWER,** le maintien sur écart est activé si **Output** moins **Process_Val** est supérieur à **HB_Deviation.**

UPPER (2): en mode UPPER, le maintien sur écart est activé si **Process_Val** moins **Output** est supérieur à **HB_Deviation.**

BAND (3) : en mode **BAND,** le maintien sur écart est activé si la valeur absolue de **Output** moins **Process_Val** est supérieure à **HB_Deviation.**

Attributs du bloc fonction

Type : 3F 01
Classe : .....PROGRAMMATEUR
Tâche par défaut : .....Task_2
Liste résumée : .....Mode, Setpoint, Rate output
Mémoire nécessaire : .....84 octets
Temps d'exécution : .....282 µsec

Description des paramètres

Mode (M)

Mode définit le mode de fonctionnement du bloc fonction, voir description précédente.

Setpoint (SP)

Setpoint est la valeur cible vers laquelle la sortie progresse en rampe.

Rate (R)

Le paramètrer **Rate** définit la vitesse d'évolution de **Output**. Son unité est définie par le paramètre **Rate_Units.**

Reset_Output (ROP)

Le paramètre **Reset_Output** définit la valeur affectée au paramètre **Output** lorsque **Mode** est Reset.

HB_Mode (HBM)

Le paramètre **HB_Mode** définit le mode de fonctionnement du maintien sur écart.

HB_Deviation (HBD)

Le paramètre **HB_Deviation** définit la valeur de l'écart permis entre **Output** et **Process_Val** avant activation du maintien sur écart.

Process_Val (PV)

Le paramètre **Process_Val** est associé à **HB_Deviation** pour déterminer si le maintien sur écart doit être activé. Ce paramètre n'est pas utilisé si **HB_Mode** est mis sur Off (0).

Rate_Units (RU)

Le paramètre **Rate_Units** sert à définir l'unité dans laquelle la vitesse d'évolution de **Output** est définie.

Output (OP)

Le paramètre **Output** est la sortie Real du bloc fonction Ramp_Rate et progresse vers **Setpoint** quand **Mode** est mis sur Run (1) et si le bloc fonction n'est pas en maintien sur écart. **Output** est égal à **Reset_Output** si **Mode** est mis sur Reset (0).

Ramp_End (RE)

Le paramètre **Ramp_End** indique quand **Output** a terminé sa progression en rampe vers **Setpoint.** Si **Mode** est égal à Reset (0), **Ramp_End** est mis à Faux (0). Lorsque Mode est égal à Run (1), **Ramp_End** n'est mis à Vrai (1) que si **Output** est égal à **Setpoint.** Si **Setpoint** est modifié après que **Ramp_End** soit devenu Vrai (1), **Ramp_End** devient alors Faux (0), jusqu'à ce que **Output** soit à nouveau égal à **Setpoint**. A l'état bloqué, le fonctionnement de **Ramp_End** est inchangé, **Ramp_End** étant mis à Faux (0), jusqu'à ce que **Output** soit égal à **Setpoint.**

Ramp_Act (RA)

Ramp_Act définit si **Output** est en rampe vers **Setpoint.** Si Mode est mis sur Reset (0), **Ramp_Act** est mis à No (0). Si **Mode** est mis sur Run (1) ou Hold (2), **Ramp_Act** est mis sur Yes (1) si **Output** n'est pas égal à **Setpoint** et sur No (0) si **Output** est égal à **Setpoint.**

HB_Active (HBA)

Le paramètre **HB_Active** est un indicateur pour la fonction de maintien sur écart. Si le paramètre **Mode** est mis sur Reset (0) ou Hold (2), **HB_Active** est égal à No (0). Si le paramètre **Mode** est mis sur Run (1) et **HB_Mode** est mis sur Lower (1), Upper (2) ou Band (3), **HB_Active** est égal à Yes (1) si le bloc est en maintien sur écart.

Time_Remain (TR)

Indique le temps nécessaire pour que **Output** atteigne **Setpoint.** Ce paramètre est remis à jour à chaque cycle d'exécution du bloc fonction.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
HB_Active	**BOOL**	No (0)	Oper	Blocc	Sens	No (0) Yes (1)
HB_Deviation	**REAL**	0.0	Oper	Oper	Lim. haute Lim. basse	999,999 -99,999
HB_Mode	**ENUM**	Off (0)	Oper	Oper	Sens	Off (0) Lower (1) Upper (2) Band (3)
Mode	**ENUM**	Reset (0)	Oper	Oper	Sens	Reset (0) Run (1) Hold (2)
Output	**REAL**	0.0	Oper	Bloc	Lim. haute Lim. basse	999,999 -99,999
Process_Val	**REAL**	0.0	Oper	Oper	Lim. haute Lim. basse	999,999 -99,999
Ramp_Act	**BOOL**	No (0)	Oper	Bloc	Sens	No (0) Yes (1)
Ramp_End	**BOOL**	False (0)	Oper	Bloc	Sens	Faux (0) Vrai (1)
Rate	**REAL**	0.0	Oper	Oper	Lim. haute Lim. basse	100,000 0
Rate_Units	**ENUM**	/Second (0)	Oper	Oper	Valeurs indiquées	/Seconde (0) /Minute (1) /Heure (2) /Jour (3)
Reset_Output	**REAL**	0.0	Oper	Oper	Lim. haute Lim. basse	999,999 -99,999
Setpoint	**REAL**	0.0	Oper	Oper	Lim. haute Lim. basse	999,999 -99,999
Time_Remain	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms

Tableau 15-2 Attributs des paramètres Ramp_Rate

BLOC FONCTION PROG8RATE

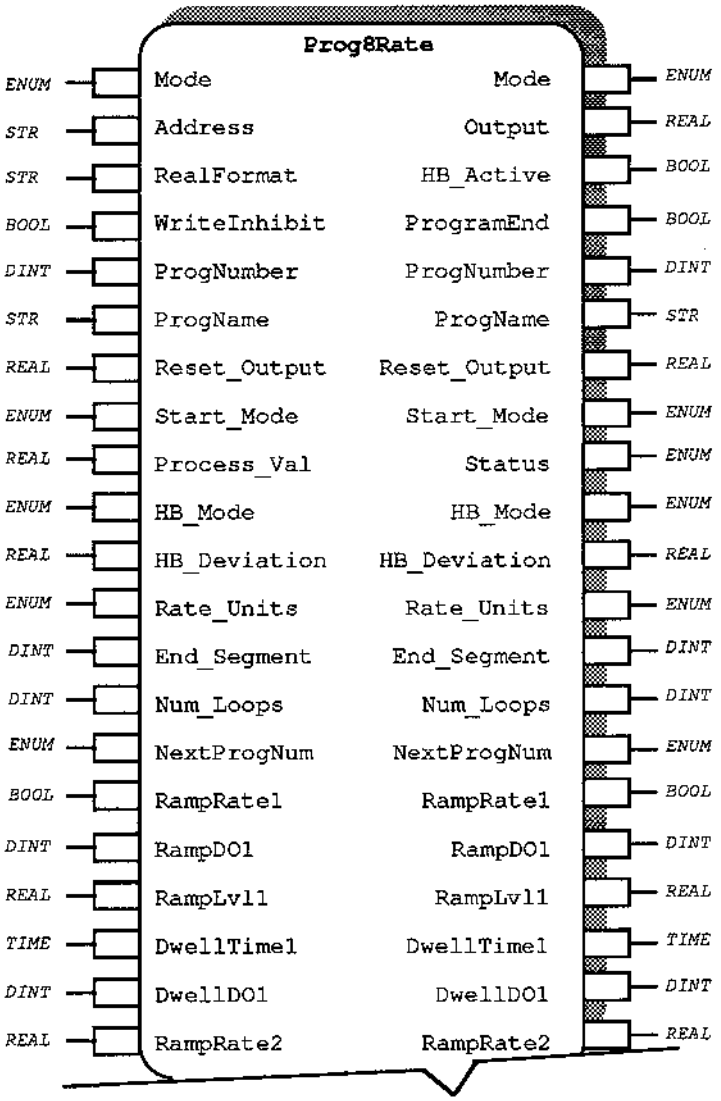


Figure 15-5 Schéma du bloc fonction Prog8Rate

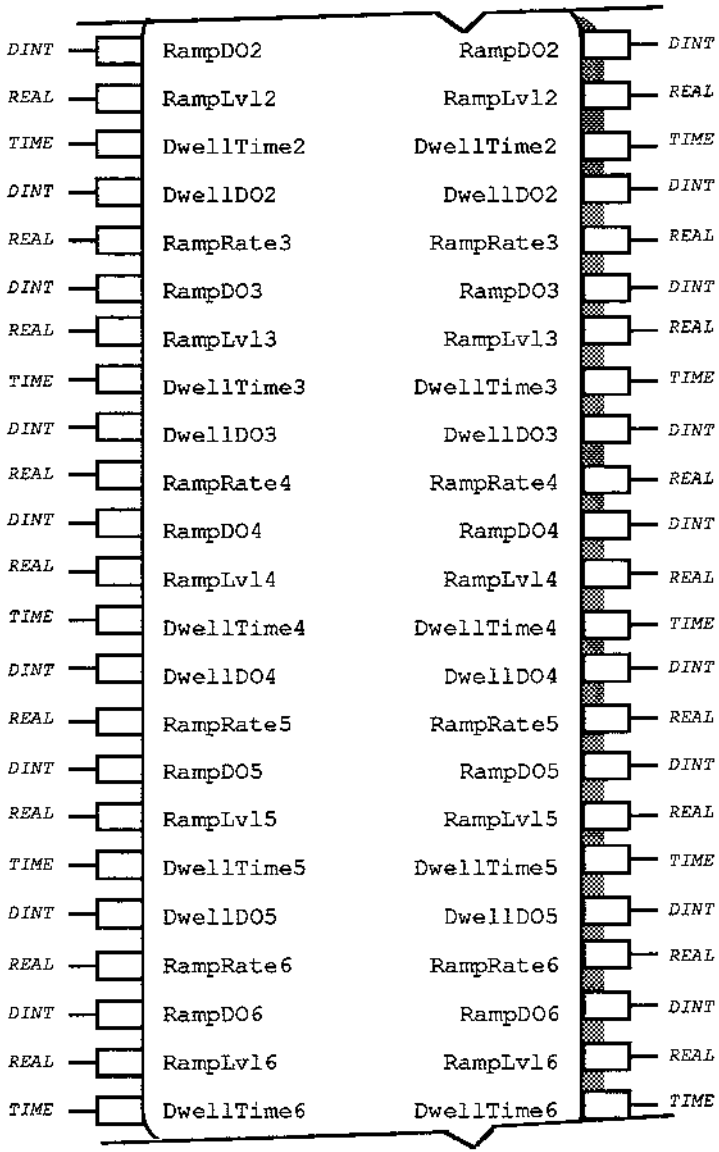


Figure 15-5 Schéma du bloc fonction Prog8Rate(suite)

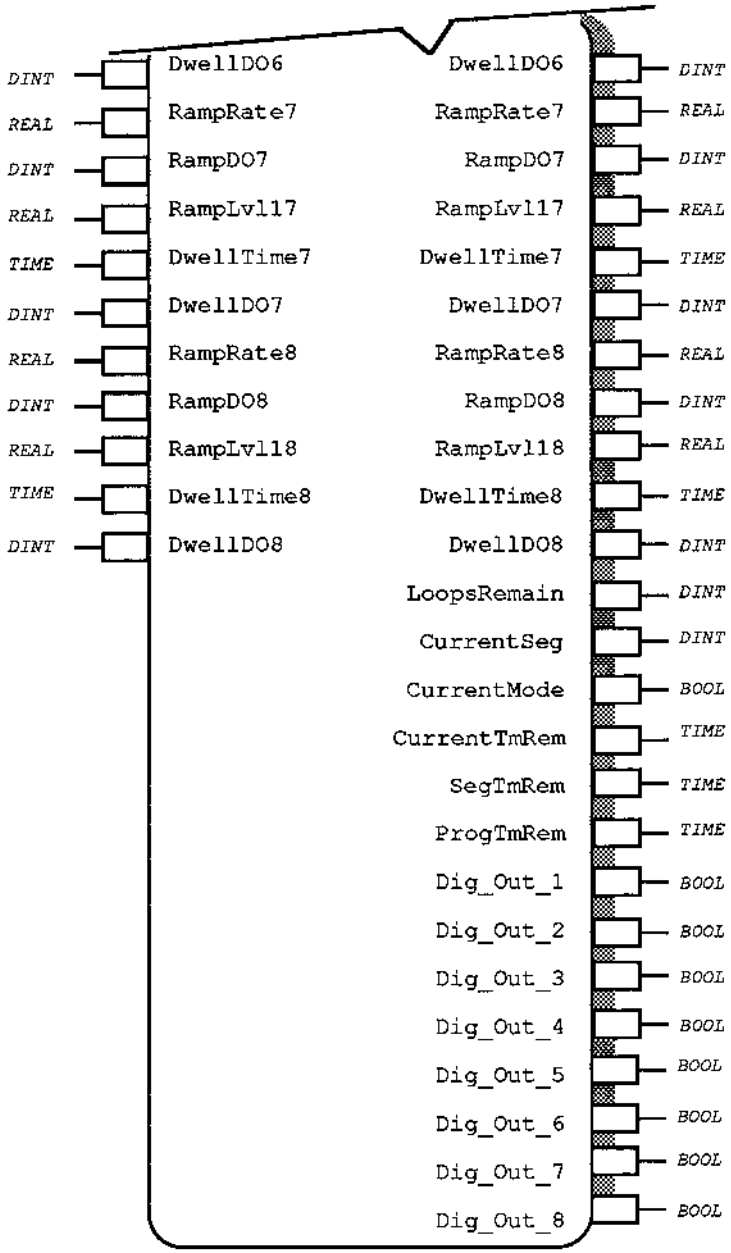


Figure 15-5 Schéma du bloc fonction Prog8Rate

Description fonctionnelle

Le bloc fonction **Prog8Rate** fournit une sortie **Output** pouvant servir de profil de consigne pour piloter une boucle de régulation. Par "profil", on entend une variation prédéterminée du point de consigne en fonction du temps. Cette fonction est semblable à celle que réalisent les appareils à programmation multiple Eurotherm 818/902.

Il est possible de mémoriser des profils (avec certains autres réglages associés) sous des noms de programme, dans le système d'enregistrement de fichiers du PC3000.

Un profil se compose d'un maximum de huit segments - le nombre de segments utilisés pour un profil donné est spécifié par l'entrée **End_Segment.** Chaque segment est normalement défini par une rampe suivie par un palier, comme pour les segments 1, 2, 3 et 8 de la Fig. 15-3.

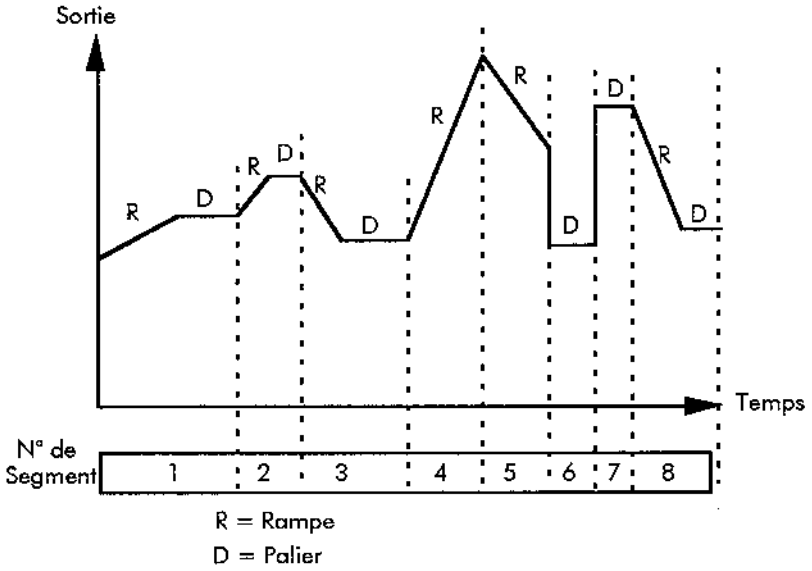


Figure 15-6 Profil de consigne : sortie Prog8Rate en fonction du temps

Un segment peut également être défini comme une rampe seule, comme pour les segments 4 et 5 ci-dessus, c'est-à-dire que le segment suivant part de la fin de la rampe. Ceci permet de créer des pics et des changements de pente dans un profil donné.

Un segment peut aussi n'être constitué que par un palier, c'est-à-dire que le niveau de la sortie n'évolue qu'au changement de segment. Voir segments 6 et 7 ci-dessus.

Un programme part toujours du segment 1, à moins que le paramètre **Start_Mode** n'indique autre chose. Voir plus loin, pour plus de détails à ce sujet. Le programme se déroule alors de façon séquentielle en suivant chaque segment jusqu'à atteindre le dernier segment configuré ou le segment 8. Le numéro du segment en cours et son mode (rampe ou palier) sont indiqués.

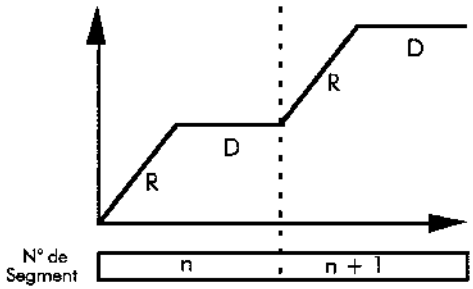
La valeur de départ de **Output** peut être préréglée au moyen de l'entrée **Reset_Output**. A chaque fois que l'entrée **Mode** est mise sur Reset, **Output** est mis à **Reset_Output**. Un segment part de la valeur actuelle de **Output** (voir Fig. 15-4) et il peut par conséquent être nécessaire de passer en **Reset** et à nouveau sur **Run** pour que le premier segment parte d'une position connue.

Avec le bloc fonction **Prog8Rate**, chaque segment de rampe d'un profil est spécifié par une vitesse de rampe (**RampRate1** à **RampRate8**) et une consigne cible (**RampLvl1** à **RampLvl8**). L'unité de pente, par exemple par seconde, par heure etc. peut également être définie.

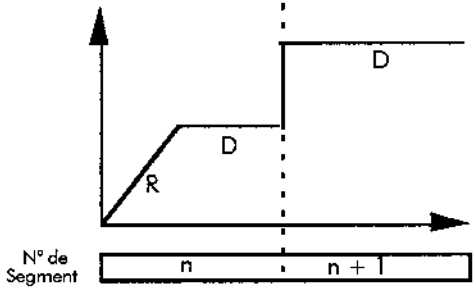
Un profil peut être répété jusqu'à 999 fois. Un profil se répète toujours par un saut du dernier segment configuré (ou du segment 8) au segment 1. Le nombre actuel de boucles restantes (sans compter la boucle en cours) est toujours indiqué. C'est-à-dire que lorsque le segment indiqué par **End_Segment** est terminé ,**Output** est alors piloté de sa valeur actuelle vers **RampLvl1** à la vitesse spécifiée par **RampRate1**. Pour qu'un profil se répète exactement, la valeur de **RampLvl8** doit être égale à la valeur de départ de **Output.**

Une vitesse de rampe 0 est interprétée comme une marche ou une rampe nulle : la sortie est instantanément mise au niveau cible. Une durée de 0 sec est interprétée comme un palier nul. Aucun temps n'est nécessaire pour exécuter une rampe nulle ou un palier nul.

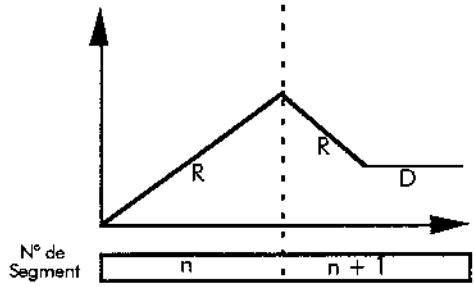
Un segment est complètement sauté si une marche (rampe nulle) et un palier nul sont configurés dans un même segment. Les sorties ne sont affectées en aucune manière par un segment sauté.



Séquence normale Rampe/palier



Une marche peut être obtenue en mettant à zéro la rampe du segment (n+1)



Deux rampes successives peuvent être obtenues en mettant à zéro le palier du segment n

Il n'y a pas de limite au nombre de segments consécutifs sautés, de rampes nulles ou de paliers nuls. Lorsque la rampe ou le palier non nul précédent est terminé, le programme saute immédiate(s)ment les rampe(s) et palier(s) nuls et passe à l'exécution de la (les) rampe ou du palier suivant on nul(le), en sautant tous les

segments nuls. Si les 8 segments sont nuls, le programme s'arrête immédiatement sans tenir compte du nombre de boucles restantes.

Quand un profil est terminé (y compris toutes les répétitions), l'indicateur **ProgramEnd** devient vrai. Quand un programme est terminé, l'état de toutes les sorties reste celui atteint immédiatement avant la fin du programme, jusqu'à la réinitialisation du programmateur.

Un programme peut être écourté en indiquant le dernier segment à exécuter au moyen du paramètre **End_Segment.** Le segment terminal par défaut est le segment 8. Tout segment suivant le segment terminal indiqué est toujours sauté : par exemple si **End_Segment** est 3, les segments 4 à 8 seront sautés et le programme s'arrêtera lorsque le segment 3 sera terminé. Si des répétitions ont été configurées, le programme se répète toujours en repassant du segment 3 au segment 1.

Un chaînage de programmes est possible au moyen du paramètre **NextProgNum.** Lorsque le programme en cours est terminé (par exemple quand **ProgramEnd** devient vrai), si le paramètre **NextProgNum** n'est pas nul, le programmateur passe automatiquement sur **HOLD**, le programme correspondant au numéro de programme indiqué par **NextProgNum** est chargé, après quoi le nouveau programme est lancé conformément au **Start_Mode** du nouveau programme.

Une liste de programmes peut être définie à l'aide des paramètres **ProgName** et **ProgNumber**.

Pour rendre simple la réalisation du type de régulation standard "PID avec programmateur", la fonction de maintien sur écart est une option intégrée au bloc fonction **Prog8Rate**. Ce qui signifie qu'il est fait en sorte que la sortie de **Prog8Rate** ne puisse s'écarter de plus d'une quantité préréglée de la valeur actuelle de la variable physique régulée. Cette dernière valeur est renvoyée au programmateur par l'entrée **Process_Val.** Trois modes de maintien sur écart sont possibles et le maintien sur écart peut être configuré à l'aide des paramètres **HB_Mode** et **HB_Deviation.**

Un jeu de 8 sorties numériques est prévu. L'état de chacune de ces sorties peut être programmé séparément pour chaque rampe ou palier de chaque segment d'un profil. Leurs états font partie du programme. Les sorties numériques ne sont actualisées qu'au départ d'une rampe ou d'un palier non nuls(différent de 0) et ne sont pas affectées par des segments nuls.

Des sorties sont prévues pour indiquer l'état du programme en cours, y compris des compteurs de temps pour le segment en cours et pour le programme total (y compris les répétitions), ainsi que l'indication du nombre de répétions restantes.

Si un paramètre quelconque évolue, tous les temps restants (sauf celui du programme total) sont immédiatement recalculés et mis à jour en fonction de la nouvelle durée. La durée restante pour le programme n'est recalculée qu'en cas de réinitialisation, de maintien sur écart, de suivi ou lors d'un changement de segment (y compris au passage d'une rampe à un palier).

Le paramètre **Mode** peut servir, non seulement pour commander le déroulement du programme en cours, mais également pour sauvegarder et récupérer des programmes dans l'enregistrementde fichier du PC3000. Les programmes sont

traités en tant que fichiers texte. Chaque programme est sauvegardé et chargé en utilisant son nom : les noms sont spécifiques et se composent de 8 caractères au maximum et d'un point suivi d'une extension sur 3 caractères. Ceci est très proche du système d'apellation des fichiers du standard MSDOS. Pour éviter de surcharger l'UC, le chargement et la sauvegarde des programmes sont répartis sur 50 cycles d'exécution, ce qui donne une temporisation type de 5 secondes avant le chargement complet d'un programme (en comptant 100 msec par tâche). La sortie Status indique l'état d'avancement des opérations de sauvegarde et de chargement.

Un programme comporte les éléments d'information suivants :

HB_Mode

HB_Deviation

Rate_Units

Reset_Output

Num_Loops

End_Segment

NextProgNum

RampRate1 - RampRate8

RampDO1 - RampDO8

RampLvl1 - RampLvl8

DwellTime1 - DwellTime8

DwellDO1 - DwellDO8

Quand un programme est extrait de l'enregistrement-fichier PC3000, tous les paramètres ci-dessus sont définis par les valeurs que contiennent les fichiers stockés. Noter qu'un programme sauvegardé par Prog8Rate ne peut pas être chargé par Prog8Time et vice versa.

Toute modification des paramètres des profils, que ce soit par le lien de communication (défini plus loin) ou par modification directe par le programme utilisateur, devient immédiatement effective si le programme est en mode **Run**. Il faut, par conséquent s'assurer avec soin que la modification de paramètres du programme est effectuée uniquement si aucun effet néfaste n'est à craindre pour l'exploitation. Pendant la progression en rampe, si la pente de la rampe ou le niveau final évolue, la rampe doit être immédiatement recalculée à partir de la valeur **Output** en cours, en utilisant les informations relatives à la nouvelle vitesse et au nouveau niveau final. Au cours d'un palier, si la durée du palier est changée, le palier est immédiatement recalculé, la durée TOTALE du palier sera la nouvelle durée de palier requise et si la nouvelle durée de palier requise est inférieure à celle déjà écoulée pour le palier, le palier s'arrête immédiatement. Si le niveau final est modifié pendant un palier, le niveau de sortie passe immédiatement au nouveau niveau final. Toutefois, les sorties numériques ne sont mises à jour qu'à l'entrée dans la rampe ou le palier donné : si la configuration de la sortie numérique pour la rampe ou le palier en cours d'exécution est modifiée, les sorties numériques ne sont pas remises à jour en conséquence, tant que ce segment n'a pas été réintroduit.

Le bloc fonction **Prog8Rate** intègre un ensemble de paramètres **Slave_Vars** pouvant servir à commander le programmateur et à mettre à jour les programmes. Le programme en cours peut être modifié via une lien de communication et stocké dans l'enregistrement-fichier du PC3000. Un programme peut être rappelé de l'enregistrement fichier pour être activé ou modifié. Le tableau 15-3 ci-dessous recense les paramètres et leurs adresses.

Nom du paramètre	Accès	Type de données	Adresse Europanel	Adresse Bisync	Adresse JBus
Mode	Lect./ Ecrit.	DINT	MD	00	00R1
Status	Lect. seule	DINT	SS	01	01R1
Program Number	Lect./ Ecrit.	DINT	PN	02	02R1
Edit Segment Number	Lect./ Ecrit.	DINT	SN	03	03R1
Edit Ramp rate	Lect./ Ecrit.	REAL	RR	04	04 (R2)
Edit Ramp Digital Output	Lect./ Ecrit.	BOOL 8	RD	05	05 (Bit Space)
Edit Ramp Level	Lect./ Ecrit.	REAL	RL	14	14 (R2)
Edit Dwell Time	Lect./ Ecrit.	TIME	DT	16	16 (R2)
Edit Dwell Digital Output	Lect./ Ecrit.	BOOL 8	DD	18	18 (Bit Space)
Reset Output	Lect./ Ecrit.	REAL	RO	27	27 (R2)
Start Mode	Lect./ Ecrit.	DINT	SM	29	29R1
Hold Back Mode	Lect./ Ecrit.	DINT	HM	30	30R1
Hold Back Deviation	Lect./ Ecrit.	REAL	HD	31	31 (R2)
End Segment Number	Lect./ Ecrit.	DINT	ES	33	33R1
Number of Loops	Lect./ Ecrit.	DINT	NL	34	34R1
Next Program Number	Lect./ Ecrit.	DINT	NP	35	35R1
Output	Lect. seule	REAL	OP	36	36 (R2)
Process Value	Lect. seule	REAL	PV	38	38 (R2)
Hold Back Active	Lect. seule	BOOL	HA	40	40 (Bit Space)
Current Active Segment	Lect. seule	DINT	CS	41	41R1
Current Active Mode	Lect. seule	DINT	CM	42	42R1
Current Time Remaining	Lect. seule	TIME	CT	43	43 (R2)
Segment Time Remaining	Lect. seule	TIME	ST	45	45 (R2)
Program Time Remaining	Lect. seule	TIME	PT	47	47 (R2)
Current Loops Remaining	Lect. seule	DINT	LR	49	49R1
Program Name	Lect. seule	STRING	NM	50	50R7
Current Digital Output	Lect. seule	STRING	DO	64	64R4

Tableau 15-3 Adresses des paramètres esclaves internes

Le paramètre d'entrée **Address** à 4 caractères sert à indiquer le protocole de communication à utiliser et à former l'adresse de base de toutes les variables esclaves internes. Par exemple EPP1 indique qu'un afficheur Eurotherm a été utilisé pour communiquer avec les esclaves (EP) et que les variables à utiliser dans OIFL doivent avoir le préfixe P1, par exemple P1MD, P1SS etc. Autre exemple, EB01 indiquerait que la communication s'est faite en utilisant le protocole Eurotherm Bisync (EB), UID étant 0 et le numéro de voie étant 1. Dans cet exemple, le format à virgule flottante peut être changé à l'aide du paramètre d'entrée **RealFormat**. Voir Présentation des communications PC3000 et les documents concernés, pour tout détail sur le paramétrage des communications du PC3000.

Pour accéder aux données des segments à partir des variables esclaves, le numéro de modification du segment est utilisé pour pointer le segment recherché et les 5 paramètres de données du segment permettent d'accéder aux sorties **ramp rate** et **ramp digital** et aux sorties numériques **ramp level, dwell time** et **dwell.** Les paramètres esclaves de lecture/écriture peuvent voir leur accès temporairement inhibé en écriture par le paramètre **WriteInhibit.**

Attributs du bloc fonction

Type : 3F20

Classe : ..... PROGRAMMATEUR

Tâche par défaut : ..... Task_2

Liste résumée : ..... Mode output, HB_Active, ProgramEnd

Mémoire nécessaire : ..... 5676 octets

Description des paramètres

Pour commentaires détaillés, utiliser l'aide en ligne, CTRL F1.

Mode (MD)

Ce paramètre commande le fonctionnement du bloc fonction et sert également au transfert des programmes vers - et hors de - l'enregistrement fichier.

Reset(0) : **Output** prend la valeur de **Reset_Output** et toutes les sorties numériques sont ramenées à Off (0). Le programme en cours est examiné à chaque cycle d'exécution et les sorties de compte rendu d'avancement sont mises à leur valeur de départ. Elles indiquent la première rampe ou le premier palier non nul(e), la durée de la rampe ou du palier actuel(e), la durée du segment en cours et le temps total d'exécution du programme. En l'absence de segment non nul,

l'indicateur **ProgramEnd** devient vrai et le programme ne peut plus tourner tant qu'un segment non nul n'a pas été configuré.

Run(1) : La sortie **Output** est pilotée en fonction du profil défini par **RampRate1** à **RampRate8**, **RampLvl1** à **RampLvl8** et **DwellTime1** à **DwellTime8**. Ce profil peut être modifié par la fonction de maintien sur écart. En partant du mode **Reset** , **Output** démarre sur la première rampe ou le premier palier non nul(e). Si le programme ne comporte que des segments nuls, le programme s'arrête aussitôt, quel que soit le nombre de boucles configurées.

En partant des modes **Hold** ou **Track**, le bloc fonction poursuit aussitôt l'exécution de la rampe ou du palier en cours. Le programme continue à tourner jusqu'à ce que le segment terminal soit atteint et qu'aucun programme suivant ne soit configuré, l'indicateur de fin de programme devenant vrai. Lorsqu'un programme s'arrête, toutes les sorties restent à l'état qu'elles avaient atteint immédiatement avant la fin, à moins que le programme ne soit réinitialisé. De nouveaux programmes peuvent être chargés pendant que **Mode** est mis sur Run.

Hold(2) : Toutes les sorties sont gelées à leurs valeurs actuelles. Tout se passe comme si le profil avait un palier de durée indéterminée inséré à la position actuelle. De nouveaux programmes peuvent être chargés pendant que **Mode** est mis à Hold. Un programme peut être mis à tout moment en mode **Hold**. En partant du mode **Reset**, le programme commence à exécuter le programme en cours comme si le mode était Run, mais il est immédiatement placé en mode **Hold**.

Track(3) : L'action du programme sur **Output** est arrêtée et **Output** est piloté directement par **Process_Val**. Lorsque **Mode** repasse à **Run**, **Output** se déplace de sa valeur actuelle vers le niveau de rampe du segment en cours avec la vitesse de rampe du segment en cours. Lorsqu'il atteint le niveau de rampe approprié, le prgramme se poursuit.

Si le programme est dans la partie rampe du segment quand **Mode** passe à Track, **CurrentMode** reste à Ramp et les 3 sorties de temps restant sont mises à jour en continu pour donner le temps qui aurait été nécessaire pour progresser en rampe de la valeur actuelle de **Output** jusqu'au niveau de rampe du segment en cours avec la vitesse de rampe de ce segment. Les sorties numériques indiquent les états résultant de la valeur **RampDO** pour le segment en cours.

Si le programme est dans la partie palier du segment quand **Mode** passe à Track, **CurrentMode** retourne à Ramp et les 3 sorties de temps restant sont mises à jour en continu pour donner le temps qui aurait été nécessaire pour progresser en rampe de la valeur actuelle de **Output** jusqu'au niveau de rampe du segment en cours avec la vitesse de rampe de ce segment. Les sorties numériques indiquent les états résultant de la valeur **DwellDO** pour le segment en cours.

Un programme peut être placé à tout moment en mode **Track**. S'il était auparavant en mode **Reset**, l'exécution du programme actuel commence comme si le mode était Run, mais il passe aussitôt en mode **Track**.

Skip Seg(4) : Si l'on est en mode **Run**, le reste du segment en cours d'exécution est sauté et l'exécution passe au début du segment suivant.

Un segment ne peut être sauté qu'en mode **Run**. Si cette demande est faite dans tout autre mode, la demande de saut de segment est n'est pas prise en compte et le mode précédent n'est pas affecté.

NxtUpSg(5) : Si l'on est en mode **Reset**, le programme démarre aussitôt. Toutefois, **Output** suit immédiatement l'entrée **Process Value** et au lieu de démarrer au premier segment, le programme démarre immédiatement l'exécution du premier segment dont le niveau final est SUPERIEUR à **Process Value**. Mode repasse aussitôt à **Run**.

NxtDnSg(6) : Si l'on est en mode **Reset**, le programme démarre aussitôt. Toutefois, **Output** suit immédiatement l'entrée **Process Value** et au lieu de démarrer au premier segment, le programme démarre immédiatement l'exécution du premier segment dont le niveau final est INFERIEUR à **Process Value**. Mode repasse aussitôt à **Run**.

Load(7) : Normalement, cette opération ne doit s'effectuer qu'en mode **Reset**. Les réglages actuels du programme sont remplacés par les valeurs stockées sous forme de programme dans l'enregistrement de fichier PC3000 sous le nom attribué par **ProgName**. Après chargement correct du programme, **Mode** est automatiquement ramené à Reset, si c'était le mode en cours lorsque l'opération de chargement a été demandée.

Pour le chargement, le mode du programme en cours passe immédiatement à **Hold**. Toutes les sorties sont maintenues à leur valeur précédente. La sortie **Status** indique la progression du chargement. Si le mode précédent était **Reset**, **Hold** ou **Track**, le mode retourne alors à **Reset**, **Hold** ou **Track**, respectivement.

Si **Mode**, au départ de l'opération de chargement, était **Run**, le mode après chargement est alors toujours fixé en fonction du **Start_Mode** du programme qui vient d'être chargé. Toutefois, si le programme précédent était terminé (c'est-à-dire si **ProgramEnd** est Yes avant la sélection de Load), le nouveau programme est chargé et, une fois le chargement terminé, le mode repasse à **Run**, mais l'indicateur de fin de programme reste vrai et toutes les sorties conservent les valeurs atteintes avant la demande de chargement.

Pendant le chargement, aucun nouveau changement de **Mode** n'est pris en compte. Lorsque le chargement est achevé, le mode est fixé conformément à l'état de **Mode** avant la demande de chargement. Si le nom de fichier indiqué n'existe pas dans l'enregistrement fichier, si le nom du fichier n'est pas conforme aux conventions de l'enregistrement fichier ou si aucun enregistrement fichier n'a été créée **Status** indique

une erreur et **Mode** repasse automatiquement à **Reset**, sans intervention du programme utilisateur.

Save(8): Les réglages actuels du programme sont sauvegardés dans l'enregistrement fichier PC3000 sous le nom donné par **ProgName** et le mode actuel est automatiquement restauré sans aucune influence sur le programme en cours.

Si le nom donné n'est pas conforme aux conventions de l'enregistrement fichier ou si aucun enregistrement fichier n'a pas été créé, **Status** indique une erreur. La sortie Status indique l'avancement de l'opération de sauvegarde.

Pendant la sauvegarde, tout changement de **Mode** est immédiatement actif, la sauvegarde n'est pas affectée. La seule exception concerne la demande de chargement qui est totalement ignorée. Si un programme se termine pendant la sauvegarde et si un nouveau programme est configuré, le nouveau programme requis est ignoré et le programme se termine immédiatement comme si aucun programme suivant n'était configuré.

Address (ADR)

Les deux premiers caractères de ce paramètre indiquent le protocole de communication à utiliser pour communiquer avec les variables esclaves intégrées au bloc fonction. Les deux caractères suivants définissent l'adresse de base de toutes les variables esclaves. Par exemple EB01 définirait l'utilisation du protocole Eurotherm Bisync avec tous les esclaves ayant un UID de 0 et un numéro de voie de 1.

Pour plus de détails, utiliser l'aide en ligne CTRL F1.

RealFormat (RLF)

Sélectionne le format à utiliser pour les nombres réels transmis sur la ligne de communication, avec le protocole Eurotherm Bisync. Pour plus de détails, se reporter à la description du bloc fonction EIBisync Slave.

L'utilisation avec le bloc fonction EuroPanel 2 nécessite que ce paramètre soit impérativement un caractère unique, pour assurer un affichage correct des sorties numériques.

Pour plus de détails, utiliser l'aide en ligne CTRL F1.

WriteInhibit (WI)

Il est possible de protéger les paramètres du programme contre toute écriture lorsqu'ils sont transmis sur une ligne de communication, en mettant cette entrée sur **Rd_Only**. La lecture des paramètres du programme reste possible par la ligne de communication.

Si ce paramètre est mis sur **Rd_Wr** les paramètres du programme peuvent être à la fois écrits et lus par la ligne de communication.

ProgNumber (PN)

Le bloc fonction **Prog8Rate** établit les références croisées entre 32 numéros de programme et 32 noms de programme associés. Lorsqu'un **ProgNumber** donné est choisi, le nom de programme qui a été associé en dernier à ce numéro de programme est affiché sur l'entrée/sortie **ProgName**.

ProgName (PNM)

Entrée/sortie servant à entrer le nom du programme associé au **ProgNumber** actuel. Le nom entré doit être un nom de fichier valide pour le système de fichiers PC3000, de telle sorte que les programmes puissent être sauvegardés et lus dans l'enregistrement fichier.

Reset_Output (ROP)

Valeur indiquée par **Output** quand **Mode** est mis sur **Reset**. Tant que le mode est Reset, toute modification de **Reset_Output** se répercute sur la sortie.

Start_Mode (STM)

Quand un programme est chargé, si le mode était **Run** au début de l'opération de chargement, le mode après chargement sera fixé conformément au **Start_Mode** du nouveau programme chargé. Les valeurs possibles pour **Start_Mode** sont :

Reset(0) :	Après chargement du programme, Mode est mis sur Reset.
Run(1) :	Après chargement du programme, Mode est mis sur Run.
Hold(2) :	Après chargement du programme, Mode est mis sur Hold.
Track(3) :	Après chargement du programme, Mode est mis sur Track.
SkipSeg(4) :	Après chargement du programme, Mode est mis sur SkipSeg.
NxtUpSeg(5) :	Après chargement du programme, Mode est mis sur NxtUpSeg.
NxtDnSeg(6) :	Après chargement du programme, Mode est mis sur NxtDnSeg.

Toutefois, si le programme précédent était terminé (c'est-à-dire si **ProgramEnd** est Yes avant la sélection Load), le nouveau programme est chargé et, une fois le chargement fini, **Mode** retourne à Run, mais l'indicateur de fin de programme reste vrai et toutes les sorties conservent les valeurs atteintes avant que le chargement ne soit demandé.

Process_Val (PV)

Normalement, la **Process_Val** de la boucle PID actuellement pilotée par le bloc fonction PROGRAMMATEUR doit être câblée sur cette entrée. Celle-ci donne des informations au bloc fonction qui valide les décisions à prendre en fonction des performances de l'application. **Output** peut être réglé pour suivre cette entrée ou bien la rampe peut être temporairement arrêtée si le processus régulé s'écarte de

la consigne de plus d'une quantité donnée (ce que l'on appelle le "maintien sur écart"). L'application détermine si une connexion est nécessaire sur cette entrée ou non.

HB_Mode (HM)

En modifiant cette entrée, il est possible de choisir la façon dont le maintien sur écart opère. Le maintien sur écart agit au cours d'une rampe - il n'y a pas de maintien sur écart automatique au cours d'un palier.

Off(0) : Pas de maintien sur écart en cours .

Lower(1) : Si **HB_Mode** est mis sur Lower, le maintien sur écart est activé si **Process_Val** est inférieure ou égale à Output moins **HB_Deviation.** En d'autres termes **Output** s'arrête à sa position actuelle jusqu'à ce que **Process_Val** augmente. La sortie **HB_Active** est également vraie dans ces circonstances. Ce mode est normalement utilisé quand il y a un risque de voir la consigne (c'est-à-dire la sortie du bloc fonction du programmateur) prendre de l'avance sur le procédé dans les rampes ascendantes.

Upper(2) : Si **HB_Mode** est mis sur Upper, le maintien sur écartt est activé si **Process_Val** est supérieure ou égale à Output plus **HB_Deviation.** En d'autres termes, **Output** s'arrête à sa position actuelle jusqu'à ce que **Process_Val** diminue. La sortie **HB_Active** est également vraie dans ces circonstances. Ce mode est normalement utilisé quand il y a un risque de voir la consigne (c'est-à-dire la sortie du bloc fonction du programmateur) prendre de l'avance sur le procédé dans les rampes descendantes.

Band(3) : Si l'écart absolu entre **Process_Val** et Output est supérieur ou égal à **HB_Deviation,** le maintien sur écart devient actif si **HB_Mode** est mis sur Band. En d'autres termes, **Output** s'arrête à sa position actuelle jusqu'à ce que cette situation ne soit plus vraie. La sortie **HB_Active** est également vraie dans ces circonstances. Ce mode est utilisé pour empêcher le procédé d'être trop en retard sur la consigne (c'est-à-dire la sortie du bloc fonction du programmateur) sur les rampes ascendantes ou descendantes.

HB_Deviation (HD)

Utilisé en association avec **Process_Val** et **Output** pour déterminer si le maintien sur écart doit être activé, selon le tparamétrage de **HB_Mode**. Il définit les limites par rapport à Output dans lesquelles **Process_Val** doit rester.

Rate_Units (RU)

Cette entrée sert à interpréter les réglages de **RampRate1** à **RampRate8** comme étant, soit en unité de sortie par seconde, par minute, par heure ou par jour. Tous les segments doivent avoir le même **Rate_Units.**

End_Segment (ES)

Le dernier segment à considérer comme faisant partie d'un profil. Ceci permet à un profil d'être arrêté avant que tous les segments non nuls aient été exécutés. Tout changement apporté à ce paramètre alors que le programme tourne, devient effectif si le segment désigné pour être le segment terminal n'a pas encore été exécuté en totalité.

Num_Loops (NL)

Nombre de fois que le profil en cours doit être répété avant de passer au programme suivant ou d'indiquer que le programme est terminé si NextProgNum est None (aucun).

NextProgNum (NPN)

Cette entrée permet le chargement d'un nouveau programme à partir de l'enregistrement fichier, une fois le programme en cours terminé.

None(0) : Aucun programme ne sera chargé, une fois le programme en cours terminé.

Prog_1(1) : Le programme dont le nom est donné en tant que **ProgName** quand **ProgNumber**=1 sera chargé à partir de l'enregistrement fichier PC3000 quand le programme en cours sera terminé. S'il n'y a pas de **ProgName** associé à **ProgNumber**=1, **ProgStatus** affiche **LoadErr** et **Mode** passe à Reset.

Prog_2(2) : Le programme dont le nom est donné en tant que **ProgName** quand **ProgNumber**=2 sera chargé à partir de l'enregistrement fichier PC3000 quand le programme en cours sera terminé. S'il n'y a pas de **ProgName** associé à **ProgNumber**=2, **ProgStatus** affiche **LoadErr** et **Mode** passe à Reset.

"

etc. jusqu'à

Prog_32 (32)

RampRate1 (RR1)

Vitesse à laquelle **Output** se déplace de sa valeur actuelle au niveau de la rampe du segment 1 en **Output units** par **Rate_Units**.

RampDO1 (RD1)

Il donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Rate à un état particulier pendant la rampe du segment 1. La valeur de cette variable est interprétée comme une configuration de 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

RampLvl1 (RL1)

Niveau de **Output** qui, lorsqu'il est atteint, provoque le démarrage du palier du segment 1.

DwellTime1 (DT1)

Durée pendant laquelle **Output** reste stationnaire à **RampLvl1** lorsque la partie rampe du segment 1 est achevée.

DwellDO1 (DD1)

Il donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Rate à un état particulier pendant le palier du segment 1. La valeur de cette variable est interprétée comme une configuration de 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

RampRate2 (RR2)

Vitesse à laquelle **Output** se déplace de sa valeur actuelle au niveau de la rampe du segment 2 en **Output units** par **Rate_Units.**

RampDO2 (RD2)

Il donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Rate à un état particulier pendant la rampe du segment 2. La valeur de cette variable est interprétée comme une configuration de 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

RampLvl2 (RL2)

Niveau de **Output** qui, lorsqu'il est atteint, provoque le démarrage du palier du segment 2.

DwellTime2 (DT2)

Durée pendant laquelle **Output** reste stationnaire à **RampLvl1** lorsque la partie rampe du segment 2 est achevée.

DwellDO2 (DD2)

Il donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Rate à un état particulier pendant le palier du segment 2. La valeur de cette variable est interprétée comme une configuration de 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

"

etc. jusqu'à

DwellDO8 (DD8)

Output (OP)

Valeur actuelle du profil en cours de réalisation par le programme actuel. Cette sortie est normalement câblée sur l'entrée **Setpoint** d'une boucle PID ou utilisée de toute autre manière en tant que consigne d'une action de commande.

HB_Active (HA)

Si **HB_Active** est vrai, ceci indique que la valeur actuellement envoyée sur l'entrée **Process_Val** s'écarte de la valeur actuelle de **Output** d'une quantité supérieure à ce qui est défini par **HB_Mode** et **HB_Deviation**.

ProgramEnd (PE)

Cette sortie signale la fin du programme actuel ou bien, en cas de chaînage de programmes par **NextProgNum**, la fin du dernier programme de la chaîne.

Status (SS)

Cette sortie signale la réussite ou autre résultat de la dernière opération d'enregistrement ou de chargement d'un fichier de l'enregistrement fichier PC3000 ou bien, pendant une opération sur un fichier, elle indique son avancement.

Ok(0) : La dernière opération, sauvegarde ou chargement, effectuée sur un fichier, a été réussie.

Saving(1) : Une opération de sauvegarde dans l'enregistrement fichier est en cours.

Loading(2) : Une opération de chargement est en cours à partir de l'enregistrement fichier.

SaveErr(3) : La dernière opération dde sauvegarde dans l'enregistrement fichier a échoué. Les raisons peuvent être diverses : pas d'enregistrement fichier, enregistrement fichier plein, nom de fichier indiqué dans **ProgName** incorrect, etc.

LoadErr(4) : La dernière opération de chargement à partir de l'enregistrement fichier a échoué. Les raisons peuvent être diverses : pas d'enregistrement fichier, nom de fichier indiqué dans **ProgName** incorrect ou n'est pas un fichier de recette, etc. Cette erreur est également signalée en cas de chargement d'un programme sauvegardé par un bloc fonction Prog8Time.

LoopsRemain (LR)

Nombre de boucles (répétitions) du programme en cours qui restent à parcourir, une fois la boucle actuelle terminée.

CurrentSeg (CS)

Numéro du segment du programme actuel en cours d'exécution. Ce peut être soit la partie rampe soit la partie palier du segment.

CurrentMode (CM)

Il indique si c'est la partie rampe ou la partie palier du segment actuel qui est en cours d'exécution. Normalement, la partie rampe d'un segment est suivie par un palier, puis l'exécution du segment suivant commence.

Toutefois, si **Mode** est mis sur Track alors que le palier a déjà commencé, **CurrentMode** passe à Ramp. Si **Mode** repasse à Run;, **Output** augmente en rampe jusqu'au niveau **RampLvl** du segment actuel, avec la pente **RampRate** du segment actuel, puis **CurrentMode** indique à nouveau Dwell. La durée de cette seconde période de palier est la durée totale **DwellTime** du segment en cours.

CurrentTmRem (CTR)

Temps nécessaire pour achever l'opération indiquée par **CurrentMode** (c'est-à-dire une rampe ou un palier) en supposant qu'il n'y a pas d'interruption due à l'activation du maintien sur écart ou au passage à une autre mode que Run.

Le calcul se fait d'après la valeur actuelle de **Output** et d'après les paramètres de profil du segment en cours.

En cas d'activation du maintien sur écart ou de changement de mode, **CurrentTmRem** indique le temps requis pour finir l'opération indiquée par **CurrentMode** si le maintien sur écart devenait immédiatement inactif ou si **Mode** repassait immédiatement sur Run.

SegTmRem (STR)

Temps nécessaire pour achever le segment en cours en supposant qu'il n'y a pas d'interruption due à l'activation du maintien sur écart ou au passage à une autre mode que Run. Le calcul se fait d'après la valeur actuelle de **Output** et d'après les paramètres de profil du segment en cours.

En cas d'activation du maintien sur écart ou de changement de mode, **SegTmRem** indique le temps requis pour achever le segment en cours si le maintien sur écart devenait immédiatement inactif ou si **Mode** repassait immédiatement sur Run.

ProgTmRem (PTR)

Temps nécessaire pour finir le nombre restant de boucles du profil en cours en supposant qu'il n'y a pas d'interruption due à l'activation du maintien sur écart ou au passage à une autre mode que Run. Le calcul se fait d'après la valeur actuelle de **Output**, les paramètres de profil du segment en cours et les paramètres **Num_Loops**.

En cas d'activation du maintien sur écart ou de changement de mode, **ProgTmRem** indique le temps requis pour achever le segment en cours si le

maintien sur écart devenait immédiatement inactif ou si **Mode** repassait immédiatement sur Run.

Si **Mode** est mis sur Run et si des paramètres de programme sont modifiés, **ProgTmRem** n'est recalculé qu'au départ du segment suivant ou au passage d'une rampe à un palier.

Dig_Out_1 (DO1)

Sortie numérique pilotée par le bit 0 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 0 de **RampDO4** est à 1 et le bit 0 de **DwellDO4** est à 0, **Dig_Out_1** est alors activée pendant la rampe du segment 4 et **Dig_Out_1** est désactivée pendant le palier du segment 4.

Dig_Out_2 (DO2)

Sortie numérique pilotée par le bit 1 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 1 de **RampDO4** est à 1 et le bit 1 de **DwellDO4** est à 0, **Dig_Out_2** est activée pendant la rampe du segment 4 et **Dig_Out_2** est désactivée pendant le palier du segment 4.

Dig_Out_3 (DO3)

Sortie numérique pilotée par le bit 2 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 2 de **RampDO4** est à 1 et le bit 2 de **DwellDO4** est à 0, **Dig_Out_3** est activée pendant la rampe du segment 4 et **Dig_Out_3** est désactivée pendant le palier du segment 4.

Dig_Out_4 (DO4)

Sortie numérique pilotée par le bit 3 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 3 de **RampDO4** est à 1 et le bit 3 de **DwellDO4** est à 0, **Dig_Out_4** est activée pendant la rampe du segment 4 et **Dig_Out_4** est désactivée pendant le palier du segment 4.

Dig_Out_5 (DO5)

Sortie numérique pilotée par le bit 4 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 4 de **Ramp5O4** est à 1 et le bit 4 de **DwellDO4** est à 0, **Dig_Out_5** est activée pendant la rampe du segment 4 et **Dig_Out_5** est désactivée pendant le palier du segment 4.

Dig_Out_6 (DO6)

Sortie numérique pilotée par le bit 5 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 5 de **RampDO4** est à 1 et le bit 5 de **DwellDO4** est à 0, **Dig_Out_6** est activée pendant la rampe du segment 4 et **Dig_Out_6** est désactivée pendant le palier du segment 4.

Dig_Out_7 (DO7)

Sortie numérique pilotée par le bit 6 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 6 de **RampDO4** est à 1 et le bit 6 de **DwellDO4** est à 0, **Dig_Out_7** est activée pendant la rampe du segment 4 et **Dig_Out_7** est désactivée pendant le palier du segment 4.

