Système de mesure et de contrôle CR1000

Manuel d'utilisation

2.1.07
Traduction Février 2007

CSL 666

Garantie

La centrale de mesure CR1000 est garantie contre tout vice de matériau, de façon et de logiciel. Cette garantie demeurera en vigueur pendant une période de 3 ans (trente six mois) mois à compter de la date de livraison. Nous nous engageons à réparer ou à remplacer les produits jugés défectueux pendant la période de garantie, à condition qu'ils nous soient renvoyés port payé. Cette garantie ne pourra être appliquée :

- A aucun équipement modifié ou altéré de quelque manière que ce soit sans une autorisation écrite de Campbell Scientific.
- Aux batteries.
- A aucun produit soumis à une utilisation abusive, un mauvais entretien, aux dégâts naturels ou endommagements lors du transport.

Campbell Scientific renverra les équipements sous garantie par voie de terre, frais de transport payés. Campbell Scientific ne remboursera ni les frais de démontage ni les frais de réinstallation du matériel. Cette garantie et les obligations de la société citées ci-dessous remplacent toute autre garantie explicite ou implicite, y compris l'aptitude et l'adéquation à une utilisation particulière. Campbell Scientific décline toute responsabilité en cas de dommages indirects.

Avant de renvoyer un équipement, veuillez nous en informer pour obtenir un numéro de référence de réparation, que les réparations soient effectuées ou non dans le cadre de la garantie. Veuillez préciser la nature du problème le plus clairement possible et, si l'appareil n'est plus sous garantie, joindre un bon de commande. Un devis pour les réparations sera fourni sur demande.

Le numéro de référence de réparation doit être indiqué clairement à l'extérieur du carton utilisé pour renvoyer tout équipement.

Veuillez noter que les produits envoyés par avion sont sujets à des frais de dédouanement que Campbell Scientific facturera au client. Ces frais sont bien souvent plus élevés que le prix de la réparation proprement dite.

Campbell Scientific Ltd France
1 rue de Terre Neuve
Miniparc du Verger, Bât. H
91967 COURTABŒUF CEDEX
FRANCE

Tél. : (+33) 1 69 29 96 77
Fax : (+33) 1 69 29 96 65
Email : info@campbellsci.fr
http://www.campbellsci.fr

A propos de ce manuel – A LIRE EN PREMIER

Merci de noter que ce manuel est une version Française d'un document édité par Campbell Scientific Inc et destiné au marché Américain. Certaines orthographes, unités de mesures ou autre références peuvent refléter cette origine.
Voici de ce fait quelques facteurs de conversion qui pourront être utiles :

Surface : 1 in^2 (square inch) = 645mm^2

Longueur : 1 in. (inch) = 25,4mm
1 ft (foot) = 304,8mm
1 yard = 0,914m

Poids : 1 oz. (ounce) = 28,35g
1 lb (pound weight) = 0,454kg

Pression : 1 psi (lb/in^2) = 68,95mb

Volume : 1 UK pint = 568,3ml
1 UK gallon = 4,546 litres
1 US gallon = 3,785 litres

La plupart des informations du manuel sont valables pour tous les pays, mais certaines sont spécifiques aux marché Nord Américain et ne pourront pas être appliquées au marché Européen. Ceci est particulièrement vrai au sujet des alimentations externes et des adaptateurs secteurs, qui en Europe ne sont pas comprises avec les PS100 mais doivent être commandées séparément. *Notez tout de même que les adaptateurs secteurs proposés pour votre pays seront compatibles avec les alimentations externes proposées*.
Quelques alimentations et coffrets listés dans ce manuel peuvent ne pas être vendus en Europe; dans certains cas des alternatives sont proposées. Les détails sur les alternatives proposées seront décrits dans des manuels séparés.
Pour de plus amples informations, merci de contacter Campbell Scientific Ltd.