Dig_Out_8 (DO8)

Sortie numérique pilotée par le bit 7 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 7 de **RampDO4** est à 1 et le bit 7 de **DwellDO4** est à 0, **Dig_Out_8** est activée pendant la rampe du segment 4 et **Dig_Out_8** est désactivée pendant le palier du segment 4.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Mode	**ENUM**	Reset (0)	Oper	Oper	Cf. liste de paramètres	
Address	**STR**	'EPP1'	Super	Super	4 caractères maxi	
RealFormat	**STR**	"	Super	Super	Caractère unique	
WriteInhibit	**BOOL**	Rd_Wr(0)	Super	Super	Sens	Rd-Wr (0) Rd_Only (1)
ProgNumber	**DINT**	1	Oper	Oper	Lim. haute Lim. basse	32 1
ProgName	**STR**	"	Oper	Oper	12 caractères maxi	
Reset_Output	**REAL**	0	Super	Super	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
Start_Mode	**ENUM**	Run (1)	Oper	Oper	Voir liste de paramètres	
Process_Val	**REAL**	0	Oper	Oper	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
HB_Mode	**ENUM**	Off (0)	Oper	Oper	Cf. liste de paramètres	
HB_Deviation	**REAL**	0	Oper	Oper	Cf. liste de paramètres	
Rate_Units	**ENUM**	/Second (0)	Super	Super	Cf. liste de paramètres	
End_Segment	**DINT**	8	Super	Super	Lim. haute Lim. basse	8 1
Num_Loops	**DINT**	1	Super	Super	Lim. haute Lim. basse	999 0
NextProgNum	**ENUM**	None(0)	Super	Super	Cf. liste de paramètres	
RampRate1 -to RampRate8	**REAL**	0	Super	Super	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
RampDO1 to RampDO8	**DINT**	0	Super	Super	Lim. haute Lim. basse	256 0
RampLv11 to RampLv18	**REAL**	0	Super	Super	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
DwellTime1 to DwellTime8	**TIME**	0ms	Super	Super	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
DwellD01-to DwellD08	**DINT**	0	Super	Super	Lim. haute Lim. basse	256 0

Tableau 15-4 Attributs des paramètres Prog8Rate (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Output	**REAL**	0	Oper	Bloc	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
HB_Active	**BOOL**	No(0)	Oper	Bloc	Sens	No(0) Yes(1)
ProgramEnd	**BOOL**	Yes(1)	Oper	Bloc	Sens	No(0) Yes(1)
Status	**ENUM**	Ok(0)	Oper	Bloc	Cf. liste de paramètres	
LoopsRemain	**DINT**	0	Oper	Bloc	Lim. haute Lim. basse	999 0
CurrentSeg	**DINT**	8	Oper	Bloc	Lim. haute Lim. basse	8 1
CurrentMode	**BOOL**	Dwell(1)	Oper	Bloc	Sens	Ramp(0) Dwell(1)
CurrentTmRem	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0sms
SegTmRem	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
ProgTmRem	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
Dig_Out_1 -to Dig_Out_8	**BOOL**	Off(0)	Oper	Bloc	Sens	Off(0) On(1)

Tableau 15-4 Attributs des paramètres Prog8Rate

BLOC FONCTION PROG8TIME

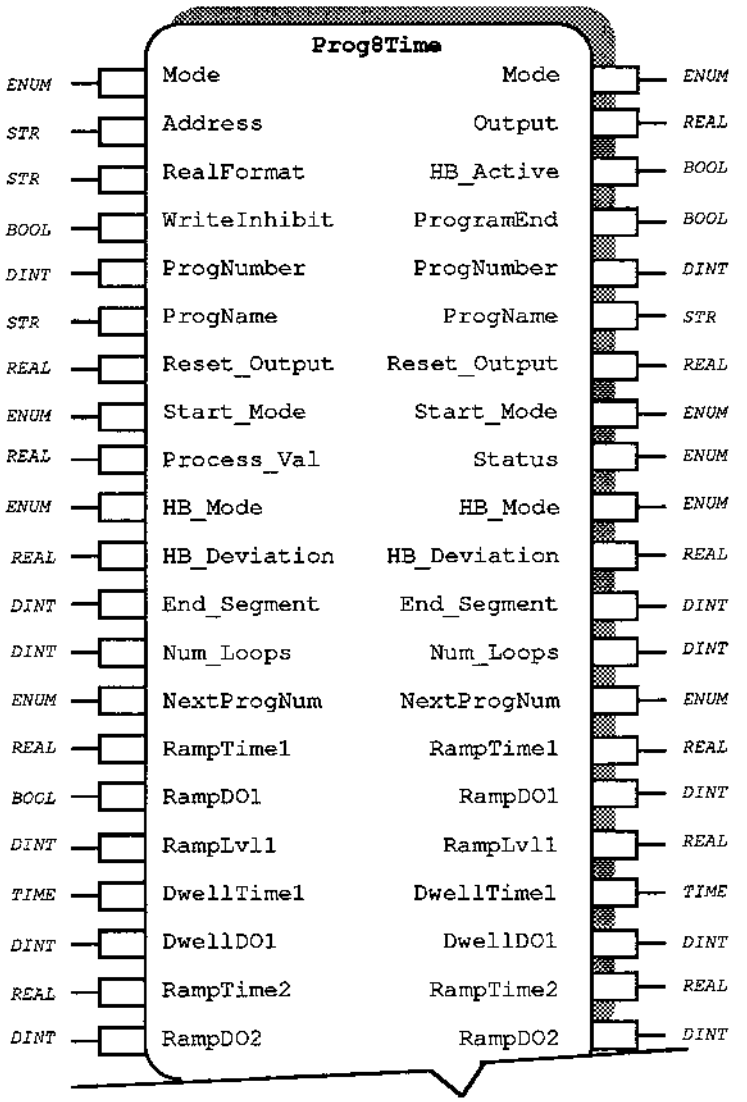


Figure 15-8 Schéma du bloc fonction Prog8Time

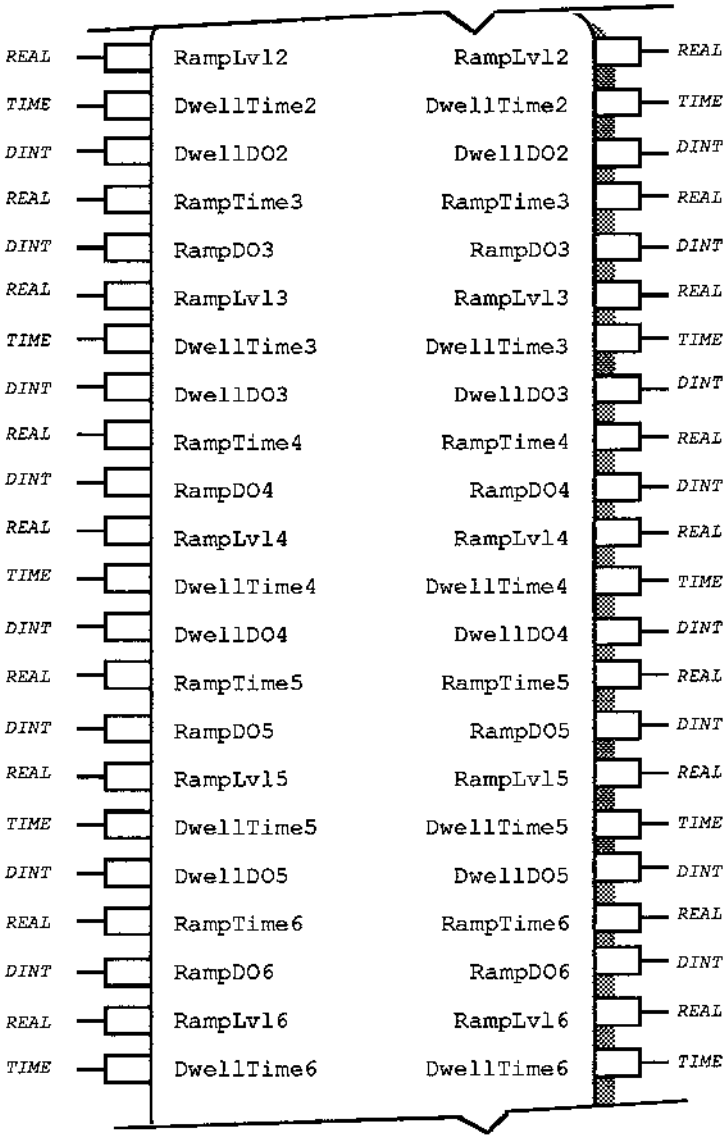


Figure 15-8 Schéma du bloc fonction Prog8Time (suite)

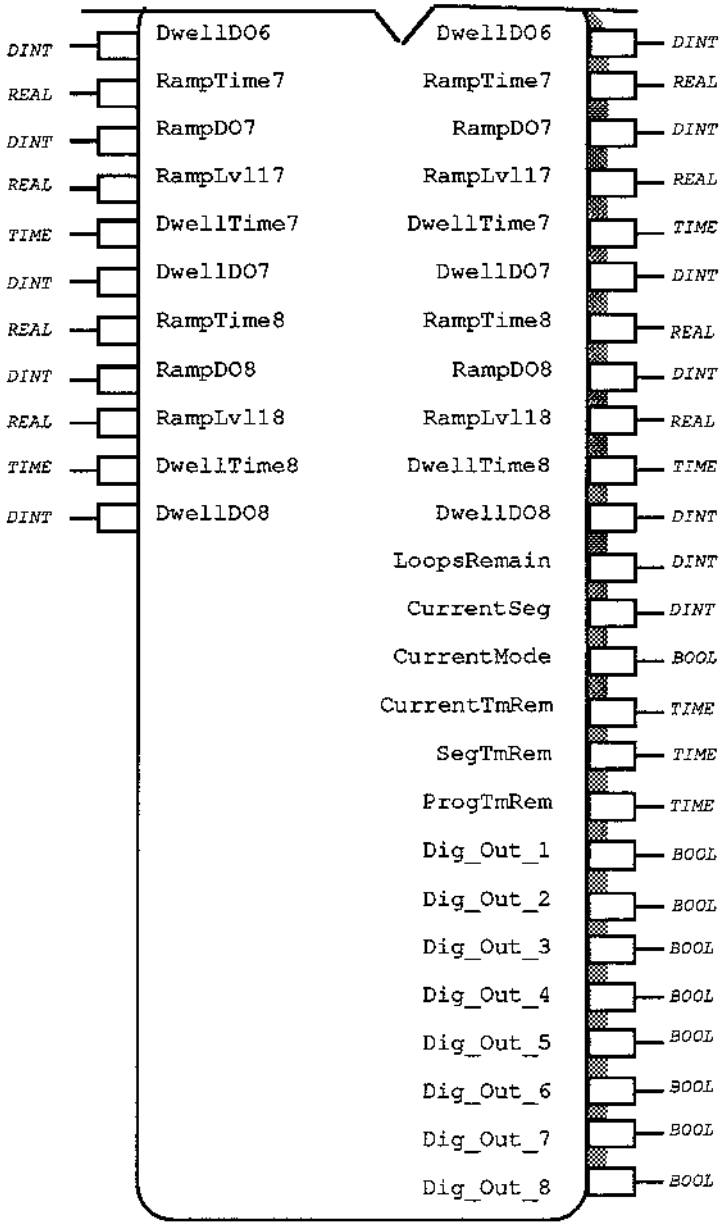


Figure 15-8 Schéma du bloc fonction Prog8Time

Description fonctionnelle

Le bloc fonction **Prog8Time** fournit une sortie **Output** pouvant servir de profil de consigne pour piloter une boucle de régulation. Par "profil", on entend une variation prédéterminée de la consigne en fonction du temps. Cette fonction est semblable à celle que réalisent les appareils à programmation multiple Eurotherm 818/902.

Il est possible de mémoriser des profils (avec certains autres réglages associés) sous des noms de programme, dans le système d'enregistrement fichier du PC3000.

Un profil se compose de huit segments au maximum- le nombre de segments utilisés pour un profil donné est spécifié par l'entrée **End_Segment.** Chaque segment est normalement défini par une rampe suivie par un palier, comme pour les segments 1, 2, 3 et 8 de la figure. 15-9 ci-dessous.

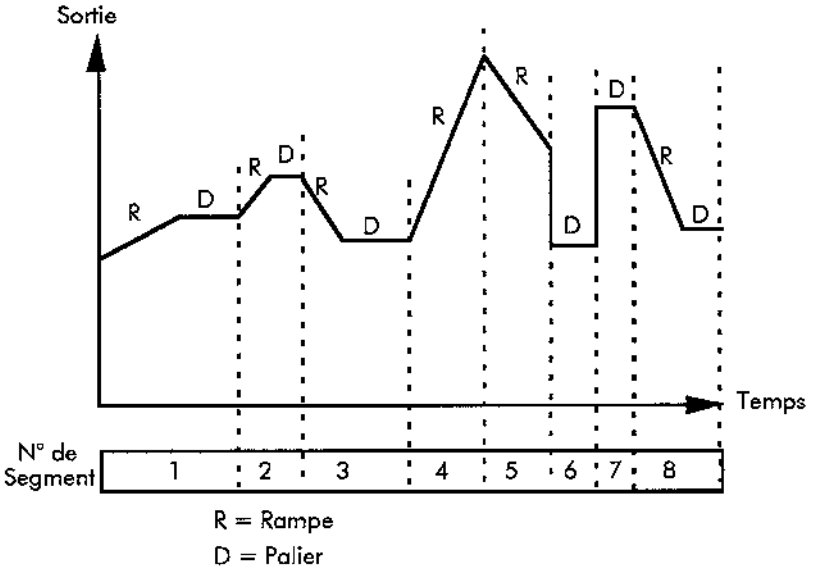


Figure 15-9 Profil de consigne: Sortie Prog8Time en fonction du temps

Un segment peut également être défini comme une rampe seule, comme pour les segments 4 et 5 ci-dessus, c'est-à-dire que le segment suivant part de la fin de la rampe. Ceci permet de créer des pics et des changements de pente dans un profil donné.

Un segment peut aussin'être constitué que par un palier, c'est-à-dire que le niveau de la sortie n'évolue qu'au changement de segment. Voir segments 6 et 7 ci-dessus.

Un programme part toujours du segment 1, à moins que le paramètre **Start_Mode** n'indique autre chose. Voir plus loin, pour plus de détails à ce sujet. Le programme

se déroule alors de façon séquentielle en suivant chaque segment jusqu'à atteindre le dernier segment configuré ou le segment 8. Le numéro du segment en cours et son mode (rampe ou palier) sont indiqués.

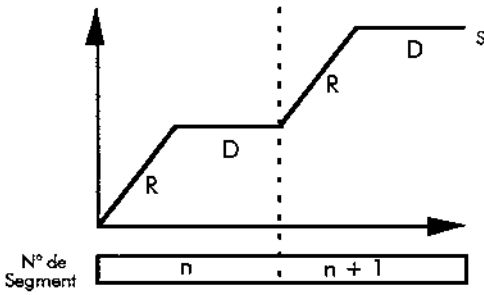
La valeur de départ de **Output** peut être préréglée au moyen de l'entrée **Reset_Output**. A chaque fois que l'entrée **Mode** est mise sur Reset, **Output** est mis à **Reset_Output**. Un segment part de la valeur actuelle de **Output** (voir Fig. 15-11) et il peut par conséquent être nécessaire de passer en **Reset** et à nouveau sur **Run** pour que le premier segment parte d'une position connue.

Avec le bloc fonction **Prog8Time**, chaque segment de rampe d'un profil est spécifié au moyen du temps nécessaire pour finir la rampe (**RampTime1** à **RampTime8**) et par la consigne cible (**RampLvl1** à **RampLvl8**).

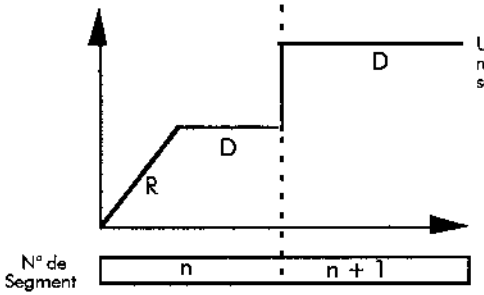
Un profil peut être répété jusqu'à 999 fois. Un profil se répète toujours par un saut du dernier segment configuré (ou du segment 8) au segment 1. Le nombre actuel de boucles restantes (sans compter la boucle en cours) est toujours indiqué. C'est-à-dire que lorsque le segment indiqué par **End_Segment** est terminé ,**Output** est alors piloté de sa valeur actuelle vers **RampLvl1** dans le temps spécifié par **RampTime1**. Pour qu'un profil se répète exactement, la valeur de **RampLvl8** doit être égale à la valeur de départ de **Output.**

Une durée de rampe 0 sec est interprétée comme une marche ou une rampe nulle : la sortie est instantanément mise au niveau final. Une durée de palier 0 sec est interprétée comme un palier nul. Aucun temps n'est nécessaire pour exécuter une rampe nulle ou un palier nul.

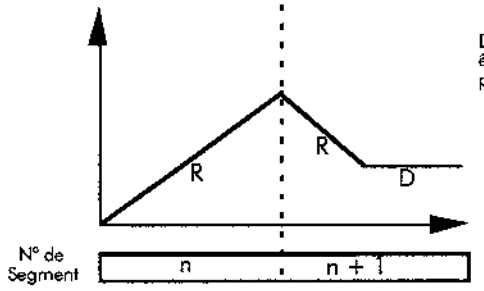
Un segment est complètement sauté si une marche (rampe nulle) et un palier nul sont configurés dans un même segment. Les sorties ne sont affectées en aucune manière par un segment sauté.



Séquence normale Rampe/palier



Une marche peut être obtenue en mettant à zéro la rampe du segment (n+1)



Deux rampes successives peuvent être obtenues en mettant à zéro le palier du segment n

Il n'y a pas de limite au nombre de segments consécutifs sautés, de rampes nulles ou de paliers nuls. Lorsque la rampe ou le palier non nul précédent est terminé, le programme saute immédiatement les rampe(s) et palier(s) nuls et passe à l'exécution de la rampe ou du palier suivant(e) non nul(le) en sautant tous les segments nuls. Si les 8 segments sont nuls, le programme s'arrête immédiatement sans tenir compte du nombre de boucles restantes.

Quand un profil est terminé (y compris toutes les répétitions), l'indicateur **ProgramEnd** devient vrai. Quand un programme est terminé, l'état de toutes les sorties reste celui atteint immédiatement avant la fin du programme, jusqu'à la réinitialisation du programmateur.

Un programme peut être écourté en indiquant le dernier segment à exécuter au moyen du paramètre **End_Segment.** Le segment terminal par défaut est le segment 8. Tout segment suivant le segment terminal indiqué est toujours sautés : par exemple si **End_Segment** est 3, les segments 4 à 8 seront ignorés et le programme s'arrêtera lorsque le segment 3 sera terminé. Si des répétitions ont été configurées, le programme se répète toujours en repassant du segment 3 au segment 1.

Un chaînage de programmes est possible au moyen du paramètre **NextProgNum.** Lorsque le programme en cours est terminé (c'est-à-dire quand **ProgramEnd** devient vrai), si le paramètre **NextProgNum** nest pas nul, le programmateur passe automatiquement sur **HOLD,** le programme correspondant au numéro de programme indiqué par **NextProgNum** est chargé, après quoi le nouveau programme est lancé conformément au **Start_Mode** du nouveau programme.

Une liste de noms de programmes peut être définie à l'aide des paramètres **ProgName** et **ProgNumber.**

Pour faciliter la réalisation du type de régulation standard "PID avec programmateur", la fonction de maintien sur écart est une option intégrée au bloc fonction **Prog8Rate.** Ce qui signifie qu'il est fait en sorte que la sortie de **Prog8Rate** ne puisse s'écarter de plus d'une quantité préréglée de la valeur actuelle de la variable physique régulée. Cette dernière valeur est renvoyée au programmateur par l'entrée **Process_Val.** Trois modes de maintien sur écart sont possibles et le maintien sur écart peut être configuré à l'aide des paramètres **HB_Mode** et **HB_Deviation.**

Un jeu de 8 sorties numériques est prévu. L'état de chacune de ces sorties peut être programmé séparément pour chaque rampe ou palier de chaque segment d'un profil. Leurs états font partie du programme. Les sorties numériques ne sont actualisées qu'au départ d'une rampe ou d'un palier non nuls (différents de 0) et ne sont pas affectées par des segments en rampe ou en palier nuls.

Des sorties sont prévues pour indiquer l'état du programme en cours, dont des compteurs de temps pour le segment en cours et pour le programme total (y compris les répétitions), ainsi que l'indication du nombre de répétions restantes.

Si un paramètre quelconque évolue, tous les temps restants (sauf celui du programme total) sont immédiatement recalculés et mis à jour en fonction de la nouvelle durée. La durée restante pour le programme n'est recalculée qu'en cas de réinitialisation, de maintien sur écart, de suivi ou lors d'un changement de segment (y compris au passage d'une rampe à un palier).

Le paramètre **Mode** peut servir, non seulement pour commander le déroulement du programme en cours, mais également pour sauvegarder et récupérer des programmes dans l'enregistrement-fichier du PC3000. Les programmes sont traités en tant que fichiers de texte. Chaque programme est sauvegardé et chargé en utilisant son nom : les noms dépendent de la casse des caractères et se composent de 8 caractères et d'un point suivi d'une extension sur 3 caractères. Ceci est très

proche du système d'apellation des fichiers standard MSDOS. Pour éviter de surcharger l'UC, le chargement et la sauvegarde des programmes sont répartis sur 50 cycles d'exécution, ce qui donne une attente type de 5 secondes avant le chargement complet d'un programme (en comptant 100 ms par tâche). La sortie Status indique l'état d'avancement des opérations de sauvegarde et de chargement.

Un programme comporte les éléments d'information suivants:

HB_Mode

HB_Deviation

Rate_Units

Num_Loops

End_Segment

NextProgNum

RampTime1 - RampTime8

RampDO1 - RampDO8

RampLvl1 - RampLvl8

DwellTime1 - DwellTime8

DwellDO1 - DwellDO8

Quand un programme est extrait de l'enregistrement fichier PC3000, tous les paramètres sont définis par les valeurs que contiennent les fichiers stockés. Noter qu'un programme sauvegardé par Prog8Rate ne peut pas être chargé par Prog8Time et vice versa.

Toute modification des paramètres des profils, que ce soit par le lien de communication (défini plus loin) ou par modification directe par le programme utilisateur, devient immédiatement effectifve si le programme est en mode **Run**. Il faut, par conséquent s'assurer avec soin que la modification de paramètres du programme ne soit effectuée que si aucun effet néfaste n'est à craindre pour l'exploitation. Pendant la progression en rampe, si la durée de la rampe ou le niveau final évolue, la rampe doit être immédiatement recalculée à partir de la valeur **Output** en cours, en utilisant les informations relatives à la nouvelle vitesse et au nouveau niveau final. Au cours d'un palier, si la durée du palier est changée, le palier est immédiatement recalculé, la durée TOTALE du palier sera la nouvelle durée de palier requise et si la nouvelle durée de palier requise est inférieure à celle déjà écoulée pour le palier, le palier s'arrête immédiatement. Si le niveau final est modifié pendant un palier, le niveau de sortie passe immédiatement au nouveau niveau final. Toutefois, les sorties numériques ne sont mises à jour qu'à l'entrée dans la rampe ou la palier donné : si la configuration de la sortie numérique pour la rampe ou le palier en cours d'exécution est modifiée, les sorties numériques ne sont pas remises à jour en conséquence, tant que ce segment n'a pas été réintroduit.

Le bloc fonction **Prog8Rate** intègre un jeu de **Slave_Vars** pouvant servir à commander le programmateur et à mettre à jour les programmes. Le programme en cours peut être modifié via une lien de communication et stocké dans l'enregistrement fichier du PC3000. Un programme peut être rappelé de

l'enregistrement fichier pour être activé ou modifié. Le tableau ci-dessous recense les paramètres et leurs adresses.

Nom du paramètre	Accès	Type de données	Adresse Europanel	Adresse Bisync	Adresse JBus
Mode	Lect. / Ecrit.	DINT	MD	00	00R1
Status	Lect. seule	DINT	SS	01	01R1
Program Number	Lect. / Ecrit.	DINT	PN	02	02R1
Edit Segment Number	Lect. / Ecrit.	DINT	SN	03	03R1
Edit Ramp Time	Lect. / Ecrit.	TIME	RT	04	04 (R2)
Edit Ramp Digital Output	Lect. / Ecrit.	BOOL 8	RD	05	05 (Bit Space)
Edit Ramp Level	Lect. / Ecrit.	REAL	RL	14	14 (R2)
Edit Dwell Time	Lect. / Ecrit.	TIME	DT	16	16 (R2)
Edit Dwell Digital Output	Lect. / Ecrit.	BOOL 8	DD	18	18 (Bit Space)
Reset Output	Lect. / Ecrit.	REAL	RO	27	27 (R2)
Start Mode	Lect. / Ecrit.	DINT	SM	29	29R1
Hold Back Mode	Lect. / Ecrit.	DINT	HM	30	30R1
Hold Back Deviation	Lect. / Ecrit.	REAL	HD	31	31 (R2)
End Segment Number	Lect. / Ecrit.	DINT	ES	33	33R1
Number of Loops	Lect. / Ecrit.	DINT	NL	34	34R1
Next Program Number	Lect. / Ecrit.	DINT	NP	35	35R1
Output	Lect. seule	REAL	OP	36	36 (R2)
Process Value	Lect. seule	REAL	PV	38	38 (R2)
Hold Back Active	Lect. seule	BOOL	HA	40	40 (Bit Space)
Current Active Segment	Lect. seule	DINT	CS	41	41R1
Current Active Mode	Lect. seule	DINT	CM	42	42R1
Current Time Remaining	Lect. seule	TIME	CT	43	43 (R2)
Segment Time Remaining	Lect. seule	TIME	ST	45	45 (R2)
Program Time Remaining	Lect. seule	TIME	PT	47	47 (R2)
Current Loops Remaining	Lect. seule	DINT	LR	49	49R1
Program Name	Lect. seule	STRING	NM	50	50R7
Current Digital Output	Lect. seule	STRING	DO	64	64R4

Tableau 15-5 Adresses des paramètres esclaves internes

Le paramètre d'entrée **Address** à 4 caractères sert à indiquer le protocole de communication à utiliser et à former l'adresse de base de toutes les variables esclaves internes. Par exemple EPP1 indique qu'un afficheur Eurotherm a été utilisé pour communiquer avec les esclaves (EP) et que les variables à utiliser dans

OIFL doivent avoir le préfixe P1, par exemple P1MD, P1SS etc. Autre exemple, EB01 indiquerait que la communication s'est faite en utilisant le protocole Eurotherm Bisync (EB), UID étant 0 et le numéro de voie étant 1. Dans cet exemple, le format à virgule flottante peut être changé à l'aide du paramètre d'entrée **RealFormat**. Voir Présentation des communications PC3000 et les documents concernés, pour tout détail sur le paramétrage des communications du PC3000.

Pour accéder aux données des segments à partir des variables esclaves, le numéro de modification du segment est utilisé pour pointer le segment recherché et les 5 premiers paramètres du segment permettent d'accéder aux sorties **ramp rate** et **ramp digital** et aux sorties numériques **ramp level, dwell time** et **dwell**. Les paramètres esclaves de lecture/écriture peuvent voir leur accès temporairement inhibé en écriture par le paramètre **WriteInhibit.**

Attributs du bloc fonction

Type : 3F22

Classe :.....PROGRAMMATEUR

Tâche par défaut : .....Task_2

Liste résumée : .....Mode output, HB_Active, ProgramEnd

Mémoire nécessaire : .....5640 octets

Description des paramètres

Pour commentaires détaillés, utiliser l'aide en ligne CTRL F1.

Mode (MD)

Ce paramètre commande le fonctionnement du bloc fonction et sert également au transfert des programmes vers - et hors de - l'enregistrement fichier.

Reset(0) : **Output** prend la valeur de **Reset_Output** et toutes les sorties numériques sont ramenées à Off (0). Le programme en cours est examiné à chaque cycle d'exécution et les sorties de compte rendu d'avancement sont mises à leur valeur de départ. Elles indiquent la première rampe ou le premier palier non nul(le), la durée de la rampe ou du palier actuel(le), la durée du segment en cours et le temps total écoulé pour le programme. En l'absence de segment non nul, l'indicateur **ProgramEnd** devient vrai et le programme ne peut plus tourner tant qu'un segment non-nul n'a pas été configuré.

Run(1) : La sortie **Output** est pilotée en fonction du profil défini par **RampTim1** à **RampTime8, RampLvl1** à **RampLvl8** et **DwellTime1**

à **DwellTime8**. Ce profil peut être modifié par la fonction de maintien sur écart.

En partant du mode **Reset, Output** démarre sur la première rampe ou le premier palier non nul(le). Si le programme ne comporte que des segments nuls, le programme s'arrête aussitôt, quel que soit le nombre de boucles configurées.

En partant des modes **Hold** ou **Track**, le bloc fonction poursuit aussitôt l'exécution de la rampe ou du palier en cours.

Le programme continue à tourner jusqu'à ce qu'un segment terminal soit atteint et qu'aucun programme suivant ne soit configuré, l'indicateur de fin de programme devenant vrai. Lorsqu'un programme s'arrête, toutes les sorties restent à l'état atteint immédiatement avant la fin, jusqu'à ce que que le programme soit réinitialisé. De nouveaux programmes peuvent être chargés pendant que **Mode** est mis à Run.

Hold(2) : Toutes les sorties sont gelées à leur valeur actuelle Tout se passe comme si le profil avait un palier de durée indéterminée inséré à la position actuelle.

De nouveaux programmes peuvent être chargés pendant que **Mode** est mis à Hold. Un programme peut être mis à tout moment en mode **Hold**. En partant du mode **Reset**, le programme commence à exécuter le programme en cours comme si le mode était Run, mais il est immédiatement placé en mode **Hold**.

Track(3) : L'action du programme sur **Output** est arrêtée et **Output** est pilotée directement par **Process_Val**. Lorsque **Mode** repasse à **Run, Output** se déplace de sa valeur actuelle vers le niveau de rampe du segment en cours avec la durée de rampe du segment en cours. Lorsqu'il atteint le niveau de rampe approprié, le programme se poursuit.

Si le programme est dans la partie rampe du segment quand **Mode** passe à Track, **CurrentMode** reste à Ramp et les 3 sorties de temps restant sont mises à jour en continu pour donner le temps qui aurait été nécessaire pour progresser en rampe de la valeur actuelle de **Output** jusqu'au niveau de rampe du segment en cours avec la durée de rampe de ce segment. Les sorties numériques indiquent les états résultant de la valeur **RampDO** pour le segment en cours.

Si le programme est dans la partie palier du segment quand **Mode** passe à Track, **CurrentMode** retourne à Ramp et les 3 sorties de temps restantes sont mises à jour en continu pour donner le temps qui aurait été nécessaire pour progresser en rampe de la valeur actuelle de **Output** jusqu'au niveau de rampe du segment en cours avec la durée de ce segment. Les sorties numériques indiquent les états résultant de la valeur **DwellDO** pour le segment en cours.

Un programme peut être placé à tout moment en mode **Track**. S'il était auparavant en mode **Reset**, l'exécution du programme actuel commence comme si le mode était Run, mais il passe aussitôt en mode **Track**.

Skip Seg(4) : Si l'on est en mode **Run**, le reste du segment en cours d'exécution est sauté et l'exécution passe au début du segment suivant.

Un segment ne peut être sauté qu'en mode **Run**. Si cette demande est faite dans tout autre mode, la demande de saut de segment n'est pas prise en compte et le mode précédent n'est pas affecté.

NxtUpSg(5) : Si l'on est en mode **Reset**, le programme démarre aussitôt. Toutefois, **Output** suit immédiatement l'entrée **Process Value** et ,au lieu de démarrer au premier segment, le programme démarre immédiatement l'exécution du premier segment dont le niveau final est SUPERIEUR à **Process Value**. Le mode repasse aussitôt à **Run**.

NxtDnSg(6) : Si l'on est en mode **Reset**, le programme démarre aussitôt. Toutefois, **Output** suit immédiatement l'entrée **Process Value** et ,au lieu de démarrer au premier segment, le programme démarre immédiatement l'exécution du premier segment dont le niveau final est INFERIEUR à **Process Value**. Le mode repasse aussitôt à **Run**.

Load(7): Normalement, cette opération ne doit s'effectuer qu'en mode **Reset**. Les réglages actuels du programme sont remplacés par les valeurs stockées sous forme de programme dans l'enregistrement fichier PC3000 sous le nom attribué par **ProgName**. Après chargement correct du programme, **Mode** est automatiquement mis à Reset, si c'était le mode en cours lorsque l'opération de chargement a été demandée.

Pour le chargement, le mode du programme en cours passe immédiatement à **Hold**. Toutes les sorties sont maintenues à leur valeur précédente.

La sortie **Status** indique la progression du chargement. Si le mode précédent était **Reset**, **Hold** ou **Track**, le mode retourne alors à **Reset**, **Hold** ou **Track**.

Si **Mode**, au départ de l'opération de chargement, était **Run**, le mode après chargement est alors toujours fixé en fonction du **Start_Mode** du programme qui vient d'être chargé. Toutefois, si le programme précédent était terminé (c'est-à-dire si **ProgramEnd** est Yes avant la sélection de Load), le nouveau programme est chargé et , une fois le chargement terminé, le mode repasse à **Run**, mais l'indicateur de fin de programme reste vrai et toutes les sorties conservent les valeurs atteintes avant la demande de chargement.

Pendant le chargement, aucun nouveau changement de **Mode** n'est pris en compte. Lorsque le chargement est achevé, le mode est fixé conformément à l'état de **Mode** avant la demande de chargement.

Si le nom de fichier indiqué n'existe pas dans l'enregistrement fichier, si le nom du fichier n'est pas conforme aux conventions de l'enregistrement fichier ou si aucun enregistrement fichier n'a été créé, **Status** indique une erreur et **Mode** repasse automatiquement à **Reset**, sans intervention du programme utilisateur.

Save(8) : Les réglages actuels du programme sont sauvegardés dans l'enregistrement fichier PC3000 sous le nom donné par **ProgName** et le mode actuel est automatiquement restauré sans aucune influence sur le programme en cours.

Si le nom donné n'est pas conforme aux conventions de l'enregistrement fichier ou si une mémoire de fichiers n'a pas été créée, **Status** indique une erreur. La sortie Status indique l'avancement de l'opération de sauvegarde.

Pendant la sauvegarde, tout changement de **Mode** est immédiatement actif, la sauvegarde n'est pas affectée. La seule exception concerne la demande de chargement qui n'est pas pris en compte Si un programme se termine pendant la sauvegarde et si un nouveau programme est configuré, le nouveau programme requis n'est pas pris en compte et le programme se termine immédiatement comme si aucun programme suivant n'était configuré.

Address (ADR)

Les deux premiers caractères de ce paramètre indiquent le protocole de communication à utiliser pour communiquer avec les variables esclaves intégrées au bloc fonction. Les deux caractères suivants définissent l'adresse de base de toutes les variables esclaves. Par exemple EB01 définirait l'utilisation du protocole Eurotherm Bisync avec tous les esclaves ayant un UID de 0 et un numéro de voie de 1.

Pour plus de détails, utiliser l'aide en ligne CTRL F1.

RealFormat (RLF)

Sélectionne le format à utiliser pour les nombres réels transmis sur la ligne de communication, avec le protocole Eurotherm Bisync. Pour plus de détails, se reporter à la description du bloc fonction EIBisync Slave.

L'utilisation avec le bloc fonction EuroPanel 2,nécesssite que ce paramètre soit impérativement un caractère unique, pour assurer un affichage correct des sorties numériques.

Pour plus de détails, utiliser l'aide en ligne CTRL F1.

WriteInhibit (WI)

Il est possible de protéger les paramètres du programme contre toute écriture lorsqu'ils sont transmis sur une ligne de communication, en mettant cette entrée sur **Rd_Only**. La lecture des paramètres du programme reste possible par la ligne de communication.

Si ce paramètre est mis sur **Rd_Wr** les paramètres du programme peuvent être à la fois écrits et lus par la ligne de communication.

ProgNumber (PN)

Le bloc fonction **Prog8Rate** établit les références croisées entre 32 numéros de programme et 32 noms de programme associés. Lorsqu'un **ProgNumber** donné est choisi, le nom de programme qui a été associé en dernier à ce numéro de programme est affiché sur l'entrée/sortie **ProgName**.

ProgName (PNM)

Entrée/sortie servant à entrer le nom du programme associé au **ProgNumber** actuel. Le nom entré doit être un nom de fichier valable pour le système de fichiers PC3000, de telle sorte que les programmes puissent être sauvegardés et lus dans l'enregistrement fichier.

Reset_Output (ROP)

Valeur indiquée par **Output** quand **Mode** est mis sur **Reset**. Tant que le mode est Reset, toute modification de **Reset_Output** se répercute sur la sortie.

Start_Mode (STM)

Quand un programme est chargé, si le mode était **Run** au début de l'opération de chargement, le mode après chargement sera fixé conformément à **Start_Mode** du nouveau programme chargé. Les valeurs possibles pour **Start_Mode** sont :

Reset(0) :	Après chargement du programme, Mode est mis sur Reset.
Run(1) :	Après chargement du programme, Mode est mis sur Run.
Hold(2) :	Après chargement du programme, Mode est mis sur Hold.
Track(3) :	Après chargement du programme, Mode est mis sur Track.
SkipSeg(4) :	Après chargement du programme, Mode est mis sur SkipSeg.
NxtUpSeg(5) :	Après chargement du programme, Mode est mis sur NxtUpSeg.
NxtDnSeg(6) :	Après chargement du programme, Mode est mis sur NxtDnSeg.

Toutefois, si le programme précédent était terminé (c'est-à-dire si **ProgramEnd** est Yes avant la sélection de Load), le nouveau programme est chargé et, une fois le chargement fini, **Mode** retourne à Run, mais l'indicateur de fin de programme reste vrai et toutes les sorties conservent les valeurs atteintes avant que le chargement ne soit demandé.

Process_Val (PV)

Normalement, la **Process_Val** de la boucle PID actuellement pilotée par le bloc fonction PROGRAMMATEUR doit être câblée sur cette entrée. Celle-ci donne des informations au bloc fonction qui permet de prendre des décisions d'après les conditions réelles de fonctionnement. **Output** peut être réglée pour suivre cette entrée ou bien la rampe peut être temporairement arrêtée si le procédé piloté

s'écarte de la consigne de plus d'une quantité donnée (ce que l'on appelle le "maintien sur écart"). L'application détermine si une connexion est nécessaire sur cette entrée ou non.

HB_Mode (HM)

En modifiant cette entrée, il est possible de choisir la façon dont le maintien sur écart opère. Holdback agit au cours d'une rampe - il n'y a pas de maintien sur écart automatique au cours d'un palier.

Off(0) : Pas de maintien sur écart en fonctionnement.

Lower(1) : Si **HB_Mode** est mis sur Lower, le maintien sur écart est activé si **Process_Val** est inférieure ou égale à Output moins **HB_Deviation.** En d'autres termes, **Output** s'arrête à sa position actuelle jusqu'à ce que **Process_Val** augmente. La sortie **HB_Active** est également vraie dans ces circonstances. Ce mode est normalement utilisé quand il y a un risque de voir la consigne (c'est-à-dire la sortie du bloc fonction du programmateur) prendre de l'avance sur le procédé dans les rampes ascendantes.

Upper(2) : Si **HB_Mode** est mis sur Upper, le maintien sur écart est activé si **Process_Val** est supérieure ou égale à Output plus **HB_Deviation.** En d'autres termes, **Output** s'arrête à sa position actuelle jusqu'à ce que **Process_Val** diminue. La sortie **HB_Active** est également vraie dans ces circonstances. Ce mode est normalement utilisé quand il y a un risque de voir la consigne (c'est-à-dire la sortie du bloc fonction du programmateur) prendre de l'avance sur le procédé dans les rampes descendantes.

Band(3) : Si l'écart absolu entre **Process_Val** et Output est supérieur ou égal à **HB_Deviation,** le maintien sur écart devient actif si **HB_Mode** est mis sur Band. En d'autres termes, **Output** s'arrête à sa position actuelle jusqu'à ce que cette situation ne soit plus vraie. La sortie **HB_Active** est également vraie dans ces circonstances. Ce mode est utilisé pour empêcher le procédé d'être trop en retard sur la consigne (c'est-à-dire la sortie du bloc fonction du programmateur) sur les rampes ascendantes ou descendantes.

HB_Deviation (HD)

Utilisé en association avec **Process_Val** et **Output** pour déterminer si le maintien sur écart doit être activé, compte tenu du paramétrage de **HB_Mode.** Il définit les limites par rapport à Output dans lesquelles **Process_Val** doit rester.

End_Segment (ES)

Le dernier segment à considérer comme faisant partie d'un profil. Ceci permet à un profil d'être arrêté avant que tous les segments non nuls aient été exécutés. Tout changement apporté à ce paramètre alors que le programme tourne devient effectif

si le segment désigné pour être le segment terminal n'a pas encore été exécuté en totalité.

Num_Loops (NL)

Nombre de fois où le profil en cours doit être répété avant de passer au programme suivant ou d'indiquer que le programme est terminé si NextProgNum est None (aucun).

NextProgNum (NPN)

Cette entrée permet le chargement d'un nouveau programme à partir de l'enregistrement fichier, une fois le programme en cours terminé.

None(0) : Aucun programme ne sera chargé, une fois le programme en cours terminé.

Prog_1(1) : Le programme dont le nom est donné en tant que **ProgName** quand **ProgNumber**=1 sera chargé à partir de l'enregistrement fichier PC3000 quand le programme en cours sera terminé. S'il n'y a pas de **ProgName** associé à **ProgNumber**=1, **ProgStatus** affiche **LoadErr** et **Mode** passe à Reset.

Prog_2(2) : Le programme dont le nom est donné en tant que **ProgName** quand **ProgNumber**=2 sera chargé à partir de l'enregistrement fichier PC3000 quand le programme en cours sera terminé. S'il n'y a pas de **ProgName** associé à **ProgNumber**=2, **ProgStatus** affiche **LoadErr** et **Mode** passe à Reset.

"

etc. jusqu'à

Prog_32 (32).

RampTime1 (RT1)

Temps nécessaire à **Output** pour progresser de sa valeur actuelle au niveau de la rampe du segment 1 dans le format de temps standard PC3000.

RampDO1 (RD1)

Donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Time à un état particulier pendant la rampe du segment 1. La valeur de cette variable est interprétée comme une configuration 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

RampLvl1 (RL1)

Niveau de **Output** qui, lorsqu'il est atteint provoque le démarrage du palier du segment 1.

DwellTime1 (DT1)

Durée pendant laquelle **Output** reste stationnaire à **RampLvl1** lorsque la partie rampe du segment 1 est achevée.

DwellDO1 (DD1)

Donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Time à un état particulier pendant le palier du segment 1. La valeur de cette variable est interprétée comme une configuration 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

RampTime2 (RT2)

Temps nécessaire à **Output** pour progresser de sa valeur actuelle au niveau de la rampe du segment 2 dans le format de temps standard PC3000.

RampDO2 (RD2)

Donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Time à un état particulier pendant la rampe du segment 2. La valeur de cette variable est interprétée comme une configuration 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

RampLvl2 (RL2)

Niveau de **Output** qui, lorsqu'il est atteint, provoque le démarrage du palier du segment 2.

DwellTime2 (DT2)

Durée pendant laquelle **Output** reste stationnaire à **RampLvl1** lorsque la partie rampe du segment 2 est achevée.

DwellDO2 (DD2)

Donne la possibilité de mettre chacune des 8 sorties numériques du bloc fonction Prog8Timeà un état particulier pendant le palier du segment 2. La valeur de cette variable est interprétée comme une configuration 8 bits, le bit le plus faible réglant **Dig_Out_1** et le bit le plus fort réglant **Dig_Out_8**.

"

etc. jusqu'à

DwellDO8 (DD8)

Output (OP)

Valeur actuelle du profil en cours de réalisation par le programme en cours. Cette sortie est normalement câblée sur l'entrée **Setpoint** d'une boucle PID ou utilisée de toute autre manière en tant que consigne d'une action de régulation.

HB_Active (HA)

Si **HB_Active** est vrai, ceci indique que la valeur actuellement envoyée sur l'entrée **Process_Val** s'écarte de la valeur actuelle de **Output** d'une quantité supérieure à ce qui est défini par **HB_Mode** et **HB_Deviation.**

ProgramEnd (PE)

Cette sortie signale la fin du programme actuel ou bien, en cas de chaînage de programmes par **NextProgNum**, la fin du dernier programme de la chaîne.

Status (SS)

Cette sortie signale la réussite ou tout autre résultat de la dernière opération d'enregistrement ou de chargement d'un fichier de l'enregistrement fichier PC3000 ou bien, pendant une opération sur un fichier, elle indique son avancement.

Ok(0) : La dernière opération, enregistrement ou chargement, effectuée sur un fichier, a été réussie.

Saving(1) : Une opération de sauvegarde en enregistrement fichier est en cours.

Loading(2) : Une opération de chargement est en cours à partir de l'enregistrement fichier.

SaveErr(3) : La dernière opération de sauvegarde en enregistrement fichier a échoué. Les raisons peuvent être diverses : pas d'enregistrement fichier, enregistrement fichier plein, nom de fichier indiqué dans **ProgName** incorrect etc.

LoadErr(4) : La dernière opération de chargement à partir de l'enregistrement fichier a échoué. Les raisons peuvent être diverses : pas d'enregistrement fichier, nom de fichier indiqué dans **ProgName** incorrect ou n'est pas un fichier de recette etc. Cette erreur est également signalée en cas de chargement d'un programme sauvegardé par un bloc fonction Prog8Rate.

LoopsRemain (LR)

Nombre de boucles (répétitions) du programme en cours qui restent à effectuer, une fois la boucle actuelle terminée.

CurrentSeg (CS)

Numéro du segment du programme actuel en cours d'exécution. Ce peut être soit la partie rampe soit la partie palier du segment.

CurrentMode (CM)

Indique si c'est la partie rampe ou la partie palier du segment qui est en cours d'exécution. Normalement, la partie rampe d'un segment est suivie par un palier, puis l'exécution du segment suivant commence.

Toutefois, si **Mode** est mis sur Track alors que le palier a déjà commencé, **CurrentMode** passe à Ramp. Si **Mode** repasse à Run, **Output** augmente en rampe jusqu'au niveau **RampLvl** du segment actuel, avec la durée **RampTime** du segment actuel, puis **CurrentMode** indique à nouveau Dwell. La durée de cette seconde période de palier est la durée totale **DwellTime** du segment en cours.

CurrentTmRem (CTR)

Temps nécessaire pour achever l'opération indiquée par **CurrentMode** (c'est-à-dire une rampe ou un palier) en supposant qu'il n'y a pas d'interruption due à l'activation du maintien sur écart ou au passage à un autre mode que Run.

Le calcul se fait d'après la valeur actuelle de **Output** et d'après les paramètres de profil du segment en cours.

En cas d'activation du maintien sur écart ou de changement de mode, **CurrentTmRem** indique le temps requis pour finir l'opération indiquée par **CurrentMode** si le maintien sur écart devenait immédiatement inactif ou si **Mode** repassait immédiatement sur Run.

SegTmRem (STR)

Temps nécessaire pour achever le segment en cours en supposant qu'il n'y a pas d'interruption due à l'activation du maintien sur écart ou au passage à un autre mode que Run. Le calcul se fait d'après la valeur actuelle de **Output** et d'après les paramètres de profil du segment en cours.

En cas d'activation du maintien sur écart ou de changement de mode, **SegTmRem** indique le temps requis pour achever le segment en cours si le maintien sur écart devenait immédiatement inactif ou si **Mode** repassait immédiatement sur Run.

ProgTmRem (PTR)

Temps nécessaire pour finir le nombre restant de boucles du segment en cours en supposant qu'il n'y a pas d'interruption due à l'activation du maintien sur écart ou au passage à un autre mode que Run. Le calcul se fait d'après la valeur actuelle de **Output**, les paramètres de profil du segment en cours et le paramètre **Num_Loops**.

En cas d'activation du maintien sur écart ou de changement de mode, **ProgTmRem** indique le temps requis pour achever le segment en cours si le maintien sur écart devenait immédiatement inactif ou si **Mode** repassait immédiatement sur Run.

Si **Mode** est mis sur Run et si des paramètres de programme sont modifiés, **ProgTmRem** n'est recalculé qu'au départ du segment suivant ou au passage d'une rampe à un palier.

Dig_Out_1 (DO1)

Sortie numérique pilotée par le bit 0 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 0 de **RampDO4** est à 1 et le bit 0 de **DwellDO4** est à 0, **Dig_Out_1** est alors activée pendant la rampe du segment 4 et **Dig_Out_1** est désactivée pendant le palier du segment 4.

Dig_Out_2 (DO2)

Sortie numérique pilotée par le bit 1 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 1 de **RampDO4** est à 1 et le bit 1 de **DwellDO4** est à 0, **Dig_Out_2** est activée pendant la rampe du segment 4 et **Dig_Out_2** est désactivée pendant le palier du segment 4.

Dig_Out_3 (DO3)

Sortie numérique pilotée par le bit 2 des entrées/sorties **2ampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 2 de **RampDO4** est à 1 et le bit 2 de **DwellDO4** est à 0, **Dig_Out_3** est activée pendant la rampe du segment 4 et **Dig_Out_3** est désactivée pendant le palier du segment 4.

Dig_Out_4 (DO4)

Sortie numérique pilotée par le bit 3 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 3 de **RampDO4** est à 1 et le bit 3 de **DwellDO4** est à 0, **Dig_Out_4** est activée pendant la rampe du segment 4 et **Dig_Out_4** est désactivée pendant le palier du segment 4.

Dig_Out_5 (DO5)

Sortie numérique pilotée par le bit 4 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 4 de **RampDO4** est à 1 et le bit 4 de **DwellDO4** est à 0, **Dig_Out_5** est activée pendant la rampe du segment 4 et **Dig_Out_5** est désactivée pendant le palier du segment 4.

Dig_Out_6 (DO6)

Sortie numérique pilotée par le bit 5 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 5 de **RampDO4** est à 1 et le bit 5 de **DwellDO4** est à 0, **Dig_Out_6** est activée pendant la rampe du segment 4 et **Dig_Out_6** est désactivée pendant le palier du segment 4.

Dig_Out_7 (DO7)

Sortie numérique pilotée par le bit 6 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 6 de **RampDO4** est à 1 et le bit 6 de **DwellDO4** est à 0, **Dig_Out_7** est activée pendant la rampe du segment 4 et **Dig_Out_7** est désactivée pendant le palier du segment 4.

Dig_Out_8 (DO8)

Sortie numérique pilotée par le bit 7 des entrées/sorties **RampDO** et **DwellDO** du segment actuel. Par exemple, si le bit 7 de **RampDO4** est à 1 et le bit 7 de **DwellDO4** est à 0, **Dig_Out_8** est activée pendant la rampe du segment 4 et **Dig_Out_8** est désactivée pendant le palier du segment 4.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Mode	ENUM	Reset (0)	Oper	Oper	Cf. liste de paramètres	
Address	STR	'EPP1'	Super	Super	4 caractères maxi	
RealFormat	STR	"	Super	Super	Caractère unique	
WriteInhibit	BOOL	Rd_Wr(0)	Super	Super	Sens	Rd_Wr(0) Rd_Only(1)
ProgNumber	DINT	1	Oper	Oper	Lim. haute Lim. basse	32 1
ProgName	STR	"	Oper	Oper	12 caractères maxi	
Reset_Output	REAL	0	Super	Super	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
Start_Mode	ENUM	Run (1)	Oper	Oper	Cf. liste de paramètres	
Process_Val	REAL	0	Oper	Oper	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
HB_Mode	ENUM	Off (0)	Oper	Oper	Cf. liste de paramètres	
HB_Deviation	REAL	0	Oper	Oper	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
End_Segment	DINT	8	Super	Super	Lim. haute Lim. basse	8 1
Num_Loops	DINT	1	Super	Super	Lim. haute Lim. basse	999 0ms
NextProgNum	ENUM	None(0)	Super	Super	Cf. liste de paramètres	
RampTime1 to RampTime8	REAL	0	Super	Super	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
RampDO1 to RampDO8	DINT	0	Super	Super	Lim. haute Lim. basse	256 0
RampLv11 to RampLv18	REAL	0	Super	Super	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38
DwellTime1 to DwellTime8	TIME	0ms	Super	Super	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
DwellD01-to DwellD08	DINT	0	Super	Super	Lim. haute Lim. basse	256 0
Output	REAL	0	Oper	Bloc	Lim. haute Lim. basse	+3.402823E+38 -3.402823E+38

Tableau 15-6 Attributs des paramètres Prog8Time (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
HB_Active	**BOOL**	No(0)	Oper	Bloc	Sens	No(0) Yes(1)
ProgramEnd	**BOOL**	Yes(1)	Oper	Bloc	Sens	No(0) Yes(1)
Status	**ENUM**	Ok(0)	Oper	Bloc	Cf. liste de paramètres	
LoopsRemain	**DINT**	0	Oper	Bloc	Lim. haute Lim. basse	999 0
CurrentSeg	**DINT**	8	Oper	Bloc	Lim. haute Lim. basse	8 1
CurrentMode	**BOOL**	Dwell(1)	Oper	Bloc	Sens	Ramp(0) Dwell(1)
CurrentTmRem	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
SegTmRem	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
ProgTmRem	**TIME**	0ms	Oper	Bloc	Lim. haute Lim. basse	23d23h59m59s999ms 0ms
Dig_Out_1 to Dig_Out_8	**BOOL**	Off(0)	Oper	Bloc	Sens	Off(0) On(1)

Tableau 15-5 Attributs des paramètres Prog8Time

Chapitre 16

ETAPES

Edition 1

Vue d'ensemble

Ce chapitre décrit les blocs fonctions, étapes et étapes de Macro qui sont créés dans le cadre d'un GRAFCET (SFC).

Contrairement aux autres blocs fonctions, les étapes et macros tournent toujours sur la même tâche, c'est-à-dire la tâche associée à l'exécution du Grafcet d'ensemble dont ils font partie.

Ces blocs fonctions sont un moyen pratique d'établir les informations associées aux étapes et étapes de Macro de GRAFCET, et peuvent servir pour synchroniser des actions dans des branches parallèles du programme séquentiel.

BLOC FONCTION MACRO

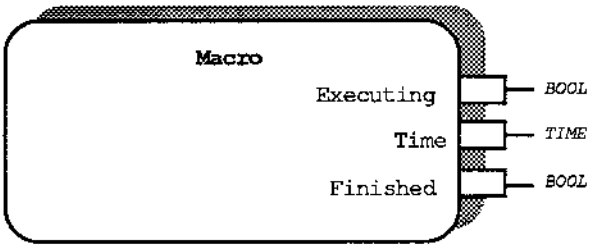


Figure 16-1 Schéma du bloc fonction Macro

Description fonctionnelle

Le bloc fonction Macro met à la disposition de l'utilisateur un jeu de paramètres qui définissent l'état de l'exécution d'une étape de Macro. Un bloc fonction est créé avec chaque nouvelle définition de Macro, dans le GRAFCET. Le bloc indique la durée pendant laquelle la Macro a été en cours d'exécution, et si l'exécution a été achevée.

Attributs du bloc fonction

Type : 40 10
Classe : ..... ETAPES
Tâche par défaut : ..... Task_2
Liste récapitulative : ..... Executing, Time, Finished
Besoins de capacité mémoire : ..... 6 octets

Description des paramètres

Executing (X)

Executing (en cours d'exécution) indique si la Macro à laquelle fait référence le bloc fonction est actuellement active. Si Executing est mis à Yes (1), la Macro est active. Si Executing est mis à No (0), la Macro est inactive.

Time (T)

Time indique la durée pendant laquelle la Macro a été en cours d'exécution. Si la Macro n'est pas active actuellement, Time indique la durée de l'exécution précédente.

Finished (F)

Finished indique si la Macro est actuellement en cours d'exécution. Si Finished est mis à No (0), la Macro est en cours d'exécution ou n'a pas encore été exécutée, et n'a pas encore terminé toutes ses étapes. Si Finished est mis à Yes (1), la Macro a terminé l'exécution de toutes ses étapes.

Attributs des paramètres

Nom	Type	Câblage	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Executing	BOOL	Sortie	No (0)	Oper		Détect.	No (0) Yes (1)
Time	TIME	Sortie	0	Oper			
Finished	BOOL	Sortie	No (0)	Oper		Détect.	No (0) Yes (1)

Tableau 16-1 Attributs des paramètres de Macro

BLOC FONCTION STEP

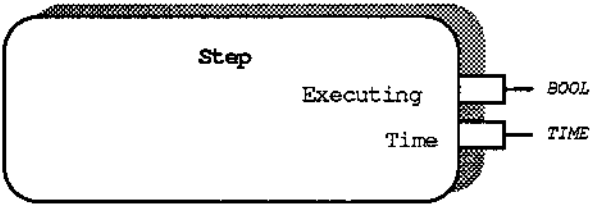


Figure 16-2 Schéma du bloc fonction Step

Description fonctionnelle

Le bloc fonction Step (étape) met à la disposition de l'utilisateur un jeu de paramètres qui définissent l'état d'exécution d'une étape. Un bloc fonction est créé avec chaque nouvelle définition de Step, dans le GRAFCET. Le bloc indique la durée pendant laquelle Step a été exécuté, et si Step est en cours d'activité.

Attributs du bloc fonction

Type : 40 20

Classe : ..... STEPS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Executing, Time

Besoins de capacité mémoire : ..... 6 octets

Description des paramètres

Executing (X)

Executing (en cours d'exécution) indique si l'étape (Step) à laquelle fait référence le bloc fonction est actuellement active. Si Executing est mis à Yes (1), l'étape est active. Si Executing est mis à No (0), l'étape est inactive.

Time (T)

Time indique la durée pendant laquelle l'étape (Step) a été en cours d'exécution. Si l'étape n'est pas active actuellement, Time indique la durée de l'exécution précédente.

Attributs des paramètres

Nom	Type	Câblage	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Executing	BOOL	Sortie	No (0)	Oper		Détect.	No (0) Yes (1)
Time	TIME	Sortie	0	Oper			

Tableau 16-2 Attributs des paramètres de Step

Chapitre 17

COMPTEURS

Edition 1

Vue d'ensemble

Ce chapitre décrit les blocs fonctions de la classe des COMPTEURS qui fournissent une variété de fonctions de comptage d'usage général, en particulier des compteurs, des décompteurs, et des compteurs réversibles.

BLOC FONCTION UP_COUNTER

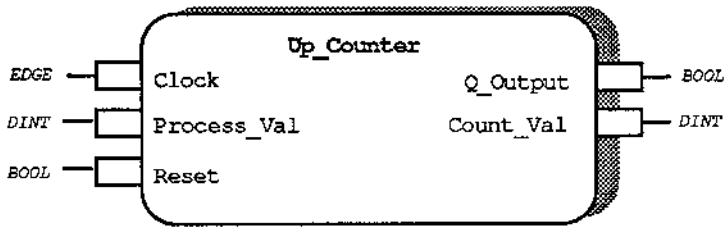


Figure 17-1 Schéma du bloc fonction Up_Counter

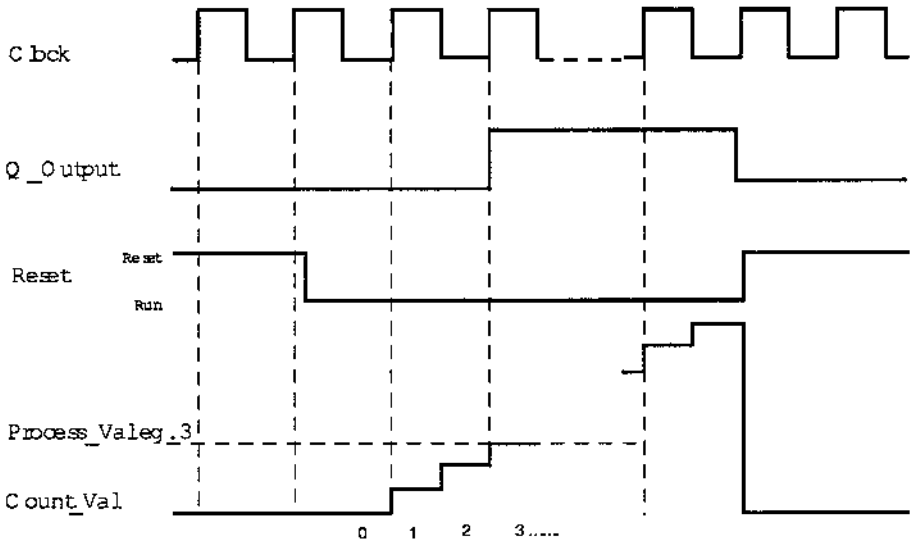
Description fonctionnelle

Le bloc fonction Up_Counter incrémente un compteur à chaque fois que son entrée passe de 0 logique à 1 logique. Le bloc a deux modes de fonctionnement, qui sont définis par le paramètre Reset. La valeur de comptage est conservée dans Count_Val. Dans les deux modes de fonctionnement, si Count_Val est inférieur à Process_Val, Q_Output est mis à Off (0), et si Count_Val est égal ou supérieur à Process_Val, Q_Output est mis à On (1).

Modes de fonctionnement

Run (0) : En mode Run (Marche), la valeur de Count_Val s'incrémente à chaque fois que l'entrée Clock (Horloge) passe de Tock (0) à Tick (1). La vitesse de changement doit être inférieure à deux fois la durée de la tâche, car il est nécessaire de repasser Clock de Tick (1) à Tock (0) entre les comptages.

Reset (1) : En mode Reset, Count_Val est remis à 0 et le comptage est inhibé.



Attributs du bloc fonction

Type : 72 16

Classe : ..... COMPTEURS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Reset, Process_Val, Q_Out, Count_Val

Besoins de capacité mémoire : ..... 12 octets

Durée d'exécution : ..... 19,3 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Clock	BOOL	Tock (0)	Oper	Oper	Détect	Tock (0) Tick (1)
Process_Val	DINT	0	Oper	Oper	Lim. haute Lim. basse	2,147,483,646 0
Reset	BOOL	Run (0)	Oper	Oper	Détect	Run (0) Reset (1)
Q_Output	BOOL	Off (0)	Oper	Bloc	Détect	Off (0) On (1)
Count_Val	DINT	0	Oper	Bloc	Lim. haute Lim. basse	2,147,483,646 0

Tableau 17-1 Attributs des paramètres de Up_Counter

BLOC FONCTION DN_COUNTER

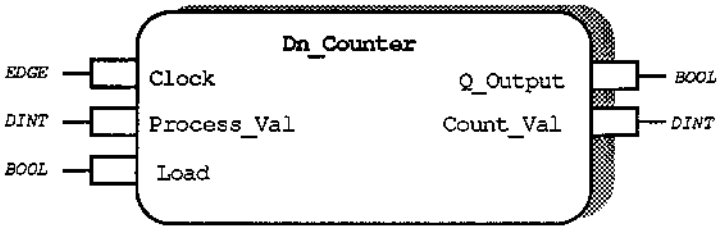


Figure 17-2 Schéma du bloc fonction Dn_Counter

Description fonctionnelle

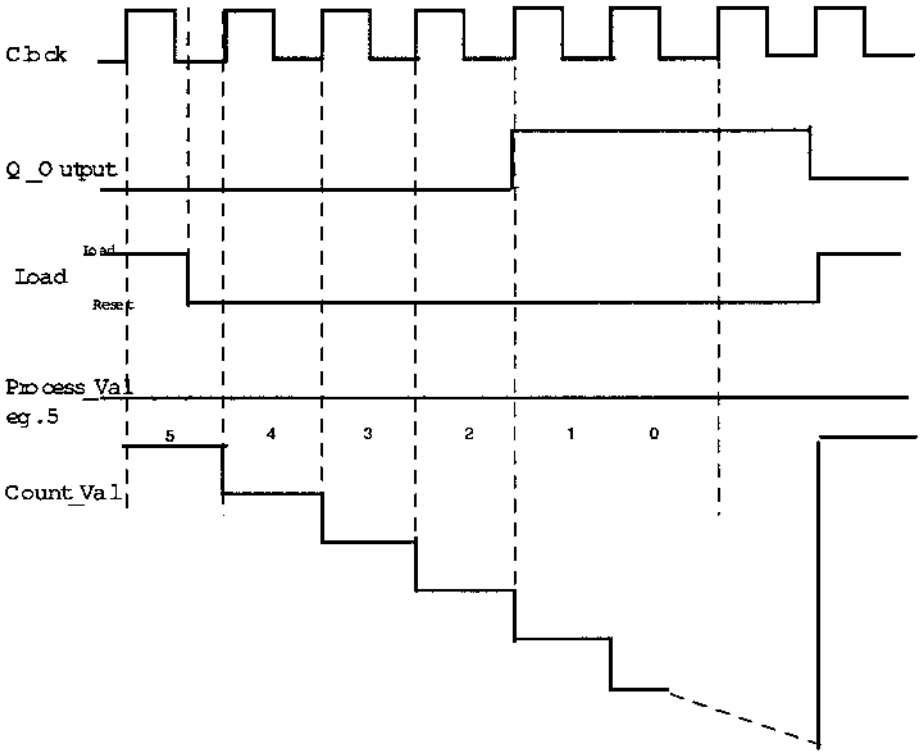
Le bloc fonction Dn_Counter décrémente un compteur à chaque fois que son entrée passe de 1 logique à 0 logique. Le bloc a deux modes de fonctionnement, qui sont définis par le paramètre Load (Chargement).

Modes de fonctionnement

Run (0) : En mode Run (Marche), la valeur de Count_Val s'incrémente à chaque fois que l'entrée Clock (Horloge) passe de Tock (0) à Tick (1). Si Count_Val est supérieur à 0, Q_Output est mis à Off (0). Si Count_Val est égal à 0 ou négatif, Q_Output est mis à On (1).

Note: La vitesse de changement de l'entrée Clock doit être inférieure à deux fois la durée de la tâche, car il est nécessaire de repasser Clock de Tick (1) à Tock (0) entre les comptages.

Load (1) : En mode Load, la valeur entrée dans Process_Val (Valeur de processus) est chargée dans Count_Val (valeur de comptage) pour former la valeur initiale du comptage. Si Process_Val (et Count_Val) est supérieur à 0, Q_Output est mis Off (0). Si Process_Val (et Count_Val) est égal à 0 ou négatif, Q_Output est mis à On (1).



Attributs du bloc fonction

Type : 72 32
Classe : ..... COUNTERS
Tâche par défaut : ..... Task_1
Liste récapitulative : ..... Load, Process_Val, Q_Output, Count_Val
Besoins de capacité mémoire : ..... 12 octets
Durée d'exécution : ..... 18,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Clock	BOOL	Tock (0)	Oper	Oper	Détect	Tock (0) Tick (1)
Process_Val	DINT	0	Oper	Oper	Lim. haute Lim. basse	2,147,483,646 0
Reset	BOOL	Run (0)	Oper	Oper	Détect	Run (0) Reset (1)
Q_Output	BOOL	Off (0)	Oper	Bloc	Détect	Off (0) On (1)
Count_Val	DINT	0	Oper	Bloc	Lim. haute Lim. basse	2,147,483,646 0

Tableau 17-2 Attributs des paramètres de Dn_Counter

BLOC FONCTION UP_DN_COUNT

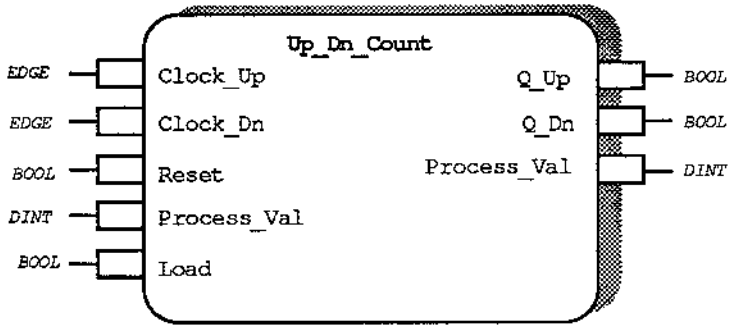


Figure 17-3 Bloc fonction Up_Dn_Count

Description fonctionnelle

Le bloc fonction Up_Dn_Count compte sur le front positif de Clock_Up et décompte sur le front positif de Clock_Dn. La valeur du comptage est conservée dans Count_Val. Le bloc a quatre modes de fonctionnement, qui sont définis par les paramètres Reset et Load. Dans tous les modes de fonctionnement la relation suivante existe entre les valeurs de Count_Val, Process_Val et 0 :

Si Count_Val est supérieur ou égal à Process_Val, Q_Up est mis à On (1). Si Count_Val est inférieur à Process_Val, Q_Up est mis à Off (0).

Si Count_Val est supérieur à 0, Q_Dn est mis à Off (0). Si Count_Val est inférieur ou égal à 0, Q_Dn est mis à On (1).

Se reporter au diagramme des temps de Up_Counter et Dn_Counter.

Modes de fonctionnement

{i} Reset = Run (0) et Load = Run (0)

La valeur de Count_Val est incrémentée à chaque fois que Clock_Up passe de Tock (0) à Tick (1) et est décrémentée à chaque fois que Clock_Dn passe de Tock (0) à Tick (1). Il y a lieu de noter que la vitesse de changement doit être inférieure à deux fois la durée de la tâche, car il est nécessaire de repasser Clock de Tick (1) à Tock (0) entre les comptages.

{ii} Reset = Run (0) et Load = Load (1)

La valeur de Process_Val est chargée dans Count_Val. La modification de Clock_Up ou Clock_Dn n'affecte pas la valeur de Count_Val.

(iii) Reset = Reset (1) et Load = Run (0)

La valeur de Count_Val est remise à 0 et maintenue à O. La modification de Clock_Up ou Clock_Dn n'affecte pas la valeur de Count_Val.

(iv) Reset = Reset (1) et Load = Load (1)

La valeur de Count_Val est remise à 0 et maintenue à O. La modification de Clock_Up ou Clock_Dn n'affecte pas la valeur de Count_Val.

Attributs du bloc fonction

Type : 72 48

Classe : ..... COUNTERS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Process_Val, Q_Up, Q_Dn, Count_Val

Besoins de capacité mémoire : ..... 16 octets

Durée d'exécution : ..... 33,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Clock_Dn	BOOL	Tock (0)	Oper	Oper	Détect	Tock (0) Tick (1)
Clock_Up	BOOL	Tock (0)	Oper	Oper	Détect	Tock (0) Tick (1)
Count_Val	DINT	0	Oper	Bloc	Lim. haute Lim. basse	Comme pour Process_Val Comme pour Process_Val
Load	BOOL	Run (0)	Oper	Oper	Détect	Run (0) Load (1)
Process_Val	DINT	0	Oper	Oper	Lim. haute Lim. basse	2,147,483,646 -2,147,483,646
Q_Dn	BOOL	Off (0)	Oper	Bloc	Détect	Off (0) On (1)
Q_Up	BOOL	Off (0)	Oper	Bloc	Détect	Off (0) On (1)
Reset	BOOL	Run (0)	Oper	Oper	Détect	Run (0) Reset (1)

Tableau 17-3 Attributs des paramètres de Up_Dn_Count

Chapitre 18

SELECT

Edition 1

Sommaire (suite)

Vue d'ensemble

Ce chapitre décrit une classe de blocs fonctions qui réalisent une opération de multiplexage. La valeur de sortie est choisie parmi 16 valeurs, en fonction de la valeur d'une entrée indexée. Des systèmes de recettes simples peuvent être constitués à partir de ces blocs fonctions. Les données pouvant être accueillies sont :

booléennes (Boolean), réelles (Real), entières (Integer), temps (Time) et chaîne (String).

BLOC FONCTION SELECT_REAL

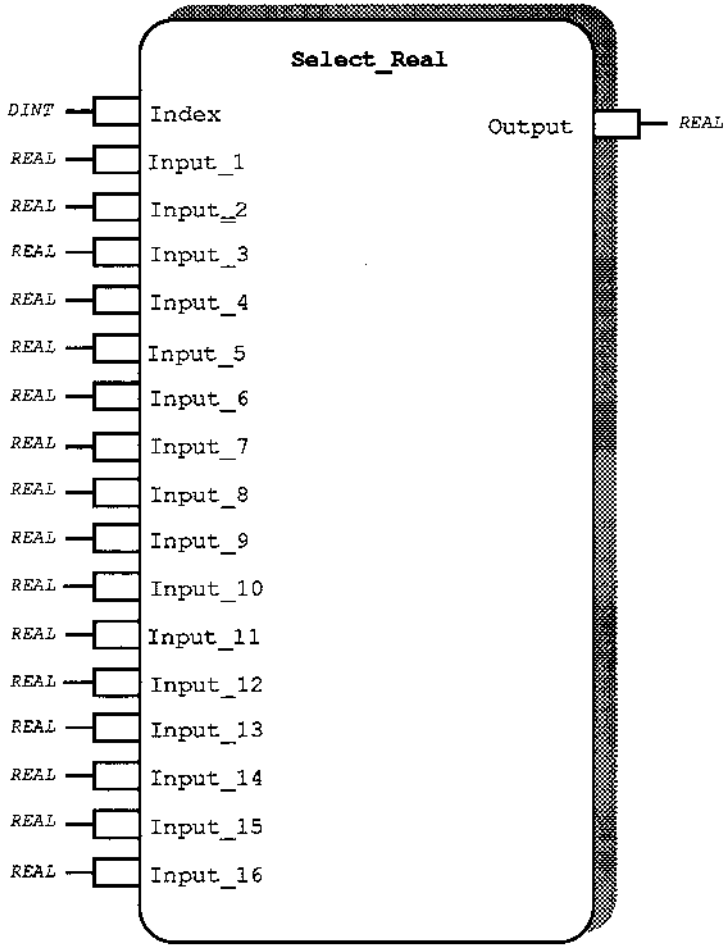


Figure 18-1 Schéma du bloc fonction Select_Real

Description fonctionnelle

Le bloc fonction Select_Real est un multiplexeur à 16 canaux pour nombres réels (REAL) qui fournit un moyen de sélectionner une sortie à virgule flottante (REAL) à partir de 16 entrées à virgule flottante (REAL). Le bloc fonction dispose d'une entrée entière (Index), de 16 points d'entrées à virgule flottante (REAL) (Input_1 à Input_16) et d'une sortie à virgule flottante (REAL) (Output). La valeur de Output est choisie parmi les 16 entrées réelles selon la relation :

Index = n

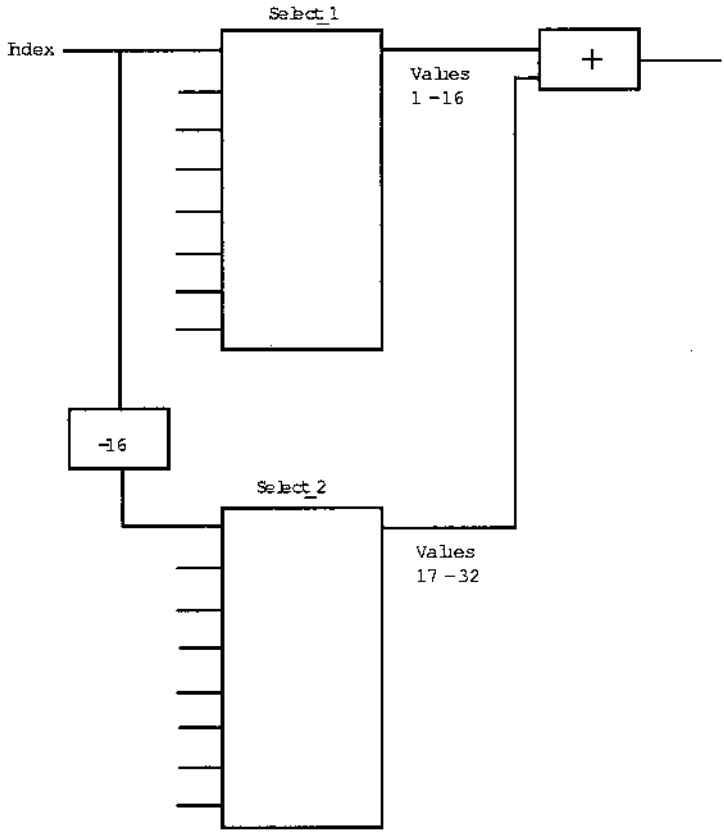
si $\leq n < 16$

Output : = Input - n

si $n \leq 0$ ou $n > 16$

Output : = 0

Cette dernière fonction permet de constituer un système de recettes simple en utilisant les blocs SELECT, par exemple :



Dans l'exemple, Select_1 fournit les 16 premières valeurs, et la sortie de Select_2 est à 0. Si l'index est supérieur à 16, Select_2 fournit les 16 valeurs suivantes et la sortie de Select_1 est à 0. Les deux sorties peuvent être simplement additionnées pour obtenir la valeur utilisée par la stratégie de commande.

Attributs du bloc fonction

Type : 80 16

Classe : ..... SELECT

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Index, Output

Besoins de capacité mémoire : ..... 72 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Index	DINT	0	Oper	Oper	Lim. haute Lim. basse	16 0
Input_1 to Input_16	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	999 999 -999 999
Output	REAL	0,0	Oper	Bloc	Lim. haute Lim. basse	999 999 -999 999

Tableau 18-1 Attributs des paramètres de Select_Real

BLOC FONCTION SELECT_INT

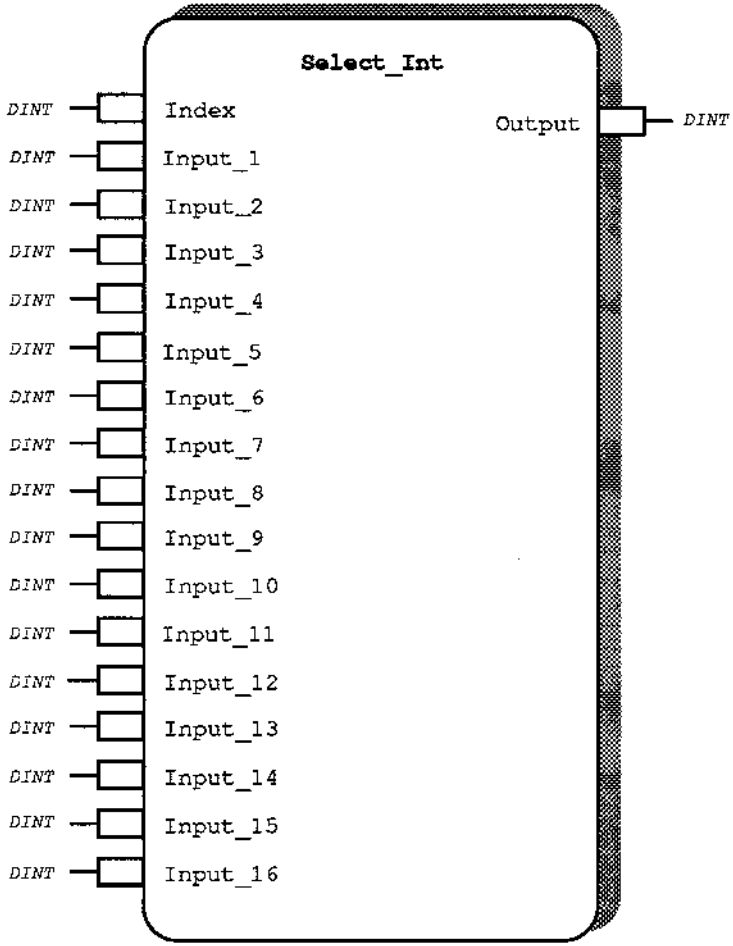


Figure 18-2 Schéma du bloc fonction Select_Int

Description fonctionnelle

Le bloc fonction Select_Int est un multiplexeur à 16 canaux pour des nombres entiers (DINT) qui fournit un moyen de sélectionner une sortie entière parmi 16 entrées entières. Le bloc fonction dispose de 17 entrées entières (Index et Input_1 à Input_16) et d'une sortie entière (Output). La valeur de Output est choisie parmi les 16 entrées entières, Input_1 à Input_16 selon la relation :

Index = n

si $1 \leq n < 16$

Output : = Input - n

si $n \leq 0$ ou $n > 16$

Output : = 0

Voir exemple d'utilisation de cette fonction, dans Select_Real.

Attributs du bloc fonction

Type : 50 20

Classe : .....SELECT

Tâche par défaut : .....Task_2

Liste récapitulative : .....Index, Output

Besoins de capacité mémoire : .....72 octets

Durée d'exécution : .....19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Index	DINT	0	Oper	Oper	Lim. haute Lim. basse	16 0
Input_1 to Input_16	DINT	0	Oper	Oper	Lim. haute Lim. basse	999 999 -999 999
Output	DINT	0	Oper	Bloc	Lim. haute Lim. basse	999 999 -999 999

Tableau 18-2 Attributs des paramètres de Select_Int

BLOC FONCTION SELECT_TIME

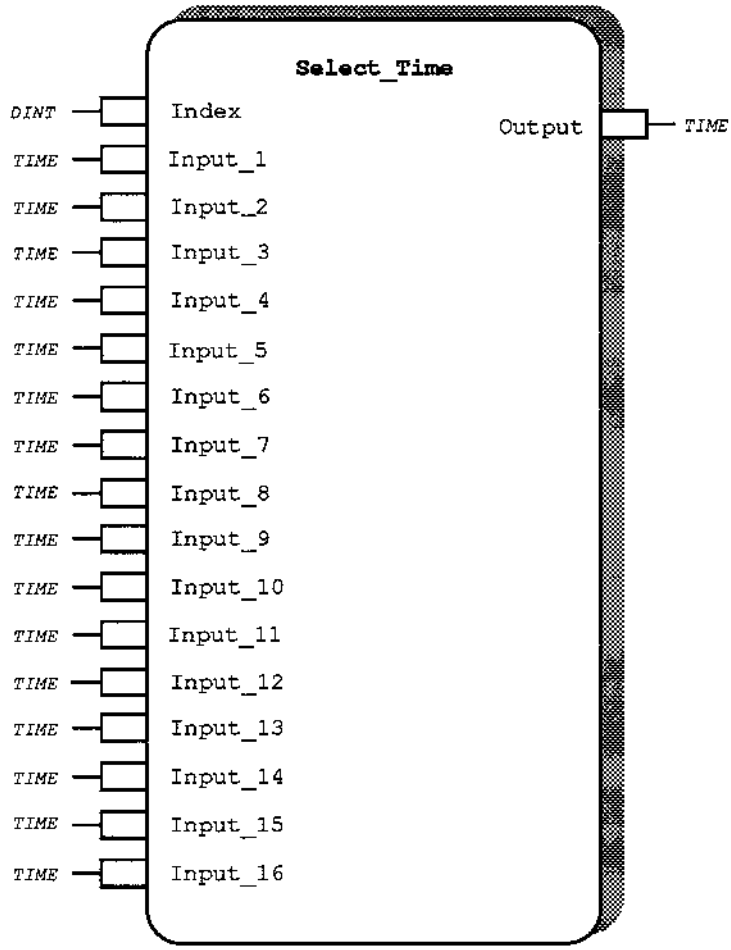


Figure 18-3 Schéma du bloc fonction Select_Time

Description fonctionnelle

Le bloc fonction Select_Time est un multiplexeur à 16 canaux pour variables de durée (TIME) qui fournit un moyen de sélectionner une sortie de durée parmi 16 entrées de durée. Le bloc fonction dispose d'une entrée entière (Index), de 16 entrées de durée (Input_1 à Input_16) et d'une sortie de durée (Output). La valeur de Output est choisie parmi les 16 entrées de durée, selon la relation :

Index = n

si $1 \leq n < 16$

Output : = Input - n

si $n \leq 0$ ou $n > 16$

Output : = 0ms

Voir exemple d'utilisation de cette fonction, dans Select_Real.

Attributs du bloc fonction

Type : 50 30

Classe : .....SELECT

Catégorie : .....Normale

Tâche par défaut : .....Task_2

Liste récapitulative : .....Index, Output

Besoins de capacité mémoire : .....72 octets

Durée d'exécution : ..... 19,5 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Index	DINT	0	Oper	Oper	Lim. haute Lim. basse	16 0
Input_1 to Input_16	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23d_23h _59ms 0 ms
Output	TIME	0 ms	Oper	Bloc	Lim. haute Lim. basse	23d_23h _59m 0 ms

Tableau 18-3 Attributs des paramètres de Select_Time

BLOC FONCTION SELECT_BOOL

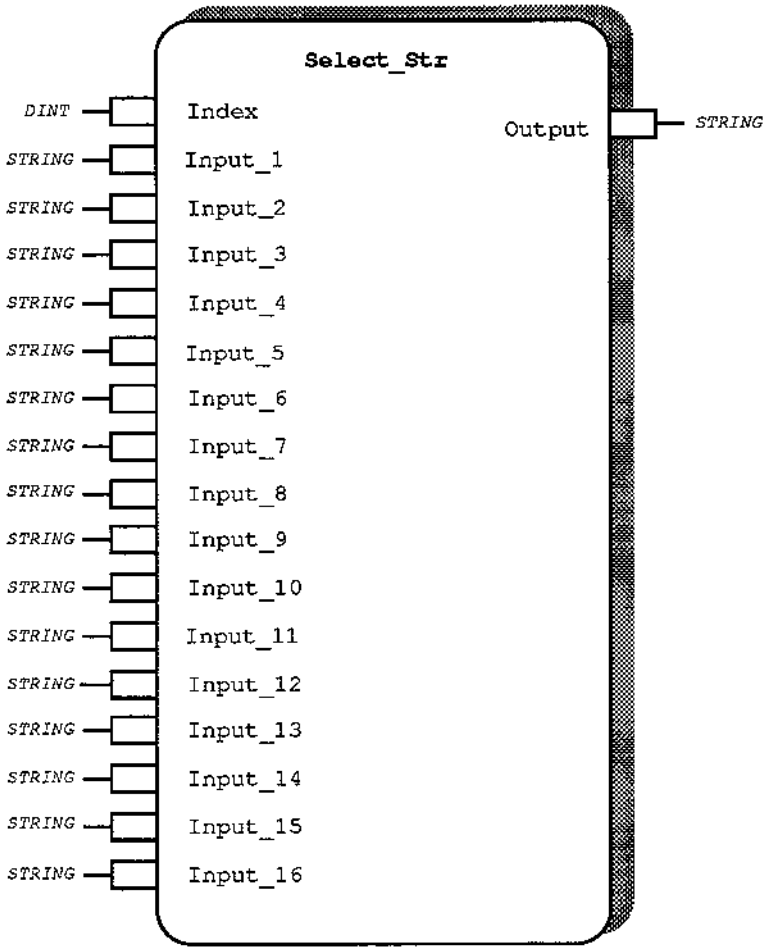


Figure 18-4 Schéma du bloc fonction Select_Bool

Description fonctionnelle

Le bloc fonction Select_Bool est un multiplexeur à 16 canaux pour variables booléennes (BOOL) qui fournit un moyen de sélectionner une sortie booléenne (BOOL) parmi 16 entrées booléennes (BOOL). Le bloc fonction dispose d'une entrée entière (Index), de 16 entrées booléennes (BOOL) (Input_1 à Input_16) et d'une sortie booléenne (BOOL) (Output). La valeur de Output est choisie parmi les 16 entrées booléennes (BOOL), selon la relation :

Index = n

si $1 \leq n < 16$

Output : = Input - n

si $n \leq 0$ ou $n > 16$

Output : = OFF (0)

Voir exemple d'utilisation de cette fonction, dans Select_Real.

Attributs du bloc fonction

Type : 50 40

Classe : ..... SELECT

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Index, Output

Besoins de capacité mémoire : ..... 22 octets

Durée d'exécution : ..... 17,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Index	DINT	0	Oper	Oper	Lim. haute Lim. basse	16 0
Input_1 to Input_16	BOOL	OFF (0)	Oper	Oper	Détect.	OFF (0) ON (1)
Output	BOOL	OFF (0)	Oper	Bloc	Détect.	OFF (0) ON (1)

Tableau 18-4 Attributs des paramètres de Select_Bool

BLOC FONCTION SELECT_STR

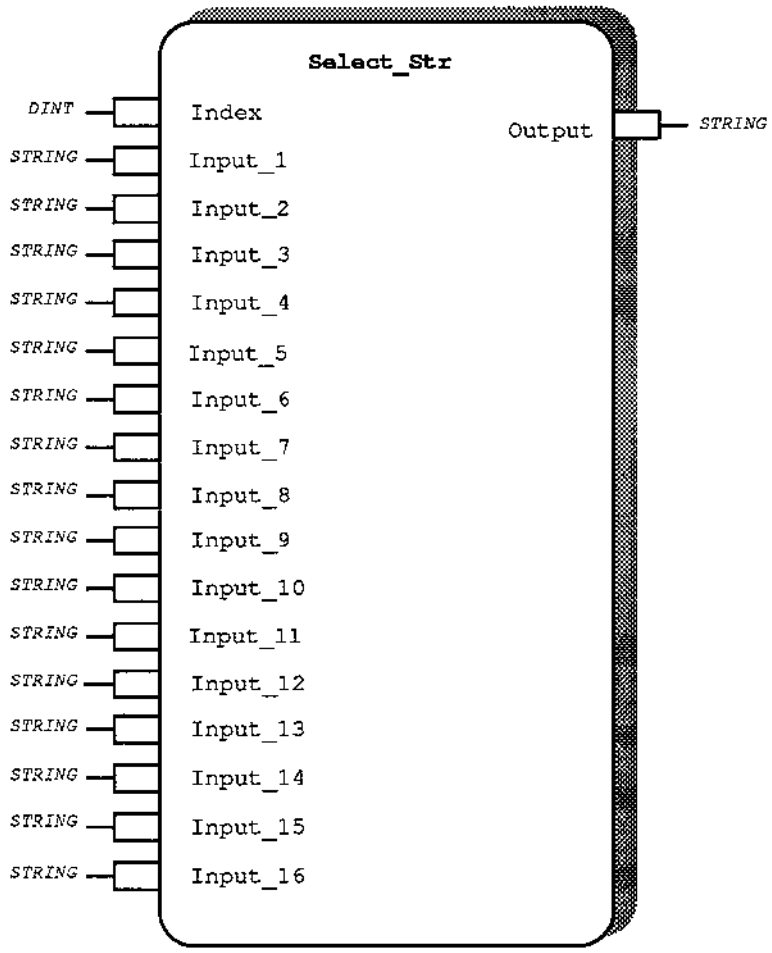


Figure 18-5 Schéma du bloc fonction Select_Str

Description fonctionnelle

Le bloc fonction Select_Str est un multiplexeur à 16 canaux pour variables de chaîne (STRING) qui fournit un moyen de sélectionner une sortie de chaîne (STRING) parmi 16 entrées de chaîne (STRING). Le bloc fonction dispose d'une entrée entière (Index), de 16 entrées de chaîne (Input_1 à Input_16) et d'une sortie de chaîne (STRING) (Output). La valeur de Output est choisie parmi les 16 entrées de chaîne, selon la relation :

Index = n

si $1 \leq n < 16$

Output : = Input - n

si $n \leq 0$ ou $n > 16$

Output : = '' (chaîne vide)

Voir exemple d'utilisation de cette fonction, dans Select_Real. La concaténation des sorties de plusieurs blocs SELECT peut s'effectuer au moyen de la fonction ST, CONCAT. Pour plus de détails, voir Manuel des fonctions PC3000.

Attributs du bloc fonction

Type : 50 80

Classe : ..... SELECT

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Index, Output

Besoins de capacité mémoire : ..... 1398 octets

Durée d'exécution : ..... 68,2 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Index	DINT	0	Oper	Oper	Lim. haute Lim. basse	16 0
Input_1 to Input_16	STRING	''	Oper	Oper	Longueur maximale	80 caractères
Output	STRING	''	Oper	Bloc	Longueur maximale	80 caractères

Tableau 18-5 Attributs des paramètres de Select_Str

BLOC FONCTION SET_REAL

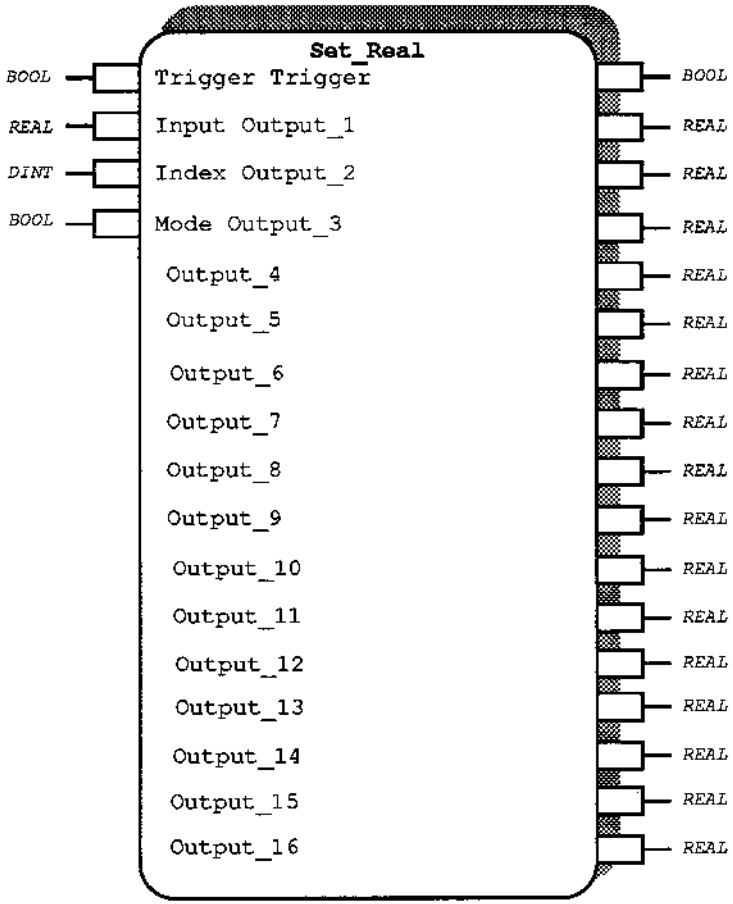


Figure 18-6 Schéma du bloc fonction Set_Real

Description fonctionnelle

Un ensemble de seize registres de type réel est maintenu sous la forme de seize sorties Output_1 à Output_16. Les valeurs contenues dans ces registres sont saisies à l'aide d'Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée Index.

Ce bloc fonction peut fonctionner dans un mode parmi deux : les registres peuvent être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou ils sont uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Attributs du bloc fonction

Type : 50 90
Classe : ..... Select
Tâche par défaut : ..... Task_2
Liste raccourcie : ..... Input, Index, Mode, Trigger
Mémoire nécessaire : 78 octets

Description des paramètres

Trigger (TRG)

Lorsque **Mode** est positionné sur Trig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **Index**. Par exemple, si **Input** est égal à 10,3 et Index est égal à 5, lorsque Trigger est activé, **Output_5** affiche également la valeur 10,3, à condition que **Mode** soit sur **Trig**.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Input (IN)

Valeur copiée dans le registre spécifié par **Index.**

Index (I)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur Trig.

Mode (M)

Détermine le mode de mise à jour des registres. Le registre indiqué par **Index** prend immédiatement la valeur d'**Input** si **Mode** est sur Cont mais, si **Mode** est sur Trig, ce changement se produira lors du prochain positionnement de **Trigger** sur On.

Output_1 (O1) à Output_16 (O16)

Valeurs des seize registres dans lesquels **Input** peut être entré.

Seul le registre indiqué par **Index** est touché par le changement de valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Input	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Mode	**BOOL**	Trig (0)	Oper	Oper	Sens	Trig (0) Cont (1)
Output_1 à Output_16	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 18-6 Attributs des paramètres Set_Real

BLOC FONCTION SET_DINT

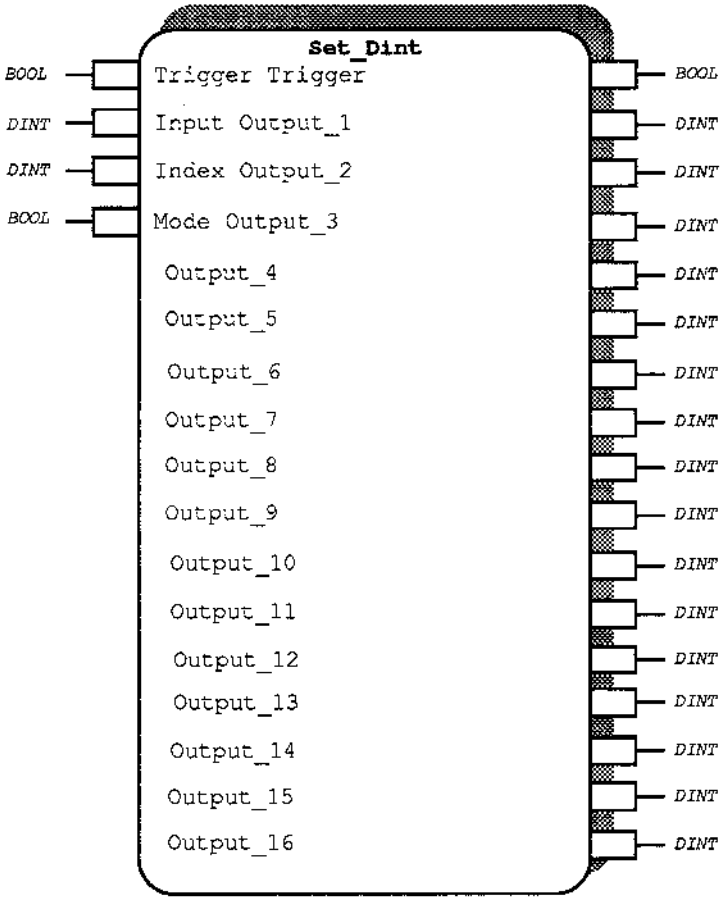


Figure 18-7 Schéma du bloc fonction Set_Dint

Description fonctionnelle

Un ensemble de seize registres de type entier double est maintenu sous la forme de seize sorties Output_1 à Output_16. Les valeurs contenues dans ces registres sont saisies à l'aide d'Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée Index.

Ce bloc fonction peut fonctionner dans un mode parmi deux : les registres peuvent être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou ils sont uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Attributs du bloc fonction

Type : 5092
Classe : .....SELECT
Tâche par défaut : .....Task_2
Liste raccourcie : .....Input, Index, Mode, Trigger
Mémoire nécessaire : 78 octets

Description des paramètres

Trigger (TRG)

Lorsque **Mode** est positionné sur Trig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **Index.** Par exemple, si **Input** est égal à 37 et Index est égal à 5, lorsque Trigger est activé, **Output_5** affiche également la valeur 37, à condition que **Mode** soit sur Trig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Input (IN)

Valeur copiée dans le registre spécifié par **Index.**

Index (I)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur Trig.

Mode (M)

Détermine le mode de mise à jour des registres. Le registre indiqué par **Index** prend immédiatement la valeur d'**Input** si **Mode** est sur Cont mais, si **Mode** est sur Trig, ce changement se produira lors du prochain positionnement de **Trigger** sur On.

Output_1 (O1) à Output_16 (O16)

Valeurs des seize registres dans lesquels **Input** peut être entré.

Seul le registre indiqué par **Index** est touché par le changement de valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Input	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Mode	**BOOL**	Trig (0)	Oper	Oper	Sens	Trig (0) Cont (1)
Output_1 à Output_16	**REAL**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648

Tableau 18-7 Attributs des paramètres Set_Dint

BLOC FONCTION SET_TIME

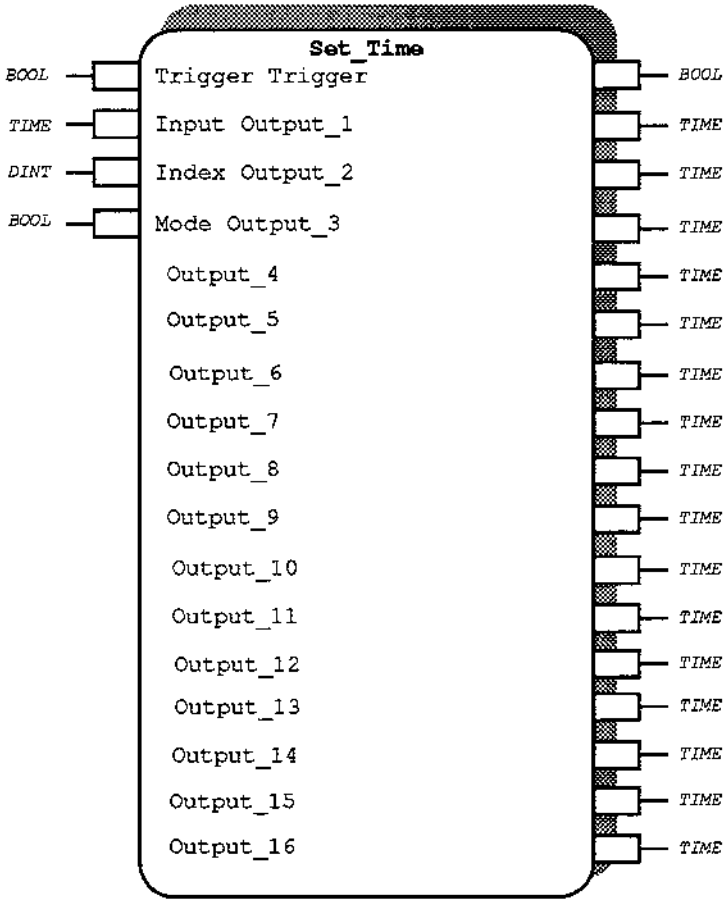


Figure 18-8 Schéma du bloc fonction Set_Time

Description fonctionnelle

Un ensemble de seize registres de type temps est maintenu sous la forme de seize sorties Output_1 à Output_16. Les valeurs contenues dans ces registres sont saisies à l'aide d'Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée Index.

Ce bloc fonction peut fonctionner dans un mode parmi deux : les registres peuvent être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou ils sont uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Attributs du bloc fonction

Type : 50 94
Classe : ..... SELECT
Tâche par défaut : ..... Task_2
Liste raccourcie : ..... Input, Index, Mode, Trigger
Mémoire nécessaire : 78 octets

Description des paramètres

Trigger (TRG)

Lorsque **Mode** est positionné sur Trig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **Index.** Par exemple, si **Input** est égal à 55s et Index est égal à 5, lorsque Trigger est activé, **Output_5** affiche également la valeur 55s, à condition que **Mode** soit sur Trig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Input (IN)

Valeur copiée dans le registre spécifié par **Index.**

Index (I)

Registre dans lequel doit être copié **Input,** soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur Trig.

Mode (M)

Détermine le mode de mise à jour des registres. Le registre indiqué par **Index** prend immédiatement la valeur d'**Input** si **Mode** est sur Cont mais, si **Mode** est sur Trig, ce changement se produira lors du prochain positionnement de **Trigger** sur On.

Output_1 (O1) à Output_16 (O16)

Valeurs des seize registres dans lesquels **Input** peut être entré.

Seul le registre indiqué par **Index** est touché par le changement de valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Input	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999m 0
Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Mode	**BOOL**	Trig (0)	Oper	Oper	Sens	Trig (0) Cont (1)
Output_1 à Output_16	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999m 0

Tableau 18-8 Attributs des paramètres Set_Time

BLOC FONCTION SET_BOOL

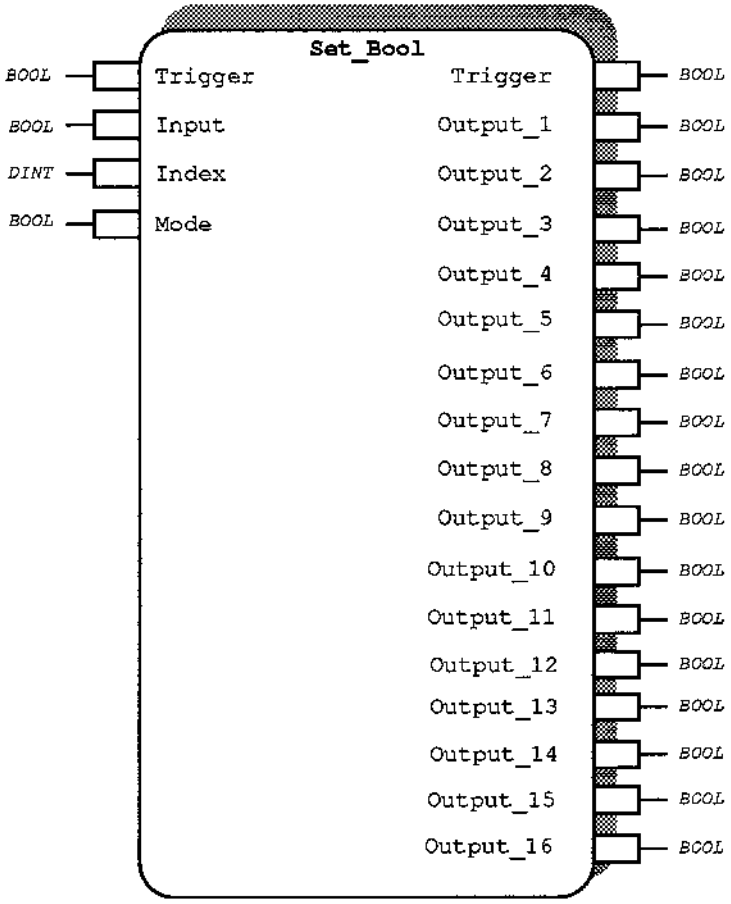


Figure 18-9 Schéma du bloc fonction Set_Bool

Description fonctionnelle

Un ensemble de seize registres de type booléen est maintenu sous la forme de seize sorties Output_1 à Output_16. Les valeurs contenues dans ces registres sont saisies à l'aide d'Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée Index.

Ce bloc fonction peut fonctionner dans un mode parmi deux : les registres peuvent être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou ils sont uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Attributs du bloc fonction

Type : 5096
Classe : ..... SELECT
Tâche par défaut : ..... Task_1
Liste raccourcie : ..... Input, Index, Mode, Trigger
Mémoire nécessaire : 28 octets

Description des paramètres

Trigger (TRG)

Lorsque **Mode** est positionné sur Trig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **Index.** Par exemple, si **Input** est sur On et Index est égal à 5, lorsque Trigger est activé, **Output_5** affiche également la valeur On, à condition que **Mode** soit sur Trig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Input (IN)

Valeur copiée dans le registre spécifié par **Index.**

Index (I)

Registre dans lequel doit être copié **Input,** soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur Trig.

Mode (M)

Détermine le mode de mise à jour des registres. Le registre indiqué par **Index** prend immédiatement la valeur d'**Input** si **Mode** est sur Cont mais, si **Mode** est sur Trig, ce changement se produira lors du prochain positionnement de **Trigger** sur On.

Output_1 (O1) à Output_16 (O16)

Valeurs des seize registres dans lesquels **Input** peut être entré.

Seul le registre indiqué par **Index** est touché par le changement de valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Trigger	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Input	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Index	DINT	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Mode	BOOL	Trig (0)	Oper	Oper	Sens	Trig (0) Cont (1)
Output_1 to Output_16	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)

Tableau 18-9 Attributs des paramètres Set_Bool

BLOC FONCTION SET_STR

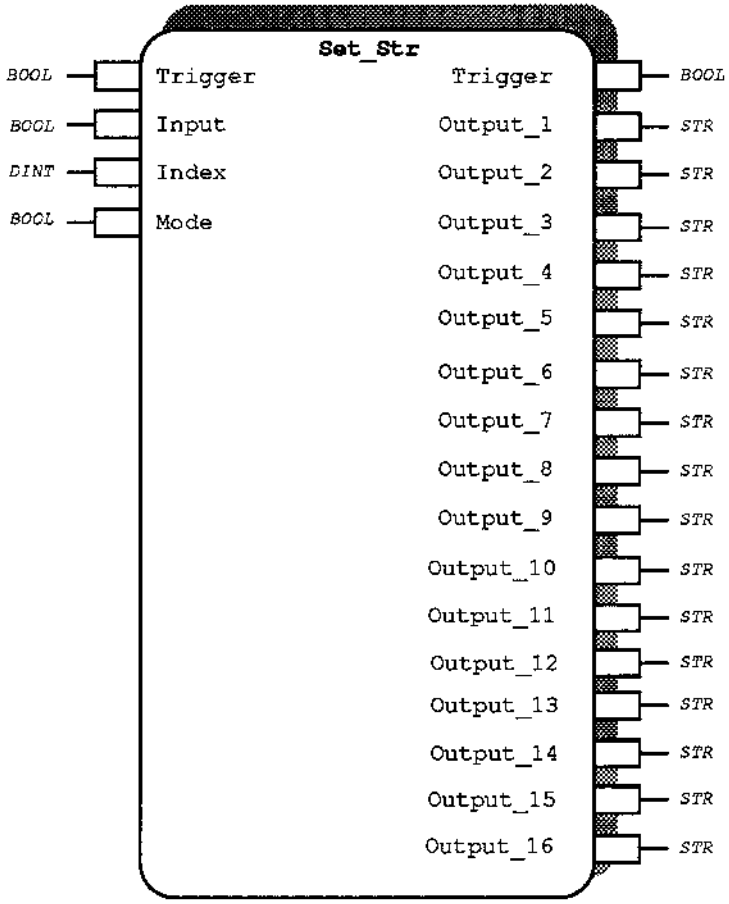


Figure 18-10 Schéma du bloc fonction Set_Str

Description fonctionnelle

Un ensemble de seize registres de type chaîne de caractères est maintenu sous la forme de seize sorties Output_1 à Output_16. Les valeurs contenues dans ces registres sont saisies à l'aide d'Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée Index.

Ce bloc fonction peut fonctionner dans un mode parmi deux : les registres peuvent être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou ils sont uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Attributs du bloc fonction

Type : 5098
Classe : ..... SELECT
Tâche par défaut : ..... Task_2
Liste raccourcie : ..... Input, Index, Mode, Trigger
Mémoire nécessaire : ..... 1404 octets

Description des paramètres

Trigger (TRG)

Lorsque **Mode** est positionné sur Trig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **Index.** Par exemple, si Input est sur "Bonjour" et Index est égal à 5, lorsque Trigger est activé, **Output_5** affiche également la valeur "Bonjour", à condition que **Mode** soit sur Trig. **Trigger** est immédiatement ramené à l'état Off (désactivé) par le bloc fonction. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Input (IN)

Valeur copiée dans le registre spécifié par **Index.**

Index (I)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur Trig.

Mode (M)

Détermine le mode de mise à jour des registres. Le registre indiqué par **Index** prend immédiatement la valeur d'**Input** si **Mode** est sur Cont mais, si **Mode** est sur Trig, ce changement se produira lors du prochain positionnement de **Trigger** sur On.

Output_1 (O1) à Output_16 (O16)

Valeurs des seize registres dans lesquels **Input** peut être entré.

Seul le registre indiqué par **Index** est touché par le changement de valeurs d'**Input**. Les autres registres conservent leurs valeurs actuelles.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Trigger	BOOL	0	Oper	Oper	Sens	Off (0) On (1)
Input	STR	"	Oper	Oper	Néant	
Index	DINT	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Mode	BOOL	Trig (0)	Oper	Oper	Sens	Trig (0) Cont (1)
Output_1 to Output_16	STR	"	Oper	Oper	Néant	

Tableau 18-10 Attributs des paramètres Set_Str

BLOC FONCTION REGISTER_REAL

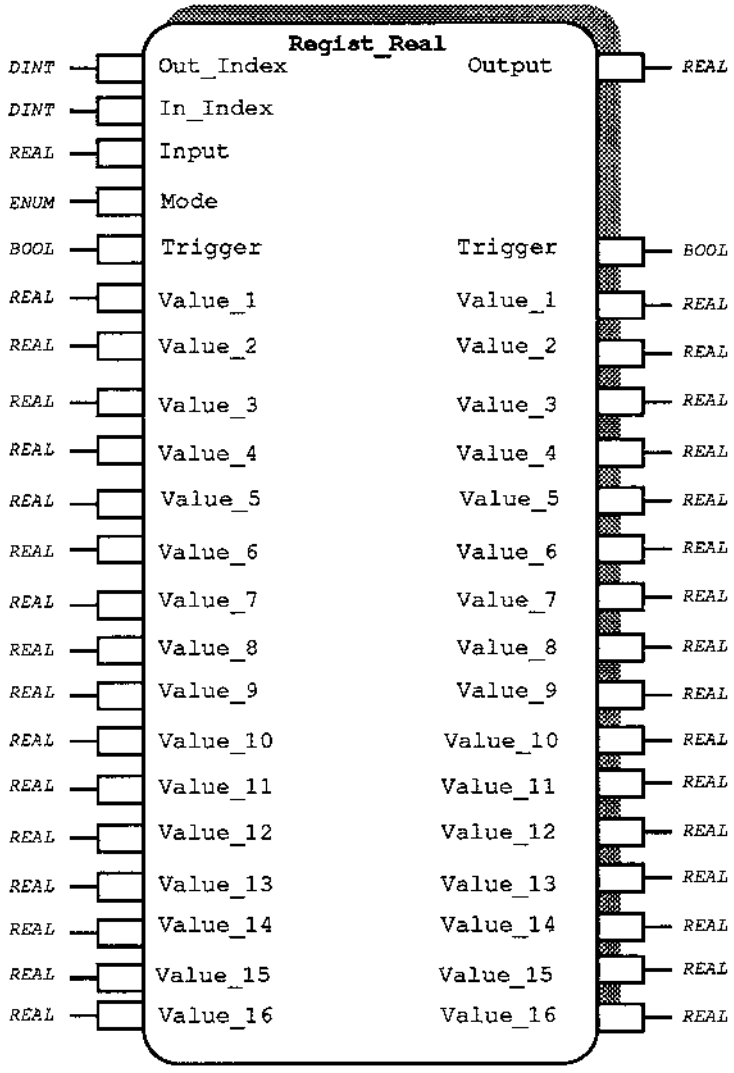


Figure 18-11 Schéma du bloc fonction Regist_Real

Description fonctionnelle

La fonction offerte par ce bloc est pour l'essentiel celle d'une matrice 16 x 1. Il est possible de stocker et de récupérer les nombres en fonction d'un index numérique.

Un ensemble de seize registres de type réel est maintenu. Les valeurs actuellement dans ces registres se présentent sous la forme de seize entrées/sorties Value_1 à Value_16. Par conséquent, il est possible de modifier directement les valeurs contenues dans ces registres en écrivant dans une de ces entrées/sorties.

Il est possible de modifier la valeur d'un de ces registres à l'aide du paramètre Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée In_Index. Cette mise à jour d'un registre peut avoir lieu d'une manière parmi deux : le registre peut être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou il est uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Il existe aussi une sortie qui indique la valeur actuelle d'un des registres. Le registre dont la valeur est indiquée est déterminé par l'entrée Out_Index.

Attributs du bloc fonction

Type : 50A0

Classe : ..... SELECT

Tâche par défaut : ..... Task_2

Liste raccourcie : ..... In_Index, Out_Index, Input, Output

Mémoire nécessaire : 86 octets

Description des paramètres

Out_Index (OID)

Détermine le registre dont provient la valeur d'**Output**. S'il est fixé à zéro, Output affiche également la valeur zéro.

In_Index (IID)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur OnTrig. **In_Index** n' aaucun effet si **Mode** est positionné sur DoNot.

Input (IN)

Valeur qui sera copiée dans le registre spécifié par **In_Index**, à condition que **Mode** soit sur Cont ou OnTrig.

Mode (M)

Détermine l'effet d'**Input** et de **Trigger** sur le registre référencé par **In_Index.**

DoNot	**Input** et **Trigger** n'ont aucun effet, quelle que soit la valeur d'**In_Index.**
Cont	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque cycle d'exécution de ce bloc fonction.
OnTrig	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque fois que **Trigger** est sur "vrai".

Trigger (TRG)

Lorsque **Mode** est positionné sur OnTrig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **In_Index.** Par exemple, si **Input** est égal à 10,3 et **In_Index** est égal à 5, lorsque Trigger est activé, **Value_5** affiche également la valeur 10,3, à condition que **Mode** soit sur OnTrig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction lorsque **Mode** est sur Cont ou sur OnTrig. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Value_1 (V1)à Value_16 (V16)

Valeurs des seize registres dans lesquels **Input** peut être entré et **Output** peut être obtenu.

Seul le registre indiqué par **In_Index** est touché par le changement des valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles sauf si l'on écrit directement dedans en positionnant sur 1 la valeur associée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Out_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
In_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Input	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Mode	**ENUM**	DoNo (0)	Oper	Oper	Cf. liste des paramètres	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Value1 à Value_16	**REAL**	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Output	**REAL**	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38

Tableau 18-11 Attributs des paramètres Regist_Real

BLOC FONCTION REGISTER_DINT

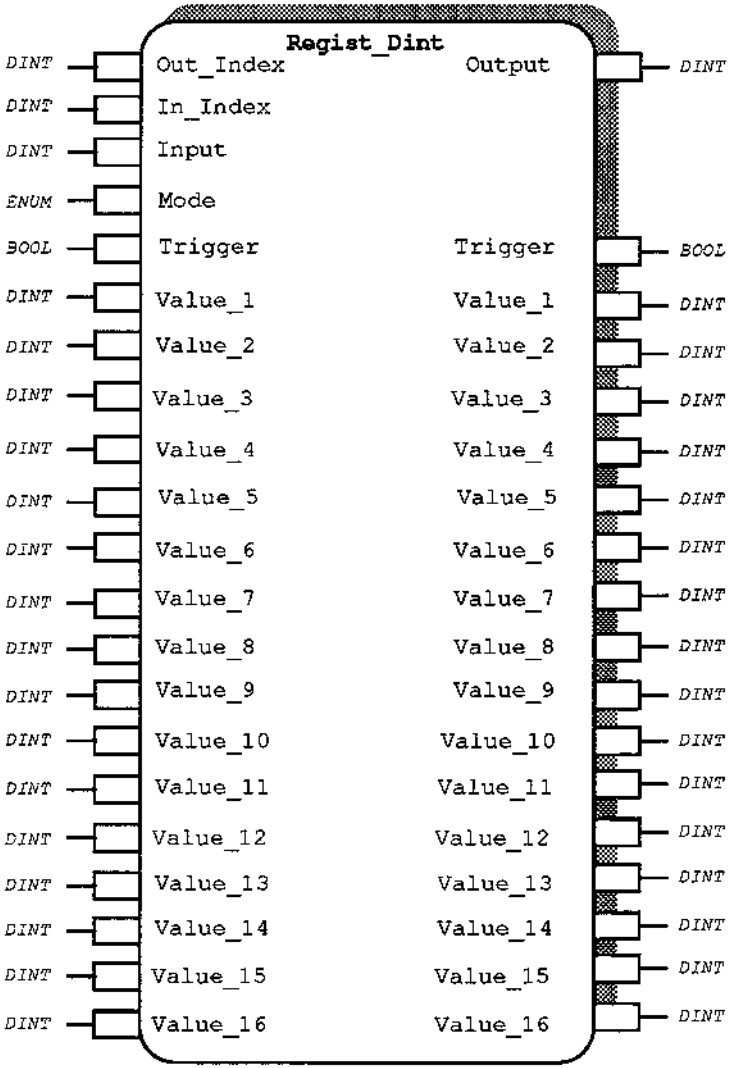


Figure 18-12 Schéma du bloc fonction Regist_Dint

Description fonctionnelle

La fonction offerte par ce bloc est pour l'essentiel celle d'une matrice 16 x 1. Il est possible de stocker et de récupérer les nombres en fonction d'un index numérique.

Un ensemble de seize registres de type entier double est maintenu. Les valeurs actuellement dans ces registres se présentent sous la forme de seize entrées/sorties Value_1 à Value_16. Par conséquent, il est possible de modifier directement les valeurs contenues dans ces registres en écrivant dans une de ces entrées/sorties. Il est aussi possible de modifier la valeur d'un de ces registres à l'aide du paramètre Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée In_Index. Cette mise à jour d'un registre peut avoir lieu d'une manière parmi deux : le registre peut être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou il est uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Il existe aussi une sortie qui indique la valeur actuelle d'un des registres. Le registre dont la valeur est indiquée est déterminé par l'entrée Out_Index.

Attributs du bloc fonction

Type : 50A2

Classe : .....SELECT

Tâche par défaut : .....Task_2

Liste raccourcie : .....In_Index, Out_Index, Input, Output

Mémoire nécessaire : 86 octets

Description des paramètres

Out_Index (OID)

Détermine le registre dont provient la valeur d'**Output**. S'il est fixé à zéro, **Output** affiche également la valeur zéro.

In_Index (IID)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur OnTrig. **In_Index** n' aucun effet si **Mode** est positionné sur DoNot.

Input (IN)

Valeur qui sera copiée dans le registre spécifié par **In_Index**, à condition que **Mode** soit sur Cont ou OnTrig.

Mode (M)

Détermine l'effet d'**Input** et de **Trigger** sur le registre référencé par **In_Index.**

DoNot	Input et Trigger n'ont aucun effet, quelle que soit la valeur d'**In_Index.**
Cont	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque cycle d'exécution de ce bloc fonction.
OnTrig	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque fois que **Trigger** est sur "vrai".

Trigger (TRG)

Lorsque **Mode** est positionné sur OnTrig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **In_Index.** Par exemple, si **Input** est égal à 37 et **In_Index** est égal à 5, lorsque Trigger est activé, **Value_5** affiche également la valeur 37, à condition que **Mode** soit sur OnTrig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction lorsque **Mode** est sur Cont ou sur OnTrig. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Value_1 (V1)à Value_16 (V16)

Valeurs des seize registres dans lesquels **Input** peut être entré et **Output** peut être obtenu.

Seul le registre indiqué par **In_Index** est touché par le changement des valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles sauf si l'on écrit directement dedans en positionnant sur 1 la valeur associée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Out_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
In_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Input	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Mode	**ENUM**	DoNo (0)	Oper	Oper	Cf. liste des paramètres	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Value_1 to Value_16	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Output	**DINT**	0	Oper	Block	Limite haute Limite basse	+2147483647 -2147483648

Tableau 18-12 Attributs des paramètres Regist_Dint

BLOC FONCTION REGISTER_TIME

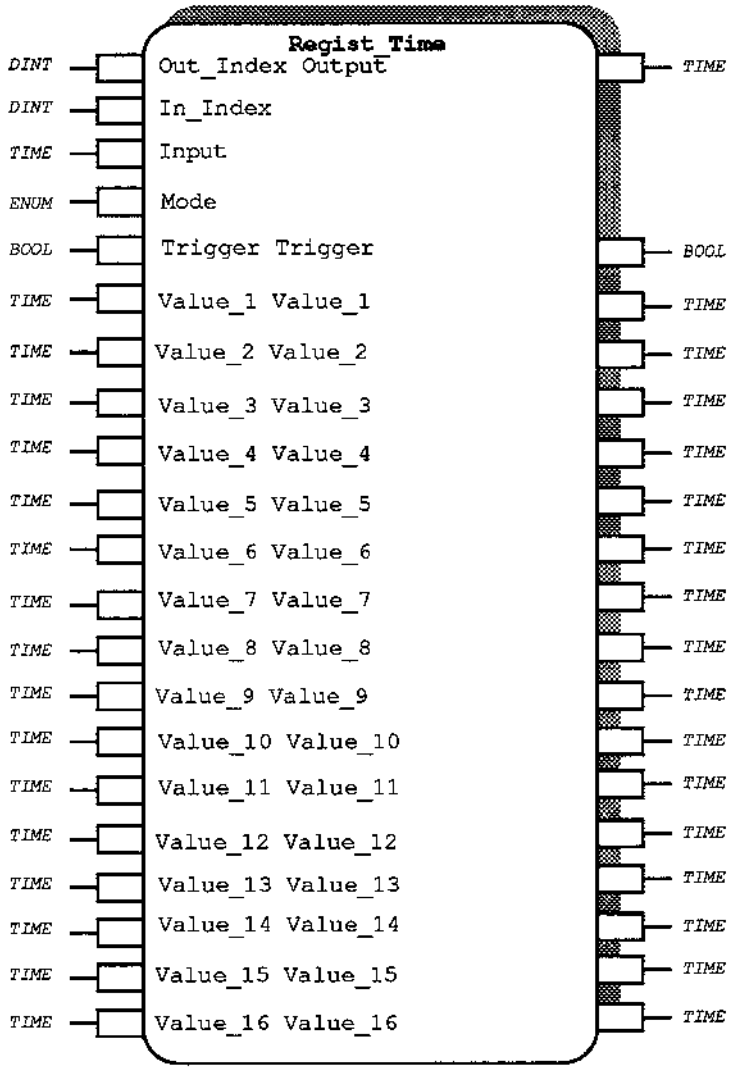


Figure 18-13 Schéma du bloc fonction Regist_Time

Description fonctionnelle

La fonction offerte par ce bloc est pour l'essentiel celle d'une matrice 16 x 1. Il est possible de stocker et de récupérer les nombres en fonction d'un index numérique.

Un ensemble de seize registres de type temps est maintenu. Les valeurs actuellement dans ces registres se présentent sous la forme de seize entrées/sorties Value_1 à Value_16. Par conséquent, il est possible de modifier directement les valeurs contenues dans ces registres en écrivant dans une de ces entrées/sorties.

Il est aussi possible de modifier la valeur d'un de ces registres à l'aide du paramètre Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée In_Index. Cette mise à jour d'un registre peut avoir lieu d'une manière parmi deux : le registre peut être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou il est uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Il existe aussi une sortie qui indique la valeur actuelle d'un des registres. Le registre dont la valeur est indiquée est déterminé par l'entrée Out_Index.

Attributs du bloc fonction

Type : 50A4

Classe : ..... SELECT

Tâche par défaut : ..... Task_2

Liste raccourcie : ..... In_Index, Out_Index, Input, Output

Mémoire nécessaire : 86 octets

Description des paramètres

Out_Index (OID)

Détermine le registre dont provient la valeur d'**Output**. S'il est fixé à zéro, **Output** affiche également la valeur zéro.

In_Index (IID)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur OnTrig. **In_Index** n'a aucun effet si **Mode** est positionné sur DoNot.

Input (IN)

Valeur qui sera copiée dans le registre spécifié par **In_Index**, à condition que **Mode** soit sur Cont ou OnTrig.

Mode (M)

Détermine l'effet d'**Input** et de **Trigger** sur le registre référencé par **In_Index.**

DoNot	Input et Trigger n'ont aucun effet, quelle que soit la valeur d'**In_Index.**
Cont	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque cycle d'exécution de ce bloc fonction.
OnTrig	le registre indiqué par **In_Index** est mis à jour avec la valeur d'Input à chaque fois que **Trigger** est sur "vrai".

Trigger (TRG)

Lorsque **Mode** est positionné sur OnTrig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **In_Index.** Par exemple, si **Input** est égal à 55s et **In_Index** est égal à 5, lorsque Trigger est activé, **Value_5** affiche également la valeur 55s, à condition que **Mode** soit sur OnTrig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction lorsque **Mode** est sur Cont ou sur OnTrig. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Value_1 (V1)à Value_16 (V16)

Valeurs des seize registres dans lesquels **Input** peut être entré et **Output** peut être obtenu.

Seul le registre indiqué par **In_Index** est touché par le changement des valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles sauf si l'on écrit directement dedans en positionnant sur 1 la valeur associée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Out_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
In_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Input	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Mode	**ENUM**	DoNot(0)	Oper	Oper	Cf. liste des paramètres	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Value_1 à Value_16	**TIME**	0	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Output	**TIME**	0	Oper	Block	Limite haute Limite basse	23d23h59m59s999m 0

Tableau 18-13 Attributs des paramètres Regist_Time

BLOC FONCTION REGISTER BOOL

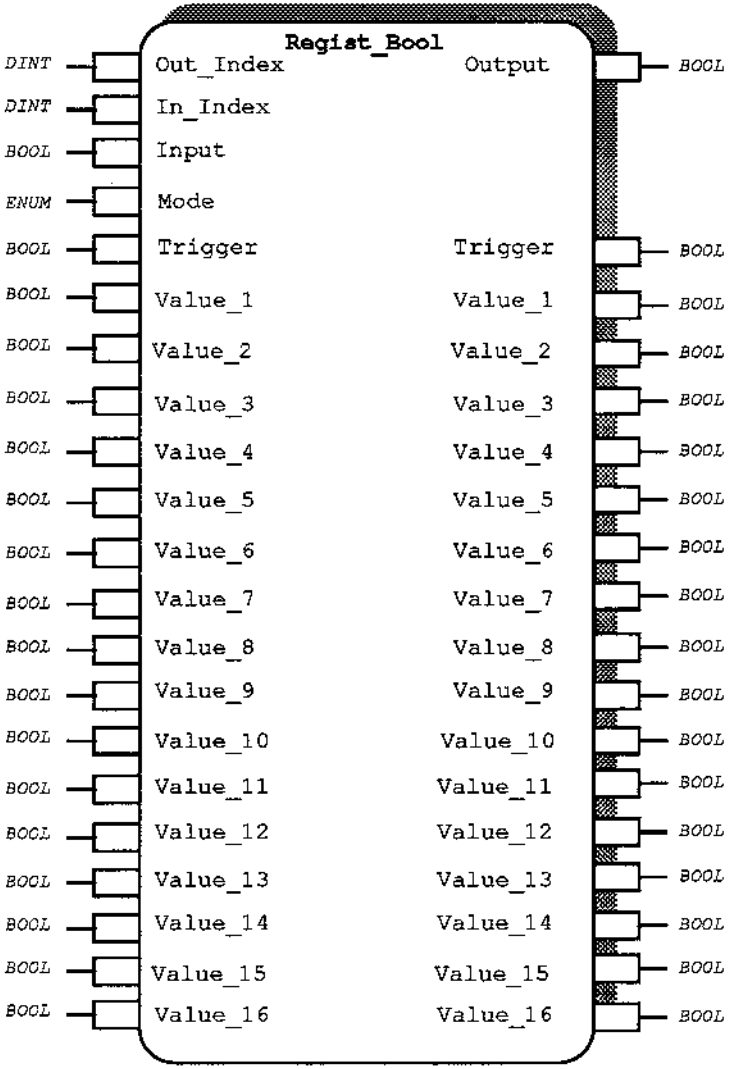


Figure 18-14 Schéma du bloc fonction Regist_Bool

Description fonctionnelle

La fonction offerte par ce bloc est pour l'essentiel celle d'une matrice 16 x 1. Il est possible de stocker et de récupérer les nombres en fonction d'un index numérique.

Un ensemble de seize registres de type booléen est maintenu. Les valeurs actuellement dans ces registres se présentent sous la forme de seize entrées/sorties Value_1 à Value_16. Par conséquent, il est possible de modifier directement les valeurs contenues dans ces registres en écrivant dans une de ces entrées/sorties.

Il est aussi possible de modifier la valeur d'un de ces registres à l'aide du paramètre Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée In_Index. Cette mise à jour d'un registre peut avoir lieu d'une manière parmi deux : le registre peut être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou il est uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Il existe aussi une sortie qui indique la valeur actuelle d'un des registres. Le registre dont la valeur est indiquée est déterminé par l'entrée Out_Index.

Attributs du bloc fonction

Type : 50A6

Classe :.....SELECT

Tâche par défaut :.....Task_2

Liste raccourcie : .....In_Index, Out_Index, Input, Output

Mémoire nécessaire :.....32 octets

Description des paramètres

Out_Index (OID)

Détermine le registre dont provient la valeur d'**Output**. S'il est fixé à zéro, **Output** affiche également la valeur zéro.

In_Index (IID)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si **Mode** est positionné sur OnTrig. **In_Index** n'a aucun effet si **Mode** est positionné sur DoNot.

Input (IN)

Valeur qui sera copiée dans le registre spécifié par **In_Index**, à condition que Mode soit sur Cont ou OnTrig.

Mode (M)

Détermine l'effet d'**Input** et de **Trigger** sur le registre référencé par **In_Index.**

DoNot	Input et **Trigger** n'ont aucun effet, quelle que soit la valeur d'**In_Index.**
Cont	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque cycle d'exécution de ce bloc fonction.
OnTrig	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque fois que **Trigger** est sur "vrai".

Trigger (TRG)

Lorsque **Mode** est positionné sur OnTrig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **In_Index.** Par exemple, si **Input** est sur On et **In_Index** est égal à 5, lorsque Trigger est activé, **Value_5** affiche également la valeur On, à condition que **Mode** soit sur OnTrig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction lorsque **Mode** est sur Cont ou sur OnTrig. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Value_1 (V1)à Value_16 (V16)

Valeurs des seize registres dans lesquels **Input** peut être entré et **Output** peut être obtenu.

Seul le registre indiqué par **In_Index** est touché par le changement des valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles sauf si l'on écrit directement dedans en positionnant sur 1 la valeur associée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Out_Index	DINT	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
In_Index	DINT	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Input	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Mode	ENUM	DoNo (0)	Oper	Oper	Cf. lise des paramètres	
Trigger	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Value_1 to Value_16	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Output	BOOL	Off (0)	Oper	Block	Sens	Off (0) On (1)

Tableau 18-14 Attributs des paramètres Regist_Bool

BLOC FONCTION REGISTER STR

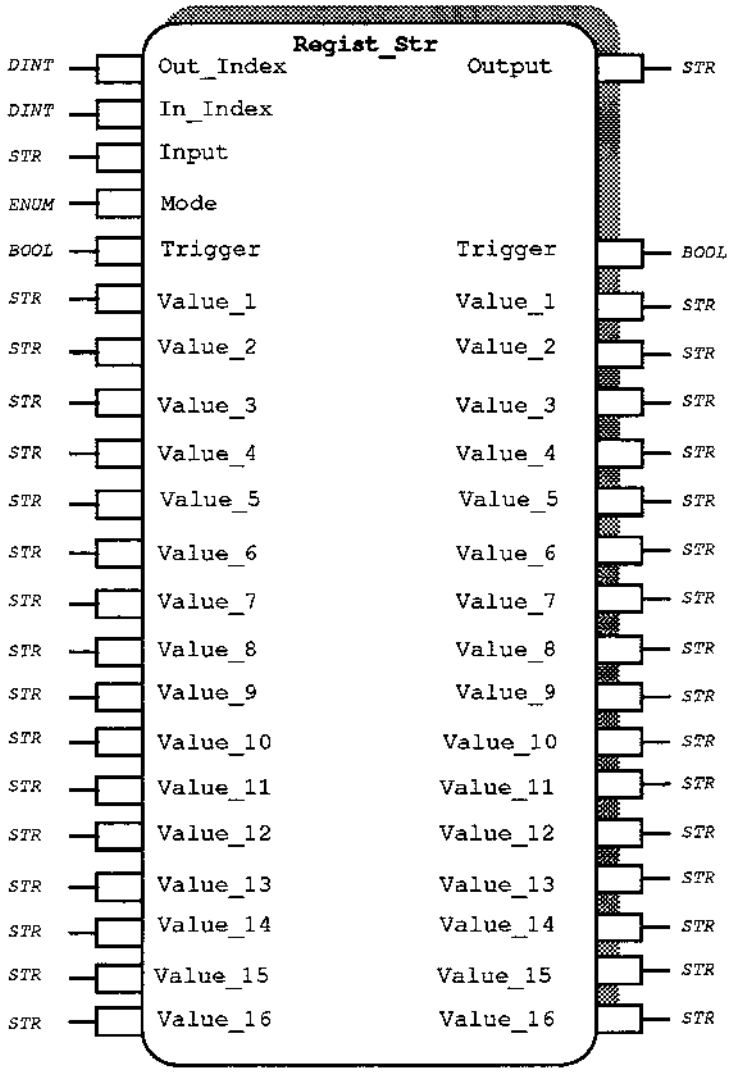


Figure 18-15 Schéma du bloc fonction Regist_Str

Description fonctionnelle

La fonction offerte par ce bloc est pour l'essentiel celle d'une matrice 16 x 1. Il est possible de stocker et de récupérer les nombres en fonction d'un index numérique.

Un ensemble de seize registres de type chaîne de caractères est maintenu. Les valeurs actuellement dans ces registres se présentent sous la forme de seize entrées/sorties Value_1 à Value_16. Par conséquent, il est possible de modifier directement les valeurs contenues dans ces registres en écrivant dans une de ces entrées/sorties.

Il est aussi possible de modifier la valeur d'un de ces registres à l'aide du paramètre Input, le registre dans lequel Input est dirigé étant déterminé par l'entrée In_Index. Cette mise à jour d'un registre peut avoir lieu d'une manière parmi deux : le registre peut être mis à jour en continu, c'est-à-dire à chaque cycle d'exécution du bloc fonction, ou il est uniquement mis à jour lorsque l'entrée Trigger passe de l'état faux à l'état vrai.

Il existe aussi une sortie qui indique la valeur actuelle d'un des registres. Le registre dont la valeur est indiquée est déterminé par l'entrée Out_Index.

Attributs du bloc fonction

Type : 50A8

Classe : .....SELECT

Tâche par défaut : .....Task_2

Liste raccourcie : .....In_Index, Out_Index, Input, Output

Mémoire nécessaire : .....1490 octets

Description des paramètres

Out_Index (OID)

Détermine le registre dont provient la valeur d'**Output**. S'il est fixé à zéro, **Output** affiche également la valeur zéro.

In_Index (IID)

Registre dans lequel doit être copié **Input**, soit immédiatement si **Mode** est sur Cont soit lorsque **Trigger** est sur On si Mode est positionné sur OnTrig. **In_Index** n'a aucun effet si **Mode** est positionné sur DoNot.

Input (IN)

Valeur qui sera copiée dans le registre spécifié par **In_Index**, à condition que **Mode** soit sur Cont ou OnTrig.

Mode (M)

Détermine l'effet d'**Input** et de **Trigger** sur le registre référencé par **In_Index.**

DoNot	**Input** et **Trigger** n'ont aucun effet, quelle que soit la valeur d'**In_Index.**
Cont	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque cycle d'exécution de ce bloc fonction.
OnTrig	le registre indiqué par **In_Index** est mis à jour avec la valeur d'**Input** à chaque fois que **Trigger** est sur "vrai".

Trigger (TRG)

Lorsque **Mode** est positionné sur OnTrig, le fait de positionner cette entrée sur 1 provoque la copie de la valeur actuellement sur **Input** dans le registre dont le nombre figure sur **In_Index.** Par exemple, si **Input** est sur "Bonjour" et **In_Index** est égal à 5, lorsque Trigger est activé, **Value_5** affiche également la valeur "Bonjour", à condition que **Mode** soit sur OnTrig.

Trigger est immédiatement ramené à l'état Off (désactivé) par le bloc fonction lorsque **Mode** est sur Cont ou sur OnTrig. Aucune action externe du programme utilisateur n'est nécessaire pour parvenir à ce résultat.

Value_1 (V1)à Value_16 (V16)

Valeurs des seize registres dans lesquels **Input** peut être entré et **Output** peut être obtenu.

Seul le registre indiqué par **In_Index** est touché par le changement des valeurs d'**Input.** Les autres registres conservent leurs valeurs actuelles sauf si l'on écrit directement dedans en positionnant sur 1 la valeur associée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Out_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
In_Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	+2147483647 -2147483648
Input	**STR**	"	Oper	Oper	Néant	
Mode	**ENUM**	DoNot(0)	Oper	Oper	Cf. liste des paramètres	
Trigger	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Value_1 to Value_16	**STR**	"	Oper	Oper	Néant	
Output	**STR**	"	Oper	Block	Néant	

Tableau 18-15 Attributs des paramètres Regist_Str

Chapitre 19

FILTRES

Edition 1

Vue d'ensemble

Vue d'ensemble

Ce chapitre décrit les blocs fonctions de la classe des FILTERS qui fournissent des filtres de signal.

BLOC FONCTION LAG_1

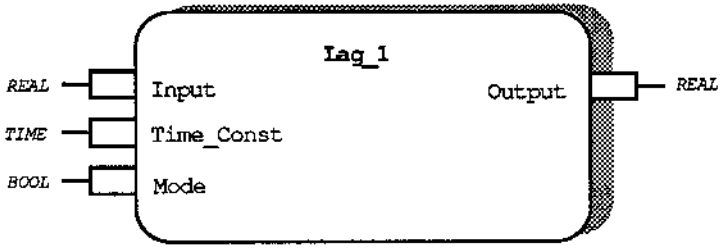


Figure 19-1 Schéma du bloc fonction Lag1

Description fonctionnelle

Le bloc fonction Lag1 (retard) est un filtre passe-bas de premier ordre entre l'entrée (Input) et la sortie (Output), qui peut servir à éliminer le bruit parasite haute fréquence. Le bloc a deux modes de fonctionnement qui peuvent être définis par la paramètre Mode.

Modes de fonctionnement :

Track (0) : en mode Track (suivi), la sortie suit l'entrée sans retard.

Limit (1) : en mode Limit (limite), un retard de premier ordre est introduit entre Output et Input, conformément à la loi de Laplace :

$$\frac{\text{Output}}{\text{Input}} = \frac{1}{1+TC.S}$$

où TC est le paramètre Time_Const. Il émule un filtre RC de premier ordre dont la constante de temps est fixée par Time_Const.

Attributs du bloc fonction

Type : 88 16

Classe : ..... FILTRES

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Input, Mode, Time_Const, Output

Besoins de capacité mémoire : ..... 18 octets

Durée d'exécution : ..... 393 µs

Description des paramètres

Input (IN)

Input est l'entrée du bloc fonction.

Time_Const (TC)

Time_Const est la constante de temps du retard de premier ordre.

Mode (M)

Mode définit le mode de fonctionnement du bloc fonction. Son utilisation a été décrite ci-dessus.

Output (OP)

Output est la sortie du bloc fonction. Si Mode est mis à Track (0), Output est lue directement dans Input. Si Mode est mis à Limit (1), un retard de premier ordre est introduit entre Output et Input.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Input	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	100 000 -100 000
Mode	BOOL	Track (0)	Oper	Super	Détect.	Track (0) Limit (1)
Output	REAL	0,0	Oper	Bloc	Lim. haute Lim. basse	100 000 -100 000
Time_Const	TIME	0	Oper	Oper	Lim. haute Lim. basse	23d_23h _59m_59s 0

Tableau 19-1 Attributs des paramètres de Lag1

Chapitre 20

ALARMES

Edition 2

Présentation

Un ensemble de blocs fonctions d'alarme a été ajouté à la bibliothèque de Blocs fonctions du système PC3000. Ceux-ci permettent l'enregistrement de situations exceptionnelles et d'événements. Les blocs sont prévus pour des sources multiples d'alarme / événement et enregistrent l'heure d'occurrence de l'alarme, la valeur à l'occurrence de l'alarme, ainsi qu'un texte de message optionnel pour chaque entrée. L'Horloge Temps Réel du PC3000 est utilisée pour horodater chaque entrée d'alarme. Le système d'alarme repose principalement sur deux types de blocs.

Capteurs d'alarme - Ce sont des blocs à une entrée booléenne qui enregistrent l'état de l'alarme. La situation réelle qui génère l'alarme est définie en dehors du bloc, par un câblage logiciel du programme utilisateur. On obtient ainsi une adaptabilité maximale pour la définition de l'origine de l'alarme ou de l'événement. A titre d'exemple simple, la situation suivante peut être testée : Loop_1.PV > Max Temp. Si l'alarme change d'état, l'état est transmis à un bloc tampon d'alarme associé.

Tampons d'alarmes - Ce sont des blocs qui mémorisent ou enregistrent des informations relatives à chaque alarme. Deux types sont prévus :

Cur_Alms	Il enregistre les données relatives aux alarmes en cours ou "actives". Ce bloc fournit des informations concernant l'état (STATE) d'une alarme.
History	Il enregistre les alarmes historiques, c'est-à-dire acquittées, pour permettre une analyse différée ou une analyse au redémarrage après défaillance.

Quatre états sont possibles pour une entrée d'alarme dans le tampon d'alarmes en cours.

Etat de l'alarme	Description
Active, non acquittée	Le capteur d'alarme a détecté une situation d'alarme. L'interface opérateur / superviseur n'a pas réagi.
Active, acquittée	Le capteur d'alarme a détecté une situation d'alarme. L'interface opérateur / superviseur a acquitté l'alarme, mais la situation est toujours présente.
Non active, non acquittée	Le capteur d'alarme a détecté une situation d'alarme. L'interface opérateur / superviseur n'a pas acquitté l'alarme, la situation d'alarme n'est plus présente (alarme "fugitive" ou non saisie).
Non active, acquittée	Le capteur d'alarme a détecté une situation d'alarme. L'interface opérateur / superviseur a acquitté l'alarme, la situation d'alarme a été supprimée (CLEARED) et devient une alarme historique dans le tampon d'alarmes historiques.

Tableau 20-1 Etats d'entrée d'alarme

Les blocs Sensor disposent de deux paramètres d'entrée pouvant servir à identifier la source de l'alarme / événement. Ces paramètres définissent le type (Type) et la zone (Area). Ces paramètres peuvent être utilisés dans les cas suivants :

1. Lorsque des alarmes doivent être regroupées, par exemple pour toutes les zones du cylindre d'une extrudeuse, le paramètre Area peut servir à classer toute alarme provenant de cette partie de machine. Si des alarmes différentes peuvent avoir lieu, par exemple une alarme de dépassement de température et une alarme d'écart de consigne, une identification complémentaire est possible avec le paramètre Type. L'alarme de dépassement de température peut être décrite comme étant de Type 1 et l'alarme d'écart de consigne, comme étant de Type 2.

2. Si un regroupement d'alarmes n'est pas requis, les paramètres Type et Area peuvent être combinés pour créer un numéro d'alarme spécifique à chaque alarme.

   N.B. : Deux alarmes ne peuvent pas avoir les mêmes paramètres Type et Area car il serait impossible de déterminer la provenance de l'alarme.

En plus du bloc Sensor, un bloc Detector permet une stratégie standard de détection d'alarme, pour les alarmes absolues, les alarmes d'écart de consigne ou de vitesse de variation. Le bloc surveille une variable d'entrée, par exemple PV et la compare aux entrées consigne et limites. Les limites détectées sont : alarme haute, alarme basse, alarme écart supérieur, alarme écart inférieur, alarme bande et alarme vitesse de variation.

Sur la Fig. 20-1, l'entrée analogique (entrée soit de surveillance soit de commande) a été câblée sur un bloc fonction Détecteur. Celui-ci compare la valeur de l'entrée analogique au seuil d'alarme du 650 et si elle dépasse, la sortie Hi_Alarm est positionnée. Cet état est perçu par le capteur d'alarme relié au détecteur et horodaté. Le paramètre de configuration BufId du capteur associe le capteur au tampon d'alarmes en cours qui partage le même ID (identificateur). Le bloc Cur_Alms comporte également une entrée configuration, LogId, qui permet d'associer le tampon History dans lequel entreront les alarmes lorsqu'elles auront été remises à zéro.

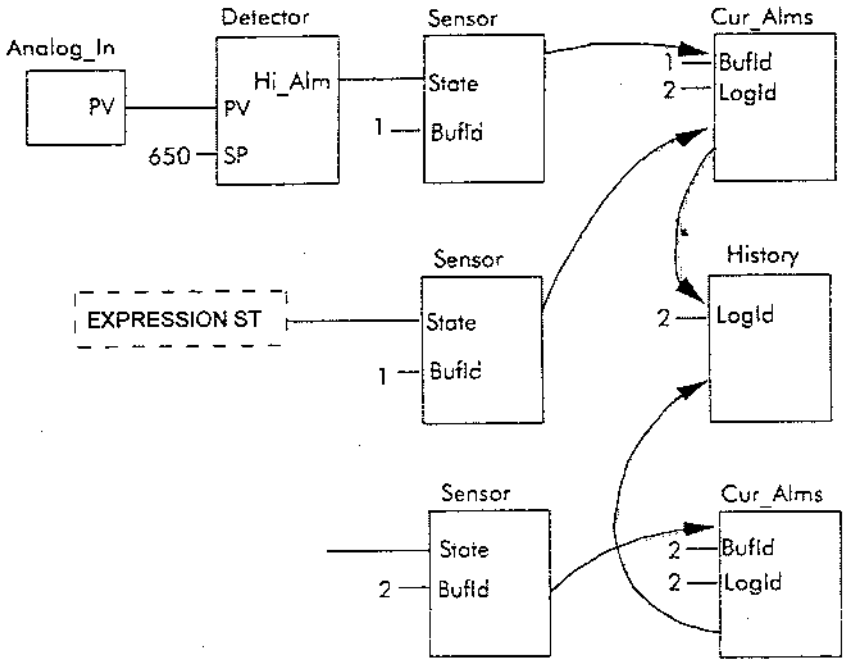


Fig. 20-1 Présentation du système d'alarme

Le schéma montre également comment un bloc capteur d'alarme peut être utilisé en l'associant à une instruction de câblage de texte structuré pour la mise en oeuvre de stratégies d'alarme non standard, par exemple :

```
Sensor.State:= tc1.PV >= 250 and BatchNo.Val= 10 AND
                        Proctime.Elapsed_Time > T#5h25m;
```

Le schéma montre finalement un troisième capteur avec un BufId différent. Ceci permet à des alarmes provenant, le cas échéant, d'un autre procédé, d'être associées à un tampon d'alarme distinct. Dans ce cas, le tampon est identifié par un BufId de 2.

BLOC FONCTION CUR_ALMS

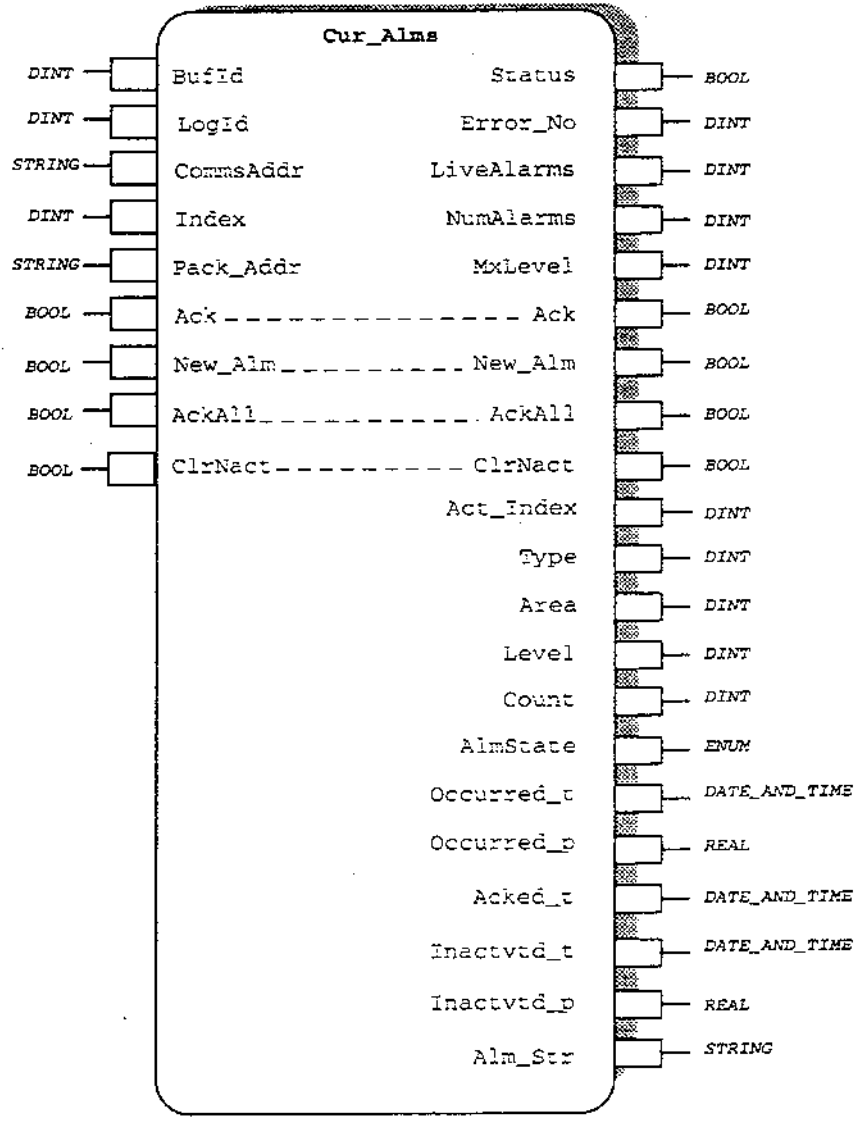


Fig. 20-2 Schéma du bloc fonction Cur_Alms

Description fonctionnelle

Le bloc fonction Cur_Alms contient un tableau des alarmes en cours non acquittées ou réelles et permet l'accès aux éléments de ce tableau au moyen d'une entrée indexée. Il est prévu pour être utilisé avec des blocs fonctions capteurs d'alarme, pour réaliser un système d'enregistrement d'alarmes et d'événements définis par l'utilisateur au niveau du régulateur PC3000.

Le bloc comporte également deux variables esclaves internes de communication. L'une d'elle fournit une interface Eurotherm Bisync (lorsqu'elle est associée à un module de communication esclave) et permet d'accéder à l'ensemble des données que comporte le bloc fonction. Elle est, en général, utilisée avec des ordinateurs de surveillance tel que l'ESP. L'autre variable esclave fournit une interface MODBUS/JBUS et donne accès à une chaîne condensée ne contenant que les informations relatives au type d'alarme, à la zone et à l'état. Cette seconde variable est prévue pour fournir un accès simple aux données essentielles et principalement pour l'utilisation des terminaux 9" et 12".

Les alarmes peuvent être acquittées en utilisant l'un ou l'autre de ces deux paramètres esclaves ou bien les paramètres du bloc fonction lui-même. Ceci permet d'utiliser des systèmes de supervision, des tableaux Xycom et l'afficheur EUROTHERM, en tant qu'interfaces du système de traitement des alarmes.

Le tampon des alarmes en cours est associé à plusieurs capteurs d'alarme (blocs fonctions qui fournissent les données relatives aux situations d'alarme) par l'intermédiaire d'un identificateur de tampon. Ceci permet l'utilisation de plusieurs tampons d'alarmes réelles dans le PC3000. Des alarmes provenant de différentes parties de la machine ou du procédé peuvent être associées à leur propre tampon d'alarmes en cours.

Le tampon d'alarmes en cours comporte également un moyen d'interfaçage à un tampon d'alarmes historiques (History). Ce dernier mémorise les alarmes acquittées. Un tampon d'alarmes historiques est associé à un tampon d'alarme en cours par un identificateur de journal.

Attributs du bloc fonction

Type : 6670

Classe : ..... ALARMS

Tâche par défaut : ..... Task_1

Liste résumée : ..... BufID, LogID, NumAlarms, MxLevel

Mémoire nécessaire : ..... 6 888 octets

Description des paramètres

Le bloc fonction Cur_Alms dispose des paramètres suivants :

BufId (BID)

C'est l'identificateur (ID) des tampons d'alarmes. LTID doit être unique. C'est la référence permettant aux capteurs d'être reliés ou associés au bloc fonction Cur_Alms.

LogId (LID)

C'est le paramètre qui est utilisé pour associer un bloc fonction History au tampon d'alarme en cours. Les alarmes acquittées sont transférées au bloc fonction History avec l'identificateur de journal (LogID) qui établit la concordance avec cette valeur.

CommsAddr (ADD)

C'est l'adresse de communication pour le paramètre esclave composite ou à éléments multiples. Cette adresse donne accès, via le protocole Eurotherm Bisync, aux informations contenues dans le tableau d'alarmes. L'accès aux données peut se faire sous la forme d'un paramètre unique à plusieurs éléments ou de plusieurs éléments distincts. La lecture des données à l'adresse esclave renvoie le paramètre unique à plusieurs éléments. L'accès à des adresses successives ne renvoie qu'un élément. Il n'est pas possible de spécifier le format des données pour les éléments individuels, le format par défaut étant imposé.

Les données que contient le paramètre esclave se présentent sous la forme suivante :

Décalage	Paramètre	Type
0	Paramètre composé	
1	Index	Entier
2	Type	Entier
3	Zone	Entier
4	Etat	Entier
5	Niveau	Entier
6	Comptage	Entier
7	Moment de l'occurrence	Entier
8	Position à l'occurrence	Réel
9	Moment de l'acquittement	Entier
10	Moment de la désactivation	Entier
11	Position à la désactivation	Réel

Le décalage se rapporte au décalage à partir de l'adresse de départ. Par exemple, si l'adresse esclave indiquée est "EBA90" et si le bloc esclave Bisync associé a un paramètre GID égal à "0", les adresses applicables sont les suivantes :

Adresse Eurotherm Bisync	Paramètre	Type
0	Paramètre composé	
1	Index	Entier
2	Type	Entier
3	Zone	Entier
4	Etat	Entier
5	Niveau	Entier
6	Comptage	Entier
7	Moment de l'occurrence	Entier
8	Position à l'occurrence	Réel
9	Moment de l'acquittement	Entier
10	Moment de la désactivation	Entier
11	Position à la désactivation	Réel

Pour plus de détails sur l'adressage des alarmes, se reporter au chapitre traitant de la communication des alarmes.

Index (I)

Ce paramètre offre un moyen d'accès aux données stockées dans le tableau d'alarmes. Les sorties du bloc fonction affichent l'entrée d'alarme à la position indexée, si elle existe. Par exemple, si la dixième alarme doit être choisie, l'indice sera mis à 9 et les sorties du bloc fonction indiqueront les données relatives à cette alarme.

Pack_Addr (PA)

C'est l'adresse de communication du paramètre esclave condensé. Ce paramètre est prévu pour émuler plusieurs applications utilisant les terminaux Xycom de 9" et 12" pour l'affichage et l'acquittement des alarmes. Si ces terminaux sont utilisés, l'adresse doit être compatible JBus, c'est-à-dire "JB****".

Pour plus de détails sur l'adressage des alarmes, se reporter au chapitre traitant de la communication des alarmes.

Ack (ACK)

C'est une entrée booléenne permettant à l'alarme actuellement sélectionnée d'être acquittée. Elle est positionnée par l'utilisateur et automatiquement remise à zéro par le bloc.

New_Alm (NAL)

C'est un paramètre booléen d'état qui signale l'occurrence d'une nouvelle alarme. Il est positionné par le bloc et doit être remis à zéro par l'utilisateur. Ce paramètre peut servir à forcer l'affichage d'un écran d'alarme sur le système ou tableau de supervision, lorsqu'une nouvelle alarme est reçue.

Status (ST)

Il indique l'état du bloc fonction. En fonctionnement normal, le paramètre indique l'état GO. Toutefois, si une erreur a été détectée, le paramètre bascule à l'état NOGO et la raison de l'erreur est indiquée par le paramètre Error_No.

Error_No (ERR)

Il indique le numéro d'erreur du bloc fonction. Si une erreur est détectée lorsque le programme PC3000 tourne, l'erreur est signalée comme indiqué dans le tableau ci-dessous et l'état du bloc devient NOGO.

N° d'erreur	Erreur	Cause et remède
105	Trop de tampons	Il y a plus de 20 tampons (blocs fonctions Cur_Alms ET History). Un tampon signalant cette erreur ne sera pas opérationnel et le nombre de tampons doit être réduit.
106	ID du tampon hors échelle	LogID est inférieur à zéro ou supérieur à 9. Changer l'ID du tampon et l'ID des capteurs associés à ce tampon.
107	ID en double	Un tampon existe déjà avec cet ID. Changer l'ID du tampon et l'ID des capteurs associés à ce tampon.

Tableau 20-2 Erreurs pouvant être détectées et signalées

Live Alarms (LA)

Il indique le nombre total d'alarmes non acquittées présentes dans le tampon d'alarmes.

NumAlarms (NA)

Ce paramètre indique le nombre total d'alarmes présentes dans le tampon.

MxLevel (MXL)

Il indique le niveau d'alarme le plus élevé dans le tampon, c'est-à-dire l'alarme ayant la priorité la plus haute.

Act_Index (AI)

Il est égal à Index si le paramètre Index est dans la plage suivante :

```
0 < Index <= NumLogged
```

Il est égal à zéro pour les valeurs Index sortant de cette plage.

Type (TYP)

Il indique le type d'alarme à la position du journal spécifiée par Act_Index.

Area (A)

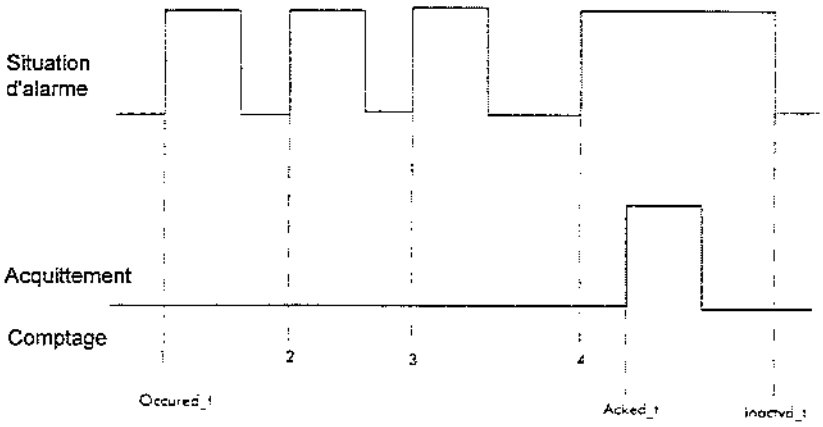
Ce paramètre se réfère à la zone dont provient l'entrée d'alarme actuellement spécifiée par Act_Index.

Level (L)

Niveau d'alarme à la position spécifiée par Act_Index. Level est utilisé pour indiquer la priorité de l'alarme. Il permet de rendre prioritaires différentes alarmes et de gérer en conséquence des messages d'avertissement et des actions correctrices. Le niveau d'alarme le plus élevé dans le tampon est indiqué par le paramètre MxLevel.

Count (C)

Nombre d'activations de l'alarme de la position spécifiée par Act_Index. Une alarme peut être enregistrée plusieurs fois sans acquittement de l'alarme (en particulier lors du contrôle de mise en service). Plutôt que de créer une entrée distincte dans le tampon d'alarmes en cours pour chaque occurrence, l'alarme est enregistrée à sa première activation et le nombre de fois où elle est ensuite passée de l'état inactivé à l'état activé est compté.



AlmState (ALS)

Ce paramètre indique l'état de l'alarme à la position spécifiée par Act_Index.

Valeur	Etat	Signification
0	Vide	Pas d'alarme à cette position. C'est l'état d'une alarme qui a été acquittée et qui n'est plus active, c'est-à-dire une alarme morte.
1	NactNak	Non active, non acquittée. Alarme qui a disparu avant d'être acquittée.
2	ActAck	Alarme actuellement active, mais qui a été acquittée.
3	ActNak	Alarme actuellement active et qui n'a pas encore été acquittée.

Tableau 20-3 Etats pouvant être affichés

Occurred_t (OCT)

Moment où l'alarme actuellement spécifiée par Act_Index est devenue active.

Occurred_p (OCP)

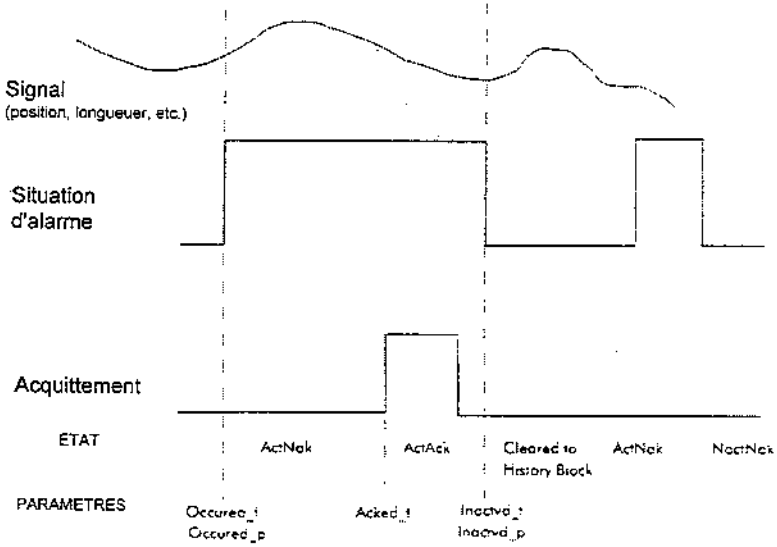
Valeur indiquant la position, par exemple la longueur à laquelle l'alarme spécifiée par Act_Index est devenue active.

Acked_t (AKT)

Moment où l'alarme (spécifiée par Act_Index) a été désactivée.

Inactvd_p (IAP)

Valeur indiquant la position, par exemple la longueur, pour laquelle l'alarme spécifiée par Act_Index est devenue inactive. Le schéma suivant résume la fonction de ces paramètres :



Alm_Str (STR)

Valeur de la chaîne d'alarme pour l'alarme spécifiée par Act_Index. La chaîne d'alarme est utilisée normalement pour mémoriser un texte de message associé à cette alarme.

Ack_All (AA)

Entrée booléenne pouvant être utilisée pour acquitter toutes les alarmes de ce tampon. Prévue essentiellement pour le contrôle de mise en service lorsque plusieurs alarmes peuvent être actives.

Clr_Nact (CCN)

Entrée booléenne qui permet d'acquitter (et donc de supprimer) toutes les alarmes inactives de ce tampon.

Utilisation générale

Examen des alarmes

Lorsqu'il y a plusieurs alarmes dans le tampon d'alarmes, elles peuvent être examinées une par une en paramétrant l'entrée Index. Si cette entrée est mise sur un numéro d'alarme existante, tous les paramètres correspondants sont affichés sur les sorties du bloc.

Noter que le moment de l'acquittement, la position inactive et le moment de désactivation sont à zéro tant que l'action appropriée (acquittement ou arrêt de l'alarme) n'a pas eu lieu.

Noter que l'alarme 1 est toujours l'alarme la plus récente, à moins que le module d'alarmes ne soit plein (128 alarmes). Si le module d'alarmes est plein, une nouvelle alarme n'est pas enregistrée, à moins que son niveau ne soit supérieur au niveau le plus bas présent du tampon.

Exemple : l'alarme 20 du tampon est de niveau 3, toutes les autres alarmes sont de niveau 4 et le tampon est plein. Si une autre alarme de niveau 3 apparaît, elle ne sera pas enregistrée. Si une alarme de niveau 4 apparaît, l'alarme 20 est supprimée et la nouvelle alarme entre à la position 1.

Acquittement des alarmes

Pour acquitter l'alarme actuellement affichée, l'entrée Ack est positionnée. Cette entrée est supprimée par le bloc.

Toutes les alarmes du tampon peuvent être acquittées en positionnant le paramètre Ack_All. La remise à zéro se fait par le bloc.

Pour enlever du tampon toutes les alarmes inactives, positionner l'entrée Clr_Nact. La remise à zéro se fait par le bloc.

Détection d'alarmes nouvelles

Une nouvelle alarme peut être détectée en surveillant l'entrée/sortie New_Alm. Elle est positionnée par le bloc et remise à zéro par le programme utilisateur.

Extrait de ST montrant l'accès au bloc d'alarme par le tableau.

```
PROGRAMME PANEL1 (*16-Nov-1992-14:43:41*)

VAR

   (*SYSTEM*)
   PocsSTATE:PcsSTATE ;
   Tsk_10ms:Task          (Interval  :=T#10ms
                          Priority    :=0) ;
   Tsk100ms:Task          (Priority   :=1) ;
   Messages:Messages ;
   RT_Clock:RT_Clocks ;

   (*COMMS*)
   panel   :Euro_Panel  (Port   :='0A') ;

   (*USER_VAR*)
   level  :Boolean   ;
   alm    :Integer
```

```
(*SLAVE_VARS*)
bool1          :Slave_Bool     (Address : ='EPbool1') ;
bool2          :Slave_Bool     (Address : ='EPbool2') ;
int1           :Slave_Int      (Address : ='EPint1') ;
int2           :Slave_Int      (Address : ='EPint2') ;
time1          :Slave_Time     (Address : ='EPtime1') ;
time2          :Slave_Time     (Address : ='EPtime2') ;
str1           :Slave_Str      (Address : ='EPstr1') ;
str2           :Slave_Str      (Address : ='EPstr2') ;

(*STEPS*)
MAIN           :Macro  ;
PANEL          :Macro  ;
Init           :Step           ;
Display        :Step           ;
End            :Step           ;

(*ALARMS*)
buff   :Cur_Alms       (BufId          :=1,
                                       LogId   :=1,
                                       CommsAddr:='EBr00) ;
high   :Sensor16       (Type           :=1,
                       AreaStart       :=1,
                       Alm_Str         :='Process Value Over
Range Loop') ;
low    :Sensor16       (Type           :=2,
                       AreaStart       :=1,
                       BufId           :=1,
                       Alm_Str         :='Process Value Under
Range Loop') ;

(*Internal Variables*)
   (*SYSTEM*)

   (*COMMS*)

   (*USER_VAR*)

   (*SLAVE_VARS*)
   real1_Value                 :REAL ;
   real2_Value                 :REAL ;
   real3_Value                 :REAL ;
   int1_Value                  :DINT ;
   time1_Value                 :TIME ;
   time2_Value                 :TIME ;
   str1_Value                  :STRING ;
   str2_Value                  :STRING ;
```

```
   (*STEPS*)
   (*LOADS*)

   (*ALARMS*)
   buff_Index                :DINT ;

END-VAR

(*function block instantiations*)
INITIAL_STEP EXECUTE_100
buff          (Index          :=buff_Index) ;
str2          (Value          :=str2_Value) ;
str1          (Value          :=str1_Value) ;
bool2         ( ) ;
bool1         ( ) ;
alm           ( ) ;
low           (AlmState1      :=load1.Main_PV<-50,
              AlmState2       :=load2.Main_PV<-50,
              AlmState3       :=load3.Main_PV<-50,
              AlmState4       :=load4.Main_PV<-50)
high          (AlmState1      :=load1.Main_PV>30,
              AlmState2       :=load2.Main_PV>30,
              AlmState3       :=load3.Main_PV>30,
              AlmState4       :=load4.Main_PV>50)
panel         ( ) ;
RT_Clock      ( ) ;
Messages      ( ) ;
Tsk100ms      ( ) ;
PcsSTATE      ( ) ;
END_STEP

INITIAL_STEP EXECUTE_10
level         ( ) ;
Tsk10ms       ( ) ;
END_STEP

      ......................

(*MACRO:ALARMS*)
STEP ALARMS:
```

```
(*

| s | Init |
        T1
        |

| c | Display |

        !

| e | End |

*)

(*SINGLE SHOT*)
INITIAL STEP :
   (*Set up the display and parse the page*)
   panel_Key_Pressed :=22 (*No_Key*) ;
   buff_Index                :=1
   (Display the contents of str2 on the top line*)
   (*with enumerated output of alarm states*)
   panel_Format_A
      :='str2:32C,int1(_,NactNak,ActAck,ActNack) ' ;
   (*Place "Acknowledge" above Fa, "Exit" above F3*)
   panel_Format_B            :='@0:1,"Acknowledge",@30,
   "Exit" ' ;
   panel_Format_C            :=" ;
   panel_Change_Page :=3 (Nxt_Pge*) ;
END STEP

TRANSITION
   FROM INIT
   TO DISPLAY
:=1 ; (*NULL transition - default TRUE*)
END_TRANSITION
(*CONTINUOUS*)
STEP Display
   (*Clockwise for next alarm*)
   IF panel.Key_Pressed=10 (*Clkwise*) THEN
   buff_Index                :=buff.Index + 1 ;
   panel_Key_Pressed :=22 (*No_Key*) ;
   END_IF ;
   (*Anticlockwise for previous alarm*)
   IF panel.Key_Pressed=11 (*AClkwse*) THEN
   buff-Index                          :=buff.Index - 1 ;
   panel-Key_Pressed :=22 (*Nc_Key*) ;
   END IF ;
```

```
   (*F1 for acknowledge*)
   IF panel.key_Pressed      =14 (*F1*) THEN
   buff_Ack                  :=1 (*On*) ;
   panel_Key_Pressed :=22 (*No_Key*) ;
   END_IF ;
   (*Display "No Alarms" if there are no alarms !*)
   IF buff.NumAlarms = 0 THEN
   str2_Value                :='No Alarms' ;
   int1_Value                :=0 ;
ELSE
   (*Limit Index to between 1 and buff.NumAlarms*)
   IF buff.Index = 0 THEN
   buff_Index                :=1 ;
   ELSIF buff.Index>buff.NumAlarms ;
   buff_Index                :=buff.NumAlarms ;
   END_IF ;
   (*Display alarm string (set on sensor) and current
   state*)
   str2_Value                :=CONCAT (IN1 := BUFF.ALM_STR
                             ,IN22:= DINT_TO_STRING
   (IN:=buff.Area)) ;
   int1_Value                :=buff.State ;
   END-IF ;
END-STEP

TRANSITION
   FROM Display
   To End
:=
      panel.Key_Pressed = 16 (F3*) ;
END_TRANSITION

(*SINGLE SHOT*)
STEP End:
END_STEP

END_STEP (*ALARMS*)

TRANSITION
   FROM ALARMS (*MACRO*)
   TO TOP
:=1 ; (*NULL transition -default TRUE*)
END_TRANSITION

END_STEP (*PANEL*)
END_STEP (*MAIN*)
END_PROGRAM
```

Communication d'alarmes

Des détails complémentaires sont donnés ci-après concernant l'accès au contenu du tampon d'alarmes en cours, via les interfaces de communication.

Interface Eurotherm Bisync

L'accès aux données se fait au moyen du paramètre esclave, en paramétrant l'adresse "CommsAdd". Les champs du paramètre composé décrit précédemment sont dans le même ordre que les entrées recensées dans les tableaux.

Exemple : pour lire l'entrée 3 de la liste d'alarmes, le paramètre "Index" est mis à 3 :

EOT0000STXA90RS>3ETX?

Le paramètre composé peut alors être relu :

EOT0000A90ENQ

La réponse est :

STXA90RS>3us>typeus>areaus>stateus>level

us>countus>otimeus@oproduced

us>atimeus>itimeus>@iproduced

Pour acquitter l'alarme TYPE 1, AREA 4 :

EOT0000STXA90RS US>1us>4ETX?

Accès alarme condensée (MODBUS/JBus ou Eurotherm Bisync)

La chaîne d'alarme condensée se présente sous forme d'une succession de registres dans le cas de JBus ou d'une chaîne de caractères dans le cas d'Eurotherm Bisync. Chaque entrée d'alarme comprend 4 octets, de telle sorte que 32 alarmes peuvent être lues avec un accès à 128 octets. La longueur maximale d'une chaîne de caractères est de 128 octets.

Chaque entrée se présente sous la forme suivante :

Active (255 pour active, 0 pour inactive)

Type

Acquittement (128 pour acquittée, 0 pour non acquittée)

Zone

Le type d'alarme sert d'indice dans une matrice de chaînes. Par exemple, le type 1 peut être une alarme de dépassement de température, le type 2 une alarme basse, le type 3 un écart etc. La zone est alors annexée à cette chaîne pour constituer le message d'alarme générale, par exemple "type 2 zone 3" peut être lu comme étant une "alarme basse en zone 3".

L'acquittement des alarmes se place dans les 2 premiers octets de la chaîne ou, dans le cas d'une communication MODBUS/JBus, dans le premier registre. Le premier octet est le type, le second est la zone.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
BuiId	**DINT**	1	Oper	Oper	Limite haute Limite basse	255 1
LogId	**DINT**	1	Oper	Oper	Limite haute Limite basse	255 1
CommsAddr	**STRING**	"	Oper	Oper	13 caract. maxi	2
Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 1
Ack	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Pack_Addr	**STRING**	"	Oper	Oper	13 caract. maxi	
New_Alm	**BOOL**	No (0)	Oper	Oper	Sens	No (0) New_Alm (1)
Ack-All	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) Ack_All(1)
Clr_Nact	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) Clr_Nact (1)
Status	**BOOL**	NOGO (0)	Oper	Oper	Sens	NoGo (0) Go (1)
Erroe_No	**DINT**	0	Oper	Block	Limite haute Limite basse	255 0
LiveAlarms	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
NumAlarms	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
MxLevel	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
ActIndex	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Type	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0

Tableau 20-4 Attributs des paramètres Cur_Alms

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Area	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Level	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Count	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
AlmState	**ENUM**	Empty (0)	Oper	Oper	Sens	Empty (0) NactNak (1) ActAck (2) ActNak (3)
Occured_t	**DATE_AND_TIME**		Oper	Oper	Limite haute Limite basse	
Occured_p	**REAL**	0	Oper	Oper	Limite haute Limite basse	
Acked_t	**DATE_AND_TIME**		Oper	Oper	Limite haute Limite basse	
Inactvtd_t	**DATE_AND_TIME**		Oper	Oper	Limite haute Limite basse	
Inactvtd_p	**REAL**		Oper	Oper	Limite haute Limite basse	
Alm_Str	**STRING**	0	Oper	Oper	Longueur maxi.	255 caractères

Tableau 20-4 Attributs des paramètres Cur_Alms (suite)

BLOC FONCTION HISTORY

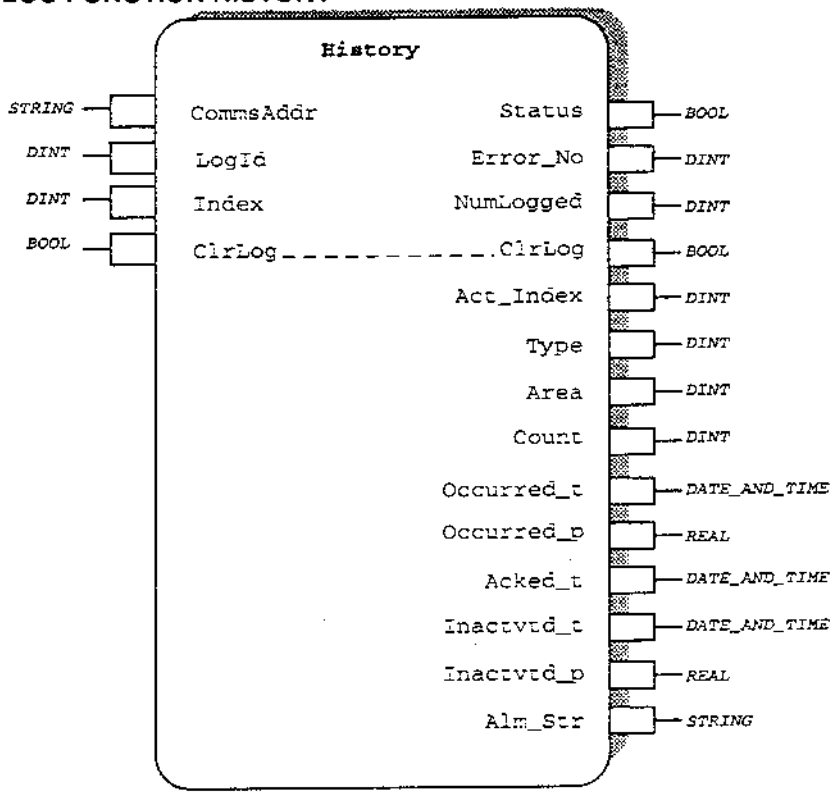


Fig. 20-3 Schéma du bloc fonction History

Description fonctionnelle

Le bloc fonction History contient un tableau des 128 dernières alarmes remises à zéro et permet l'accès aux éléments de ce tableau au moyen d'une entrée indexée. Il est prévu pour être utilisé avec des blocs fonctions Cur_Alms, pour réaliser un système d'enregistrement d'alarmes historiques et d'événements définis par l'utilisateur au niveau du régulateur PC3000.

Le bloc comporte également une variable esclave interne de communication. Celle-ci fournit une interface Eurotherm Bisync et donne accès à un paramètre composé ou à éléments multiples contenant toutes les informations d'alarme. L'accès via MODBUS/JBus n'est pas possible avec ce bloc (voir "Tampon d'alarmes en cours").

Le tampon d'alarmes historiques est associé à un ou plusieurs blocs fonctions Cur_Alms (blocs fonctions avec données mémorisées associées à des situations d'alarmes réelles ou en cours) par l'intermédiaire d'un identificateur de journal. Ceci permet l'utilisation de plusieurs tampons d'alarmes historiques dans le système PC3000. Il permet également à des tampons d'alarmes en cours multiples de stocker des alarmes historiques dans le même journal historique. Dans ce cas, tous les tampons d'alarmes en cours seront paramétrés avec le même identificateur de journal.

Attributs du bloc fonction

Type : 6671

Classe : ..... ALARMS

Tâche par défaut : ..... Task_1

Liste résumée : ..... LogID, Index, NumLogged, Status

Mémoire nécessaire : ..... 5 348 octets

Description des paramètres

LogID (LID)

C'est l'identificateur (ID) de journal d'alarmes. Il sert à relier ou à associer plusieurs gestionnaires d'alarmes à un journal d'alarmes. Des gestionnaires d'alarmes multiples peuvent enregistrer leurs alarmes dans un seul journal d'alarmes en ayant le même LogID. La valeur de LogID doit nécessairement être dans la plage de 0 à 9.

Index (I)

C'est un indice du tableau des alarmes enregistrées. Il peut prendre toute valeur dans la plage de 0 à 127. Toutefois, le journal d'alarmes ne contient que des données d'alarmes historiques valides dans la plage de 0 à NumLogged et la valeur du paramètre Index doit être limitée à cette plage.

CommsAddr (ADD)

C'est l'adresse de communication d'un paramètre esclave composé ou à éléments multiples. Cette adresse donne accès, via le protocole Eurotherm Bisync, aux informations contenues dans le tableau d'alarmes. L'accès aux données peut se faire sous la forme d'un paramètre unique à plusieurs éléments ou de plusieurs éléments distincts. La lecture des données à l'adresse esclave renvoie le paramètre unique à plusieurs éléments. L'accès à des adresses successives ne renvoie qu'un élément. Il n'est pas possible de spécifier le format des données pour les éléments individuels, le format par défaut étant imposé.

Les données que contient le paramètre esclave se présentent sous la forme suivante

Décalage	Paramètre	Type
0	Paramètre composé	
1	Index	Entier
2	Type	Entier
3	Zone	Entier
4	Etat	Entier
5	Niveau	Entier
6	Comptage	Entier
7	Moment de l'occurrence	Entier
8	Position à l'occurrence	Réel
9	Moment de l'acquittement	Entier
10	Moment de la désactivation	Entier
11	Position à la désactivation	Réel

Le décalage se rapporte au décalage à partir de l'adresse de départ. Par exemple, si l'adresse esclave indiquée est "EBA90" et si le bloc esclave Bisync associé a un GID égal à "0", les adresses applicables sont les suivantes :

Adresse Eurotherm Bisync	Paramètre	Type
00 à 90	Paramètre composé	
00 à 91	Index	Entier
00 à 92	Type	Entier
00 à 93	Zone	Entier
00 à 94	Etat	Entier
00 à 95	Niveau	Entier
00 à 96	Comptage	Entier
00 à 97	Moment de l'occurrence	Entier
00 à 98	Position à l'occurrence	Réel
00 à 99	Moment de l'acquittement	Entier
00 à A9;	Moment de la désactivation	Entier
00 à A9;	Position à la désactivation	Réel

Pour plus de détails sur l'adressage des alarmes, se reporter au chapitre traitant de la communication des alarmes.

ClrLog (CLR)

C'est une entrée/sortie servant à effacer le journal d'alarmes. Elle est positionnée par l'utilisateur et remise à zéro par le bloc.

Status (ST)

Il indique l'état du bloc fonction et peut prendre l'état GO en fonctionnement normal et NOGO en cas d'erreur. L'erreur est spécifiée par le paramètre Error_No.

Error_No (ERR)

Il indique le numéro d'erreur du bloc fonction. Si une erreur est détectée lorsque le programme PC3000 tourne, l'erreur est signalée comme indiqué dans le tableau ci-dessous et l'état du bloc passe à NOGO.

Les erreurs suivantes sont détectées et indiquées :

N° d'erreur	Erreur	Cause et remède
105	Trop de tampons	Il y a plus de 20 tampons (blocs fonctions Cur_Alms ET History). Un tampon signalant cette erreur ne sera pas opérationnel et le nombre de tampons doit être réduit.
106	ID du tampon hors échelle	LogID est inférieur à zéro ou supérieur à 9. Changer l'ID du tampon et l'ID des capteurs associés à ce tampon.
107	ID en double	Un tampon existe déjà avec cet ID. Changer l'ID du tampon et l'ID des capteurs associés à ce tampon.

NumLogged (NL)

Indique le nombre total d'alarmes dans le tampon.

Act_Index (AI)

Il est égal à Index si le paramètre Index est dans la plage suivante :

$$0 < \text{Index} <= \text{NumLogged}$$

Egal à zéro pour les valeurs Index sortant de cette plage.

Type (TYP)

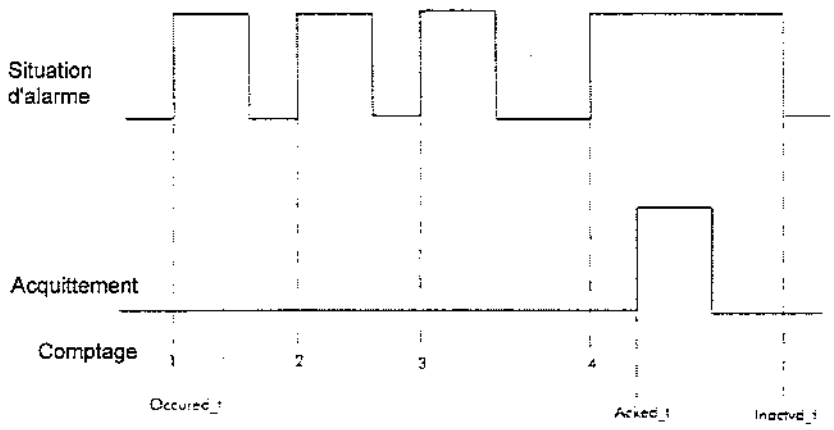
Indique le type d'alarme à la position de journal spécifiée par Act_Index.

Area (A)

Ce paramètre se rapporte à la zone dont provient l'entrée d'alarme actuellement spécifiée par Act_Index.

Count (C)

Nombre d'activations de l'alarme de la position spécifiée par Act_Index.



Occured_t (OCT)

Moment où l'alarme actuellement spécifiée par Act_Index est devenue active.

Occured_p (OCP)

Valeur indiquant la position, par exemple la longueur à laquelle l'alarme spécifiée par Act_Index est devenue active.

Acked_t (AKT)

Moment où l'alarme spécifiée par Act_Index a été acquittée.

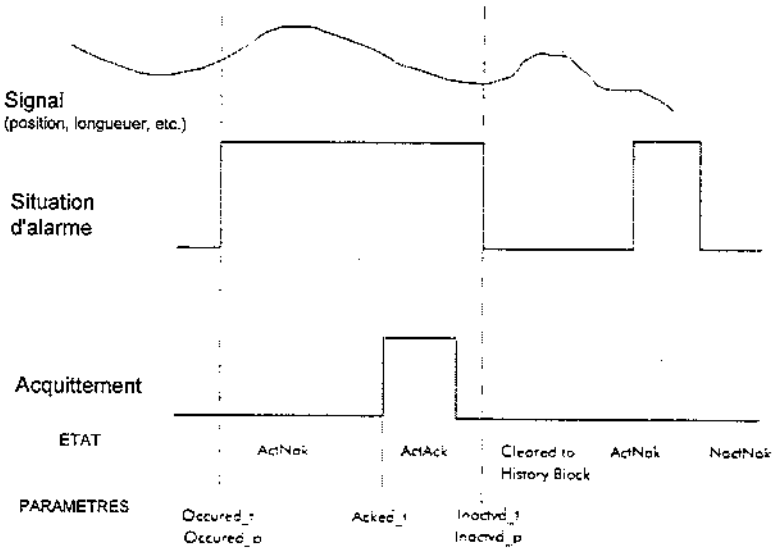
Inactvtd_t (IAT)

Moment où l'alarme spécifiée par Act_Index a été désactivée.

Inactvtd_P (IAP)

Valeur indiquant la position, par exemple la longueur, pour laquelle l'alarme spécifiée par Act_Index est devenue inactive.

Le schéma suivant résume la fonction de ces paramètres :



Alm_Str (STR)

Valeur actuelle de la chaîne d'alarme pour l'alarme spécifiée par Act_Index. La chaîne d'alarme est utilisée normalement pour mémoriser un texte de message associé à cette alarme.

Communication d'alarmes

Des détails complémentaires sont donnés ci-après concernant l'accès au contenu du tampon d'alarmes historiques, via les interfaces de communication.

Interface Eurotherm Bisync

L'accès aux données se fait au moyen du paramètre esclave, en paramétrant l'adresse "CommsAddr". Les champs du paramètre composé décrit précédemment sont dans le même ordre que les entrées recensées dans les tableaux.

Exemple : pour lire l'entrée 3 de la liste d'alarmes, le paramètre "Index" est mis à 3:

EOT0000 STX A90 RS> 3ETX?

Le paramètre composé peut alors être relu :

EOT0000A90 ENQ

La réponse est :

STX A90RS> 3 us> type us> area us. state us> level
us> count us> otime us@oproduced
us> atime us> itime us>@ iproduced

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
LogId	**DINT**	1	Oper	Oper	Limite haute Limite basse	255 1
Index	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 1
ClcLog	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Status	**BOOL**	NOGO (0)	Oper	Oper	Sens	NoGo (0) Go (1)
Erroe_No	**DINT**	0	Oper	Block	Limite haute Limite basse	255 0
NumLogged	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
ActIndex	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Type	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Area	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Count	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Occured_t	DATE_AND_TIME		Oper	Oper	Limite haute Limite basse	
Occured_p	**REAL**	0	Oper	Oper	Limite haute Limite basse	
Acked_t	DATE_AND_TIME		Oper	Oper	Limite haute Limite basse	
Inactvtd_t	DATE_AND_TIME		Oper	Oper	Limite haute Limite basse	
Inactvtd_p	**REAL**		Oper	Oper	Limite haute Limite basse	
Alm_Str	**STRING**	0	Oper	Oper	Longueur maxi.	255 caractères

Tableau 20-5 Attributs du paramètre History

BLOC FONCTION SENSOR

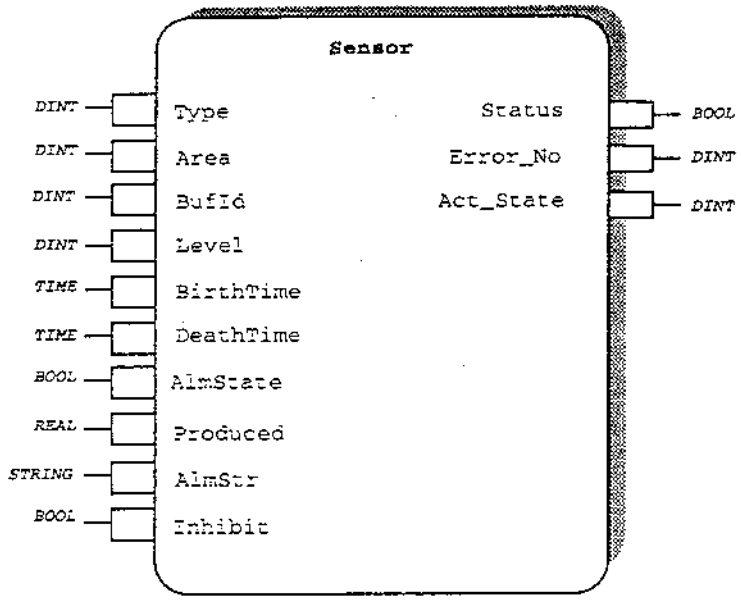


Fig. 20-4 Schéma du bloc fonction Sensor

Description fonctionnelle

Les capteurs d'alarme sont les éléments chargés de recueillir des informations dans le sous-système d'alarme. Les capteurs d'alarme recueillent des informations telles que le moment où l'alarme est apparue et la source de la situation d'alarme. Ces données sont alors enregistrées ou mémorisées dans un bloc tampon d'alarmes Cur_Alms. Ce tampon d'alarmes en cours ou réelles est associé aux capteurs d'alarme au moyen d'un paramètre d'identification de tampon.

Différents capteurs peuvent, si nécessaire, être associés à des tampons d'alarme distincts, en affectant à chaque groupe de capteurs des valeurs d'identification de tampon différentes.

Des capteurs peuvent être utilisés pour détecter une situation d'alarme ou un événement défini par l'utilisateur ou bien en association avec un bloc Detector afin de fournir des informations relatives à des stratégies d'alarme "standard". Exemples d'alarmes "standard" : seuil haut, seuil bas, écart etc. Une alarme définie par l'utilisateur peut nécessiter le dépassement d'une valeur et l'écoulement d'un certain laps de temps. Dans le cas des alarmes définies par l'utilisateur, la situation est définie par toute expression d'état (ST) qui est jugée VRAIE ou FAUSSE. Cette expression peut être câblée par programme à l'entrée état d'alarme des capteurs d'alarme.

Lorsque l'état d'alarme change, l'état est envoyé au tampon d'alarmes en cours.

Attributs du bloc fonction

Type : 6662

Classe : ..... ALARMS

Tâche par défaut : ..... Task_1

Liste résumée : ..... Type, Area, BufID, Alm_Str

Mémoire nécessaire : ..... 326 octets

Description des paramètres

Le bloc fonction Sensor dispose des paramètres suivants :

BufId (BID)

Ce tampon d'alarme mémorise temporairement l'identificateur (ID). C'est la référence permettant aux capteurs d'alarmes d'être reliés ou associés au bloc fonction Cur_Alms.

Type (TYP) et Area (AR)

Les paramètres de type et de zone peuvent s'utiliser de deux façons différentes. Dans certains systèmes d'alarme, les alarmes sont groupées selon le type (par exemple "Valeur de procédé hors échelle") et selon la zone (par exemple "Boucle a"). Les types et zones de détecteur d'alarme peuvent se combiner pour donner une référence unique (on aura, par exemple, un seul capteur de type 1 dans la zone 5). Ce système d'alarme est particulièrement utile avec les terminaux Xycom, où un message complet d'alarme (par exemple "Valeur de procédé hors échelle boucle 1") peut être généré à partir de deux octets de données.

Les systèmes d'alarme sans concept de type et de zone ne nécessitent qu'un seul nombre pour identifier chaque bloc capteur d'alarme.

Level (L)

Des alarmes différentes peuvent nécessiter des actions différentes dans le système de commande. Par exemple, une alarme d'écart par rapport à la consigne peut ne justifier qu'un avertissement, tandis qu'une valeur de procédé hors échelle peut nécessiter une mise à l'arrêt par le système de commande. Le journal d'alarmes enregistre le niveau et le niveau maximal dans le journal d'alarmes.

AlmState (S)

C'est un paramètre booléen qui est câblé avec la situation de défaut. Par exemple, une alarme de dépassement de température peut comporter :

```
sensor1.State : =AI1.Process_Val > alm_Level.Value ;
```

Produced (P)

Le paramètre produit est utilisé dans des procédés où une position ou un volume de produit doit être enregistré dans le tampon d'alarmes. Par exemple, un amorçage sur un câble nécessite l'enregistrement de la position et non du moment.

Alm_Str (STR)

Les applications utilisant de simples tableaux, comme le terminal VT 100 ou l'afficheur EUROTHERM, ne peuvent pas calculer une chaîne d'alarme à partir d'une référence numérique. Chaque bloc fonction capteur dispose par conséquent d'une entrée chaîne d'alarme pour les messages.

Inhibit (INH)

La fonction d'inhibition force l'alarme à l'état inactif. Elle sert à invalider des alarmes pendant certaines phases de procédé. Par exemple, les alarmes d'écart peuvent être inhibées quand la porte d'un four est ouverte. Si AlmState est actif alors que le paramètre Inhibit est au niveau bas, une nouvelle alarme est immédiatement émise.

Status (ST)

L'état est un simple paramètre GO / NOGO. La raison de l'erreur est spécifiée par le paramètre Error_No.

Error_No (ERR)

Il indique le numéro d'erreur du bloc fonction. Si une erreur est détectée lorsque le programme PC3000 tourne, l'erreur est signalée comme indiqué dans le tableau ci-dessous et l'état du bloc passe à NOGO.

Les erreurs suivantes sont détectées et indiquées :

N° d'erreur	Erreur	Cause et remède
103	Pas de tampon d'alarme en cours	Il n'y a pas de bloc Cur_Alms avec identificateur de tampon BufId dans le programme. Vérifier les paramètres BufId sur les blocs de capteur et d'alarmes en cours.
133	IPlus d'espace d'enregistrement	Le bloc d'alarmes en cours avec un identificateur de tampon BufId comporte 128 alarmes non remises à zéro. L'alarme qui vient juste d'apparaître n'a pas un niveau supérieur au niveau d'alarme le plus bas dans le tampon et n'a pas été ajouté. Trop d'alarmes apparaissent pendant une courte période. Si ceci a lieu au cours du démarrage ou du contrôle de mise en service, désactiver les alarmes en utilisant l'entrée d'inhibition des capteurs.

Time Hysteresis (Birth Time (BT) & Death Time (DT))

L'hystérésis de temps nécessite que l'alarme reste à un état donné pendant un certain temps avant que l'alarme ne soit activée. Le capteur a une sortie Act_State permettant de surveiller l'état de l'alarme en cours après hystérésis.

La fonction d'inhibition invalide l'alarme sans tenir compte de l'hystérésis de temps. Lorsque l'inhibition est supprimée, l'alarme doit cependant rester active pendant la durée d'établissement (Birth Time), avant qu'une alarme ne soit émise.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Type	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
Area	**DINT**	0	Oper	Oper	Limite haute Limite basse	255 0
BufID	**DINT**	1	Oper	Oper	Limite haute Limite basse	255 0
Level	**DINT**	1	Oper	Oper	Limite haute Limite basse	255 1
Birth Time	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	
Death Time	**TIME**	0ms	Oper	Oper	Limite haute Limite basse	
AlmState	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Produced	**REAL**	0	Oper	Block	Limite haute Limite basse	
Alm_Str	**STRING**	0	Oper	Oper	Longueur maxi.	255 caractères
Inhibit	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Status	**BOOL**	NOGO (0)	Oper	Oper	Sens	NoGo (0) Go (1)
Erroe_No	**DINT**	0	Oper	Block	Limite haute Limite basse	255 0
Act_State	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)

Tableau 20-6 Attributs des paramètres Sensor

BLOC FONCTION SENSOR 16

Pour réduire le temps passé à l'instanciation des blocs fonctions, un bloc d'alarme Sensor 16 a été prévu : il contient 16 blocs fonctions capteur imbriqués. Les blocs ont un type commun et des paramètres de zone contigus. Il ont également les mêmes valeur d'hystérésis, la même chaîne d'alarme et les mêmes entrées d'inhibition.

BLOC FONCTION DETECTOR

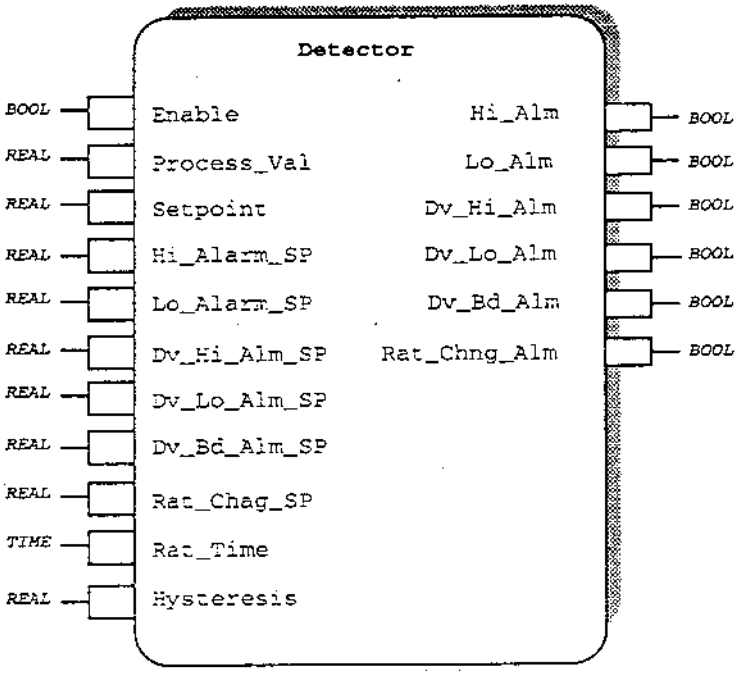


Fig. 20-5 Schéma du bloc fonction Detector

Description fonctionnelle

Le bloc fonction Detector reçoit en entrée la valeur de procédé et de consigne, ainsi qu'un jeu de limites entre lesquelles la valeur de procédé doit fonctionner. Le bloc dispose alors de sorties booléennes correspondant à chacune de ces limites. Les contrôles de limite effectués sont :

1. Limite haute
2. Limite basse
3. Limite d'écart supérieur
4. Limite d'écart inférieur
5. Limite de bande
6. Limite de vitesse de variation

Une fonction hystérésis est également intégrée à tous les contrôles de limite, sauf la vitesse de variation et une inhibition est prévue pour mettre toutes les sorties à zéro.

Le bloc fonction est utilisé associé à un bloc fonction capteur d'alarme qui enregistre les données, comme le moment d'occurrence de l'alarme etc. Les sorties du bloc détecteur concerné doivent être câblées sur une entrée AlmState de capteur d'alarme.

Attributs du bloc fonction

Type : 6674

Classe : ..... ALARMS

Tâche par défaut : ..... Task_1

Liste résumée : ..... Enable, Process_Val, Setpoint, Dv_Bd_Alm

Mémoire nécessaire : ..... 66 octets

Description des paramètres

Le bloc fonction Detector dispose des paramètres suivants :

Enable (EN)

Lorsque l'entrée Enable est à zéro (*0*), toutes les sorties sont mises à zéro (*0*) et l'alarme de vitesse moyenne de variation est remise à zéro. Lorsque Enable est à 1, toutes les sorties fonctionnent comme décrit ci-après.

Process_Val (PV)

Valeur du procédé à surveiller.

Setpoint (SP)

Consigne correspondante pour le processus, utilisée pour les alarmes d'écart.

Hi_Alarm_SP (HAS)

Niveau au-dessus duquel l'alarme haute devient active.

Lo_Alarm_SP (LAS)

Niveau au-dessous duquel l'alarme basse devient active.

Dv_Hi_Alm_SP (DHS)

Ecart par rapport à la consigne au-dessus duquel l'alarme d'écart supérieur devient active.

Dv_Lo_Alm_SP (DLS)

Ecart par rapport à la consigne en dessous duquel l'alarme d'écart inférieur devient active.

Dv_Bd_Alm_SP (DBS)

Ecart par rapport à la consigne au-delà duquel l'alarme de bande devient active.

Rat_Chang_SP (RCS)

Variation de la valeur de procédé sur la durée Rat_Time au-delà de laquelle une alarme de vitesse de variation est émise.

Rat_Time (RT)

Durée sur laquelle est calculée l'alarme de vitesse de variation.

Hysteresis (HYS)

Niveau d'hystérésis pour les alarmes : haute, basse, écart supérieur, écart inférieur et bande.

La fonction de chacune des sorties est décrite ci-après :

Hi_Alm (HA)

If Process_Val<Hi_Alm_SP - Hysteresis:	Hi_Alm=Off(*0*)
Hi_Alm_SP - Hysteresis<Process_Val<Alm_SP:	Hi_Alm=unchanged
Hi_SAlm_SP<Process_Val:	Hi_Alm=On(*0*)

Lo_Alm (LA)

If Process_Val>Lo_Alm_SP+Hysteresis: Lo_Alm=Off(*0*)

Lo_Alm_SP+Hysteresis>Process_Val>Lo_Alm_SP: Lo_Alm=unchanged

Lo_Alm_SP>Process_Val Lo_Alm=On(*1*)

Dv_Hi_Alm (DHA)

If deviation<Dv_Hi_Alm_SP-Hysteresis: Dv_Hi_Alm=Off(*0*)

Dv_Hi_Alm_SP-Hysteresis,deviation,Dv_Hi_Alm_SP:Dv_Hi_Alm=unchanged

Dv_Hi_Alm_SP<deviation: Dv_Hi_Alm=On(*1*)

Deviation = Process_Val - Setpoint

Dv_Lo_Alm (DLA)

If deviation,Dv_Lo-Alm_SP - Hysteresis: Dv_Lo_Alm=Off(*0*)

Dv_Lo_Alm_SP - Hysteresis<deviation<Dv_Lo_Alm-SP: Dv_Lo_Alm=unchanged

Dv_Lo_Alm_SP<deviation: Dv_Lo_Alm=On(*1*)

Deviation + Setpoint - Process_Val

Dv_Bd_Alm (DBA)

If deviation<Dv_Bd_Alm_SP-Hysteresis: Dv_Bd_Alm=Off(*0*)

Dv_Bd_Alm_SP-Hysteresis<deviation<Dv_Bd_Alm_SDv_Bd_Alm=unchanged

Dv_Bd_Alm_SP<deviation: Dv_Bd_Alm=On(*1*)

Deviation = Setpoint - Process - Val

Rat_Chng_Alm (RCA)

La valeur de procédé est filtrée en prenant la valeur filtrée précédente et la valeur de procédé actuelle :

$$PV_{filtered}(n)=(3*PV_{filtered}(n-1)+PV(n))4$$

N.B. :Il y a lieu de noter que ces algorithmes ne prennent pas en compte les situations de démarrage et que celles-ci doivent être prises en considération dans le programme utilisateur.

Le contrôle de limite de vitesse est effectué sur Process_Val après filtrage, en prenant la valeur absolue de la différence PVfiltered(n)-PVfiltered(n-1) et en divisant par le temps de base pour obtenir une vitesse. Celle-ci est ensuite comparée à la vitesse indiquée par Rat_Chang_SP/Rat_Time.

Pour résumer :

$$Rat\_Chng\_Alm=|PV_{filtered}(n)-PV_{filtered}(n-1)|>(Rat\_Chang\_SP/Rat\_Time)$$

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Enable	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Process_Val	REAL	0	Oper	Oper	Limite haute Limite basse	
Setpoint	REAL	1	Oper	Oper	Limite haute Limite basse	
Hi_Alm_SP	REAL	1	Oper	Oper	Limite haute Limite basse	
Lo_Alm_SP	REAL	0ms	Oper	Oper	Limite haute Limite basse	
Dv_Hi_Alm_SP	REAL	0ms	Oper	Oper	Limite haute Limite basse	
Dv_Lo_Alm_SP	REAL	Off (0)	Oper	Oper	Limite haute Limite basse	
Dv_Bd_Alm_SP	REAL	0	Oper	Block	Limite haute Limite basse	
Rat_Chang_SP	REAL	0	Oper	Oper	Limite haute Limite basse	
Rat_Time	TIME	Off (0)	Oper	Oper	Limite haute Limite basse	
Hi_Alm	BOOL	NOGO (0)	Oper	Oper	Sens	Off (0) On (1)
Lo_Alm	BOOL	0	Oper	Block	Sens	Off (0) On (1)
Dv_Hi_Alm	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Dv_Lo_Alm	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Dv_Bd_Alm	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Rat_Chng_Alm	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)

Tableau 20-7 Attributs des paramètres Detector

Chapitre 21

STATISTIQUES

Edition 3

Présentation

Présentation

Cette classe autorise les fonctions SPC (contrôle statistique) sur une valeur REELLE unique en temps réel. Les sorties peuvent être communiquées simplement à l'utilisateur par une interface utilisateur appropriée comme un terminal ou, si le procédé est suffisamment bien défini et connu, elles peuvent servir à modifier le procédé sans intervention de la part de l'utilisateur.

Le SPC (contrôle statistique) est utilisé dans les applications où il faut surveiller et réguler les conditions à long terme afin d'optimiser le rendement du procédé.

Les blocs fonctions peuvent être utilisés *indépendamment* ou en *tandem*.

Il existe trois blocs SPC (contrôle statistique) :

SPC fournit une courbe de base Shewhart pour une variable analogique ou réelle échantillonnée sur la base du TEMPS.

SPC_Event offre des fonctions identiques au bloc SPC mais l'échantillonnage a lieu sur demande (EVENEMENT).

Le bloc **Histogram** calcule la répartition de la variable réelle et fournit la moyenne courante, l'écart type et une indication de l'indice de capacité.

Le bloc final, **Statistics**, fournit simplement le minimum, le maximum, la moyenne et l'écart type d'une variable REELLE.

Il est impossible de contrôler ou de tester la qualité dans un produit, il doit être construit correctement du premier coup. Cela implique que la fabrication soit stable et que l'ensemble des personnes concernées par le procédé (opérateurs, techniciens, personnel de l'assurance qualité et direction) cherchent à améliorer en permanence les performances du procédé et diminuent les variations sur les paramètres essentiels.

La première étape de l'amélioration est la possibilité de mesurer la qualité. Il est par conséquent important de déterminer la ou les variable(s) de procédé qui est(sont) essentielle(s) pour évaluer la qualité et les variables qu'il est possible de manipuler pour influer sur la qualité.

Afin d'évaluer la qualité du produit, il existe un certain nombre d'outils statistiques simples que l'on peut utiliser :

Courbes

Histogrammes

Fiches de contrôle

Analyse de Pareto

Schémas de concentration des défauts

Schémas des causes et des effets

Courbes de dispersion et corrélation

Les courbes et les histogrammes sont des outils qui peuvent être utilisés en ligne et les blocs fonctions PC3000 offrent une possibilité d'archivage de routine associée à l'utilisation d'une courbe pour les variables (par opposition aux attributs).

La figure 21-1 montre une courbe type.

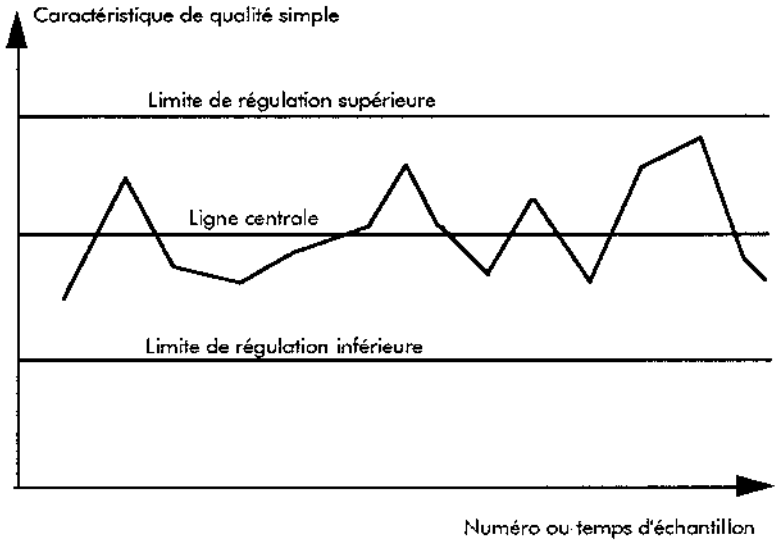


Figure 21-1 Courbe type

La méthode type de production d'une courbe consiste à mesurer le signal intéressant à intervalles réguliers (si l'échantillonnage doit s'effectuer par rapport au temps). Les échantillons sont ensuite regroupés en sous-groupes de taille petite à moyenne (dans le cas du PC3000, la taille du sous-groupe est comprise entre 2 et 10).

Certaines statistiques du sous-groupe sont calculées (dans le cas du PC3000, la moyenne, l'écart type et la plage [la plage est définie comme la différence entre les valeurs minimale et maximale du sous-groupe]).

Ces statistiques sont ensuite tracées et étudiées selon certaines règles spécifiées pour ce qui est de l'archivage de la courbe. Dans le PC3000, l'archivage est automatique (par exemple les tests pour les échantillons incontrôlables, les dérives, les erreurs systématiques, etc.).

L'utilisateur doit définir la méthodologie de groupe :

Sub_Time	définit l'intervalle de base entre les échantillons de sous-groupes,
Sub_Size	définit le nombre total d'éléments d'un sous-groupe,
Group_Time	définit la différence de temps entre les échantillons de sous-groupes et
Group_Size	fixe le nombre total de sous-groupes utilisés pour l'évaluation (le maximum est de 125). Cf. figure 21-2.

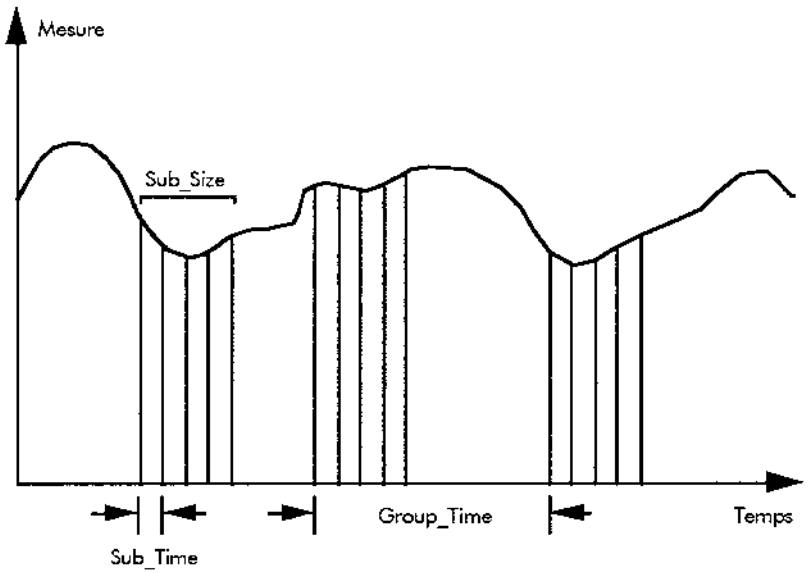


Figure 21-2 Procédé d'échantillonnage type

Si les limites de la machine ou du procédé étudié(e) sont inconnues, il est possible d'utiliser les données collectées pour obtenir les limites des courbes. Le bloc fonction PC3000 effectue le calcul nécessaire. Toutefois, la plupart du temps, les limites sont connues ou fixées et le bloc sert à effectuer l'archivage de base d'une courbe Shewhart.

Un autre procédé d'échantillonnage courant pour les données séquentielles consiste à utiliser une fenêtre mobile, où un sous-groupe est composé des n derniers points de données, n étant la taille du sous-groupe. On utilise habituellement cette méthode pour détecter les variations de moyenne du procédé. Malheureusement, avec cette méthode d'échantillonnage, le seul contrôle fiable est le contrôle hors limites car il y a une corrélation inter-échantillons élevée, ce qui rend les tests

d'erreur systématique trop sensibles avec un degré élevé de fausses alarmes. La figure 21-3 montre cette méthode d'échantillonnage.

Il est parfois nécessaire de mettre à jour les limites de régulation en raison des variations du procédé, ce qui peut se faire de deux manières. La première consiste à redémarrer la phase de collecte des données. La deuxième, tout comme les fenêtres mobiles pour les données brutes, permet de calculer les limites de régulation sur la base des fenêtres mobiles des sous-groupes.

Dans ce cas, les limites sont calculées à l'aide de la dernière taille de sous-groupes. De cette manière, les limites passent lentement des valeurs actuelles aux nouvelles. Bien sûr, au cours de la mise à jour des limites, la plupart des tests ne sont pas valables mais il est possible d'effectuer des tests hors limites.

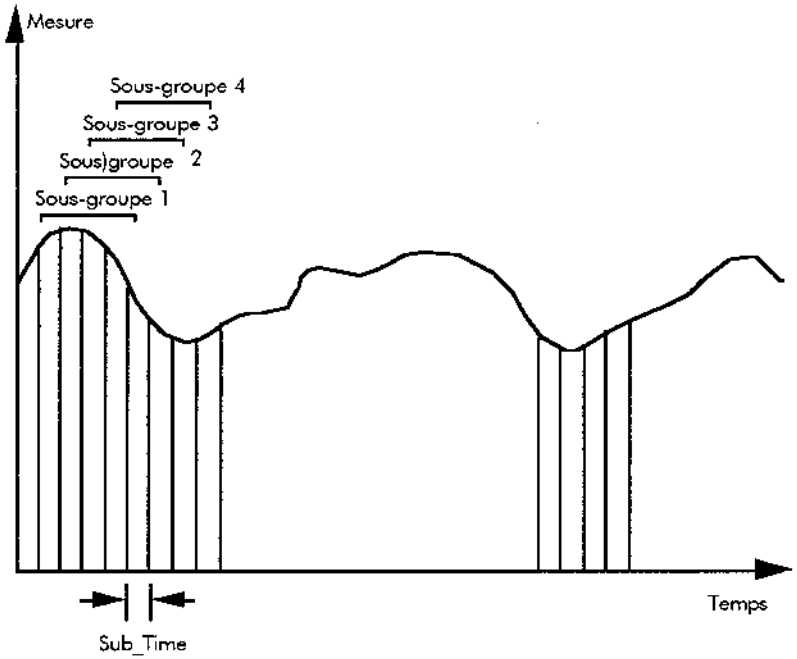


Figure 21-3 Procédé d'échantillonnage avec fenêtre mobile

L'échantillonnage peut bien entendu s'effectuer sur la base du temps ou de tout autre événement. Pour les cas où l'échantillonnage doit être synchronisé avec des événements autres que le temps, il est possible d'utiliser le bloc fonction **SPC_Event**.

Un autre aspect important de la surveillance statistique de la qualité et de la régulation qui en résulte est associé à la répartition des mesures. Le bloc fonction **Histogram** donne un histogramme en ligne des mesures. Il fournit également à l'utilisateur les moyennes courantes, les écarts types et les mesures de désalignement et de forme (arrondie, pointue, etc.) de la répartition. En outre, des contrôles d'anomalie sont effectués sur la base du désalignement et de la forme (arrondie, pointue, etc.) de la répartition. Un indice brut de capacité est également calculé sur la base du nombre d'éléments de chacun des segments de l'histogramme. Cela peut être utile pour l'appariement qualitatif avec les chiffres provenant des blocs fonctions **SPC** et **SPC_Event**. Ce bloc fonction fournit à l'utilisateur un outil puissant et simple qui lui permet d'examiner la répartition des variables intéressantes.

La figure 21-4 montre un exemple d'histogramme. Le bloc fonction peut être utilisé pour effectuer des échantillonnages à la demande ou à intervalles réguliers.

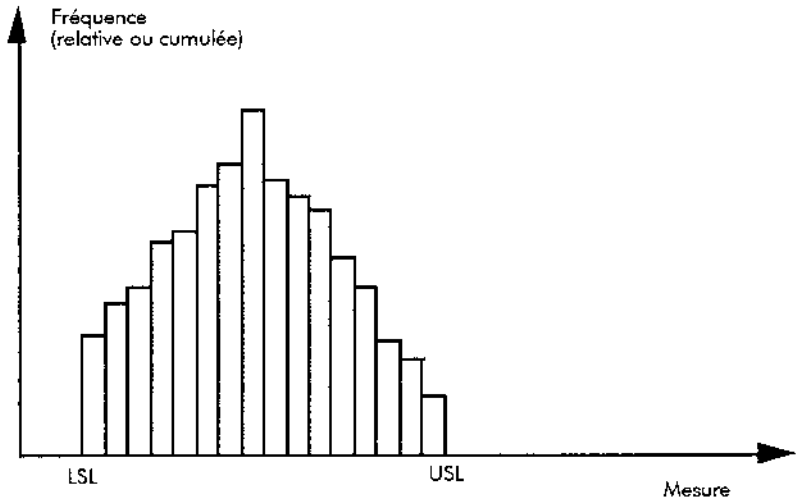


Figure 21-4 Histogramme type

Pour résumer, les blocs fonctions PC3000 traitent les données brutes pour former un ensemble d'indicateurs faciles à comprendre et à interpréter. Le reste du logiciel utilisateur peut servir à lancer certaines opérations de maintenance ou de correction afin de ramener le procédé en contrôle statistique s'il commençait à dériver.

Normes

La suite de blocs fonctions SPC suit les exemples de test de Ford Motor Company ci-après pour le logiciel SPC (contrôle statistique). Cf. la référence 3 pour avoir plus de détails.

Tests 1,2 et 3 : se rapportent aux exigences de base pour les graphiques s avec moyenne qui constituent une bonne indication des situations incontrôlables, de la précision des calculs et du calcul des indices de capacité pour les répartitions normales.

Test 10 : moyennes mobiles, graphiques *s* mobiles et *R* mobiles.

Tests 12 et 13 : bonne indication des graphiques *R* avec moyenne dont les situations sont incontrôlables et calcul de l'indice de capacité.

Les tests 8 et 9 associés au changement d'outillage et à l'usure de l'outillage sont respectés dans la mesure où l'indication des situations incontrôlables est concernée mais les corrections pour le calcul des indices de capacité ne sont pas effectués dans ce cas.

Mauvaises utilisations de SPC (contrôle statistique)

La courbe Shewhart (comme toutes les techniques statistiques) doit être utilisée dans les cas où l'on est suffisamment sûr que le procédé à surveiller répond aux caractéristiques prévues pour la dérivation des courbes. Pour que les courbes soient valables :

> Le procédé doit être statistiquement contrôlé, ce qui implique que toutes les variations systématiques aient été éliminées. Le contrôle de stabilité à la fin de la phase de collecte des données du bloc fonction SPC va dans le sens de la stabilité mais ne doit pas être utilisé comme seule indication de la stabilité du procédé.
>
> L'échantillonnage doit être tel que les sous-groupes garantissent que les variations y sont principalement localisées et que la variation inter-groupes est purement aléatoire (sauf lorsque des conditions incontrôlables apparaissent).

Deux problèmes que l'on rencontre très souvent dans les applications de SPC (contrôle statistique) sont les données en corrélation et l'effet des variations entre les échantillons. Dans les deux cas, les limites de régulation calculées par les calculs standard (comme dans le bloc fonction SPC) peuvent être entièrement fausses et il peut être nécessaire de modifier les limites après une analyse d'autopsie.

Il faut noter que le fait de continuer avec le SPC standard sur ces procédés peut aboutir à des conclusions erronées au sujet des caractéristiques du procédé et de ses capacités.

Références

(1) Statistical Process Control: Instruction Guide, Ford Motor Company, EU 880 A, 1985.

(2) Process Capability: Guidelines, Ford Motor Company, EU 882 A, 1990.

(3) Test Examples for SPC Software, Statistical Methods Office, Ford Motor Company, EU 33793 A, 1991.

(4) Statistical Quality Control, E.L. Grant and R.S. Leavenworth, 6th Edition, McGraw-Hill Book Co., 1988.

(5) Introduction to Statistical Quality Control, D.C. Montgomery, Second Edition, John Wiley and Sons, 1991.

(6) Biometrika tables for statisticians, E.S. Pearson and H.O. Hartley, Cambridge University Press, 1966.

BLOC FONCTION SPC

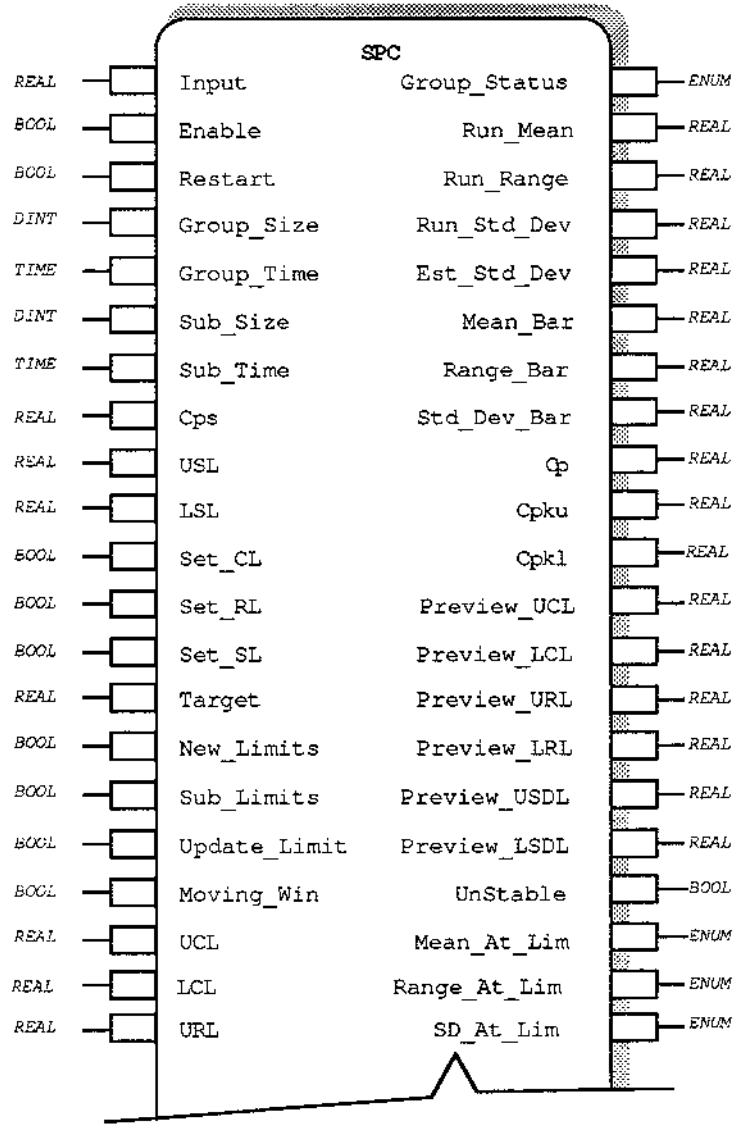


Figure 21-5 Schéma du bloc fonction SPC

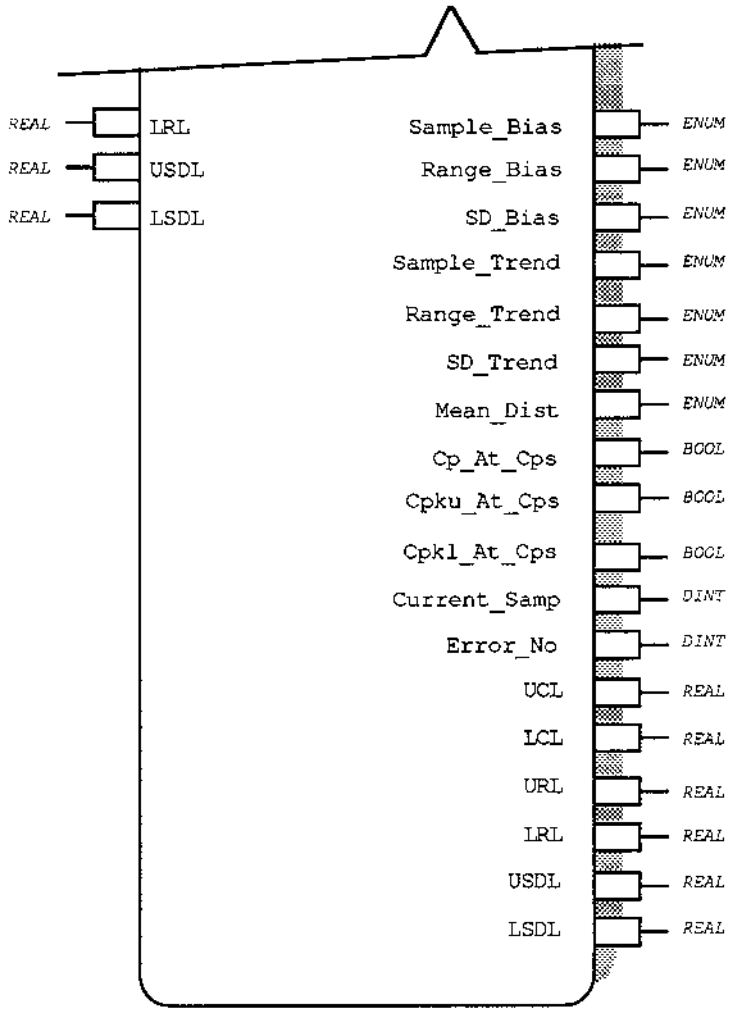


Figure 21-5 Schéma du bloc fonction SPC (suite)

Description fonctionnelle

Le bloc fonction SPC est conçu pour effectuer les tâches suivantes :

* calculer et stocker les indicateurs qui conviennent (**Run_Mean, Run_Range, Run_Std_Dev, Mean_Bar, Range_Bar, Std_Dev_Bar, Est_Std_Dev**) pour une mesure continue (entrée) ;

* calculer les limites supérieures et inférieures de régulation et de plage (**UCL, LCL, URL, LRL**) associées au procédé une fois que le paramètre

Group_Size des sous-groupes a été collecté ou bien il faut utiliser des valeurs prédéfinies pour ces limites ;

* calculer les valeurs intermédiaires des limites de régulation **Preview_LCL, Preview_UCL**, les limites de plage **Preview_LRL, Preview_URL** et les limites d'écart type (**Preview_LSDL, Preview_USDL**) à 25 %, 50 % et 75 % de la totalité du groupe ;

* mettre à jour les limites de régulation sur demande en fonction des données d'une fenêtre mobile des points de données **Group_Size** ;

* examiner si le procédé est contrôlé statistiquement (c'est-à-dire si la moyenne, la plage et les écarts types de l'échantillon sont dans les limites, ne présentent pas une erreur systématique par rapport à la moyenne calculée des moyennes des échantillons et ne dérivent pas, et si la répartition de X barre est à peu près normale, c'est-à-dire si 2/3 des valeurs sont situées dans le tiers moyen entre les limites) ;

* calculer les capacités du procédé, c'est-à-dire la capacité globale du procédé plus les capacités supérieure et inférieure du procédé (**Cp, Cpku, Cpkl**) et vérifier si ces valeurs se situent dans la spécification exigée.

* calculer les graphiques avec les moyennes, les plages et les écarts types des fenêtre mobiles.

* examiner la stabilité statistique du procédé par un contrôle rétrospectif portant sur les points de données précédents dans le cas où les limites sont calculées à partir des points de données mesurés.

Il existe deux modes de fonctionnement élémentaires pour le bloc fonction **SPC** : la collecte des données et le calcul interne des limites de régulation et de plage ou bien l'examen des statistiques des données par rapport aux limites spécifiées.

Dans les deux modes, Input et Enable doivent être fixés par l'utilisateur. Les variables **Group_Size, Group_Time, Sub_Size** et **Sub_Time** sont copiées et vérifiées soit la première fois qu'elles sont appelées soit immédiatement après un redémarrage. Les valeurs doivent être redémarrées à froid ou fixées puis le bloc fonction doit être redémarré. **Cps, USL** et **LSL** sont les limites de spécification et, si elles ne sont pas correctement définies, le bloc fonction indiquera des erreurs. Il continuera à fonctionner mais ne calculera pas les indices de capacité du procédé.

Si les valeurs des limites de régulation et de plage sont prédéfinies, il faut régler **Set_CL, Set_RL** et **Set_SL** , ce qui inhibe la mise à jour de **LCL** etc. par le bloc fonction et des tests d'échantillons hors spécification sont effectués avec les valeurs prédéfinies. Le test d'erreur systématique et de répartition pour X est effectué par rapport à la valeur de Target qui est normalement fixée à (**UCL + LCL**)/2}. Des graphiques de plage et d'écart type sont réalisés respectivement avec (**URL + LRL**)/2} et (**USDL + LSDL**)/2}.

Si les limites doivent être calculées en interne, il faut régler **New_Limits** et **Sub_Limits** de telle manière que les valeurs prévisionnelles soient mises à jour à intervalles réguliers et que les valeurs calculées de **LCL** etc. soient transférées. Il est possible de fixer **Update_Limit** afin d'effectuer une mise à jour en continu des limites de régulation et de traiter les capacités en fonction d'une fenêtre mobile de taille **Group_Size** .

Applications types

Graphiques en continu

Lorsque les limites des graphiques sont fixées et la stratégie d'échantillonnage est arrêtée :

Câbler l'entrée à la variable nécessaire pour le graphique.

Précharger les valeurs d'**UCL, LCL, Target, URL, LRL, USDL** et **LSDL.**

Précharger **Sub_Size, Group_Size, Sub_Time** et **Group_Time** dans le cas du bloc fonction **SPC**.

Dans le cas du bloc fonction **SPC**, régler l'entrée **Enable** toutes les fois que l'échantillonnage doit commencer. Dans le cas du bloc fonction **SPC_Event**, régler **Enable** toutes les fois qu'il faut prélever un seul échantillon. Le programme utilisateur doit effacer l'entrée **Enable** avant de préparer l'échantillonnage suivant.

Fixer **Set_CL, Set_RL, Set_SL.**

Le bloc (**SPC** ou **SPC_Event**) continue à effectuer les contrôles nécessaires pour les graphiques et à indiquer les erreurs éventuelles.

Il faut noter que, si une réinitialisation est nécessaire (par exemple pour modifier la stratégie d'échantillonnage), il faut modifier les paramètres nécessaires et un flanc ascendant sur le paramètre **Restart** fera redémarrer le bloc fonction avec ces nouvelles valeurs.

Calcul des limites appropriées

Dans le cas où les limites doivent être calculées :

Câbler l'entrée à la variable nécessaire pour le graphique.

Vérifier que **Set_CL, Set_SL, Set_RL** sont vides.

Précharger **Sub_Size, Group_Size, Sub_Time** et **Group_Time** dans le cas du bloc fonction **SPC**.

Dans le cas du bloc fonction **SPC**, régler l'entrée Enable toutes les fois que l'échantillonnage doit commencer. Dans le cas du bloc fonction **SPC_Event**, régler Enable toutes les fois qu'il faut prélever un seul échantillon. Le programme utilisateur doit effacer l'entrée **Enable** avant de préparer l'échantillonnage suivant.

Régler **New_Limits.**

Régler **Sub_Limits** si l'on souhaite surveiller les limites de régulation calculées avant de les utiliser. Il est possible d'utiliser les valeurs prévisionnelles pour cela.

Le bloc (**SPC** ou **SPC_Event**) collecte les données de la taille voulue à l'aide de la stratégie d'échantillonnage et calcule les limites à la fin du sous-groupe final. Le bloc scrute ensuite les données historiques pour les contrôles de stabilité standard. S'il n'y a pas de situation incontrôlable, l'indicateur **UnStable** reste vide sinon est il est positionné sur 1.

Le bloc (**SPC** ou **SPC_Event**) continue à effectuer les contrôles nécessaires pour mettre à jour les courbes Shewhart et continue à signaler les éventuelles erreurs.

Il incombe au programme utilisateur de modifier ou de ne pas prendre en compte les limites calculées.

S'il est impératif de modifier le procédé d'échantillonnage ou de relancer l'opération de collecte des données, fixer les entrées du bloc fonction sur les valeurs appropriées (modifier par exemple **Sub_Time et Group_Time**) et positionner l'entrée **Restart** du bloc fonction sur 1. Le programme utilisateur doit remettre à zéro le paramètre **Restart**.

Mise à jour des limites

S'il est impératif de mettre à jour les limites à partir de leurs valeurs actuelles vers un nouvel ensemble en fonction des données qui arrivent, il est possible de mettre à 1 l'indicateur **Update_Limit.**

Il ne faut pas confondre ce mode avec les sous-groupes de fenêtre mobile décrits dans la figure 21-3.

Il est bien entendu possible de modifier aussi les limites à tout moment en modifiant en externe les valeurs d'**UCL** etc.

Limites prédéfinies et calculées

Il est possible d'utiliser les valeurs continues pour les graphiques et de calculer simultanément des valeurs nouvelles à partir d'un ensemble spécifié de points de données. Pour cela, les entrées du bloc fonction sont positionnées sur celles nécessaires pour les graphiques continus et fixent la valeur booléenne **Sub_Limits**. Cela permet de calculer les valeurs prévisionnelles avec le graphique habituel qui continue. En cas de différences importantes, l'utilisateur peut choisir les limites qui conviennent pour le procédé.

Attributs du bloc fonction

Type : 68 10

Classe : ..... STATISTIQUES

Tâche par défaut : ..... Task_2

Liste résumée : ..... Input, Enable Cp Group_Status

Description des paramètres

Paramètres d'entrée

Input (IN)

Mesure que le bloc fonction doit surveiller. Peut être une mesure effective ou peut être déduite d'une formule. La valeur doit être continue et non on / off.

Enable (EN)

Interrupteur qui (dés)active le fonctionnement du bloc fonction. Lorsqu'il est positionné sur 1 (* Run *), le bloc fonction est entièrement fonctionnel. Lorsqu'il est positionné sur 0 (* Reset *), seul un démarrage à froid consistant en l'initialisation de certaines matrices et variables internes est effectué. Si, à un stade quelconque après l'initialisation, **Enable** est positionné sur **Reset**, le bloc fonction entre dans un état d'attente et est réactivé lorsqu'on positionne Enable sur Run .

Restart (RE)

Toutes les fois que **Restart** passe de 0 (* Non *) à 1 (* Oui *) (c'est-à-dire un flanc ascendant), un démarrage à froid est effectué. Un flanc descendant n'a aucune influence sur le bloc fonction. Un redémarrage n'est possible que si le bloc fonction est activé.

Group _Size (GS)

Ce paramètre définit le nombre de sous-groupes d'un groupe. Il ne peut avoir ni une valeur inférieure à 1 ni une valeur supérieure à 125. Ce paramètre est en lecture seule au cours d'un démarrage à froid initial ou d'un redémarrage. La modification de ce paramètre au cours du fonctionnement du bloc fonction n'a aucune influence.

Group_Time (GT)

Ce paramètre définit l'intervalle qui sépare le début de deux sous-groupes consécutifs. Il ne doit pas être inférieur à **Sub_Time** x **Sub_Size.**

Sub_Size (SS)

Ce paramètre définit le nombre d'échantillons consécutifs prélevés dans chaque sous-groupe. Il ne peut être ni inférieur à 2 ni supérieur à 10.

Sub _Time (ST)

Ce paramètre définit l'intervalle d'échantillonnage entre deux échantillons consécutifs. La limite inférieure est l'intervalle de la tâche. Le degré de finesse du temps est également l'intervalle de la tâche.

Cps (CPS)

Ce paramètre est la capacité minimale acceptable du procédé. Une valeur de 1,33 est un minimum type.

USL (USL)

Ce paramètre est la limite supérieure de spécification de la variable **Input**. Il doit être choisi de manière à être compatible avec **LSL**.

LSL (LSL)

Ce paramètre est la limite inférieure de spécification de la variable **Input** . Il doit être choisi de manière à être compatible avec **USL**.

Set _CL (SCL)

Ce paramètre, lorsqu'il est sur 1 (* Oui *), positionne **LCL** et **UCL** sur leurs valeurs affectées avant l'exécution du bloc fonction. Une valeur de 0 (* Non *) implique que le bloc fonction peut modifier la valeur des limites de régulation lors de l'exécution du bloc fonction. Lorsque Set est sur 1 (* Oui *), des tests de violation des limites et de la dérive de la moyenne des échantillons sont effectués. Le test d'erreur systématique de l'échantillon est effectué par rapport à **Target**.

Set _RL (SRL)

Ce paramètre, lorsqu'il est sur 1 (* Oui *), positionne **LRL** et **URL** sur leurs valeurs affectées avant l'exécution du bloc fonction. Une valeur de 0 (* Non *) implique que le bloc fonction peut modifier la valeur des limites de la plage lors de l'exécution du bloc fonction. Lorsque Set est sur 1 (* Oui *), des tests de violation des limites et de la dérive de la plage des échantillons sont effectués. Le test d'erreur systématique de l'échantillon est effectué par rapport à (**URL** + **LRL**)/2.

Set _SL (SSL)

Ce paramètre, lorsqu'il est sur 1 (* Oui *), positionne **LSDL** et **USDL** sur leurs valeurs affectées avant l'exécution du bloc fonction. Une valeur de 0 (* Non *) implique que le bloc fonction peut modifier la valeur des limites de l'écart type lors de l'exécution du bloc fonction. Lorsque **Set_SL** est sur 1 (* Oui *), des tests de violation des limites et de la dérive de la plage des échantillons sont effectués. Le test d'erreur systématique de l'écart type est effectué par rapport à (**USDL** + **LSDL**)/2 .

Target (T)

Valeur cible de X. Lorsque **Set_CL** est sur 1, les tests d'erreur systématique sont effectués par rapport à cette valeur. Elle est normalement fixée à (**UCL** + **LCL**)/2 .

New _Limits (NL)

Lorsqu'il est sur 1 (* Oui *), ce paramètre provoque le calcul, pour l'ensemble du groupe, des limites de régulation et de plage pour **LCL**, **UCL**, **LRL** et **URL**. Si

SetCL, Set_RL ou **Set_SL** est sur 1, les limites correspondantes *ne* sont *pas* mises à jour.

Sub _Limits (SL)

Lorsqu'il est sur 1 (* Oui *), ce paramètre provoque un calcul partiel de groupe des limites de régulation et de plage. Ces valeurs sont mises à jour à 25 %, 50 %, 75 % et 100 % de Group_Size.

Update _Limit (UL)

Lorsqu'il est sur 1 (* Oui *), ce paramètre provoque un nouveau calcul des limites de régulation et de plage en fonction d'une fenêtre glissante de données de la taille de **Group_Size** une fois que la totalité du groupe est atteinte. Lorsqu'il est sur 0 (* Non *), la mise à jour s'arrête et il reprend à partir de l'état qu'il avait auparavant lorsqu'il est remis sur 1 (* Oui *).

Moving_Win (MW)

Lorsque ce paramètre est sur 1, cela implique que les sous-groupes doivent être formés à partir d'échantillons consécutifs, contrairement à la méthode traditionnelle de groupement.

Paramètres d'entrée/sortie

UCL (UCL)

Ce paramètre est la limite supérieure de contrôle. Il est défini par l'utilisateur ou calculé par le bloc fonction. La valeur calculée est uniquement transférée sous le contrôle des paramètres **Set_CL New_Limits** et **Update_Limit**.

LCL (LCL)

Ce paramètre est la limite inférieure de contrôle. Il est défini par l'utilisateur ou calculé par le bloc fonction. La valeur calculée est uniquement transférée sous le contrôle des paramètres **Set_CL, New_Limits** et **Update_Limit**.

URL (URL)

Ce paramètre est la limite supérieure de la plage. Il est défini par l'utilisateur ou calculé par le bloc fonction. La valeur calculée est uniquement transférée sous le contrôle des paramètres **Set_CL, New_Limits** et **Update_Limit**.

LRL (LRL)

Ce paramètre est la limite inférieure de la plage. Il est défini par l'utilisateur ou calculé par le bloc fonction. La valeur calculée est uniquement transférée sous le contrôle des paramètres **Set_CL, New_Limits** et **Update_Limit**.

USDL (USD)

Ce paramètre est la limite supérieure de l'écart type. Il est défini par l'utilisateur ou calculé par le bloc fonction. La valeur calculée est uniquement transférée sous le contrôle des paramètres **Set_CL, New_Limits** et **Update_Limit**.

LSDL (LSD)

Ce paramètre est la limite inférieure de l'écart type. Il est défini par l'utilisateur ou calculé par le bloc fonction. La valeur calculée est uniquement transférée sous le contrôle des paramètres **Set_CL, New_Limits** et **Update_Limit**.

Paramètres de sortie

Group_Status (GRP)

Ce paramètre agit comme indicateur signalant que le nombre nécessaire de points de données est collecté pour un groupe complet et que les indicateurs **Mean_Bar**, **Range_Bar** et **Est_Std_Dev** ainsi que les limites de régulation et de plage sont calculés sur la base des données assimilées. Il est positionné sur NotFull, Pending ou Full. Pending est choisi lorsque le bloc fonction calcule les limites et effectue un contrôle rétrospectif des données. Le contrôle rétrospectif est uniquement effectué lorsque le bloc fonction doit calculer les limites.

Run _Mean (RM) [aussi appelé ($\bar{X}$ ou X barre]

Ce paramètre est la moyenne calculée du dernier sous-groupe de données. Il est mis à jour à chaque fois qu'un sous-groupe est complet.

Run _Range (RR)

Ce paramètre est la plage calculée du dernier sous-groupe de données. Il est mis à jour à chaque fois qu'un sous-groupe est complet.

Run _Std_Dev (RSD)

Ce paramètre est l'écart type calculé du dernier sous-groupe. Il est mis à jour à chaque fois qu'un sous-groupe est complet.

Est _Std_Dev (ESD)

Ce paramètre est l'écart type estimé de la population dont sont tirées les mesures. Il est mis à jour lorsque le groupe est plein et à toutes les fois suivantes si **Update_Limit** est sur 1.

Mean _Bar (MB) [aussi appelé ($\bar{\bar{X}}$ ou X barre barre]

Ce paramètre est la moyenne calculée des moyennes des sous-groupes. Il est évalué lorsque le groupe est plein et à toutes les fois suivantes si **Update_Limit** est sur 1.

Range _Bar (RB)

Ce paramètre est la moyenne calculée des plages des sous-groupes. Il est évalué lorsque le groupe est plein et à toutes les fois suivantes si **Update_Limit** est sur 1.

Std _Dev_Bar (STD)

Ce paramètre est la moyenne calculée des écarts types des sous-groupes. Il est évalué lorsque le groupe est plein et à toutes les fois suivantes si **Update_Limit** est sur 1.

Cp (CP)

Ce paramètre est l'indice de capacité calculé du procédé. Il est évalué lorsque le groupe est plein ou toutes les fois suivantes si **Update_Limit** est sur 1.

Cpku (CPU)

Ce paramètre est l'indice de capacité calculé supérieur du procédé. Il est évalué lorsque le groupe est plein ou toutes les fois suivantes si **Update_Limit** est sur 1.

Cpkl (CPL)

Ce paramètre est l'indice de capacité calculé inférieur du procédé. Il est évalué lorsque le groupe est plein ou toutes les fois suivantes si **Update_Limit** est sur 1.

Preview _UCL (PUC)

Ce paramètre est la limite de régulation calculée supérieure. Il est évalué à 25 %, 50 % , 75 % et 100 % de **Group_Size** si **Sub_Limit** est sur 1.

Preview _LCL (PLC)

Ce paramètre est la limite de régulation calculée inférieure. Il est évalué à 25 %, 50 % , 75 % et 100 % de **Group_Size** si **Sub_Limit** est sur 1.

Preview_URL (PUR)

Ce paramètre est la limite calculée supérieure de la plage. Il est évalué à 25 %, 50 % , 75 % et 100 % de **Group_Size** si **Sub_Limit** est sur 1.

Preview_LRL (PLR)

Ce paramètre est la limite calculée inférieure de la plage. Il est évalué à 25 %, 50 % , 75 % et 100 % de **Group_Size** si **Sub_Limit** est sur 1.

Preview _USDL (PUS)

Ce paramètre est la limite calculée supérieure de l'écart type. Il est évalué à 25 %, 50 % , 75 % et 100 % de **Group_Size** si **Sub_Limit** est sur 1.

Preview _LSDL (PLS)

Ce paramètre est la limite calculée inférieure de l'écart type. Il est évalué à 25 %, 50 % , 75 % et 100 % de **Group_Size** si **Sub_Limit** est sur 1.

UnStable (US)

Lorsque le groupe est complet, le bloc vérifie rétrospectivement les mesures et, s'il détecte des conditions incontrôlables, il met sur 1 l'indicateur **UnStable**. Cela se produit uniquement si le bloc doit calculer les limites à partir de cet ensemble de mesures.

Mean _At_Lim (MAL)

Ce paramètre est positionné sur 1 (* Bas *) lorsque la moyenne des sous-groupes est inférieure ou égale à la limite de régulation inférieure. Il est positionné sur 2 (* Haut *) lorsque la moyenne des sous-groupes est supérieure ou égale à la limite de régulation supérieure. Ce test est effectué si le groupe est plein ou si **Set_CL** est positionné sur 1.

Range _At_Lim (RAL)

Ce paramètre est positionné sur 1 (* Bas *) lorsque la plage des sous-groupes est inférieure ou égale à la limite inférieure de la plage. Il est positionné sur 2 (* Haut *) lorsque la plage des sous-groupes est supérieure ou égale à la limite supérieure de la plage. Ce test est effectué si le groupe est plein ou si **Set_RL** est positionné sur 1.

SD_At_Lim (SAL)

Ce paramètre est positionné sur 1 (* Bas *) lorsque l'écart type des sous-groupes est inférieur ou égal à la limite inférieure de l'écart type. Il est positionné sur 2 (* Haut *) lorsque l'écart type des sous-groupes est supérieur ou égal à la limite supérieure. Ce test est effectué si le groupe est plein ou si **Set_SL** est positionné sur 1.

SampleBias (SAB)

Ce paramètre est positionné sur 1 (* Bas *) lorsque 7 moyennes courantes consécutives sont inférieures à la moyenne des moyennes. Il est positionné sur 2 (* Haut *) lorsque 7 moyennes courantes consécutives sont supérieures à la moyenne des moyennes.

Range_Bias (RAB)

Ce paramètre est positionné sur 1 (* Bas *) lorsque 7 plages courantes consécutives sont inférieures à la moyenne des plages. Il est positionné sur 2 (* Haut *) lorsque 7 plages courantes consécutives sont supérieures à la moyenne des plages.

SD _Bias (SDB)

Ce paramètre est positionné sur 1 (* Bas *) lorsque 7 écarts types courants consécutifs sont inférieurs à la moyenne des écarts types. Il est positionné sur 2 (* Haut *) lorsque 7 écarts types courants consécutifs sont supérieurs à la moyenne des écarts types.

Sample _Trend (SAT)

Ce paramètre est positionné sur 1 (* Bas *) lorsque 7 moyennes courantes consécutives dérivent vers le bas. Il est positionné sur 2 (* Haut *) lorsque 7 moyennes courantes consécutives dérivent vers le haut.

Range_Trend (RAT)

Ce paramètre est positionné sur 1 (* Bas *) lorsque 7 plages courantes consécutives dérivent vers le bas. Il est positionné sur 2 (* Haut *) lorsque 7 plages courantes consécutives dérivent vers le haut.

SD_Trend (SDT)

Ce paramètre est positionné sur 1 (* Bas *) lorsque 7 écarts types courants consécutifs dérivent vers le bas. Il est positionné sur 2 (* Haut *) lorsque 7 écarts types courants consécutifs dérivent vers le haut.

Mean _Dist (MD)

Ce paramètre est positionné sur 1 si, sur 25 sous-groupes, plus de 92 % ou moins de 40 % des moyennes des sous-groupes se situent dans le tiers médian de la répartition. Lorsque plus de 92 % se situent dans le tiers médian, l'indicateur est positionné sur **Lev_92** et, lorsque moins de 40 % se situent dans le tiers médian, il est positionné sur **Lev_40**.

Cp _At_Cps (CAC)

Ce paramètre est positionné sur 1 (* On *) lorsque la capacité calculée du procédé est inférieure ou égale à la limite spécifiée. Ce test est effectué lorsque le groupe est plein.

Cpku _At_Cps (CUC)

Ce paramètre est positionné sur 1 (* On *) lorsque la capacité calculée supérieure du procédé est inférieure ou égale à la limite spécifiée. Ce test est effectué lorsque le groupe est plein.

Cpkl_At_Cps (CLC)

Ce paramètre est positionné sur 1 (* On *) lorsque la capacité calculée inférieure du procédé est inférieure ou égale à la limite spécifiée. Ce test est effectué lorsque le groupe est plein.

Current_Samp (CS)

Ce paramètre est le nombre d'échantillons depuis le dernier démarrage à froid ou redémarrage du bloc fonction.

Error_No (ERN)

Ce paramètre est le numéro d'erreur indiquant une incohérence dans l'entrée du bloc fonction ou dans un paramètre calculé du bloc fonction.

0	Normal ;
1	Erreur de la spécification de synchronisation ;
10	**Sub_Size** trop petit ou trop grand ;
100	**Group _Size** trop petit ou trop grand ;
1000	Limites de spécification **USL, LSL** incompatibles ;
10000	Ecart type calculé trop petit pour les calculs ultérieurs.

Tout nombre compris entre ces valeurs indique une combinaison de ces erreurs. Le bloc fonction continuera à fonctionner partiellement pour les deux dernières erreurs.

Attributs des paramètres

Nom	Mnémonique	Type	Démarrage à froid	Câblable	Accès en lecture	Accès en écriture	Informations propres au type	
Input	IN	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Variable
Enable	EN	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Restart	RE	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Group_Size	GS	DINT	10	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 125 Fixée à 1 Variable
Group_Time	GT	TIME	15m	Oui	Oper	Oper	Néant	
Sub_Size	SS	DINT	5	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 10 Fixée à 2 Variable
Sub_Time	ST	TIME	5s	Oui	Oper	Oper	Néant	
Cps	CPS	REAL	1.333	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Néant
USL	USL	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
LSL	LSL	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Set_CL	SCL	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Set_RL	SRL	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Set_SL	SSL	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Target	T	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée

Tableau 21-1 Attributs des paramètres d'entrée SPC

Nom	Mnémonique	Type	Démarrage à froid	Câblable	Accès en lecture	Accès en écriture	Informations propres au type	
New_Limits	NL	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Sub_Limits	SL	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Update_Limit	UL	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Moving_Win	MW	BOOL	No (0)	Oui	Oper	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)

Tableau 21-1 Attributs des paramètres d'entrée (suite)

Nom	Mnémonique	Type	Démarrage à froid	Câblable	Accès en lecture	Accès en écriture	Informations propres au type	
UCL	UCL	REAL	0		Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
LCL	LCL	REAL	0		Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
URL	URL	REAL	0		Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
LRL	LRL	REAL	0		Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
USDL	USD	REAL	0		Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
LSDL	LSD	REAL	0		Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée

Tableau 21-2 Attributs des paramètres d'entrée/sortie SPC

Nom	Mnémonique	Type	Démarrage à froid	Accès en lecture	Informations propres au type	
Group_Status	GRP	ENUM	NotFull(0)	Oper	Chaînes	NotFull (0) Pending (1) Full (2)
Run_Mean	RM	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Run_Range	RR	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Run_Std_Dev	RSD	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Est_Std_Dev	ESD	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Mean_Bar	MB	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Range_Bar	RB	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Std_Dev_Bar	STD	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Cp	CP	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Néant
Cpku	CPU	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Néant
Cpkl	CPL	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Néant
Preview_UCL	PUC	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée

Tableau 21-3 Attributs des paramètres de sortie SPC

Nom	Mnémonique	Type	Démarrage à froid	Accès en lecture	Informations propres au type	
Preview_LCL	PLC	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Preview_URL	PUR	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Preview_LRL	PLR	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Preview_USDL	PUS	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
Preview_LSDL	PLS	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
UnStable	US	BOOL	No (0)	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Mean_At_Lim	MAL	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
Range_At_Lim	RAL	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
SD_At_Lim	SAL	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
Sample_Bias	SAB	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
Range_Bias	RAB	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
SD_Bias	SDB	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
Sample_Trend	SAT	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)

Tableau 21-3 Attributs des paramètres de sortie SPC (suite)

Nom	Mnémonique	Type	Démarrage à froid	Accès en lecture	Informations propres au type	
Range_Trend	RAT	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
SD_Trend	SDT	ENUM	OK (0)	Oper	Chaînes	OK (0) Haut (1) Bas (2)
Mean_Dist	MD	ENUM	OK (0)	Oper	Chaînes	OK (0) 92_Lev (1) 40_Lev (2)
Cp_At_Cps	CAC	BOOL	No (0)	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Cpku_At_Cps	CUC	BOOL	No (0)	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Cpkl_At_Cps	CLC	BOOL	No (0)	Oper	Sens	Fixé à Oui (1) Fixé à Non (0)
Current_Samp	CS	DINT	0	Oper	Limite haute Limite basse Unités	Fixée à 2147483647 Fixée à 0 Variable
Error_No	ERN	DINT	0	Oper	Limite haute Limite basse Unités	Fixée à 11111 Fixée à 0 Variable

Tableau 21-3 Attributs des paramètres de sortie SPC (suite)

BLOC FONCTION SPC _EVENT

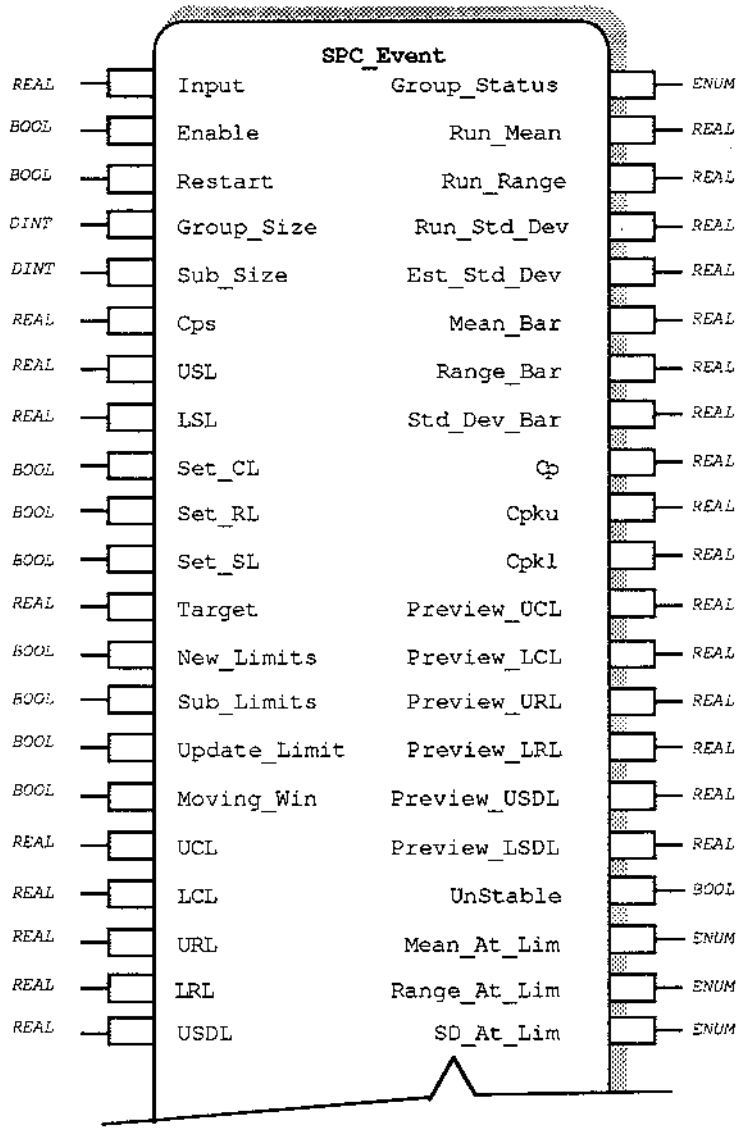


Figure 21-6 Schéma du bloc fonction SPC _Event

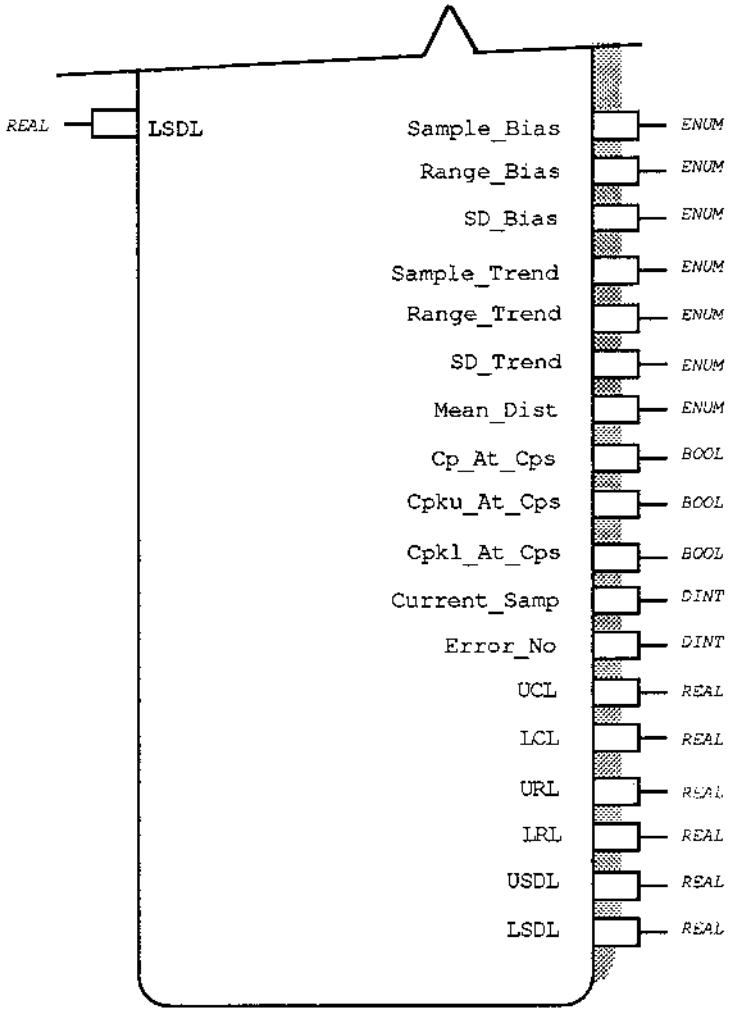


Figure 21-6 Schéma du bloc fonction SPC _Event (suite)

Description fonctionnelle

Les fonctions globales sont les mêmes que celles du bloc fonction **SPC**. **SPC_Event** est utilisé pour l'échantillonnage sur la base des événements et non sur la base du temps. Il n'y a par conséquent pas d'entrées **Group_Time** et **Sub_Time**.

Un échantillon est prélevé lorsque l'entrée **Enable** passe de 0 à 1. La mise en oeuvre doit se faire dans le programme utilisateur qui doit également réinitialiser Enable pour l'échantillon suivant.

Pour avoir des exemples, se reporter à la description du bloc fonction SPC.

Attributs du bloc fonction

Type : 68 30

Classe : .....STATISTIQUES

Tâche par défaut : ..... Task_2

Liste résumée : ..... Input, Enable Cp Group_Status

Description des paramètres

Paramètres d'entrée

Input (IN)

Mesure que le bloc fonction doit surveiller. Peut être une mesure effective ou peut être déduite d'une formule. La valeur doit être *continue* et non on / off.

Enable (EN)

Sur une transition de 0 à 1, le bloc fonction échantillonne **Input** et une autre mesure est ajoutée au sous-groupe qui convient. Le fait de laisser le signal **Enable** sur 1 n'a aucune autre influence. Le programme utilisateur doit réinitialiser **Enable** pour qu'il soit prêt pour l'échantillon suivant.

Restart (RE)

Toutes les fois que **Restart** passe de 0 (* Non *) à 1 (* Oui *) (c'est-à-dire un flanc ascendant), un démarrage à froid est effectué. Un flanc descendant n'a aucune influence sur le bloc fonction. Un redémarrage n'est possible que si le bloc fonction est activé.

Group _Size (GS)

Ce paramètre définit le nombre de sous-groupes d'un groupe. Il ne peut avoir ni une valeur inférieure à 1 ni une valeur supérieure à 125. Ce paramètre est en lecture seule au cours du démarrage à froid initial ou d'un redémarrage. La modification de ce paramètre au cours du fonctionnement du bloc fonction n'a aucune influence.

Sub_Size (SS)

Ce paramètre définit le nombre d'échantillons consécutifs prélevés dans chaque sous-groupe. Il ne peut être ni inférieur à 2 ni supérieur à 10.

Cps (CPS)

Ce paramètre est la capacité minimale acceptable du procédé. Une valeur de 1,33 est un minimum type.

Ce paramètre et tous les paramètres restants sont identiques à ceux du bloc fonction SPC standard.

Attributs des paramètres (cf. page 21-13).

Les paramètres sont les mêmes que ceux du bloc fonction SPC standard.

BLOC FONCTION HISTOGRAM

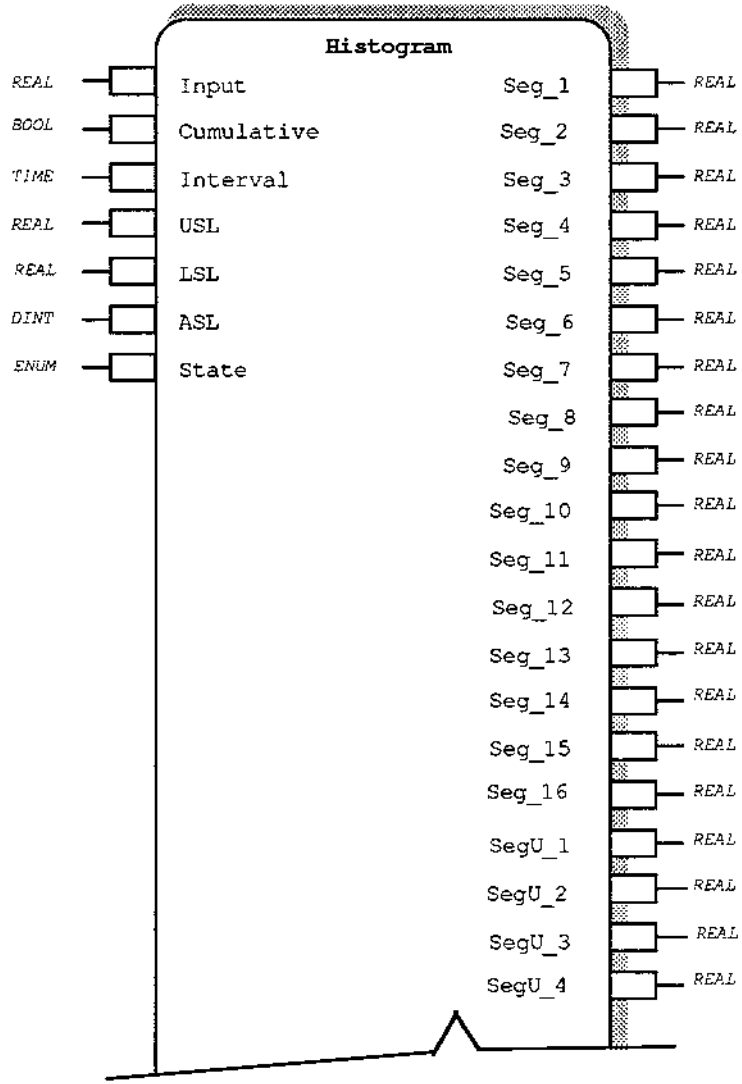


Figure 21-7 Schéma du bloc fonction Histogram

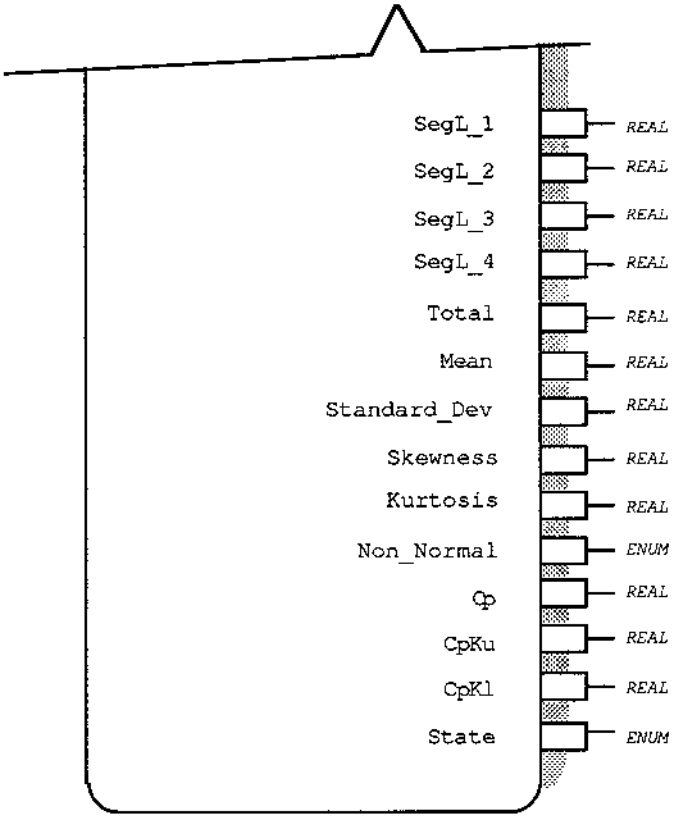


Figure 21-7 Schéma du bloc fonction Histogram (suite)

Description fonctionnelle

Le bloc fonction Histogram calcule la répartition d'une mesure REELLE. L'utilisateur spécifie les limites de spécification supérieure et inférieure de la variable et le bloc fonction divise cet intervalle en 16 segments équidistants à partir desquels la répartition de la variable est calculée.

Le bloc fonction a une mémoire dans laquelle il peut stocker 50 échantillons. A chaque changement des limites de spécification, toutes les informations (à l'exception des 50 derniers échantillons) sont effacées.

Le bloc fonction calcule la moyenne courante, l'écart type, le désalignement et la forme (arrondie, pointue, etc.) de la mesure qui commence au moment où l'échantillonnage a démarrée jusqu'à la mesure actuelle. Il est possible d'utiliser la variable ASL (Asymptotic Sample Length, longueur asymptotique d'échantillon) pour ramener *approximativement* la taille de cette fenêtre à celle des échantillons

ASL. L'approximation est due au fait que la fenêtre est exponentielle avec une constante de temps effective des échantillons ASL. Cela permet de suivre les moyennes à variation lente. La partie histogramme du bloc fonction *n'est pas* touchée par le choix d'**ASL**. Les valeurs types d'**ASL** sont comprises entre 100 et 1000 échantillons environ.

Le bloc fonction offre deux informations supplémentaires :

Il indique si la répartition des mesures échantillonnées est trop oblique, trop plate ou trop pointue pour être une répartition normale (gaussienne).

Une estimation de l'indice de capacité du procédé provenant des données de l'histogramme est également fournie. Bien que sommaire, cette estimation donne une assez bonne indication de l'indice de capacité effectif du procédé.

Il est possible d'utiliser le bloc fonction pour effectuer un échantillonnage régulier sur la base du temps ou de la demande. Il est également possible de réinitialiser le bloc à n'importe quel moment de l'exécution.

Une fois qu'il est déclenché, le bloc fonction recompile les nouvelles sorties sur 4 exécutions, c'est-à-dire 4 x l'intervalle de la tâche.

Afin d'obtenir l'histogramme d'une variable REELLE :

câbler la variable sur l'entrée.

positionner l'intervalle sur une longueur de temps souhaitée (*doit* être égal à au moins 5 fois l'intervalle de la tâche) et positionner **State** sur **SmpCont** ou sur **Demand** chaque fois qu'un échantillon est nécessaire. . Dans ce dernier cas, State revient à OK une fois que les calculs ont été effectués.

précharger **USL** et **LSL**. A chaque modification de ces valeurs, seuls les 50 derniers échantillons sont conservés.

si un abandon *continu* est nécessaire, il est possible de l'obtenir en positionnant **ASL** (Asymptotic Sample Length) sur la valeur souhaitée (100 échantillons par exemple). Cela signifie que la fenêtre effective des données pour la moyenne courante, l'écart type, etc. est de 100 échantillons. Le choix d'**ASL** n'a aucune influence sur les valeurs de fréquence dans les segments.

Il faut noter que, lorsque **Cumulative** est sur 1, le bloc fonction produit le totale cumulé (comptage effectif) des fréquences dans chaque segment. Dans le cas contraire, c'est la fréquence relative (c'est-à-dire le comptage effectif divisé par le nombre total d'échantillons prélevés) qui est donnée.

Attributs du bloc fonction

Type: 68 20

Classe : .....STATISTIQUES

Tâche par défaut : ..... Task_2

Liste résumée : ..... Input Cumulative Interval State

Description des paramètres

Paramètres d'entrée

Input (IN)

Entrée qui est échantillonnée régulièrement ou échantillonnée sur demande, selon la valeur de l'état.

Cumulative (C)

Lorsque ce paramètre est positionné sur Oui (0), chaque segment contient le nombre cumulé d'échantillons et le total des segments = Total.

Lorsque ce paramètre est positionné sur Non (1), chaque segment contient le nombre relatif d'échantillons (cumulé/total) et le total des segments = 1,00.

Interval (INT)

Intervalle de temps entre les échantillons provenant de la mesure dans le mode d'échantillonnage continu. La valeur minimale d'**Interval** est égale à 5 fois l'intervalle de la tâche. Si l'intervalle de la tâche est supérieur à une seconde, l'intervalle minimal est l'intervalle de la tâche.

USL (USL)

Niveau supérieur de spécification pour la mesure. Le fait de modifier **USL** au cours du fonctionnement du bloc fonction provoque la réalisation d'un nouveau calcul des segments et de la répartition avec les 50 derniers échantillons de données conservés en interne dans une mémoire tampon circulaire.

LSL (LSL)

Niveau inférieur de spécification pour la mesure. Le fait de modifier **LSL** au cours du fonctionnement du bloc fonction provoque la réalisation d'un nouveau calcul des segments et de la répartition avec les 50 derniers échantillons de données conservés en interne dans une mémoire tampon circulaire. Il y a 16 segments équidistants entre **USL** et **LSL** et 4 segments de chaque côté en dehors des limites.

ASL (ASL)

Longueur asymptotique d'échantillon utilisée pour l'abandon exponentiel des données dans l'histogramme. La fenêtre effective de données utilisées a une longueur égale à celle des échantillons **ASL**. L'abandon a uniquement un effet sur les valeurs de la moyenne, de l'écart type, du désalignement et de la curtosis.

Paramètres d'entrée/sortie

State (S)

Paramètre de type énumération qui peut prendre les valeurs suivantes :

0	Disable
1	SmpCont
2	Demand
3	Pending
4	OK
5	Init

Si **State** est positionné sur **SampCont**, le bloc échantillonne **Input** à intervalles réguliers définis par **Interval**. Si **State** est positionné sur **Demand**, le bloc fera passer **State** à **OK** une fois le calcul achevé, prêt pour l'échantillon suivant. Au cours du calcul, le bloc positionne **State** sur **Pending** et aucun échantillonnage de la mesure ne peut avoir lieu tant que les calculs ne sont pas terminés. Si l'utilisateur positionne **State** sur **Pending** alors qu'il n'est pas en attente, le bloc revient à **OK** et vice versa. Lors du positionnement de **State** sur **Init**, le bloc fonction réinitialise la production de l'histogramme.

Le calcul est réparti sur 4 cycles d'exécution, c'est-à-dire 4 x l'intervalle de la tâche.

Paramètres de sortie

Seg (S1)

Fréquence du segment 1 de l'histogramme.

Seg (S2)

Fréquence du segment 2 de l'histogramme.

etc.

Seg (S16)

Fréquence du segment 16 de l'histogramme.

SegU_1 (SU1)

Fréquence de l'ensemble des mesures situées en dehors d'USL et inférieures à **USL** +(**USL-LSL**)/16.

SegU_2 (SU2)

Fréquence de l'ensemble des mesures situées en dehors d'**USL** +(**USL-LSL**)/16 et inférieures à **USL** +2*(**USL-LSL**)/16.

SegU_3 (SU3)

Fréquence de l'ensemble des mesures situées en dehors d'**USL** +2*(**USL-LSL**)/16 et inférieures à **USL** + 3*(**USL-LSL**)/16.

SegU_4 (SU4)

Fréquence de l'ensemble des mesures situées en dehors d'**USL** +3*(**USL-LSL**)/16.

SegL_1 (SL1)

Fréquence de l'ensemble des mesures situées en dehors d'**LSL** - 3*(**USL-LSL**)/16.

SegL_2 (SL2)

Fréquence de l'ensemble des mesures supérieures à **LSL** - 3*(**USL-LSL**)/16 et inférieures à **LSL** - 2*(USL-LSL)/16.

SegL_3 (SL3)

Fréquence de l'ensemble des mesures supérieures à **LSL** - 2*(**USL-LSL**)/16 et inférieures à **LSL** - (**USL-LSL**)/16.

SegL (SL4)

Fréquence de l'ensemble des mesures supérieures à **LSL** - (**USL-LSL**)/16 et inférieures à **LSL**.

Total (TOT)

Nombre total de points de données collectés.

Mean (M)

Moyenne courante de la séquence **Input**.

Standard _Dev (STD)

Ecart type courant de la séquence **Input**.

Skewness (SK)

Désalignement courant de la séquence **Input**.

Kurtosis (K)

Curtosis courante de la séquence **Input**.

Non _Normal (NN)

Positionné sur **Normal** en l'absence d'indication positive d'anomalie, positionné sur **Skew** s'il y a une indication de désalignement, positionné sur **Kurtic** s'il y a une indication de curtosis haute ou basse et positionné sur S s'il y a une indication de désalignement ou de curtosis anormal(e).

Cp (CP)

Indice estimé de capacité du procédé reposant sur les segments de l'histogramme.

CpKu (CKU)

Indice estimé supérieur de capacité du procédé reposant sur les segments de l'histogramme.

CpKl (CKL)

Indice estimé inférieur de capacité du procédé reposant sur les segments de l'histogramme.

Attributs des paramètres

Nom	Mnémonique	Type	Démarrage à froid	Câblable	Accès en lecture	Accès en écriture	Informations propres au type	
Input	N	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Variable
Cumulative	C	BOOL	Oui (0)	Oui	Oper	Oper	Sens	Fixée à Non (1) Fixée à Oui (0)
Interval	INT	TIME	0ms	Oui	Oper	Oper	Néant	
USL	USL	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
LSL	LSL	REAL	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100,000 Comme l'entrée
ASL	ASL	DINT	0	Oui	Oper	Oper	Limite haute Limite basse Unités	Fixée à 12147483647 Fixée à 2 Variable

Tableau 21-4 Attributs des paramètres d'entrée Histogram

Nom	Mnémonique	Type	Démarrage à froid	Câblable	Accès en lecture	Accès en écriture	Informations propres au type	
State	S	ENUM	Disable (0)		Oper	Oper	Chaînes	Disable (0) SmpCnt (1) Demand (2) Pending (3) OK (4) Init (5)

Tableau 21-5 Attributs des paramètres d'entrée/sortie Histogram

Nom	Mnémonique	Type	Démarrage à froid	Accès en lecture	Informations propres au type	
Seg_1	S1	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
Seg_2	S2	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
			etc.			
			etc.			
Seg_16	S16	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegU_1	SU1	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegU_2	SU2	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegU_3	SU3	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegU_4	SU4	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegL_1	SL1	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegL_2	SL2	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegL_3	SL3	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
SegL_4	SL4	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant

Tableau 21-6 Bloc fonction des paramètres de sortie Histogram

Nom	Mnémonique	Type	Démarrage à froid	Accès en lecture	Informations propres au type	
Total	TOT	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à 0 Néant
Mean	M	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Comme l'entrée
Standard_Dev	STD	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Comme l'entrée
Skewness	SK	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Néant
Kurtosis	K	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Néant
Non_Normal	NN	ENUM	Normal (0)	Oper	Chaînes	Normal (0) Skew (1) Kurtic (2) S_and_K (3)
Cp	CP	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Néant
CpKu	CKU	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Néant
CpKl	CKL	REAL	0	Oper	Limite haute Limite basse Unités	Fixée à 100,000 Fixée à -100000 Néant

Tableau 21-6 Bloc fonction des paramètres de sortie Histogram (suite)

BLOC FONCTION STATISTICS

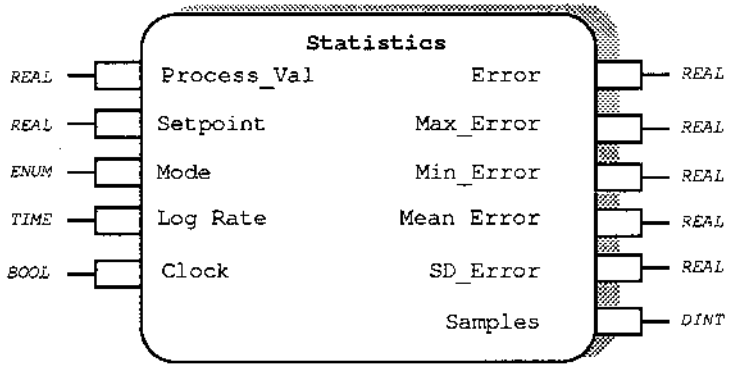


Figure 21-8 Schéma du bloc fonction Statistics

Description fonctionnelle

Ce bloc fonction fournit des informations statistiques sur les différences entre une valeur Process_Val et une valeur Setpoint. Cela peut par exemple s'appliquer aux entrées Process_Val et Setpoint d'une boucle de régulation, pour donner des informations supplémentaires au personnel chargé de l'exploitation par l'intermédiaire d'une interface utilisateur sans mettre en oeuvre toute la complexité de ce que l'on entend habituellement par le terme SPC (contrôle statistique).

Attributs du bloc fonction

Type : 68 38

Classe : .....STATISTIQUES

Tâche par défaut : ..... Task_2

Liste résumée : .....Process_Val, Setpoint, Error, SD_Error

Mémoire nécessaire : ..... 68 octets

Description des paramètres

Process_Val (PV)

Valeur de la variable value, soit directe soit dérivée, qui doit être suivie et comparée à **Setpoint.**

Setpoint (SP)

Valeur essentiellement statique par rapport à laquelle **Process_Val** est examinée pour **Error.**

Mode (M)

Le paramètre mode sert à réguler le fonctionnement du bloc fonction :

Reset : toutes les sorties sont réinitialisées et les intégrateurs statistiques sont eux aussi réinitialisés. Plus aucun calcul n'est effectué.

Hold : les sorties conservent leur état actuel et tous les intégrateurs statistiques internes sont conservés

RunLog : le bloc fonction exécute tous les calculs mais uniquement sur les échantillons provenant de **Process_Val** et **Setpoint** à des intervalles déterminés par **Log_Rate.** Le premier échantillon est prélevé après un intervalle **Log_Rate** après le démarrage du mode **RunLog**.

RunClk : le bloc fonction exécute tous les calculs mais uniquement sur les échantillons provenant de **Process_Val** et **Setpoint** à des intervalles déclenchés par l'entrée **Clock**. Les échantillons sont prélevés lors du cycle d'exécution du bloc fonction lorsque l'entrée **Clock** passe de l'état faux à l'état vrai.

RunCont le bloc fonction exécute tous les calculs mais uniquement sur les échantillons provenant de **Process_Val** et **Setpoint** à des intervalles définis par l'intervalle de tâche du bloc fonction.

Log_Rate (R)

Fréquence d'échantillonnage utilisée lorsque le bloc fonction est en mode **RunLog**.

Clock (CLK)

Déclencheur servant à provoquer le prélèvement d'un échantillon lorsqu'on est en mode **RunClk** . Le programme utilisateur doit le ramener de On(1) à Off(0) car le bloc fonction n'effectue pas cette opération.

Error (ERR)

Différence entre les échantillons les plus récents de **Process_Val** et **Setpoint.** Elle n'est pas nécessairement égale à la difference entre les valeurs actuelles des entrées **Process_Val** et **Setpoint.** L'obtention des entrées actuelles est régie par l'entrée **Mode** .

Max_Error (MXE)

Valeur maximale d'**Error** enregistrée depuis la dernière réinitialisation du bloc fonction.

Min_Error (MNE)

Valeur minimale d'**Error** enregistrée depuis la dernière réinitialisation du bloc fonction.

Mean_Error (MN)

Moyenne arithmétique de l'ensemble des valeurs d'**Error** enregistrées depuis la dernière réinitialisation du bloc fonction.

SD_Error (SDE)

Ecart type de l'ensemble des valeurs d'**Error** enregistrées depuis la dernière réinitialisation du bloc fonction. Calculé d'après l'équation des petits échantillons pour l'écart type :

$$SD\_Error = \sqrt{\left(\frac{\Sigma(Error)^2}{n-1} - \frac{(\Sigma Error)^2}{n(n-1)}\right)}$$

Samples

Nombre d'échantillons prélevés depuis la dernière réinitialisation du bloc fonction. Ce nombre d'échantillons est l'ensemble de valeurs sur lequel reposent les sorties actuelles.

Attributs des paramètres

Nom	Mnémonique	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Process_Val	PV	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Setpoint	SP	REAL	0	Oper	Oper	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Mode	M	ENUM	Reset (0)	Oper	Oper	Cf. liste des paramètres	
Log_Rate	R	TIME	1s	Oper	Oper	Limite haute Limite basse	23d23h59m59s999ms 0
Clock	CLK	BOOL	Off (0)	Oper	Block	Sens	Off (0) On (1)
Error	ERR	REAL	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Max_Error	MXE	REAL	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Min_Error	MNE	REAL	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Mean_Error	MN	REAL	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
SD_Error	SDE	REAL	0	Oper	Block	Limite haute Limite basse	+3·402823E+38 -3·402823E+38
Samples		DINT	0	Oper	Block	Limite hautes Limite basses	2147483647 0

Tableau 21-7 Attributs des paramètres Statistics

Chapitre 22

LOGGERS

Edition 1

Toutes les informations relatives à cette classe de blocs fonctionnels figureront dans la prochaine révision du présent manuel.

Ces blocs sont en cours de développement.

Chapitre 23

INSTRUMENTATION DEPORTEE

Edition 3

Présentation

Cette classe de blocs fonctions offre une interface standard avec les appareils déportés reliés au PC3000 par une liaison série. Des blocs fonctions sont fournis pour assurer l'interface avec des régulateurs de température discrets, des commandes moteur alternatif et continu et des gradateurs de puissance à thyristors utilisant le driver maître EI Bisync. L'ancien bloc fonction GenRmtInst est obsolète et il est recommandé de ne pas l'utiliser dans les nouvelles applications.

Ces blocs sont actuellement en cours de développement.

BLOC FONCTION RMTCONTROLR

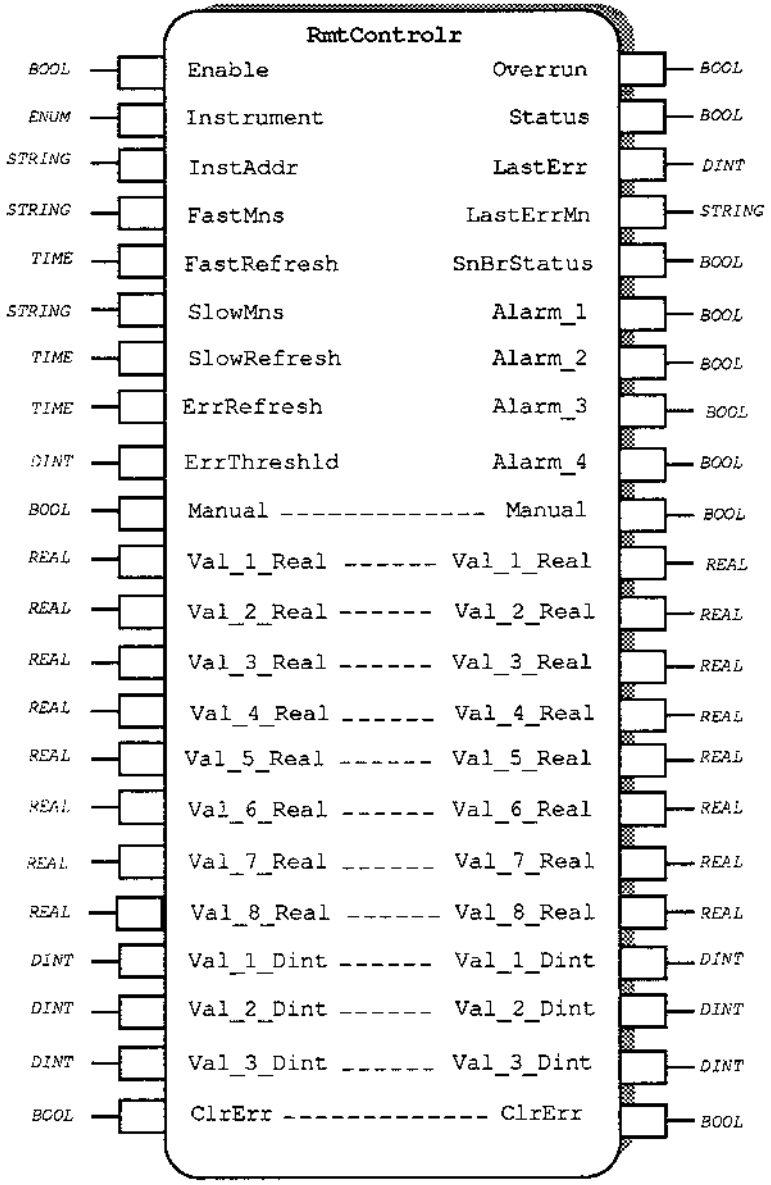


Figure 23-1 Schéma du bloc fonction RmtControlr

Description fonctionnelle.

Ce bloc sert à effectuer la liaison avec un des régulateurs Eurotherm (808, 94, 818, 902, 900EPC ou série 2000). Il donne accès à un maximum de 8 paramètres à virgule flottante et de 3 paramètres entiers à deux vitesses de scrutation : une rapide (doit être utilisée pour PV) et une lente (doit être utilisée pour les paramètres SP, PID, etc).

Ce bloc offre aussi un traitement des erreurs afin de minimiser l'interrogation des appareils qui ne sont pas en liaison et de rétablir les communications après rétablissement de la liaison.

Dans les applications à déclarations multiples du bloc, il serait prudent de décaler le réglage d'Enable sur les blocs afin qu'ils démarrent un par un. Chaque bloc permettra une seule transaction de communications à la fois. Q_Space dans le maître BiSync associé montrera l'état de la liaison.

Attributs du bloc fonction

Type DC21
Classe ..... REMOTE_INST
Tâche par défaut ..... Task_2
Liste résumée ..... InstAddr, Enable, Status, Overrun
Mémoire nécessaire..... 740 octets

Description des paramètres

Enable

Le fait de positionner cette broche sur 1 (Enable) provoque l'analyse syntaxique des paramètres définis par les mnémoniques dans les chaînes FastMns et SlowMns et le démarrage de l'interrogation de l'appareil, mnémonique par mnémonique. Status passe uniquement à 1 (Go) après que tous les mnémoniques ont été interrogés une fois avec succès.

> N.B. : les mnémoniques sont analysés UNE SEULE FOIS lorsque la broche Enable passe de 0 (Disable = désactivé) à 1 (Enable = activé). Pour que le changement d'un mnémonique soit possible, Enable doit être positionné sur 0 (Disable), la chaîne de mnémoniques doit être changée puis Enable doit être positionné sur 1 (Enable).

Le fait de positionner cette broche sur 0 (Disable) provoque l'arrêt de l'interrogation. Une fois que le bloc est désactivé, Status passe à (NOGO), indiquant que l'interrogation s'est arrêtée.

Si un paramètre est en cours d'interrogation ou d'écriture lors de la désactivation du bloc, la transaction en cours se termine avant que la désactivation entre en vigueur.

Instrument

Numéro de modèle du régulateur Eurotherm en cours d'adressage. Les types pris en charge sont les suivants :

0	(Undef)	- type quelconque
1	(E818)	- Eurotherm 818/815
2	(E902)	- Eurotherm 902/903/904
3	(E900EPC)	- Eurotherm 900EPC
4	(E94)	- Eurotherm 94c
5	(E808)	- Eurotherm 808/847
6	(E2000)	- Eurotherm série 2000

Dans le mode 0 (Undef), aucun paramètre supplémentaire n'est interrogé et les broches **Manual, SnBrStatus** et **Alarm_1** à **Alarm_4** ne sont pas prises en charge.

Dans le mode 1 (E818), le paramètre SW est ajouté à l'interrogation rapide et les fonctions suivantes sont prises en charge :

Manual	(SW bit 15)
SnBrStatus	(SW bit 1)
Alarm_1	(SW bit 10)
Alarm_2	(SW bit 8)

Alarm_3 et Alarm_4 ne sont pas utilisées.

Dans le mode 2 (E902), le paramètre SW est ajouté à l'interrogation rapide et les fonctions suivantes sont prises en charge :

Manual	(SW bit 15)
SnBrStatus	(SW bit 1)
Alarm_1	(SW bit 10)
Alarm_2	(SW bit 8)

Alarm_3 and Alarm_4 ne sont pas utilisées.

Dans le mode 3 (E900EPC), les paramètres WL et WA sont ajoutés à l'interrogation rapide et les fonctions suivantes sont prises en charge :

Manual	(WL bit 0)
SnBrStatus	(WL bit 7 & bit 15)
Alarm_1-4	(WA bits 8,9,10 et 11 respectivement pour la boucle 1). (WA bits 20, 21, 22 et 23 respectivement pour la boucle 2).

Dans le mode 4 (E94), le paramètre SW est ajouté à l'interrogation rapide et les fonctions suivantes sont prises en charge :

SnBrStatus	(SW bit 1 & bit 3)
Alarm_1	(SW bit 10)
Alarm_2	(SW bit 8)

Alarm_3, Alarm_4 et Manual ne sont pas utilisées.

Dans le mode 5 (E808), le paramètre SW est ajouté à l'interrogation rapide et les fonctions suivantes sont prises en charge :

Manual	(SW bit 15)
SnBrStatus	(SW bit 1)
Alarm_1	(SW bit 11)
Alarm_2	(SW bit 9)
Alarm_3	(SW bit 7)

Alarm_4 n'est pas utilisée.

Dans le mode 6 (E2000), les paramètres mA et FS sont ajoutés à l'interrogation rapide et les fonctions suivantes sont prises en charge :

Manual	(mA bit 0)
SnBrStatus	(FS bit 5 & bit 6 & bit 7)
Alarm_1	(FS bit 0)
Alarm_2	(FS bit 1)
Alarm_3	(FS bit 2)
Alarm_4	(FS bit 3)

InstAddr

Adresse de l'appareil. Par exemple, '0A12 ' représente

'0' =	emplacement 0 (LCM)	
'A' =	port A	
'1'	=	ID de groupe 1
'2'	=	ID d'unité
' '	=	ID de voie (espace = néant)

N;B; : il doit y avoir un caractère à la position ID de voie. Si besoin est, il faut utiliser un caractère "espace" pour indiquer qu'il n'y a pas d'ID de voie.

FastMns

Liste de mnémoniques à deux caractères séparés par une virgule à interroger à la vitesse de rafraîchissement rapide. Prévue pour les paramètres qui, comme PV, changent en permanence.

Les blocs peuvent récupérer un maximum de 8 paramètres réels et 3 paramètres entiers, interrogations rapide et lente confondues. Les paramètres supplémentaires ne sont pas pris en compte.

Se reporter à la broche Appareil pour voir les paramètres supplémentaires en fonction du type.

Les paramètres correspondent à Val_<n>_Real/Dint dans l'ordre où ils apparaissent dans les chaînes FastMns puis SlowMns. Les paramètres impliqués par le type Appareil précèdent FastMns.

Il est possible d'ajouter un suffixe supplémentaire à un caractère à chaque mnémonique pour indiquer la syntaxe bisync à utiliser (cf. l'adresse **Rmt_Real/Rmt_Dint.** dans le chapitre 12 pour avoir plus de détails). Lorsqu'aucune syntaxe n'est spécifiée, la valeur par défaut est 'f', sauf lorsqu'Appareil=4 (94) : dans ce cas, la valeur est 'g'. L'utilisation d'un caractère à syntaxe entière (par exemple Z,B,X ou Y) distingue les paramètres réels des paramètres entiers.

A chaque vitesse d'interrogation, les paramètres réels sont interrogés avant les entiers. Si le type d'appareil est 902, FastMns = 'PV, IIX, OP' et SlowMns = 'SL, OSX, SP' puis l'ordre d'interrogation serait PV, OP, SW, II pour l'interrogation rapide et SL, SP, OS pour l'interrogation lente.

FastRefresh

Vitesse de rafraîchissement pour l'interrogation des mnémoniques "rapides".

SlowMns

Liste des mnémoniques à deux caractères séparés par une virgule à interroger à la vitesse de rafraîchissement lente (cf. FastMns).

SlowRefresh

Vitesse de rafraîchissement pour l'interrogation des mnémoniques "lents".

ErrRefresh

Vitesse de rafraîchissement à utiliser lorsque l'interrogation ne fonctionne pas correctement.

Cette vitesse continuera à être utilisée jusqu'à ce que tous les paramètres aient été lus correctement une fois : ensuite, l'interrogation continuera aux vitesses d'interrogation rapide et lente et les blocs Status passeront à 1 (Go).

Il faut noter que la vitesse d'interrogation en cas d'erreur ne commence pas tant que le nombre d'erreurs spécifié par ErrThreshld ne s'est pas produit.

ErrThreshld

Nombre d'erreurs autorisé avant que les blocs **Status** passent à 0 (NOGO) et que l'interrogation passe à la vitesse **ErrRefresh**. Les interrogations rapide et lente sont comptées séparément et le premier comptage qui dépasse ce seuil d'erreur déclenche la vitesse de rafraîchissement des erreurs.

Chaque comptage est supprimé séparément une fois qu'une interrogation est terminée correctement.

Overrun

Overrun passe à 1 (vrai) lorsqu'une interrogation antérieure n'est pas terminée alors que la suivante doit avoir lieu.

Si cela se produit, l'interrogation est sautée et les données peuvent être perdues ou retardées. Les dépassements de capacité risquent de se produire si une nouvelle tentative est effectuée et prend du temps ou lorsque les vitesses de rafraîchissement sont trop rapides pour que l'interrogation des paramètres puisse avoir lieu.

Status

Indication de l'état global de la liaison avec le régulateur.

Status passe à 1 (Go) lorsque tous les paramètres ont été interrogés une fois correctement.

Status passe à (NOGO) après ErrThreshld échecs de lecture des paramètres.

Il faut noter que l'interrogation rapide, lente ou d'erreur est arrêtée à la première défaillance, même si **ErrThreshld** n'a pas été atteint.

LastErr

Code d'erreur de paramètre déporté de la dernière erreur qui s'est produite. Se reporter aux erreurs EI Bisync_M dans le chapitre 3.

LastErrMn

Mnémonique à 2 caractères du dernier paramètre qui a provoqué une erreur, suivi du caractère de syntaxe utilisé pour le lire ou l'écrire.

SnBrStatus

Etat de rupture capteur de l'appareil.

Ce paramètre correspond à toutes les alarmes de défaut de rupture capteur, rupture boucle et de charge disponibles en fonction du type d'appareil sélectionné.

Alarm_1

Première alarme (si elle est utilisée). (Se reporter à l'appareil).

Alarm_2

Deuxième alarme (si elle est utilisée). (Se reporter à l'appareil).

Alarm_3

Troisième alarme (si elle est utilisée). (Se reporter à l'appareil).

Alarm_4

Quatrième alarme (si elle est utilisée). (Se reporter à l'appareil).

Manual

Mode Auto/Manuel de l'appareil (s'il est utilisé). La valeur de cette broche correspond au bit auto/manuel dans le paramètre qui convient. Pour avoir des détails sur le bit ou sur le paramètre auquel il correspond, se reporter à l'appareil.

Pour les appareils de type 0 (Undef) et 4 (E94), la broche Manual n'est pas utilisée.

Si le type d'appareil est 1 (E818), 2 (E902), 3 (E900EPC) ou 5 (E808), une opération de lecture-modification-écriture sera effectuée sur le mot d'état correspondant toutes les fois que la valeur de Manual changera sur le bloc.

Pour l'appareil de type 6 (E2000), une écriture simple est effectuée sur le paramètre mA toutes les fois que la valeur de Manual est modifiée sur le bloc.

Val_1_Real

Premier paramètre réel.

Ce paramètre est interrogé à la vitesse d'interrogation demandée. Si la valeur du paramètre est changée, elle est écrite immédiatement à raison d'un maximum d'une fois par temps d'interrogation. Si le paramètre a été déjà écrit dans le temps d'interrogation, l'écriture sera retardée jusqu'à l'interrogation suivante. Si une écriture continue plus rapide est nécessaire, il faut augmenter la vitesse d'interrogation du paramètre à une valeur correcte.

Val_2_Real à Val_8_Real : comme Val_1_Real.

Val_1_Dint

Premier paramètre entier.

La stratégie d'interrogation et d'écriture est la même que pour les paramètres réels.

Val_2_Dint et Val_3_Dint : comme Val_1_Dint

ClrErr

Le positionnement de ce paramètre sur 1 (vrai) efface **LastErr** et **LastErrMn.** Lors de l'exécution du bloc, ce paramètre revient toujours à 0 (faux).

Les erreurs sont effacées avant le reste de l'exécution du bloc, ce qui fait qu'une erreur qui a lieu au cours de l'exécution où l'effacement est détecté reste sortie depuis le bloc.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Enable	**BOOL**	Disable	Oper	Oper	Sens	Disable (0) Enable (1)
Instrument	**ENUM**	E902	Oper	Oper	Sens	Undef (0) E818 (1) E902 (2) E900EPC (3) E94 (4) E808 (5) E2000 (6)
InstAddr	**STRING**	'0A00 '	Oper	Oper	Nombre maxi. de caractères 5	
FastMns	**STRING**	'PV,OP'	Oper	Oper	Nombre maxi. de caractères 43	
FastRefresh	**TIME**	2s	Oper	Oper	Limite haute Limite asse	2147483647 0
SlowMns	**STRING**	'SL'	Oper	Oper	Nombre maxi. de caractères 43	
SlowRefresh	**TIME**	20s	Oper	Oper	Limite haute Limite basse	123d23h59m59s999 0
ErrRefresh	**TIME**	1m	Oper	Oper	Limite haute Limite basse	123d23h59m59s999 0
ErrThreshld	**DINT**	1	Oper	Oper	Limite haute Limite basse	2147483647 1
Manual	**BOOL**	Auto	Oper	Oper	Sens	Auto (0) Manual (1)
Val_1_Real to Val_8_Real	**REAL**	0	Oper	Oper	Limite haute Limite basse	3.402823e+38 -3.402823e+38
Val_1_Dint to Val_3_Dint	**DINT**	0	Oper	Oper	Sens	2147483646 -2147483647
ClrErr	**BOOL**	False	Oper	Oper	Sens	False (0) True 1)
Overrun	**BOOL**	False	Oper	Block	Sens	False 0) True (1)
Status	**BOOL**	NOGO	Oper	Block	Sens	Go (1) NOGO (0)
LastErr	**DINT**	0	Oper	lock	Limite haute Limite basse	255 0

Tableau 23-1 Attributs des paramètres RmtControlr (suite)

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
LastErrMn	STRING	"	Oper	Block	Nombre maxi. de caractères 3	
SnBrStatus	BOOL	Off	Oper	Block	Sens	Off (0) On (1)
Alarm_1 to Alarm_4	BOOL	Off	Oper	Block	Sens	Off (0) On (1)

Tableau 23-1 Attributs des paramètres RmtControlr

BLOC FONCTION GENRMDRIVE

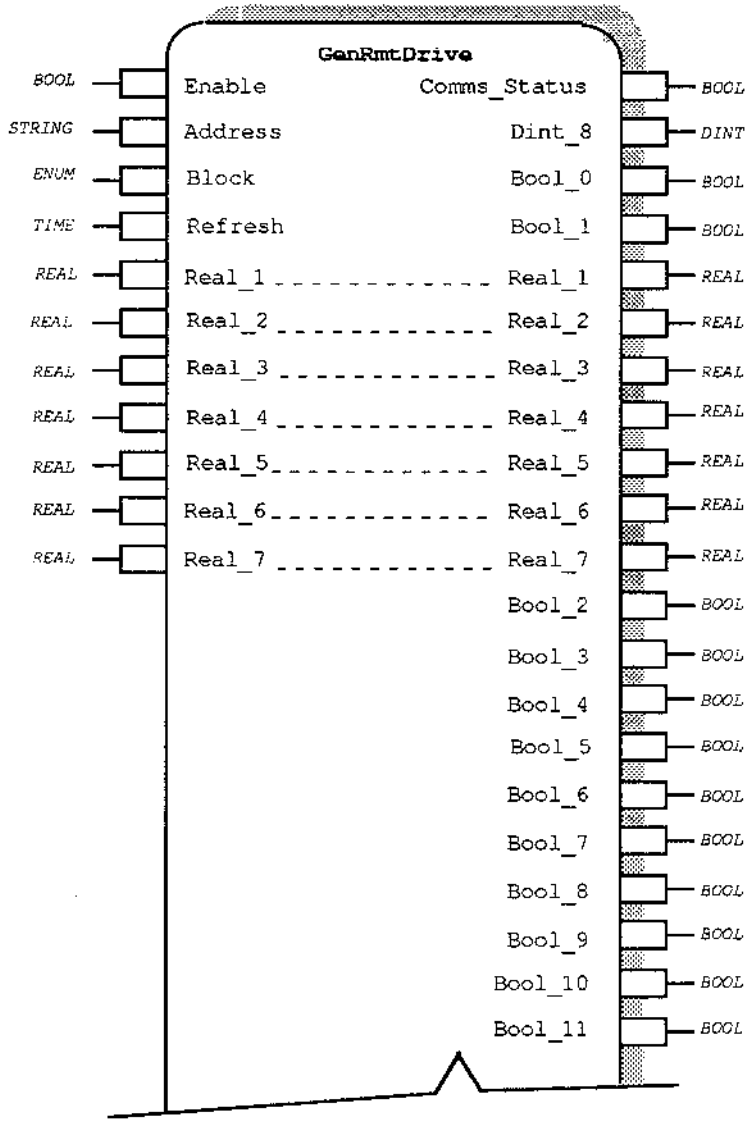


Figure 23-2 Schéma du bloc fonction Generic Remote Drive 584

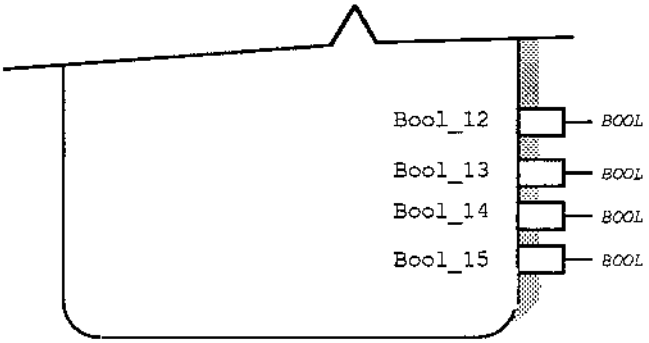


Figure 23-2 Schéma du bloc fonction Generic Remote Drive 584

Description fonctionnelle.

Bloc fonction de communications générique qui représente la syntaxe de bloc PNO utilisée dans les commandes moteur alternatif et continu d'Eurotherm Drives.

Des blocs fonctions spécifiques permettent aussi d'assurer l'interface avec les commandes moteur alternatif de la série 584 et les commandes moteur continu de la série 590. Ils offrent une interface simple avec les paramètres qui sont couramment nécessaires.

Attributs du bloc fonction

Type DC30
Classe ..... REMOTE_INSTR
Tâche par défaut ..... Task_2
Liste résumée ..... Enable, Address, Block, Comms_Status
Mémoire nécessaire ..... 1852 octets

Description des paramètres

Enable (EN)

Ce paramètre sert à activer et à désactiver la liaison de communications. Lorsqu'**Enable** est positionné sur **Enable (1)**, la liaison de communications est active et les paramètres seront interrogés.

Address (A)

Ce paramètre définit l'adresse de la commande moteur. La syntaxe de l'adresse est la suivante :

Port No. GID UID <space>

Par exemple, une adresse de 0B12<space> identifie une commande moteur dont GID = 1 et UID = 2, reliée au port B sur l'unité centrale PC3000.

Il faut également créer un bloc fonction maître EI Bisync pour chaque port de communications qui sera utilisé avec un bloc fonction d'appareil déporté.

Block (BLK)

Ce paramètre définit le numéro du bloc PNO qui est adressé. Il est possible de modifier ce numéro de bloc en ligne, ce qui permet d'utiliser un seul bloc fonction GenRmtDrive pour accéder à toute une série de blocs PNO.

Si l'un des paramètres d'un bloc n'est pas disponible, **Comms_Status** est positionné sur FAULT (0).

Refresh (R)

Cette entrée définit la vitesse à laquelle seront lus les paramètres provenant de la commande moteur déportée. Toutes les écritures dans la commande moteur ont lieu immédiatement lorsque la valeur d'une entrée change. Si la commande moteur est retirée de la liaison ou est mise hors tension, il faut positionner Enable sur Disable (0) ou Refresh sur une durée de rafraîchissement longue afin d'empêcher le bloc fonction maître EI Bisync de perdre du temps et de la largeur de bande de communications par des tentatives répétées continues.

Real_1 (R1) à Real_7 (R7)

Ces paramètres d'entrée-sortie montrent les valeurs actuelles des sept paramètres réels définis pour le bloc fonction PNO spécifié par l'entrée Block. Sur le PC3000, ces paramètres sont en lecture/écriture mais la commande moteur déportée peut empêcher les écritures dans certains paramètres.

Comms_Status (S)

Si tous les paramètres du bloc PNO sont interrogés avec succès, **Comms_Status** est positionné sur OK (1). Si des paramètres définis dans le bloc PNO ne sont pas disponibles ou si des erreurs de communications sont détectées, Comms_Status est positionné sur FAULT (1).

Dint_8

Ce paramètre montre la valeur de l'entier d'état associé au bloc PNO spécifié par l'entrée Block. Ce mot d'état est aussi scindé en seize bits séparés. Ce paramètre est en lecture seule. S'il est nécessaire d'écrire dans un bit, il faut utiliser un bloc fonction **Remote_Bool** ou **Remote_SW**.

Bool_0 (B0) à Bool_15 (B15)

Ces sorties montrent les valeurs des différents bits du mot d'état indiqué par la sortie **Dint_8.**

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Enable	BOOL	Disable (0)			Sens	Disable (0) Enable (1)
Address	STRING	'OAOO'			Maximum 5 caractères	
Block	ENUM	1 (1)			Sens	0 (0) 1 (1) 2 (2) 3 (3) 4 (4) 5 (5) 6 (6) 7 (7) 8 (8) 9 (9) A (10) B (11) C (12) D (13) E (14) F (15)
Refresh	TIME	10s			Limite haute Limite basse	123d23h59m59s999 0
Real_1 to Real_7	REAL	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Comms_Status	BOOL	FAULT (0)			Sens	FAULT (0) Ok (1)
Dint_8	DINT	0			Sens	Off (0) On (1)

Tableau 23-2 Attributs des paramètres GenRmDrive

BLOC FONCTION RMTDRIVE584

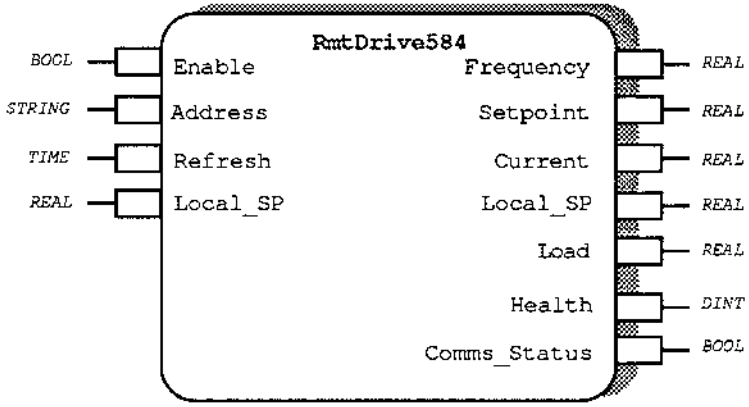


Figure 23-3 Schéma du bloc fonction Remote 584

Description fonctionnelle.

Ce bloc fonction fournit une interface simple avec une commande moteur alternatif Eurotherm Drives de la série 584 et donne accès aux paramètres les plus courants. Si d'autres paramètres sont nécessaires, il faut utiliser le bloc fonction **GenRmtDrive** pour accéder directement au numéro de bloc PNO contenant le paramètre nécessaire.

Des blocs fonctions spécifiques sont également disponibles pour assurer l'interface avec les commandes moteur alternatif de la série 584 et les commandes moteur continu de la série 590. Ces blocs fonctions offrent une interface simple avec les paramètres qui sont couramment nécessaires.

Attributs du bloc fonction

Type DC36
Classe ..... REMOTE_INSTR
Tâche par défaut ..... Task_2
Liste résumée ..... Enable, Address, Frequency, Setpoint
Mémoire nécessaire ..... 1396 octets

Description des paramètres

Enable (EN)

Ce paramètre sert à activer et à désactiver la liaison de communications. Lorsqu'**Enable** est positionné sur **Enable (1)**, la liaison de communications est active et les paramètres seront interrogés.

Address (A)

Ce paramètre définit l'adresse de la commande moteur. La syntaxe de l'adresse est la suivante :

Port No. GID UID <space>

Par exemple, une adresse de 0A34<space> identifie une commande moteur dont GID = 3 et UID = 4, reliée au port A sur l'unité centrale PC3000.

Il faut également créer un bloc fonction maître EI Bisync pour chaque port de communications qui sera utilisé avec un bloc fonction d'appareil déporté.

Refresh (R)

Cette entrée définit la vitesse à laquelle seront lus les paramètres provenant de la commande moteur déportée. Toutes les écritures dans la commande moteur ont lieu immédiatement lorsque la valeur d'une entrée change. Si la commande moteur est retirée de la liaison ou est mise hors tension, il faut positionner **Enable** sur **Disable (0)** ou **Refresh** sur une durée de rafraîchissement longue afin d'empêcher le bloc fonction maître EI Bisync de perdre du temps et de la largeur de bande de communications par des tentatives répétées continues.

Frequency (PV)

Cette sortie montre la sortie de fréquence actuelle de la commande moteur si **Comms_Status** est OK (1). Cette sortie correspond au mnémonique 0B et est en lecture seule. La valeur est lue sous la forme d'un pourcentage.

Setpoint (SP)

Cette sortie montre la consigne de vitesse actuelle de la commande moteur si **Comms_Status** est lue sous la forme d'un pourcentage.

Local_SP (SL)

Cette entrée/sortie montre la valeur actuelle de la consigne série si **Comms_Status** est OK (1). Ce paramètre correspond au mnémonique 26 et il est possible d'écrire dedans s'il a été autorisé en écriture dans la commande moteur. La valeur est exprimée sous la forme d'un pourcentage.

Current (I)

Cette sortie montre l'intensité du moteur de la commande moteur si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 08 et est en lecture seule. La valeur est exprimée sous la forme d'un pourcentage.

Load (L)

Cette sortie montre la charge du moteur de la commande moteur si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 09 et est en lecture seule. La valeur est exprimée sous la forme d'un pourcentage.

Health (H)

Cette sortie montre le mot d'état de la commande moteur si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 0F et est en lecture seule.

Comms_Status (S)

Si tous les paramètres sont lus sans erreur de communications, la sortie **Comms_Status** est positionnée sur OK (1). Si des erreurs de communications sont détectées, **Comms_Status** est positionné sur FAULT (0).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Enable	**BOOL**	Disable (0)			Sens	Disable (0) Enable (1)
Address	**STRING**	'OAOO'			Maximum 5 caractères	
Refresh	**TIME**	10s			Limite haute Limite basse	23d23h59m59s999 0
Real_1 to Real_7	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Comms_Status	**BOOL**	FAULT (0)			Sens	FAULT (0) Ok (1)
Dint_8	**DINT**	0			Sens	Off (0) On (1)

Tableau 23-3 Attributs des paramètres RmDrive 584

BLOC FONCTION RMTDRIVE590

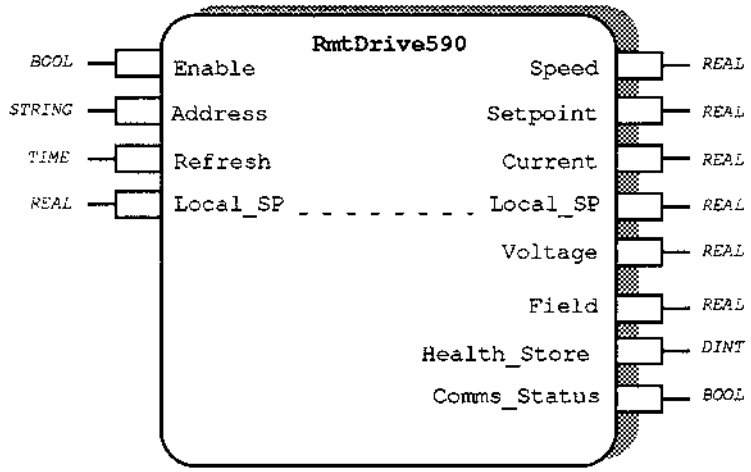


Figure 23-4 Schéma du bloc fonction Remote 590 Drive

Description fonctionnelle.

Ce bloc fonction offre une interface simple avec une commande moteur continu Eurotherm Drives de la série 590 et donne accès aux paramètres les plus courants. Si d'autres paramètres sont nécessaires, il faut utiliser le bloc fonction **GenRmtDrive** pour accéder directement au numéro de bloc PNO qui contient le paramètre nécessaire.

Attributs du bloc fonction

Type DC38
Classe ..... REMOTE_INSTR
Tâche par défaut ..... Task_2
Liste résumée ..... Enable, Address, Speed, Setpoint
Mémoire nécessaire ..... 1602 octets

Description des paramètres

Enable (EN)

Ce paramètre sert à activer et à désactiver la liaison de communications. Lorsqu'**Enable** est positionné sur **Enable (1)**, la liaison de communications est active et les paramètres seront interrogés.

Address (A)

Ce paramètre définit l'adresse de la commande moteur. La syntaxe de l'adresse est la suivante :

Port No. GID UID <space>

Par exemple, une adresse de 1C56<space> identifie une commande moteur dont GID = 5 et UID = 6, reliée au port C sur un ICM de l'emplacement 1 du rack PC3000.

Il faut également créer un bloc fonction maître EI Bisync pour chaque port de communications qui sera utilisé avec un bloc fonction d'appareil déporté.

Refresh (R)

Cette entrée définit la vitesse à laquelle seront lus les paramètres provenant de la commande moteur déportée. Toutes les écritures dans la commande moteur ont lieu immédiatement lorsque la valeur d'une entrée change. Si la commande moteur est retirée de la liaison ou est mise hors tension, il faut positionner **Enable** sur **Disable** (**0**) ou **Refresh** sur une durée de rafraîchissement longue afin d'empêcher le bloc fonction maître EI Bisync de perdre du temps et de la largeur de bande de communications par des tentatives répétées continues.

Speed (PV)

Cette sortie montre la vitesse actuelle de la commande moteur si **Comms_Status** est sur OK (1) Cette sortie correspond au mnémonique 0A (retour d'informations au sujet de la vitesse) et est en lecture seule.

Setpoint (SP)

Cette sortie montre la consigne de vitesse actuelle de la commande moteur si **Comms_Status** est sur OK (1). Cette sortie correspond au mnémonique 08 et est en lecture seule.

Local_SP (SL)

Cette entrée/sortie montre la valeur actuelle de la consigne de la liaison série si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 34 (entrée de rampe) et il est possible d'écrire dedans s'il a été autorisé en écriture dans la commande moteur.

Current (I)

Cette sortie montre l'intensité du moteur de la commande moteur si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 08 et est en lecture seule. La valeur est exprimée sous la forme d'un pourcentage.

Load (L)

Cette sortie montre la charge du moteur de la commande moteur si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 09 et est en lecture seule. La valeur est exprimée sous la forme d'un pourcentage.

Current (I)

Cette sortie montre l'intensité du moteur de la commande moteur si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique 0C (retour d'informations au sujet de l'intensité) et est en lecture seule.

Voltage (V)

Cette sortie montre la tension du moteur de la commande moteur si **Comms_Status** est sur OK (1). Cette sortie correspond au mnémonique 20 et est en lecture seule.

Field (F)

Cette sortie montre le retour d'informations au sujet du champ du moteur si **Comms_Status** est sur OK (1). Cette sortie correspond au mnémonique 0E et est en lecture seule.

Health_Store (HS)

Cette sortie montre l'enregistrement d'état de la commande moteur si **Comms_Status** est sur OK (1). Cette sortie correspond au mnémonique 1F et est en lecture seule.

Comms_Status (S)

Si tous les paramètres sont lus sans erreur de communications, la sortie **Comms_Status** est positionnée sur OK (1). Si des erreurs de communications sont détectées, **Comms_Status** est positionné sur FAULT (0).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Enable	**BOOL**	Disable (0)			Sens	Disable (0) Enable (1)
Address	**STRING**	'OAOO'			Maximum 5 caractères	
Refresh	**TIME**	10s			Limite haute Limite basse	23d23h59m59s999 0
Speed	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Setpoint	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Local_SP	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Current	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Voltage	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Load	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Field	**REAL**	0			Limite haute Limite basse	3.402823e+38 -3.402823e+38
Health_Store	**DINT**	0			Limite haute Limite basse	2147483646 -2147483647
Comms_Status	**BOOL**	FAULT (0)			Sens	FAULT (0) Ok (1)

Tableau 23-4 Attributs des paramètres RmDrive 590

BLOC FONCTION RMTTU1400

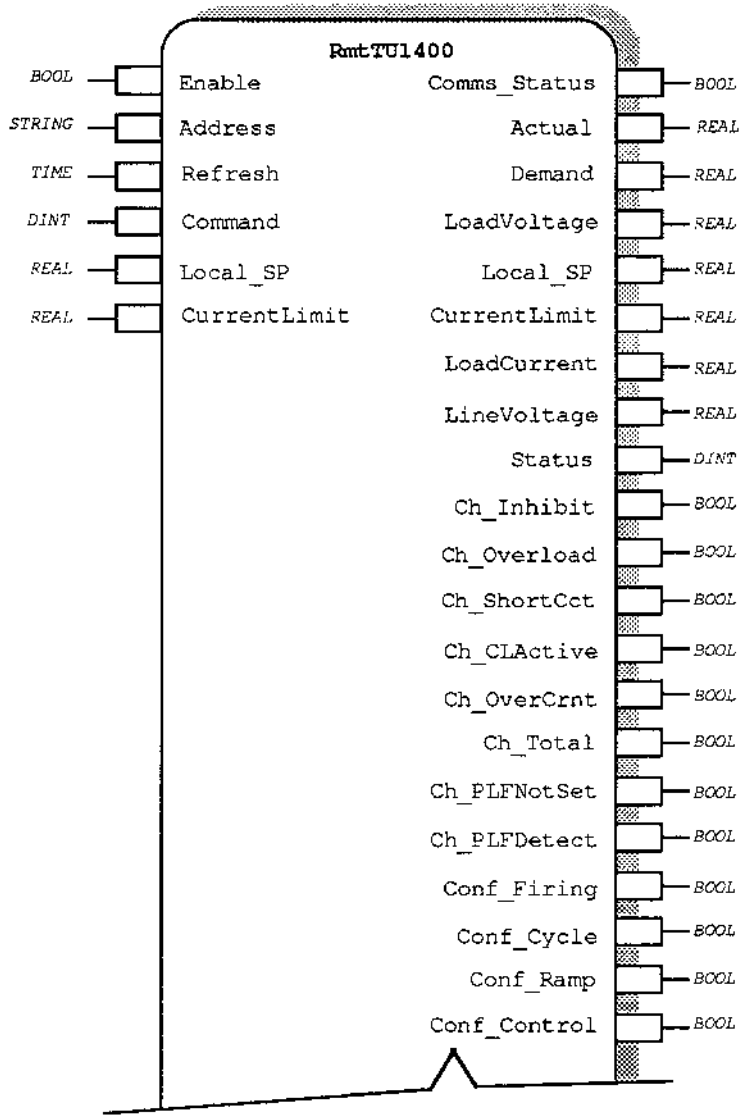


Figure 23-5 Schéma du bloc fonction Remote TU1400 (suite)

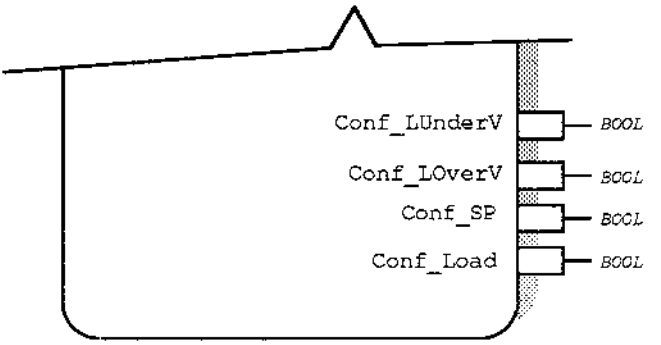


Figure 23-5 Schéma du bloc fonction Remote TU1400

Description fonctionnelle.

Ce bloc fonction offre une liaison de communications simple EI Bisync ASCII avec un gradateur de puissance à thyristors Eurotherm de la série TU1400. Il offre une interface de bloc fonction simple avec les paramètres les plus couramment utilisés.

Il doit y avoir une déclaration du bloc fonction **RmTU1400** pour chaque gradateur de puissance à thyristors. Pour assurer l'interface avec le quad, il faut créer quatre déclarations du bloc fonction.

Attributs du bloc fonction

Type DC40
Classe ..... REMOTE_INSTR
Tâche par défaut ..... Task_2
Liste résumée ..... Enable, Address, Actual, Demand
Mémoire nécessaire..... 2002 octets

Description des paramètres

Enable (EN)

Ce paramètre sert à activer et à désactiver la liaison de communications. Lorsqu'**Enable** est positionné sur **Enable (1)**, la liaison de communications est active et les paramètres seront interrogés.

Address (A)

Ce paramètre définit l'adresse de la commande moteur. La syntaxe de l'adresse est la suivante :

Port No. GID UID <space>

Par exemple, une adresse de 0B05<space> identifie une commande moteur dont GID = 0 et UID = 5, reliée au port B sur l'unité centrale PC3000. Les trois autres gradateurs de puissance à thyristors auront les adresses 0B06, 0B07 et 0B08.

Il faut également créer un bloc fonction maître EI Bisync pour chaque port de communications qui sera utilisé avec un bloc fonction d'appareil déporté.

Refresh (R)

Cette entrée définit la vitesse à laquelle seront lus les paramètres provenant de la commande moteur déportée. Toutes les écritures dans la commande moteur ont lieu immédiatement lorsque la valeur d'une entrée change. Si la commande moteur est retirée de la liaison ou est mise hors tension, il faut positionner **Enable** sur **Disable (0)** ou **Refresh** sur une durée de rafraîchissement longue afin d'empêcher le bloc fonction maître EI Bisync de perdre du temps et de la largeur de bande de communications par des tentatives répétées continues.

Command (CMD)

Ce paramètre est le mot de commande CW et est en écriture seule. Dans certains cas, les quatre gradateurs de puissance à thyristors du quad sont concernés par ce mot de commande. Dans d'autres cas, seul un gradateur isolé est concerné. Il arrive que les effets d'une écriture puissent être lus dans la sortie **Status**. Les valeurs possibles du mot de commande sont les suivantes :

0	- inhibe les quatre gradateurs du quad
1	- inhibe un gradateur isolé
2	- active les quatre gradateurs du quad
3	- active un gradateur isolé
4	- acquitte les alarmes
5	- règle PLF
6	- régulation sur V * I
7	- régulation sur V carré
8	- déclenchement de PA (gradateurs de puissance analogiques

uniquement)

9 - démarrage progressif du train d'ondes

10 - train d'ondes lent

11 - train d'ondes rapide

Comms_Status (S)

Si tous les paramètres sont lus sans erreur de communications, la sortie **Comms_Status** est positionnée sur OK (1). Si des erreurs de communications sont détectées, **Comms_Status** est positionné sur FAULT (0).

Actual (PV)

Cette sortie montre le calcul du retour d'informations de régulation sélectionné, soit V * I soit V carré. Ce paramètre correspond au mnémonique PV et est en lecture seule.

Demand (OP)

Cette sortie montre la demande sur le gradateur de puissance à thyristors si **Comms_Status** est sur Ok (1). Ce paramètre correspond au mnémonique OP et est en lecture seule.

Load Voltage (VV)

Cette sortie montre la tension de charge si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique VV et est en lecture seule.

LoadCurrent (CV)

Cette sortie montre l'intensité réelle dans la charge si **Comms_Status** est sur OK (1). Ce paramètre correspond au mnémonique CV et est en lecture seule.

LineVoltage (LV)

Cette sortie montre la tension de ligne appliquée au gradateur de puissance à thyristors si **Comms_Status** est sur OK (1). Cette sortie correspond au mnémonique LV et est en lecture seule.

Status (SW)

Cette sortie montre le mot d'état pour le gradateur de puissance à thyristors si **Comms_Status** est sur Ok (1). Ce paramètre correspond au mnémonique SW et est en lecture seule. L'entrée de la commande peut servir à modifier les réglages du gradateur de puissance. Les différents bits du mot d'état sont décodés en paramètres booléens simples énumérés ci-dessous.

Ch_Inhibit (INH)

Cette sortie montre si la voie est ou non activée et correspond à l'indicateur FGINH. Les valeurs possibles sont les suivantes :

activation (0)

inhibition (1)

Ch_Overload (OVL)

Cette sortie montre l'état de surcharge et correspond à l'indicateur FGOVL. Les valeurs possibles sont les suivantes :

Ok (0)

surcharge (1)

Ch_ShortCct (SCT)

Cette sortie montre si le gradateur de puissance à thyristors est ou non en court-circuit et correspond à l'indicateur FGSTL. Les valeurs possibles sont les suivantes :

Ok (0)

court-circuit (1)

Ch_CLActive (CLA)

Cette sortie est réglée si le gradateur de puissance à thyristors est en limite d'intensité et correspond à l'indicateur FGIMI. Les valeurs possibles sont les suivantes :

Ok (0)

I_Limit(1)

Ch_OverCrnt (OCT)

Cette sortie est réglée si le gradateur de puissance à thyristors débite une intensité excessive et correspond à l'indicateur FGOVC. Les valeurs possibles sont les suivantes :

Ok (0)

Over_I(1)

Ch_TotalLF (TLF)

Cette sortie est réglée s'il y a eu une défaillance de la charge totale et correspond à l'indicateur FGTLF. Les valeurs possibles sont les suivantes :

Ok (0)

TotalLF (1)

Ch_PLFNotSet (PNS)

Cette sortie montre si une détection de défaillance partielle de la charge (PLF) est ou non activée et correspond à l'indicateur FGNPLF. Les valeurs possibles sont les suivantes :

NotSet (0)

Ok (1)

Ch_PLFDetect (PLF)

Cette sortie est réglée si une défaillance partielle de la charge (PLF) est détectée et correspond à l'indicateur FGPLF. Les valeurs possibles sont les suivantes :

Ok (0)

PLFDet (1)

Conf_Firing (CF)

Cette sortie indique le mode de déclenchement et correspond à l'indicateur FGPA. Les valeurs possibles sont les suivantes :

Cycle (0) pour le déclenchement du cycle

Phase (1) pour le déclenchement angulaire de phase

Conf_Cycle (CCY)

Cette sortie correspond à l'indicateur FGLTO. Les valeurs possibles sont les suivantes :

simple (0) pour le cycle simple

rapide (1) pour le cycle rapide

Conf_Firing (CF)

Cette sortie indique le mode de déclenchement et correspond à l'indicateur FGPA. Les valeurs possibles sont les suivantes :

Cycle (0) pour le déclenchement du cycle

Phase (1) pour le déclenchement angulaire de phase

Conf_Ramp (CR)

Cette sortie indique le mode de rampe et correspond à l'indicateur FGRAMP. Les valeurs possibles sont les suivantes :

pur (0) pour le mode cycle pur

progressif (1) pour le mode de démarrage progressif

Conf_Control (CC)

Cette sortie indique le mode de régulation et correspond à l'indicateur FGRUGU. Les valeurs possibles sont les suivantes :

V2 (0) pour la régulation V carré

VxI (1) pour la régulation V * I

Conf_LUnderV (CLU)

Cette sortie indique une tension insuffisante sur la ligne d'alimentation et correspond à l'indicateur FGUNDV. Les valeurs possibles sont les suivantes :

Off (0) : la tension est correcte

On (1) : une tension insuffisante a été détectée

Conf_LOverV (CLO)

Cette sortie indique une surtension sur la ligne d'alimentation et correspond à l'indicateur FGUOVV. Les valeurs possibles sont les suivantes :

Off (0) : la tension est correcte

On (1) : une surtension a été détectée

Conf__SP`(CSP)

Cette sortie indique la source de la consigne et correspond à l'indicateur FGAN. Les valeurs possibles sont les suivantes :

analogique (0) travaillant à partir de la consigne analogique

numérique (1) travaillant à partir de la consigne numérique

Conf__Load (CLD)

Cette sortie indique le mode de la détection de charge partielle (PLF) et correspond à l'indicateur FGIR. Les valeurs possibles sont les suivantes :

Res_Ld (0) : détection PLF réglée pour la charge résistive

Inf_Ld (0) : détection PLF réglée pour la charge infra-rouge

Local_SP (SL)

Cette entrée/sortie sert à régler la sortie souhaitée sur le gradateur de puissance à thyristors et correspond au mnémonique SL. Il s'agit d'un paramètre en lecture/écriture.

CurrentLimit (CL)

Cette entrée/sortie sert à régler l'intensité maximale provenant du gradateur de puissance à thyristors et correspond au mnémonique CL. Il s'agit d'un paramètre en lecture/écriture qui est exprimé sous la forme d'un pourcentage de l'intensité nominale du gradateur de puissance .

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Enable	BOOL	Disable (0)			Sens	Disable (0) Enable (1)
Address	STRING	'OAOO'			Maximum 5 caractères	
Refresh	TIME	10s			Limite haute Limite basse	23d23h59m59s999 0
Command	DINT	0			Limite haute Limite basse	2147483646 -2147483647
Comms_Status	BOOL	FAULT (0)			Sens	FAULT (0) Ok (1)
Actual	REAL	0			Limite haute Limite basse	+3.402823e+38 -3.402823e+38
LocalVoltage	REAL	0			Limite haute Limite basse	+3.402823e+38 -3.402823e+38
LoadCurrent	REAL	0			Limite haute Limite basse	+3.402823e+38 -3.402823e+38
LineVoltage	REAL	0			Limite haute Limite basse	+3.402823e+38 -3.402823e+38
Status	DINT	0			Limite haute Limite basse	2147483646 -2147483647
Ch_Inhibit	BOOL	Enable (0)			Sens	Enable (0) Inhibit (1)
Ch_Overload	BOOL	Ok (0)			Sens	Ok (0) OverLd (1)
Ch_ShortCct	BOOL	Ok (0)			Sens	Ok (0) ShortCt (1)
Ch_ClActive	BOOL	Ok (0)			Sens	Ok (0) I_Limit (1)
Ch_OverICrnt	BOOL	Ok (0)			Sens	Ok (0) Over_I (1)

Tableau 23-5 Attributs des paramètres RmTU1400

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Ch_TotalLF	BOOL	Ok (0)			Sens	Ok (0) TotalLF(1)
Ch_PLFNotSet	BOOL	NotSet0)			Sens	NotSet 0) Ok (1)
Ch PLFDetect	BOOL	Ok (0)			Sens	Ok (0) PLFDet 1()
Conf_Firing	BOOL	Cycle (0)			Sens	Cycle (0) Phase (1)
Conf_Cycle	BOOL	Single 0)			Sens	Single (0) Phase (1)
Conf_Ramp	BOOL	Pure (0)			Sens	Pure (0) Soft (1)
Conf_Control	BOOL	V2 (0)			Sens	V2 (0) Vxl (1)
Conf_LUnderV	BOOL	Off (0)			Sens	Off (0) On (1)
Conf_LOverV	BOOL	Off (0)			Sens	Off (0) On (1)
Conf_SP	BOOL	Analog (0)			Sens	Analog (0) Digital (1)
Conf_Load	BOOL	Res_Ld (0)			Sens	Res_Ld (0) Inf_Ld (1)
Local_SP	REAL	0			Limite haute Limite basse	+3.402823e+38 -3.402823e+38
CurrentLimit	REAL	0			Limite haute Limite basse	+3.402823e+38 -3.402823e+38

Tableau 23-5 Attributs des paramètres RmTU1400

Chapitre 24

MANIPULATION BINAIRE

Edition 2

Présentation

Cette classe de blocs fonctions contient une famille de blocs fonctions d'usage général servant au codage d'ensembles de paramètres booléens en variables DINT et au décodage de paramètres DINT en ensembles de variables booléennes.

BLOC FONCTION BOOLTODINT

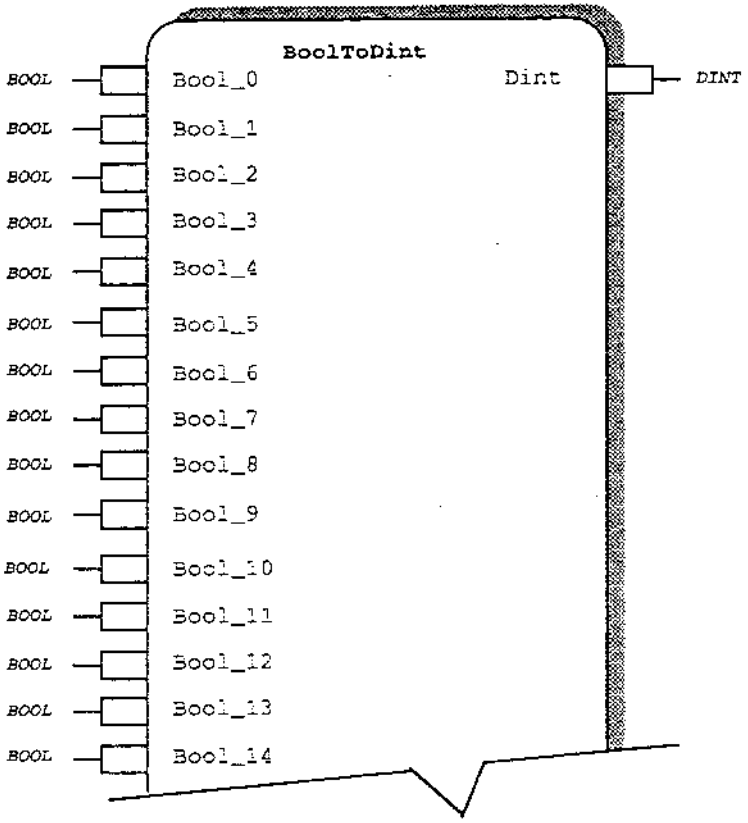


Fig. 24-1 Schéma du bloc fonction BoolToDint

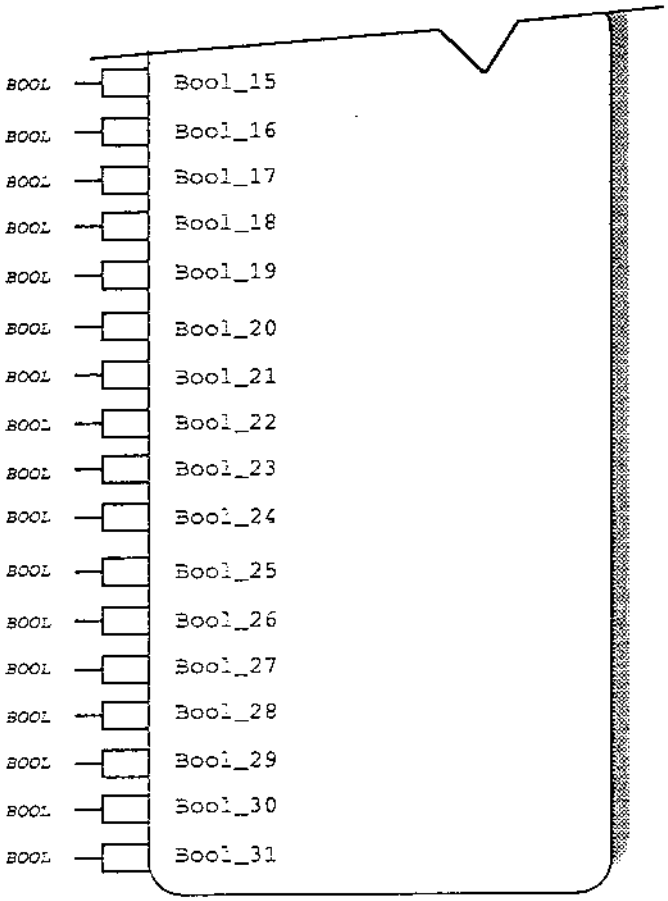


Fig. 24-1 Schéma du bloc fonction BoolToDint (Suite)

Description fonctionnelle

Ce bloc permet de regrouper jusqu'à 32 valeurs booléennes individuelles en un entier double avec signe (c'est-à-dire un nombre entier sur 32 bits). Il peut servir à compacter des données booléennes pour permettre leur transmission rapide sur une liaison ou pour regrouper des informations de code d'erreur.

Les valeurs booléennes 0 à 30 ont un poids binaire égal à leur indice, c'est-à-dire que si la quatrième entrée (Bool_3) est positionnée, elle ajoute une valeur de 8 (2 exposant 3) à la sortie Dint.

La valeur booléenne Bool_31 reçoit également un poids égal à son indice (c'est-à-dire 31) mais avec le signe moins. Bool_31 détermine par conséquent le signe de Dint, ainsi que la valeur totale. Quelques exemples de la valeur de sortie correspondant à différentes entrées sont donnés ci-dessous.

Entrées positionnées	Valeur Dint
Bool_0, Bool_1	3
Bool_0, Bool_1,Bool30	1073741827
Bool_0, Bool_1,Bool31	-2147483645
Bool_0 to Bool_31	-1

Attributs du bloc fonction

Type : E601
Classe : ..... BITMANIP
Tâche par défaut : ..... Task_2
Liste résumée : ..... Dint
Mémoire nécessaire : ..... 36 octets

Description des paramètres

Bool_0 (B0)

Entrée utilisée comme bit de poids faible de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 1.

Bool_1 (B1)

Entrée utilisée comme second bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** égal à 2.

Bool_2 (B2)

Entrée utilisée comme troisième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 4.

Bool_3 (B3)

Entrée utilisée comme quatrième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 8.

Bool_4 (B4)

Entrée utilisée comme cinquième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 16.

Bool_5 (B5)

Entrée utilisée comme sixième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 32.

Bool_6 (B6)

Entrée utilisée comme septième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 64.

Bool_7 (B7)

Entrée utilisée comme huitième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 128.

Bool_8 (B8)

Entrée utilisée comme neuvième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 256.

Bool_9 (B9)

Entrée utilisée comme dixième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 512.

Bool_10 (B10)

Entrée utilisée comme onzième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 1 024.

Bool_11 (B11)

Entrée utilisée comme douzième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 2 048.

Bool_12 (B12)

Entrée utilisée comme treizième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 4 096.

Bool_13 (B13)

Entrée utilisée comme quatorzième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 8 192.

Bool_14 (B14)

Entrée utilisée comme quinzième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 16 384.

Bool_15 (B15)

Entrée utilisée comme seizième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 32 768.

Bool_16 (B16)

Entrée utilisée comme dix-septième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 65 536.

Bool_17 (B17)

Entrée utilisée comme dix-huitième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 131 072.

Bool_18 (B18)

Entrée utilisée comme dix-neuvième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 262 144.

Bool_19 (B19)

Entrée utilisée comme vingtième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 524 288.

Bool_20 (B20)

Entrée utilisée comme vingt et unième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 1 048 576.

Bool_21 (B21)

Entrée utilisée comme vingt-deuxième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 2 097 152.

Bool_22 (B22)

Entrée utilisée comme vingt-troisième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 4 194 304.

Bool_23 (B23)

Entrée utilisée comme vingt-quatrième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 8 388 608.

Bool_24 (B24)

Entrée utilisée comme vingt-cinquième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 16 777 216.

Bool_25 (B25)

Entrée utilisée comme vingt-sixième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 33 554 432.

Bool_26 (B26)

Entrée utilisée comme vingt-septième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 67 108 864.

Bool_27 (B27)

Entrée utilisée comme vingt-huitième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 134 217 728.

Bool_28 (B28)

Entrée utilisée comme vingt-neuvième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 268 435 456.

Bool_29 (B29)

Entrée utilisée comme trentième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 536 870 912.

Bool_30 (B30)

Entrée utilisée comme trente et unième bit de **Dint**. Si c'est la seule entrée positionnée, **Dint** est égal à 1 073 741 824.

Bool_31 (B31)

Entrée utilisée comme trente-deuxième bit de **Dint**. Cette entrée est affectée du signe négatif. Si c'est la seule entrée positionnée, **Dint** est égal à - 2 147 483 648.

Dint (INT)

Somme des entrées booléennes pondérées individuellement.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Bool_0	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_1	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_2	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_3	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_4	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_5	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_6	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_7	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_8	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_9	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_10	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_11	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_12	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_13	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_14	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_15	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)

Tableau 24-1 Attributs des paramètres BoolToDint

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Bool_16	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_17	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_18	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_19	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_20	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_21	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_22	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_23	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_24	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_25	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_26	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_27	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_28	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_29	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_30	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_31	**BOOL**	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Dint	**DINT**	0	Oper	Oper	Block	2,147,483,647 -2,147,483,648

Tableau 24-1 Attributs des paramètres BoolToDint (Suite)

BLOC FONCTION DINTTOBOOL

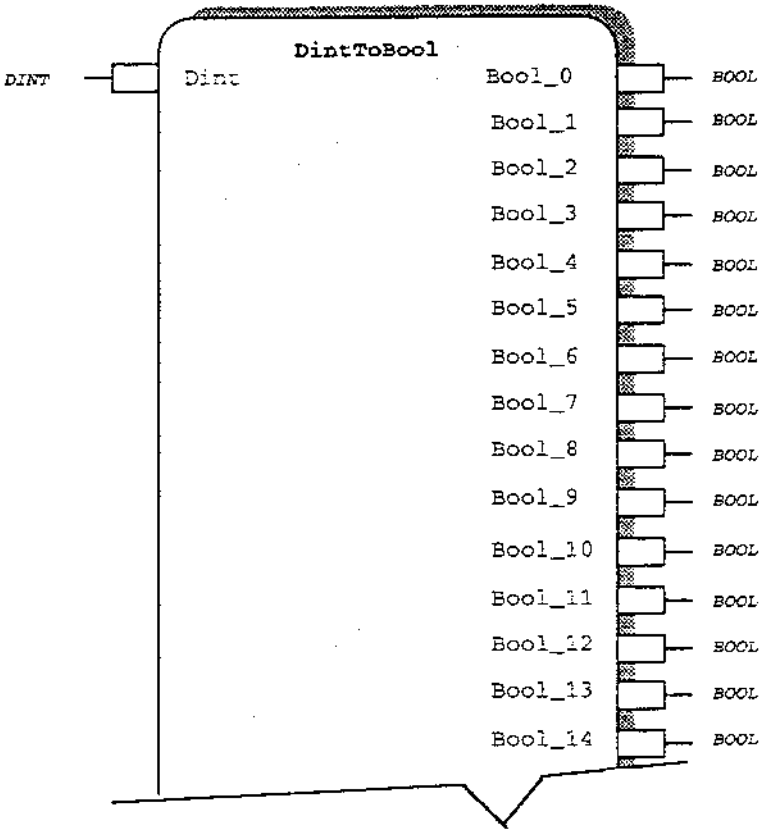


Fig. 24-2 Schéma du bloc fonction DintToBool

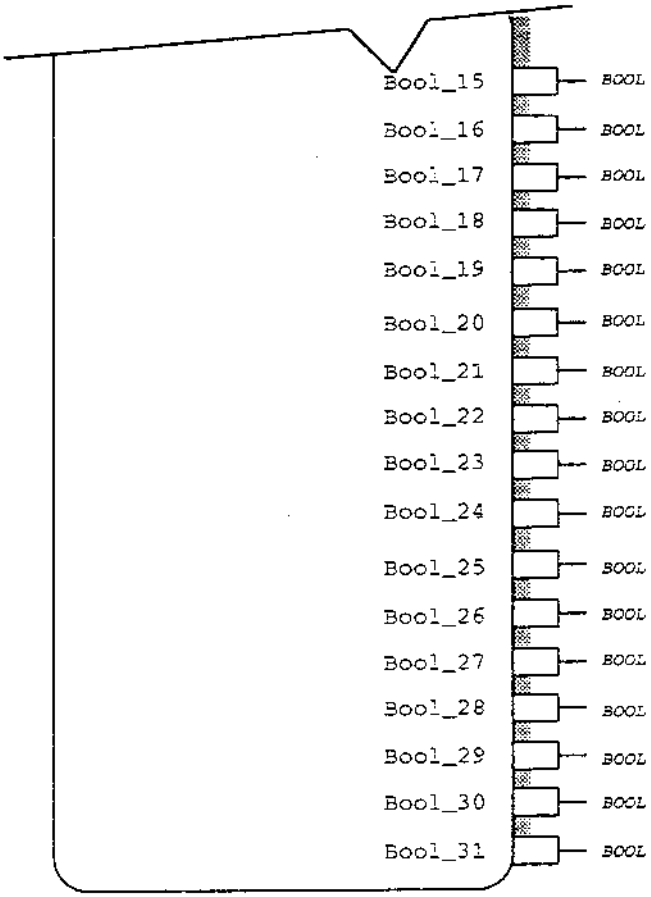


Fig. 24-2 Schéma du bloc fonction DintToBool (Suite)

Description fonctionnelle

Un entier double avec signe (c'est-à-dire un nombre entier sur 32 bits) est scindé en 32 sorties booléennes individuelles. Ceci peut être utile pour scinder une valeur entière mise à jour sur une liaison ou pour décoder des informations de code d'erreur.

Les valeurs booléennes 0 à 30 ont un poids binaire égal à leur indice, c'est-à-dire que la quatrième sortie (Bool_3) est positionnée, si l'entrée Dint a une valeur de 8 (2 exposant 3).

La valeur booléenne Bool_31 a également un poids égal à son indice (c'est-à-dire 31) mais avec le signe moins. Le signe de Dint détermine par conséquent la valeur de Bool_31, ainsi que la valeur totale. Quelques exemples montrant les sorties positionnées correspondant à différentes valeurs d'entrée sont donnés ci-dessous.

Valeur Dint	Sorties positionnées
3	Bool_0, Bool_1
1073741827	Bool_0, Bool_1,Bool30
-2147483645	Bool_0, Bool_1,Bool31
-1	Bool_0 to Bool_31

Attributs du bloc fonction

Type : E604
Classe : ..... BITMANIP
Tâche par défaut : ..... Task_2
Liste résumée : ..... Dint, Bool_0, Bool_1, Bool_2
Mémoire nécessaire : ..... 36 octets

Description des paramètres

Dint (INT)

Valeur utilisée pour commander les sorties booléennes pondérées individuellement.

Bool_0 (B0)

Sortie positionnée par le bit inférieur de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 1.

Bool_1 (B1)

Sortie positionnée par le second bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 2.

Bool_2 (B2)

Sortie positionnée par le troisième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 4.

Bool_3 (B3)

Sortie positionnée par le quatrième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 8.

Bool_4 (B4)

Sortie positionnée par le cinquième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 16.

Bool_5 (B5)

Sortie positionnée par le sixième bit de **Dint**. C'est la seule sortie positionnée, si **Dint** est égal à 32.

Bool_6 (B6)

Sortie positionnée par le septième bit de **Dint**. C'est la seule sortie positionnée, si **Dint** est égal à 64.

Bool_7 (B7)

Sortie positionnée par le huitième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 128.

Bool_8 (B8)

Sortie positionnée par le neuvième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 256.

Bool_9 (B9)

Sortie positionnée par le dixième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 512.

Bool_10 (B10)

Sortie positionnée par le onzième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 1 024.

Bool_11 (B11)

Sortie positionnée par le douzième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 2 048.

Bool_12 (B12)

Sortie positionnée par le treizième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 4 096.

Bool_13 (B13)

Sortie positionnée par le quatorzième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 8 192.

Bool_14 (B14)

Sortie positionnée par le quinzième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 16 384.

Bool_15 (B15)

Sortie positionnée par le seizième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 32 768.

Bool_16 (B16)

Sortie positionnée par le dix-septième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 65 536.

Bool_17 (B17)

Sortie positionnée par le dix-huitième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 131 072.

Bool_18 (B18)

Sortie positionnée par le dix-neuvième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 262 144.

Bool_19 (B19)

Sortie positionnée par le vingtième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 524 288.

Bool_20 (B20)

Sortie positionnée par le vingt et unième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 1 048 576.

Bool_21 (B21)

Sortie positionnée par le vingt-deuxième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 2 097 152.

Bool_22 (B22)

Sortie positionnée par le vingt-troisième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 4 194 304.

Bool_23 (B23)

Sortie positionnée par le vingt-quatrième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 8 388 608.

Bool_24 (B24)

Sortie positionnée par le vingt-cinquième bit de **Dint**. C'est la seule sortie positionnée, si **Dint** est égal à 16 777 216.

Bool_25 (B25)

Sortie positionnée par le vingt-sixième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 33 554 432.

Bool_26 (B26)

Sortie positionnée par le vingt-septième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 67 108 864.

Bool_27 (B27)

Sortie positionnée par le vingt-huitième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 134 217 728.

Bool_28 (B28)

Sortie positionnée par le vingt-neuvième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 268 435 456.

Bool_29 (B29)

Sortie positionnée par le trentième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 536 870 912.

Bool_30 (B30)

Sortie positionnée par le trente et unième bit de **Dint**. C'est la seule sortie positionnée si **Dint** est égal à 1 073 741 824.

Bool_31 (B31)

Sortie positionnée par le trente-deuxième bit de **Dint**. Cette sortie est positionnée si **Dint** est de signe négatif. C'est la seule sortie positionnée si **Dint** est égal à - 2 147 483 648.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Dint	DINT	0	Oper	Oper	Limite haute Limite basse	2,147,483,647 -2,147,483,648
Bool_1	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_2	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_3	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_4	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_5	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_6	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_7	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_8	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_9	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_10	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_11	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_12	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_13	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_14	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)
Bool_15	BOOL	Off (0)	Oper	Oper	Sens	Off (0) On (1)

Tableau 24-2 Attributs du paramètre DintToBool

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Bool_16	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_17	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_18	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_19	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_20	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_21	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_22	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_23	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_24	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_25	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_26	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_27	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_28	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_29	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_30	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)
Bool_31	**BOOL**	Off (0)	Oper	Block	Sens	Off (0) On (1)

Tableau 24-2 Attributs du paramètre DintToBool (Suite)

CHAPITRE 25

CHARGES

Edition 1

PRÉSENTATION

Ce chapitre décrit la classe CHARGES de blocs fonctions qui fournit des modèles de charges de centrales fréquemment rencontrés. Ces modèles peuvent servir à simuler le fonctionnement d'une unité de production sans qu'il soit nécessaire d'utiliser des E/S matérielles. On peut s'en servir comme d'outils de mise au point. Par exemple, une PID_Load peut être connectée à un bloc fonction PID pour simuler une boucle d'asservissement complète. L'interaction du programme utilisateur avec le bloc fonction PID peut être contrôlée sans qu'il soit nécessaire de se connecter à l'unité de production physique, un four, etc.

BLOC FONCTION PID_LOAD

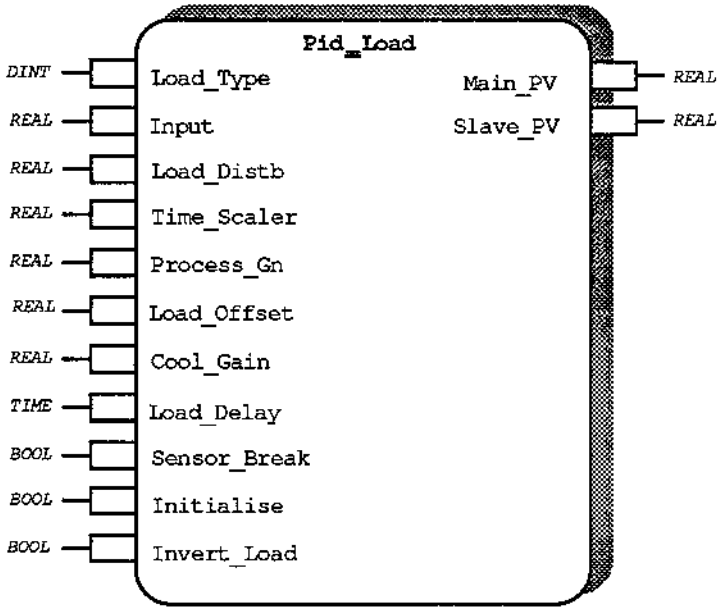


Figure 25-1 Bloc fonction PID_Load

Description fonctionnelle

Le bloc fonction PID_Load simule le comportement de trois types de charges régulées en température utilisés dans des tests de simulation de régulation en boucle fermée. Il se compose d'une série de retards en cascade, avec une échelle des temps, un gain et une charge de perturbation réglables. Le bloc fonction peut être configuré pour simuler un four, une zone de tambour d'extrudeuse ou un four avec temps de retard.

Configuration de Pid_Load pour simuler une boucle de régulation

En règle générale, lorsqu'on emploie Pid_Load dans une simulation PC3000, on construit un modèle dans lequel le bloc fonction est régulé en boucle fermée soit par le PID, soit par les blocs fonctions PID_Auto. Ci-dessous un exemple de câblage par soft pour commander la Pid_Load par le bloc fonction PID :

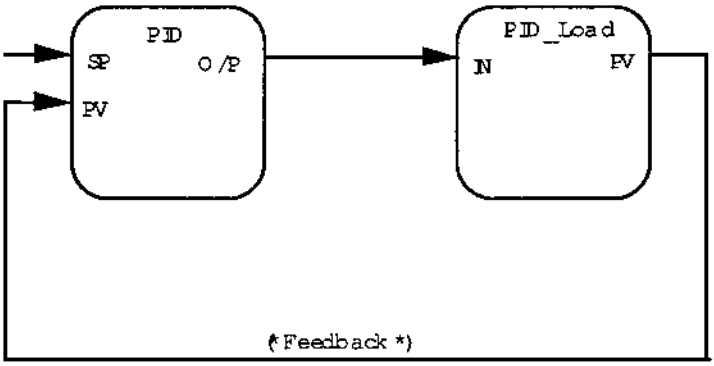


Figure 25-2 Câblage par soft de Pid_Load et PID.

Pour faire fonctionner le bloc fonction dans la configuration illustrée ci-dessus, il convient de régler correctement la commande PID pour obtenir une réponse stable. Pour ce faire, on utilise l'Autoréglant après avoir configuré la charge pour se rapprocher de la charge simulée. Des exemples de valeurs pour les types de charge 1,2 et 3 sont donnés dans le tableau ci-dessous, avec les valeurs des paramètres de charge pour lesquels le réglage PID va produire une réponse stable.

Attributs du bloc fonction

Type : ..... fO 10
Classe : ..... LOADS
Tâche par défaut : ..... Task_2
Liste récapitulative : ..... Load_Type, Main_PV, Slave_PV, Input
Besoins de capacité mémoire : ..... 1802 octets
Durée d'exécution : ..... 14,0 msec

Paramètre	Type de charge		
	1	2	3
Load_Distb	0	0	0
Time_Scaler	1	1	1
Process_Gn	2	2	2
Load_Offset	0	0	0
Cool_Gain	1	1	1
Load_Delay	0	0	20s
Prop_Band	5%	12%	30%
Integral	24s	72s	75s
Derivative	4s	12s	12s_500ms

Tableau 25-1 Configuration de charge caractéristique et valeurs PID pour une simulation de charge réglée

Description des paramètres

Load_Type (LT)

Load_Type permet de choisir le type de charge simulé par le bloc fonction. Il peut être réglé sur l'un des trois types suivants :

1: Four

2: Zone de tambour d'extrudeuse

3: Four avec temps de retard.

Input (IN)

Input est l'entrée dans le bloc fonction.

Load_Distb (LD)

Load_Distb peut servir à simuler une charge de perturbation comme une perte thermique, qui est ajoutée directement à l'entrée dans le bloc fonction.

Time_Scaler (TS)

Time_Scaler sert d'échelle sur les premiers retards d'ordre du bloc fonction. Le réglage du Time_Scaler sur 1 permet aux modèles de simulation de donner une représentation en temps réel des systèmes réels. Les valeurs inférieures à 1 signifient plus rapide qu'en temps réel et les valeurs supérieures à 1 plus lentement qu'en temps réel.

Process_Gn (PG)

Process_Gn est le gain stable du process simulé. Les valeurs stables de Main_PV et Slave_PV étant identiques à Input * Process_Gn, il est important de sélectionner la valeur de Process_Gn en tenant compte à la fois de l'étendue de l'application de commande simulée et du gain du process.

Load_Offset (LO)

Load_Offset n'a pas de fonction dans le bloc fonction PID_Load. Ce paramètre est réservé pour une utilisation ultérieure.

Cool_Gain (CG)

Cool_Gain permet d'introduire un chauffage et un refroidissement non linéaires dans la simulation de charge. Le réglage de Cool_Gain sur une valeur inférieure à 1 réduit la constante de temps de la charge si l'entrée est inférieure à 0, ce qui simule une réponse fraîche et rapide. Le réglage de Cool_Gain sur une valeur supérieure à 1 simule une réponse fraîche et lente en augmentant la constante temps de la charge pour les entrées inférieures à 1. Le réglage de Cool_Gain sur 1 simule une réponse fraîche-chaude linéaire, avec une constante de temps de charge constante pour toutes les valeurs d'entrée.

Load_Delay (DEL)

Load_Delay n'est utilisé que lorsque Load_Type est réglé sur 3. Il définit la longueur du temps de retard compris dans le modèle de simulation.

Sensor_Break (SBR)

Lorsque Sensor_Break est réglé sur BREAK (1), le bloc fonction simule une situation de rupture de capteur. Dans cet état, Main_PV monte à 200 % et Slave_PV monte à Input * Process_Gn. Dès que Sensor_Break est remis à NO_BRK (0), la simulation de charge normale reprend.

Initialise (INI)

Le réglage d'Initialise sur Init (1) gèle le fonctionnement du bloc fonction, Main_PV et Slave_PV étant maintenus à des valeurs constantes. Dès que Initialise passe du réglage Init (1) à Run (0), Main_PV et Slave_PV sont remis à zéro et la simulation reprend.

Invert_Load (IL)

Invert_Load n'a pas de fonction dans le bloc fonction PID_Load. Ce paramètre est réservé pour une utilisation future.

Main_PV (PV)

Main_PV est la principale sortie de valeur de process simulée (température simulée) du bloc fonction.

Slave_PV (SPV)

Slave_PV est une valeur de process intermédiaire prise dans la sortie d'un retard intermédiaire dans le modèle de simulation. Ce paramètre est destiné aux simulations de boucle de commande maître/esclave.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Load_Type	DINT	1	Oper	Oper	Lim. haute Lim. basse	3 1
Input	REAL	0	Oper	Oper	Lim. haute Lim. basse	1000 -1000
Load_Distb	REAL	0	Oper	Oper	Lim. haute Lim. basse	10 0
Time_Scaler	REAL	1	Oper	Oper	Lim. haute Lim. basse	30 0,3
Process_Gn	REAL	1	Oper	Oper	Lim. haute Lim. basse	100 0,1
Load_Offset	REAL	0	Oper	Oper	Lim. haute Lim. basse	100 -100
Cool_Gain	REAL	1	Oper	Oper	Lim. haute Lim. basse	10 0,1
Load_Delay	TIME	0	Oper	Oper	Lim. haute Lim. basse	25 s 0
Sensor_Break	BOOL	NO_BRK (0)	Oper	Oper	Détect.	NO_BREAK (0) BREAK (1)
Initialise	BOOL	Run (0)	Config	Config	Détect.	Run (0) Init (1)
Invert_Load	BOOL	No (0)	Config	Config	Détect.	No (0) Yes (1)
Main_PV	REAL	0	Oper		Lim. haute Lim. basse	1000 -1000
Slave_PV	REAL	0	Oper		Lim. haute Lim. basse	1000 -1000

Tableau 25-2 Attributs des paramètres de PID_Load

BLOC FONCTION VP_LOAD

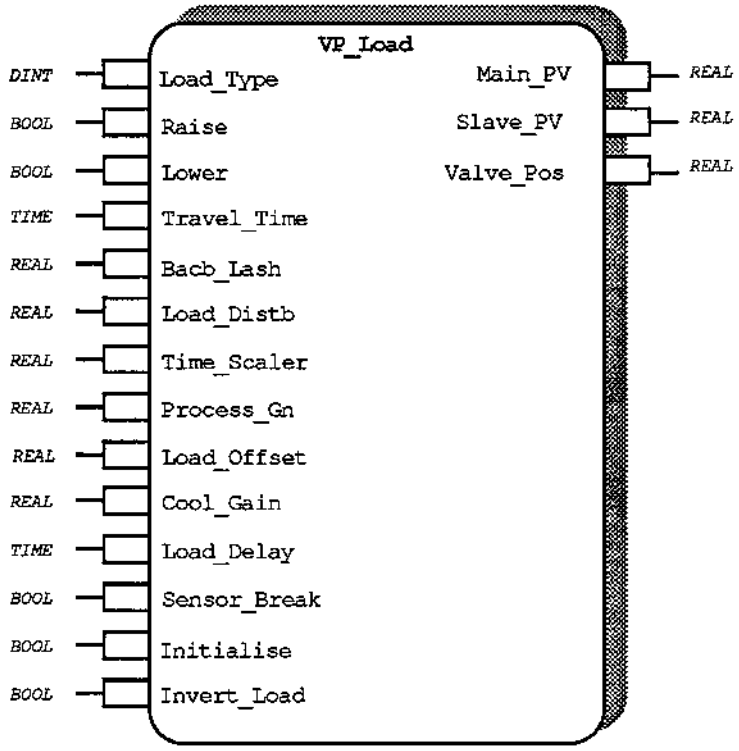


Figure 25-3 Schéma du bloc fonction VP_Load

Description fonctionnelle

Le bloc fonction VP_Load simule le comportement de trois types de charges entraînées par des vannes de positionnement, contrôlées en température, pour utilisation en essais de simulation de commande en boucle fermée. Il se compose d'un ensemble de retards en cascade, avec possibilité de définir l'échelle de temps, le gain et la charge de perturbation. Le bloc fonction peut être configuré pour simuler un four, une zone de tambour d'extrudeuse, ou un four avec temps mort.

Configuration de VP_Load en simulation de boucle de régulation

Pour utiliser VP_Load en simulation de PC3000, on construit, en général, un modèle dans lequel le bloc fonction est commandé en boucle fermé par les blocs fonctions VP ou VP_Auto. Un exemple de câblage logiciel pour réguler VP_Load par le bloc fonction VP est représenté ci-dessous.

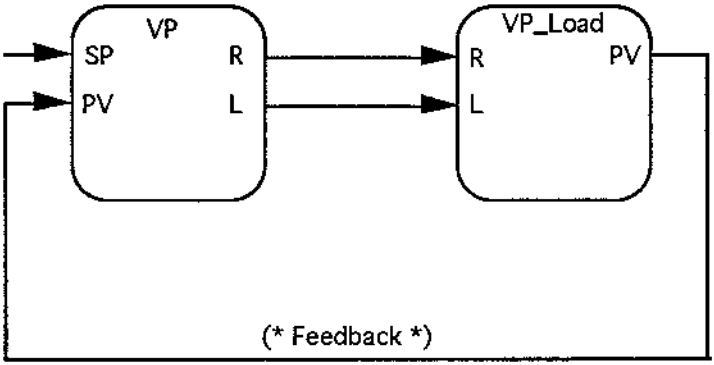


Figure 25-4 Câblage logiciel de VP_Load et VP.

Pour faire fonctionner le bloc fonction dans la configuration représentée ci-dessus, il est nécessaire de régler correctement la régulation PID, pour obtenir une réponse stable. Ceci peut être obtenu à l'aide de Autoréglant après avoir configuré une charge semblable au système à simuler. Des exemples de valeurs typiques de PID sont données dans le tableau ci-après, pour les VP_Load de type 1,2 et 3, y compris les valeurs de paramètres de charge pour lesquelles le réglage PID fournit une réponse stable.

Attributs du bloc fonction

Type : fO2O
Classe : ..... LOADS
Tâche par défaut : ..... Task_2
Liste récapitulative : ..... Raise, Lower, Main_PV, Valve_Pos
Besoins de capacité mémoire : ..... 1836 octets
Durée d'exécution : ..... 14 300 µs

Paramètre	Type de VP-Load		
	1	2	3
Travel_Time	10 s	10 s	10 s
Back_Lash	1,0 %	1,0 %	1,0 %
Load_Distb	0	0	0
Time_Scaler	1	1	1
Process_Gn	2	2	2
Load_Offset	0	0	0
Cool_Gain	1	1	1
Load_Delay	0	0	20 s
Prop_Band	10 %	17,5 %	70 %
Integral	42 s	1 m_30 s	1 m_30 s
Derivative	7 s	15 s	0

Tableau 25-3 Types de paramètres de VP_Load

Description des paramètres

Load_Type (LT)

Load_Type permet la sélection du type de charge entraînée par le positionneur de vanne que simule le bloc fonction. Trois types peuvent être choisis :

1 : Four

2 : Zone de tambour d'extrudeuse

3 : Four avec temps mort.

Raise (R)

Raise (monter) est l'entrée du bloc fonction qui permet la montée de la vanne simulée.

Lower (L)

Lower (descendre) est l'entrée du bloc fonction qui permet la descente de la vanne simulée.

Travel_Time (TT)

Travel_Time (temps de déplacement) est le temps nécessaire à la vanne simulée pour se déplacer entre les positions d'ouverture totale et de fermeture totale.

Back_Lash (BL)

Back_Lash (course morte) définit la course morte de la vanne simulée. Elle est définie en pourcentage du déplacement total de la vanne.

Load_Distb (LD)

Load_Distb (perturbation de charge) permet de simuler une charge parasite, comme une perte de chaleur, qui s'ajoute directement à l'entrée (Input) du bloc fonction.

Time_Scaler (TS)

Time_Scaler (mise à l'échelle des temps) agit en tant que mise à l'échelle des retards de premier ordre du bloc fonction. La mise à 1 de Time_Scaler permet aux modèles de simulation de fournir une représentation en temps réel de systèmes réels. Une valeur inférieure à 1 signifie une vitesse supérieure au temps réel, et une valeur supérieure à 1, représente des systèmes plus lents qu'en temps réel.

Process_Gn (PG)

Process_Gn (gain processus) est le gain à l'état stable du processus simulé. Les valeurs à l'état stable de Main_PV et Slave_PV sont égales à Valve_Pos Process_Gn, de telle sorte qu'il est important que la valeur de Process_Gn soit choisie en tenant compte à la fois de la plage utile de l'application de commande simulée, et du gain du processus.

Load_Offset (LO)

Load_Offset n'est pas utilisé dans le bloc fonction VP_Load. Il a été prévu pour un développement futur.

Load_Delay (DEL)

Load_Delay (délai de charge) n'est utilisé que si Load_Type est mis à 3. Il définit la durée du retard inclus au modèle simulé.

Sensor_Break (SBR)

Si Sensor_Break (rupture capteur) est mis à BREAK (1), le bloc fonction simule une situation de rupture de capteur. Dans cet état, Main_PV varie linéairement jusqu'à 200 % et Slave_PV jusqu'à 0. Si Sensor_Break est remis à No (0), la simulation de charge redevient normale.

Initialise (INI)

Si Initialise est mis sur Init (1), le fonctionnement du bloc fonction est "gelé", Main_PV et Slave_PV étant maintenus à une valeur constante. Sur le front avant de réinitialisation de Init (1) à Run (0), Main_PV et Slave_PV sont remis à zéro et la simulation reprend.

Invert_Load (IL)

Invert_Load n'est pas utilisé dans le bloc fonction VP_Load. Il a été prévu pour un développement futur.

Main_PV (PV)

Main_PV est la valeur principale simulée du processus (c'est-à-dire la température simulée) du bloc fonction.

Slave_PV (SPV)

Slave_PV est une valeur intermédiaire du processus, qui est prise en sortie d'un retard intermédiaire du modèle de simulation. Il est prévu pour servir dans les simulations de boucles de commande en cascade maître / esclave.

Valve_Pos (VP)

Valve_Pos indique la position de la vanne simulée.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Back_Lash	REAL	1 %	Oper	Oper	Lim. haute Lim. basse	100 % 0
Cool_Gain	REAL	1	Oper	Oper	Lim. haute Lim. basse	10 0,1
Initialise	BOOL	Run (0)	Config	Config	Détect.	Run (0) Init (1)
Invert_Load	BOOL	No (0)	Config	Config	Détect.	No (0) Yes (1)
Load_Delay	TIME	0	Oper	Oper	Lim. haute Lim. basse	25 s 0
Load_Distb	REAL	0	Oper	Oper	Lim. haute Lim. basse	10 0
Load_Offset	REAL	0	Oper	Oper	Lim. haute Lim. basse	100 -100
Load_Type	DINT	1	Oper	Oper	Lim. haute Lim. basse	3 1
Lower	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Main_PV	REAL	0	Oper		Lim. haute Lim. basse	10 000 -10 000
Process_Gn	REAL	1	Oper	Oper	Lim. haute Lim. basse	100 0,1
Raise	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Sensor_Break	BOOL	No (0)	Oper	Oper	Détect.	No (0) Break (1)
Slave_PV	REAL	0	Oper		Lim. haute Lim. basse	10 000 -10 000
Time_Scaler	REAL	1	Oper	Oper	Lim. haute Lim. basse	30 0,3
Travel_Time	TIME	10 s	Oper	Oper	Lim. haute Lim. basse	16m_40s 0
Valve_Pos	REAL	0	Oper		Lim. haute Lim. basse	100 % 0

Tableau 25-4 Attributs des paramètres de VP_Load

Chapitre 26

DIVERS

Edition 1

Sommaire (suite)

Vue d'ensemble

Ce chapitre décrit les catégories diverses de blocs fonctions qui fournissent différentes fonctions, comme les registres à décalage, les limiteurs de vitesse de variation, un générateur de rampe pour la création de point de consigne, un bloc de commande d'alarme d'usage général, et des fonctions de bascule.

BLOC FONCTION RATE_LIMIT

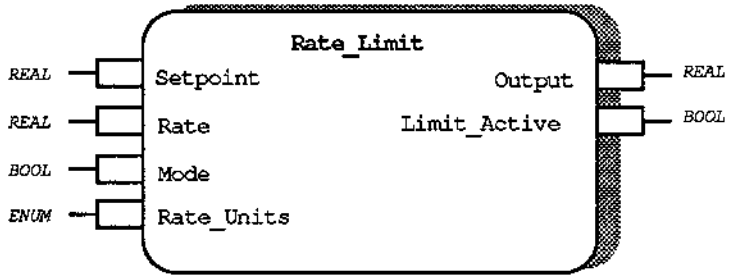


Figure 26-1 Schéma du bloc fonction Rate_Limit

Description fonctionnelle

Le bloc fonction Rate_Limit est utilisé pour limiter la vitesse maximale de variation d'un paramètre. Le paramètre dont la vitesse de variation doit être limitée est mis sur l'entrée Setpoint et la valeur du paramètre limitée en vitesse est sortie en Output. La valeur maximale de variation autorisée pour Output est définie par Rate, les unités de Rate étant définies par le paramètre Rate_Units. Lorsque la limitation de vitesse a lieu, Limit_Active est mis à Limit (1). Le bloc fonction a deux modes de fonctionnement qui sont définis par le paramètre Mode.

Modes de fonctionnement

Track (0) : en mode Track (suivi), la sortie suit l'entrée sans aucune limitation de vitesse.

Limit (1) : en mode Limit (limite), la vitesse maximale de variation de Output est limitée par la valeur indiquée par Rate.

Attributs du bloc fonction

Type : F8 20

Classe : ..... DIVERS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Setpoint, Mode, Output

Besoins de capacité mémoire : ..... 32 octets

Durée d'exécution : ..... 298 µs

Description des paramètres

Setpoint (SP)

Le paramètre Setpoint est l'entrée du bloc fonction dont la variation doit être limitée en vitesse.

Rate (R)

Le paramètre Rate définit la vitesse maximale de variation à laquelle doit être limitée Output. Les unités de Rate sont définies par Rate_Units.

Mode (M)

Le paramètre Mode définit le mode de fonctionnement du bloc fonction, selon les définitions précédentes.

Rate_Units (RU)

Le paramètre Rate_Units définit l'unité du paramètre Rate. Quatre états sont possibles pour Rate_Units :

/ Second (0) :	La vitesse est par seconde
/ Minute (1) :	La vitesse est par minute
/ Hour (2) :	La vitesse est par heure
/ Day (3) :	La vitesse est par jour

Output (OP)

Le paramètre Output est la sortie limitée en vitesse du bloc fonction. En mode Track (0), Output suit l'entrée Setpoint sans procéder à une limitation de vitesse. En mode Limit (1), Output suit l'entrée Setpoint avec une limitation de vitesse de variation définie par Rate.

Limit_Active (LA)

Le paramètre Limit_Output est un indicateur qui signale qu'une limitation de vitesse a lieu. Si le limiteur de vitesse est actif, Output est différent de Setpoint et Limit_Output est mis à Limit (1). Si Output est égal à Setpoint, le limiteur de vitesse n'est pas actif, et Limit_Output est mis à Track (0).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Limit_Active	**BOOL**	Track (0)	Oper	Bloc	Détect.	Track (0) Limit (1)
Mode	**BOOL**	Track (0)	Oper	Config	Détect.	Track (0) Limit (1)
Output	**REAL**	0,0	Oper	Bloc	Lim. haute Lim. basse	10 000 -10 000
Rate	**REAL**	0,0	Oper	Oper	Lim. haute Lim. basse	1 000 0
Rate_Units	**ENUM**	/ Second (0)	Oper	Config	Détect.	/ Second (0) / Minute (1) / Hour (2) / Day (3)
Setpoint	**REAL**	0,0	Oper	Oper	Lim. haute Lim. basse	10 000 -10 000

Tableau 26-1 Attributs des paramètres de Rate_Limit

BLOC FONCTION RAMP

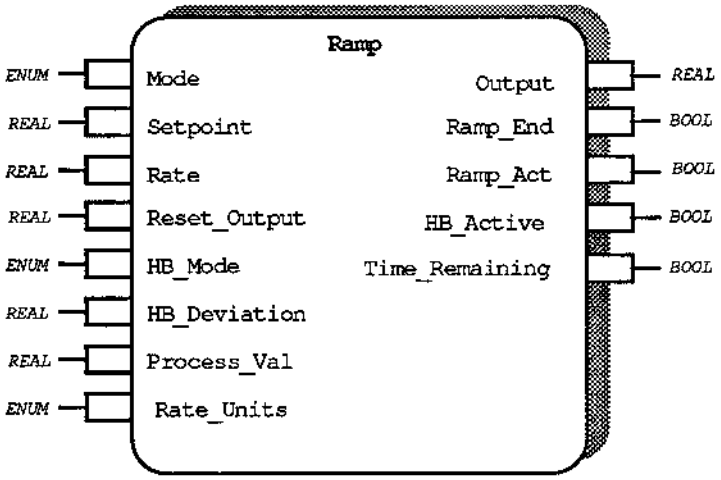


Figure 26-2 Schéma du bloc fonction Ramp

Description fonctionnelle

Le bloc fonction Ramp (rampe) fait varier linéairement la sortie Output à une vitesse Rate constante, en direction d'un point de consigne (Setpoint) visé. Le bloc a trois modes de fonctionnement qui sont définis par Mode. Une fonctionnalité de blocage (Holdback) peut être activée pour restreindre la variation en cas de Process_Val brusque.

Modes de fonctionnement

Reset (0) : en mode Reset (réinitialisation), Output est mis à Reset_Output et Ramp_Act est mis à No (0)

Hold (2) : en mode Hold (maintien), Output reste bloqué à la valeur atteinte avant d'entrer en mode Hold

Run (1) : en mode Run (marche), Output croît linéairement vers Setpoint à une vitesse définie par la paramètre Rate, et Ramp_Act est mis à Yes (1). Lorsque Output a atteint Setpoint, Ramp_End est mis à Vrai (1).

Fonctionnement du blocage

Holdback (blocage) sert à maintenir Output à une valeur constante, si l'écart entre Output et Process_Val dépasse la valeur définie par HB_Deviation. La Figure ci-après montre le fonctionnement du maintien en réponse à une Process_Val en retard sur la rampe de montée de Output. Si l'entrée Process_Val s'écarte de la rampe de Output d'une valeur supérieure à HB_Deviation, l'accroissement de Output est maintenu à une valeur constante jusqu'à ce que l'écart redevienne inférieur à HB_Deviation. Dans le cas représenté, ceci a pour effet de limiter la pente de la rampe pour qu'elle ne dépasse pas la vitesse de variation de Process_Val.

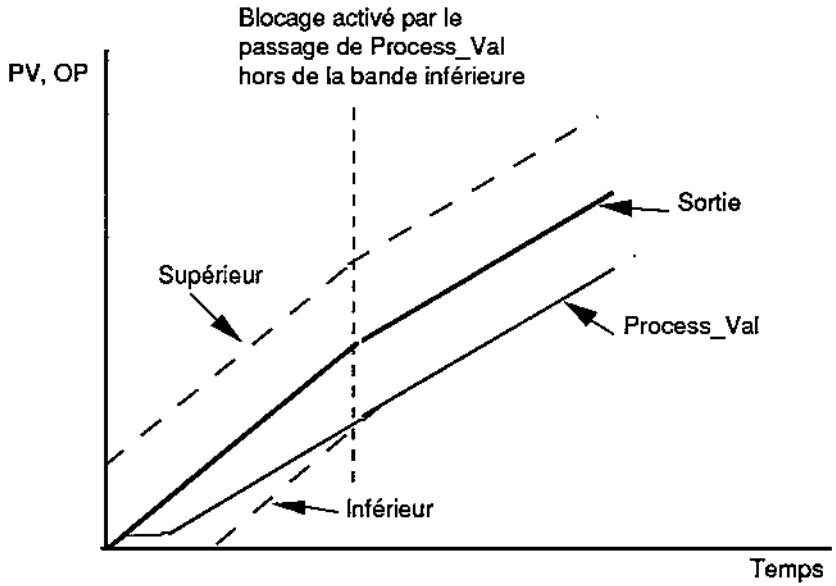


Figure 26-3 Opération de blocage avec Process_Val en retard

Modes de fonctionnement du blocage (Holdback)

OFF (0) : en mode OFF, l'option de blocage est désactivée.

LOWER (1) : en mode LOWER (inférieur) le blocage est activé si Output moins Process_Val est supérieur à HB_Deviation.

UPPER (2) : en mode UPPER (supérieur) le blocage est activé si Process_Val moins Output est supérieur à HB_Deviation.

BAND (3) : en mode BAND (bande) le blocage est activé si la valeur absolue de Output moins Process_Val est supérieure à HB_Deviation.

Attributs du bloc fonction

Type : F8 48
Classe : ..... DIVERS
Tâche par défaut : Tsk100ms
Liste récapitulative : ..... Mode, Setpoint, Rate, Output
Besoins de capacité mémoire : ..... 80 octets
Durée d'exécution : ..... 282 µs

Description des paramètres

Mode (M)

Le paramètre Mode définit le mode de fonctionnement du bloc fonction, selon les définitions précédentes.

Setpoint (SP)

Le paramètre Setpoint est la cible vers laquelle se dirige la rampe de Output.

Rate (R)

Le paramètre Rate définit la vitesse avec laquelle doit varier Output. Les unités de Rate sont définies par Rate_Units.

Reset_Output

Le paramètre Reset_Output définit la valeur écrite dans le paramètre Output lorsque Mode est le mode Reset.

HB_Mode

Le paramètre HB_Mode définit le mode de fonctionnement du blocage.

HB_Deviation

Le paramètre HB_Deviation définit la valeur de l'écart admis entre Output et Process_Val avant d'appliquer le blocage.

Process_Val (PV)

Le paramètre Process_Val fonctionne en association avec HB_Deviation pour déterminer si le blocage est actif. Ce paramètre n'est pas utilisé si HB_Mode est mis à Off (0).

Rate_Units

Le paramètre Rate_Units est utilisé pour définir l'unité dans laquelle est définie la vitesse de variation de Output.

Output (OP)

Le paramètre Output est la sortie réelle (Real) du bloc fonction Ramp qui rejoint Setpoint selon une rampe quand Mode est mis à Run (1) et quand le bloc fonction n'est pas à l'état bloqué. Output est égal à Reset_Output si Mode est mis à Reset (0).

Ramp_End (RE)

Le paramètre Ramp_End définit le point où Output a terminé sa rampe vers Setpoint. Quand Mode est égal à Reset (0), Ramp_End est mis à Faux (0). Quand Mode est égal à Run (1), Ramp_End est mis à Vrai (1) si Output est égal à Setpoint. Si Setpoint varie après que Ramp_End soit devenu Vrai (1), Ramp_End passe alors à Faux (0) jusqu'à ce que Output soit à nouveau égal à Setpoint. En mode blocage, le fonctionnement de Ramp_End est inchangé, avec Ramp_End mis à Faux (0) à moins que Output soit égal à Setpoint.

Ramp_Act (RA)

Le paramètre Ramp_Act définit si Output effectue une rampe en direction de Setpoint. Quand Mode est mis à Reset (0), Ramp_Act est mis à No (0). Quand Mode est mis à Run (1) ou Hold (2), Ramp_Act est mis à Yes (1) si Output n'est pas égal à Setpoint, et est mis à No (0) si Output est égal à Setpoint.

HB_Active

Le paramètre HB_Active est un indicateur qui signale l'action de la fonction de blocage. Quand le paramètre Mode est mis à Reset (0) ou Hold (2), HB_Active est égal à No (0). Si le paramètre Mode est mis à Run (1) et HB_Mode est mis à Lower (1), Upper (2) ou Band (3), HB_Active est égal à Yes (1) si le bloc est en situation de blocage.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
HB_Active	BOOL	No (0)	Oper	Bloc	Détect.	No (0) Yes (1)
HB_Deviation	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	999 999 - 99 999
HB_Mode	ENUM	Off (0)	Oper	Oper	Détect.	Off (0) Lower (1) Upper (2) Band (3)
Mode	ENUM	Reset (0)	Oper	Oper	Détect.	Reset (0) Run (1) Hold (2)
Output	REAL	0,0	Oper	Bloc	Lim. haute Lim. basse	999 999 - 99 999
Process_Val	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	999 999 - 99 999
Ramp_Act	BOOL	No (0)	Oper	Bloc	Détect.	No (0) Yes (1)
Ramp_End	BOOL	Faux (0)	Oper	Bloc	Détect.	Faux (0) Vrai (1)
Rate	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	100 000 0
Rate_Units	ENUM	/Second (0)	Oper	Oper	Valeurs énumérées	/Second (0) /Minute (1) /Hour (2) /Day (3)
Reset_Output	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	999 999 - 99 999
Setpoint	REAL	0,0	Oper	Oper	Lim. haute Lim. basse	999 999 - 99 999

Tableau 26-2 Attributs des paramètres de Ramp

BLOC FONCTION SHIFT_REAL

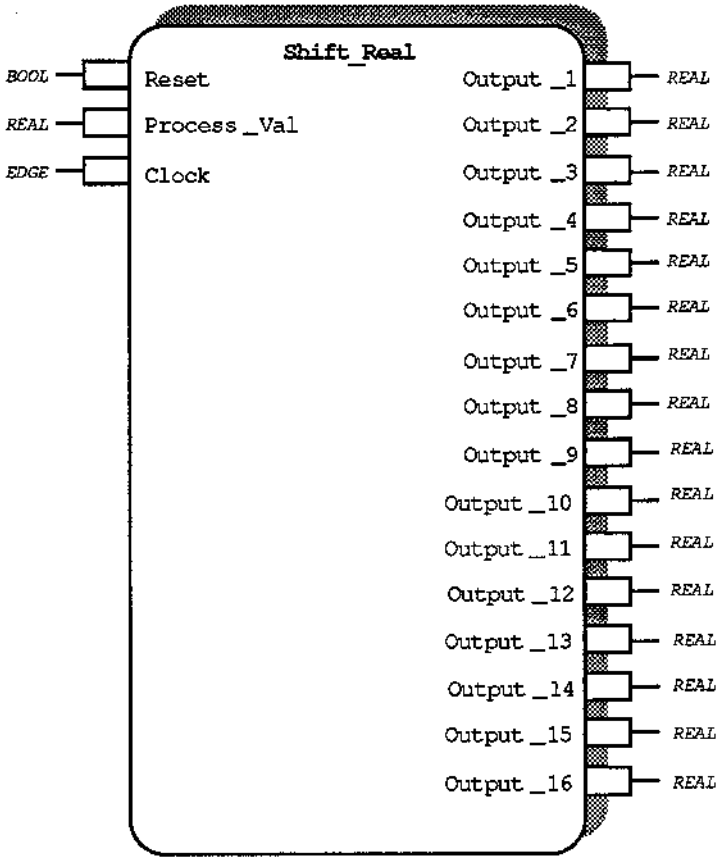


Figure 26-4 Schéma du bloc fonction Shift_Real

Description fonctionnelle

Le bloc fonction Shift_Real est un registre à décalage à 16 valeurs réelles (REAL). Le bloc fonction a 16 sorties réelles, Output_1 à Output_16, à travers lesquelles des valeurs sont décalées sur apparition d'un front montant en entrée du paramètre Clock (horloge).

Modes de fonctionnement

Le bloc fonction a deux modes de fonctionnement qui sont définis par le paramètre Reset :

Run (0) : en mode Run (Marche), l'action du registre à décalage est activée lorsque la valeur du paramètre d'entrée Clock (horloge) passe de Tock (0) à Tick (1). Sur réception de l'entrée Tick (1), les sorties du bloc fonction se décalent une fois selon la relation :

Sur Clock = Tick (1) ET DERNIER CLOCK = TOCK (0)

Output_*n* : = Output_*n*-1 pour ***n* = 2 à 16**

Output_1 : = Process_Val

Les sorties gardent ensuite leur valeur jusqu'à ce que Clock repasse de Tock (0) à Tick (1). Il est alors nécessaire de passer Clock de Tick (1) à Tock (0) entre les décalages des sorties.

Reset (1) : en mode Reset toutes les sorties sont remises à zéro. Les sorties sont alors maintenues à zéro jusqu'à ce que Mode repasse en Run (0).

Attributs du bloc fonction

Type : F8 40
Classe : ..... DIVERS
Tâche par défaut : ..... Task_2
Liste récapitulative : ..... Process_Val, Clock, Reset, Output_1
Besoins de capacité mémoire : ..... 74 octets
Durée d'exécution : ..... 22 µs

Description des paramètres

Reset (RST)

Le paramètre Reset définit le mode de fonctionnement (Mode) du bloc fonction.

Process_Val (PV)

Le paramètre Process_Val est l'entrée du bloc fonction.

Clock (CLK)

Quand le bloc fonction fonctionne en mode Run, le passage du paramètre Clock de Tock (0) à Tick (1) provoque un décalage des sorties. Il y a lieu de noter que la vitesse de variation du signal d'entrée doit être au moins le double de la durée de la tâche, car il est nécessaire de repasser Clock de Tick (1) à Tock (0) entre les décalages.

Output_1 à Output_16 (O1 à O16)

Les paramètres Output_1 à Output_16 sont les sorties dont les valeurs en virgule flottante (REAL) sont décalées en avançant d'une position à chaque fois l'entrée Clock passe de Tock (0) à Tick (1).

Output_16 peut être utilisé comme sortie de retenue. Des registres à décalage plus grands peuvent être obtenus en reliant ensemble par logiciel toutes les entrées horloge, et en raccordant le paramètre Output_15 du premier registre à décalage à l'entrée Process_Val du second. D'autres registres à décalage peuvent être raccordés de la sorte.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Reset	**BOOL**	Run (0)	Oper	Oper	Détect.	Run (0) Reset (1)
Process_Val	**REAL**	0,0	Oper	Oper	Lim. haute Lim. basse	999 999 - 99 999
Clock	**BOOL**	Tock (0)	Oper	Oper	Détect.	Tock (0) Tick (1)
Output_1 to Output_16	**REAL**	0,0	Oper	Bloc	Lim. haute Lim. basse	999 999 - 99 999

Tableau 26-3 Attributs des paramètres de Shift_Real

BLOC FONCTION SHIFT_DINT

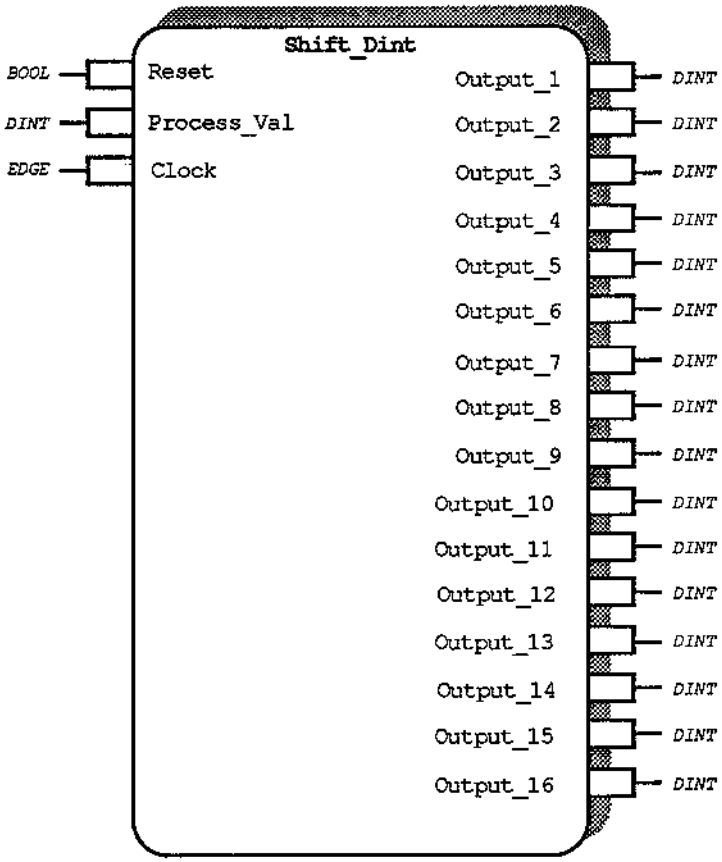


Figure 26-5 Schéma du bloc fonction Shift_Dint

Description fonctionnelle

Le bloc fonction Shift_Dint est un registre à décalage à 16 valeurs entières. Le bloc fonction a 16 sorties entières (DINT), Output_1 à Output_16, à travers lesquelles des valeurs sont décalées d'une position à la fois sur apparition d'un front montant en entrée du paramètre Clock (horloge).

Modes de fonctionnement

Le bloc fonction a deux modes de fonctionnement qui sont définis par le paramètre Reset :

Run (0) : en mode Run (Marche), l'action du registre à décalage est activée lorsque la valeur du paramètre d'entrée Clock (horloge) passe de Tock (0) à Tick (1). Sur réception de l'entrée Tick (1), les sorties du bloc fonction se décalent une fois selon la relation :

Sur Clock = Tick (1) ET DERNIER CLOCK = TOCK (0)

Output_*n* : = Output_*n*-1 pour *n* = 2 à 16

Output_1 : = Process_Val

Les sorties gardent ensuite leur valeur jusqu'à ce que Clock repasse de Tock (0) à Tick (1). Il est alors nécessaire de passer Clock de Tick (1) à Tock (0) entre les décalages des sorties.

Reset (1) : en mode Reset toutes les sorties sont remises à zeéro. Les sorties sont alors maintenues à zéro jusqu'à ce que Mode repasse en Run (0).

Attributs du bloc fonction

Type : F8 45

Classe : ..... DIVERS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Process_Val, Clock, Reset, Output_1

Besoins de capacité mémoire : ..... 74 octets

Durée d'exécution : ..... 22 µs

Description des paramètres

Reset (RST)

Le paramètre Reset définit le mode de fonctionnement (Mode) du bloc fonction.

Process_Val (PV)

Le paramètre Process_Val (valeur de processus) est l'entrée du bloc fonction.

Clock (CLK)

Quand le bloc fonction fonctionne en mode Run, le passage du paramètre Clock (horloge) deTock (0) à Tick (1) a pour effet de décaler les sorties d'une position. Il y a lieu de noter que la vitesse de variation du signal d'entrée doit être au moins le double de la durée de la tâche, car il est nécessaire de repasser Clock de Tick (1) à Tock (0) entre les décalages.

Output_1 à Output_16 (O1 à O16)

Les paramètres Output_1 à Output_16 sont les sorties dont les valeurs entières (DINT) se décalent en avançant d'une position à chaque fois que l'entrée Clock (horloge) passe de Tock (0) à Tick (1).

Output_16 peut être utilisé comme sortie de retenue. Des registres à décalage plus grands peuvent être obtenus en reliant ensemble par logiciel toutes les entrées horloge, et en raccordant le paramètre Output_15 du premier registre à décalage à l'entrée Process_Val du second. D'autres registres à décalage peuvent être raccordés de la sorte.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Reset	**BOOL**	Run (0)	Oper	Oper	Détect.	Run (0) Reset (1)
Process_Val	**DINT**	0	Oper	Oper	Lim. haute Lim. basse	999 999 - 99 999
Clock	**BOOL**	Tock (0)	Oper	Oper	Détect.	Tock (0) Tick (1)
Output_1 to Output_16	**DINT**	0	Oper	Block	Lim. haute Lim. basse	999 999 - 99 999

Tableau 26-4 Attributs des paramètres de Shift_Real

BLOC FONCTION SHIFT_16

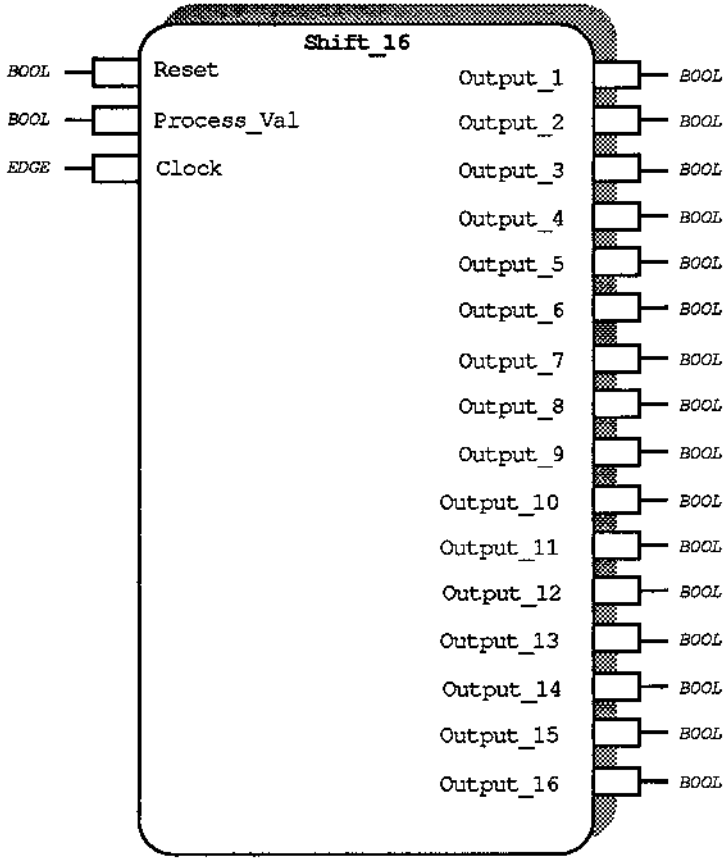


Figure 26-6 Schéma du bloc fonction Shift_16

Description fonctionnelle

Le bloc fonction Shift_16 est un registre à décalage à 16 bits. Le bloc fonction a 16 sorties booléennes, Output_1 à Output_16, à travers lesquelles des valeurs sont décalées d'une position à la fois sur apparition d'un front montant en entrée du paramètre Clock (horloge).

Modes de fonctionnement

Le bloc fonction a deux modes de fonctionnement qui sont définis par le paramètre Reset :

Run (0) : en mode Run (Marche), l'action du registre à décalage est activée lorsque la valeur du paramètre d'entrée Clock (horloge) passe de Tock (0) à Tick (1). Sur réception de l'entrée Tick (1), les sorties du bloc fonction se décalent une fois selon la relation :

Sur Clock = Tick (1) ET DERNIER CLOCK = TOCK (0)

Output_*n* : = Output_*n*-1 pour *n* = 2 to 16

Output_1 : = Process_Val

Les sorties gardent ensuite leur valeur jusqu'à ce que Clock repasse de Tock (0) à Tick (1). Il est alors nécessaire de passer Clock de Tick (1) à Tock (0) entre les décalages des sorties.

Reset (1) : en mode Reset toutes les sorties sont remises à zeéro. Les sorties sont alors maintenues à zéro jusqu'à ce que Mode repasse en Run (0).

Attributs du bloc fonction

Type : 248 16

Classe : ..... DIVERS

Tâche par défaut : ..... Tsk_10ms

Liste récapitulative : ..... Process_Val, Clock, Reset, Output_1

Besoins de capacité mémoire : ..... 20 octets

Description des paramètres

Reset (RST)

Le paramètre Reset définit le mode de fonctionnement (Mode) du bloc fonction.

Process_Val (PV)

Le paramètre Process_Val (valeur de processus) est l'entrée du bloc fonction.

Clock (CLK)

Quand le bloc fonction fonctionne en mode Run, le passage du paramètre Clock (horloge) deTock (0) à Tick (1) a pour effet de décaler les sorties d'une position. Il y a lieu de noter que la vitesse de variation du signal d'entrée doit être au moins le double de la durée de la tâche, car il est nécessaire de repasser Clock de Tick (1) à Tock (0) entre les décalages.

Output_1 à Output_16 (O1 à O16)

Les paramètres Output_1 à Output_16 sont les sorties dont les valeurs se décalent en avançant d'une position à chaque fois que l'entrée Clock (horloge) passe de Tock (0) à Tick (1).

Output_16 peut être utilisé comme sortie de retenue. Des registres à décalage plus grands peuvent être obtenus en reliant ensemble par logiciel toutes les entrées horloge, et en raccordant le paramètre Output_15 du premier registre à décalage à l'entrée Process_Val du second. D'autres registres à décalage peuvent être raccordés de la sorte.

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Reset	BOOL	Run (0)	Oper	Oper	Détect.	Run (0) Reset (1)
Process_Val	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Clock	BOOL	Tock (0)	Oper	Oper	Détect.	Tock (0) Tick (1)
Output_1 to Output_16	BOOL	Off (0)	Oper	Block	Détect.	Off (0) On (1)

Tableau 26-5 Attributs des paramètres de Shift_16

BLOC FONCTION ALARM_CNTRL

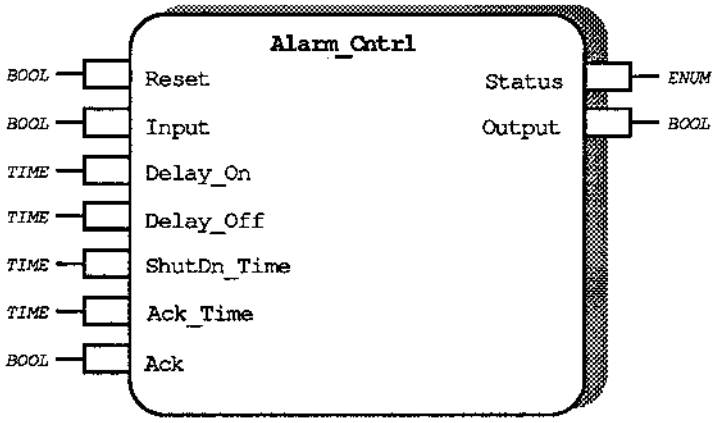


Figure 26-7 Schéma du bloc fonction Alarm_Cntrl

Description fonctionnelle

Le bloc fonction Alarm_Cntrl définit la sortie Status conformément à l'état de l'alarme câblée en entrée. Status peut avoir quatre états différents : Clear (0), Alarm (1), Ack (2) et Shut_Dn (3).

A partir de l'état initial Clear (0) (initialisation), si Input (entrée) est On (1) pendant une durée continue supérieure à Delay_On (temporisation d'activation), Status passe à Alarm (1). Si Input reste alors à On (1) pendant une autre durée continue supérieure à ShutDn_Time (temporisation de mise en alarme absolue), Status passe à Shut_Dn et Output est mis à On (1). Toutefois, si au cours de cette seconde temporisation avant mise en alarme absolue, l'alarme est acquittée, en passant Ack (aquittement) de Not_Ack (0) à Ack (1), la temporisation de mise en alarme absolue est remise à zéro et maintenue à zéro pour une durée égale à Ack_Time (temporisation d'aquittement), après quoi la temporisation de mise en alarme absolue est à nouveau relancée. La temporisation de mise en alarme absolue et la temporisation d'acquittement d'alarme ne peuvent être annulées qu'au moyen de Ack sur transition positive de Not_Ack (0) à Ack (1). Pendant la temporisation d'acquittement d'alarme, Status est mis à Ack (2).

N.B. : Si Status est mis à Alarm (1), Ack (2) ou Shut_Dn (3) et si Input repasse de On (1) à Off (0), Status ne repasse à Clear (0) que si Input est resté Off (0) pendant une durée supérieure à Delay_Off. Il en résulte que si la temporisation de Delay_Off est supérieure à celle de ShutDn_Time, le retour de Input de On (1) à Off (0) pendant un état d'alarme n'empêche pas la mise en alarme absolue, car la temporisation de mise en alarme absolue sera écoulée avant que la temporisation de réinitialisation ait atteint Delay_Off. Cette situation peut être évitée en choisissant avec soin Delay_On, Delay_Off et ShutDn_Time.

Attributs du bloc fonction

Type : ..... f 8 50

Classe : ..... DIVERS

Tâche par défaut : ..... Task_2

Liste récapitulative : ..... Input, Reset, Status, Output

Besoins de capacité mémoire : ..... 82 octets

Description des paramètres

Reset (RES)

Quand Reset est mis à Run (0) le bloc fonction fonctionne normalement. Le réglage de Reset à Reset (1) met toutes les temporisations à zéro, met Status à Clear (0) et Output à Off (0).

Input (IN)

Input est l'entrée de la condition d'alarme sur le bloc fonction.

Delay_On (DON)

Delay_On définit la durée pendant laquelle Input doit être On (1) avant que Status ne passe de Clear (0) à Alarm (1).

Delay_Off (OFD)

Si Input est mis à Off (0) pendant une durée continue supérieure ou égale à Delay_Off, Status est mis à Clear (0), quel que soit son état précédent.

ShutDn_Time (SDT)

Quand Status entre en condition Alarm (1) une temporisation de mise en alarme absolue est lancée. Si Status est à l'état Alarm (1) pendant une durée continue supérieure ou égale à ShutDn_Time, Status est mis à Shut_Dn (3).

La temporisation de mise en alarme absolue peut être remise à zéro en utilisant Ack, qui déclenche la fonctionnalité d'acquittement d'alarme.

Ack_Time (AT)

Lorsque Status est en alarme, sa temporisation de mise en alarme absolue peut être remise à zéro en utilisant Ack. Celui-ci passe ensuite Status à Ack (2) et le maintient ainsi pendant une durée égale à Ack_Time. Quand la durée de Ack_Time est écoulée, la temporisation de mise en alarme absolue repart de zéro.

Ack (ACK)

Lorsque Status est dans l'état Alarm (1), il peut être acquitté par une transition positive de Ack de Not_Ack (0) à Ack (1).

Status (ST)

Status peut prendre quatre états différents :

Clear (0) :	L'entrée à été à Off (0) pendant une durée supérieure à Delay_Off.
Alarm (1) :	L'entrée a été à On (1) pendant une durée supérieure à Delay_On. S'il n'a pas été acquitté avant que la période de mise en alarme absolue ne soit expirée, Status passe à Shut_Dn (3).
Ack (2) :	L'état d'alarme a été acquitté. Status reste à cet état jusqu'à ce que le délai d'acquittement soit écoulé, puis il repasse à Alarm (1).
Shut_Dn (3) :	L'état d'alarme a été mis à Alarm (1) pendant un délai supérieur à ShutDn_Time sans acquittement.

Output (OP)

Output est mise à On (1) si Status est Shut_Dn (3). Pour tous les autres états, Output est mis à Off (0).

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Reset	BOOL	Run (0)	Oper	Oper	Détect.	Run (0) Reset (1)
Input	BOOL	Off (0)	Oper	Oper	Détect.	Off (0) On (1)
Delay_On	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23d_23h_59m 0
Delay_Off	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23d_23h_59m 0
ShutDn_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23d_23h_59m 0
Ack_Time	TIME	0 ms	Oper	Oper	Lim. haute Lim. basse	23d_23h_59m 0
Status	ENUM	Clear (0)	Oper	Bloc	Détect.	Clear (0) Alarm (1) Ack (2) Shut_Dn (3)
Output	BOOL	Off (0)	Oper	Bloc	Détect.	Off (0) On (1)

Tableau 26-6 Attributs des paramètres de Alarm_Cntrl

BLOC FONCTION BISTABLE_SD

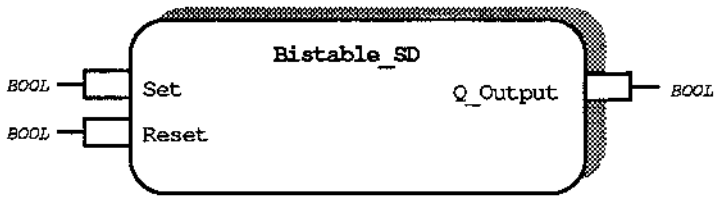


Figure 26-8 Schéma du bloc fonction Bistable_SD

Description fonctionnelle

Le bloc fonction Bistable_SD assure la fonction d'une bascule numérique bistable à S (mise à 1) dominant. Le bloc fonction a deux entrées booléennes, Set and Reset, (S = mise à un et R = mise à zéro) et une sortie booléenne (Q_Output). L'état de Q_Output dépend de celui de Set et de Reset. Si Set est Vrai (1), alors Q_Output est Vrai (1) quel que soit l'état de Reset. Si Set est Faux (0) et Reset est Vrai (1), alors Q_Output est Faux (0). Si Set and Reset sont à la fois Faux (0), alors l'état de Q_Output ne change pas de la valeur qu'elle avait avant que Set et Reset ne deviennent Faux (0).

Les états relatifs de Set, Reset et Q_Output sont résumés dans le Tableau ci-après.

Set (Mise à 1)	Reset (Mise à 0)	Q_Output (Sortie)
0	0	Pas de changement
1	0	1
1	1	1
0	1	0

Tableau 26-7 Etats de la bascule à S dominant

Attributs du bloc fonction

Type : ..... f 8 60

Classe : ..... DIVERS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Set, Reset, Q_Output

Besoins de capacité mémoire : ..... 4 octets

Durée d'exécution : ..... 10,6 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Q_Output	**BOOL**	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Reset	**BOOL**	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Set	**BOOL**	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)

Tableau 20-8 Attributs des paramètres de Bistable_SD

BLOC FONCTION BISTABLE_RD

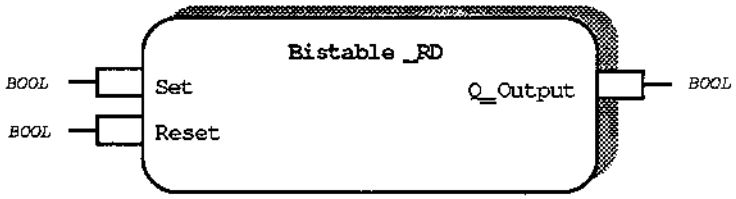


Figure 26-9 Schéma du bloc fonction Bistable_RD

Description fonctionnelle

Le bloc fonction Bistable_RD assure la fonction d'une bascule numérique bistable à R (mise à 0) dominant. Le bloc fonction a deux entrées booléennes, Set and Reset, (S = mise à un et R = mise à zéro) et une sortie booléenne (Q_Output). L'état de Q_Output dépend de celui de Set et de Reset. Si Reset est Vrai (1), alors Q_Output est Faux (0) quel que soit l'état de Set. Si Reset est Faux (0) et Set est Vrai (1), alors Q_Output est Vrai (1). Si Set and Reset sont à la fois Faux (0), alors l'état de Q_Output ne change pas de la valeur qu'elle avait avant que Set et Reset ne deviennent Faux (0).

Les états relatifs de Set, Reset et Q_Output sont résumés dans le Tableau ci-après..

Set (Mise à 1)	Reset (Mise à 0)	Q_Output (Sortie)
0	0	Pas de changement
1	0	1
1	1	0
0	1	0

Tableau 26-9 Etats de la bascule à R dominant

Attributs du bloc fonction

Type : ..... f 8 70

Classe : ..... DIVERS

Tâche par défaut : ..... Task_1

Liste récapitulative : ..... Set, Reset, Q_Output

Besoins de capacité mémoire : ..... 4 octets

Durée d'exécution : ..... 10,6 µs

Attributs des paramètres

Nom	Type	Démarrage à froid	Accès en lecture	Accès en écriture	Informations propres au type	
Q_Output	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Reset	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)
Set	BOOL	Faux (0)	Oper	Oper	Détect.	Faux (0) Vrai (1)

Tableau 26-10 Attributs des paramètres de Bistable_RD

EUROTHERM AUTOMATION SERVICE REGIONAL

SIÈGE SOCIAL ET USINE
6 chemin des Joncs
BP 55
69572 Dardilly Cedex

Tél. : 04 78 66 45 00
Fax : 04 78 35 24 90

AGENCES

Aix-en-Provence
Tél.: 04 42 39 70 31

Colmar
Tél.: 03 89 23 52 20

Lille
Tél.: 03 20 96 96 39

Lyon
Tél.: 04 78 66 45 10
04 78 66 45 12

Nantes
Tél.: 02 40 30 31 33

Paris
Tél.: 01 69 18 50 60

Toulouse
Tél.: 05 61 71 99 33

BUREAUX

Bordeaux
Clermont-Ferrand
Dijon
Grenoble
Metz
Normandie
Orléans

L'évolution de nos produits peut amener le présent document à être modifié sans préavis.

Imprimé en France 09/96