Campbell Scientific Ltd,
Campbell Park, 80 Hathern Road,
Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141
Fax: +44 (0) 1509 601091
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

Campbell Scientific Ltd France,
1 rue de Terre Neuve, ,
Miniparc du Verger, Bât. H
91967 COURTABŒUF Cedex
France
Tel: +33 (0) 169299677
Fax: 33 (0) 169299665
Email: info@campbellsci.fr
www.campbellsci.fr

Sommaire

FIGURES

TABLEAUX

Aperçu de la CR1000 version B5/06 du manuel CR1000

La CR1000 permet d'avoir des capacités de mesures de précision à partir d'un matériel robuste alimenté par batterie. La CR1000 comprend un processeur et des entrées / sorties analogiques ou numériques. Le langage de programmation se rapproche du BASIC et comprend des fonctions de traitement et d'analyse. Les logiciels PC200W 3.0, PC400 1.2, ou LoggerNet 3.0 permettent de générer ou de modifier des programmes, de récupérer des données ou de les visualiser en temps réel.

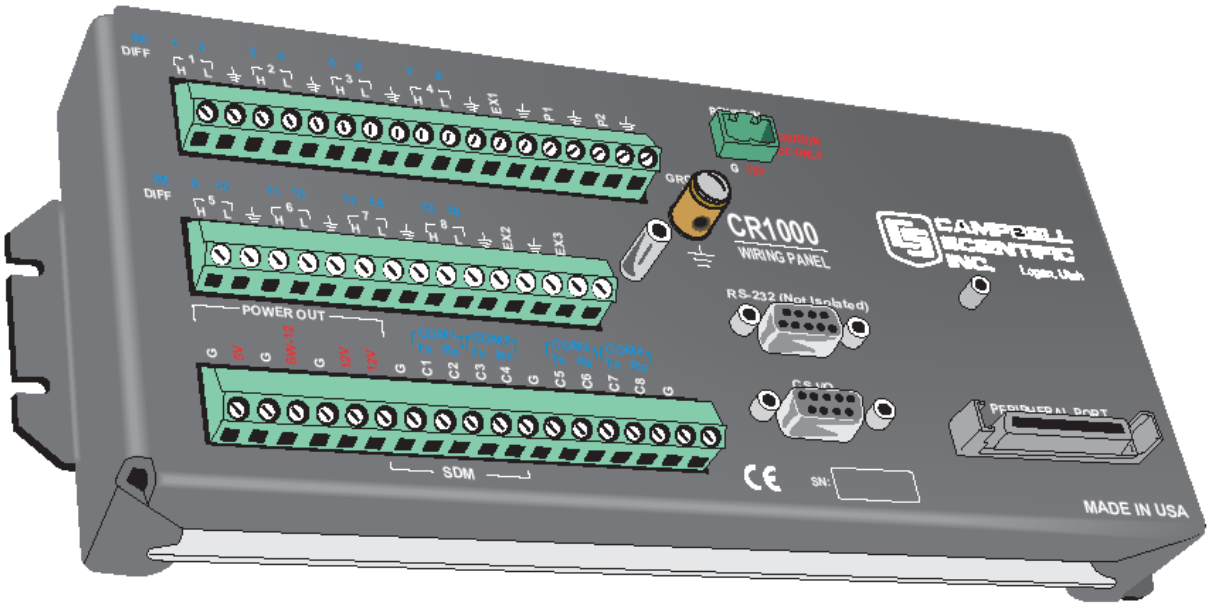


FIGURE OV1-1. Système de mesure et de contrôle CR1000

OV1. Description Physique

La figure OV1-2 montre le bornier de la CR1000 et les instructions de programme qui y sont associées. Le détail des instructions de mesure peut être trouvé au paragraphe 7.

OV1.1 Mesures en entrée

OV1.1.1 Mesures analogiques (Analog Inputs (SE 1-16, DIFF 1-8))

Il y a 8 entrées différentielles ou 16 entrées unipolaires pour effectuer des mesures de tension jusqu'à ±5 V. Une thermistance installée dans le bornier, peut être utilisée afin de mesurer la température de référence pour une mesure de thermocouple. Le bornier comprend aussi une barre de mise à la terre en cuivre, qui associée à l'architecture du bornier et de ses connecteurs, permet de réduire le gradient de température afin de fournir des mesures précises pour les thermocouples. La résolution pour l'étendue de mesure la plus sensible, est de 0,67 µV.

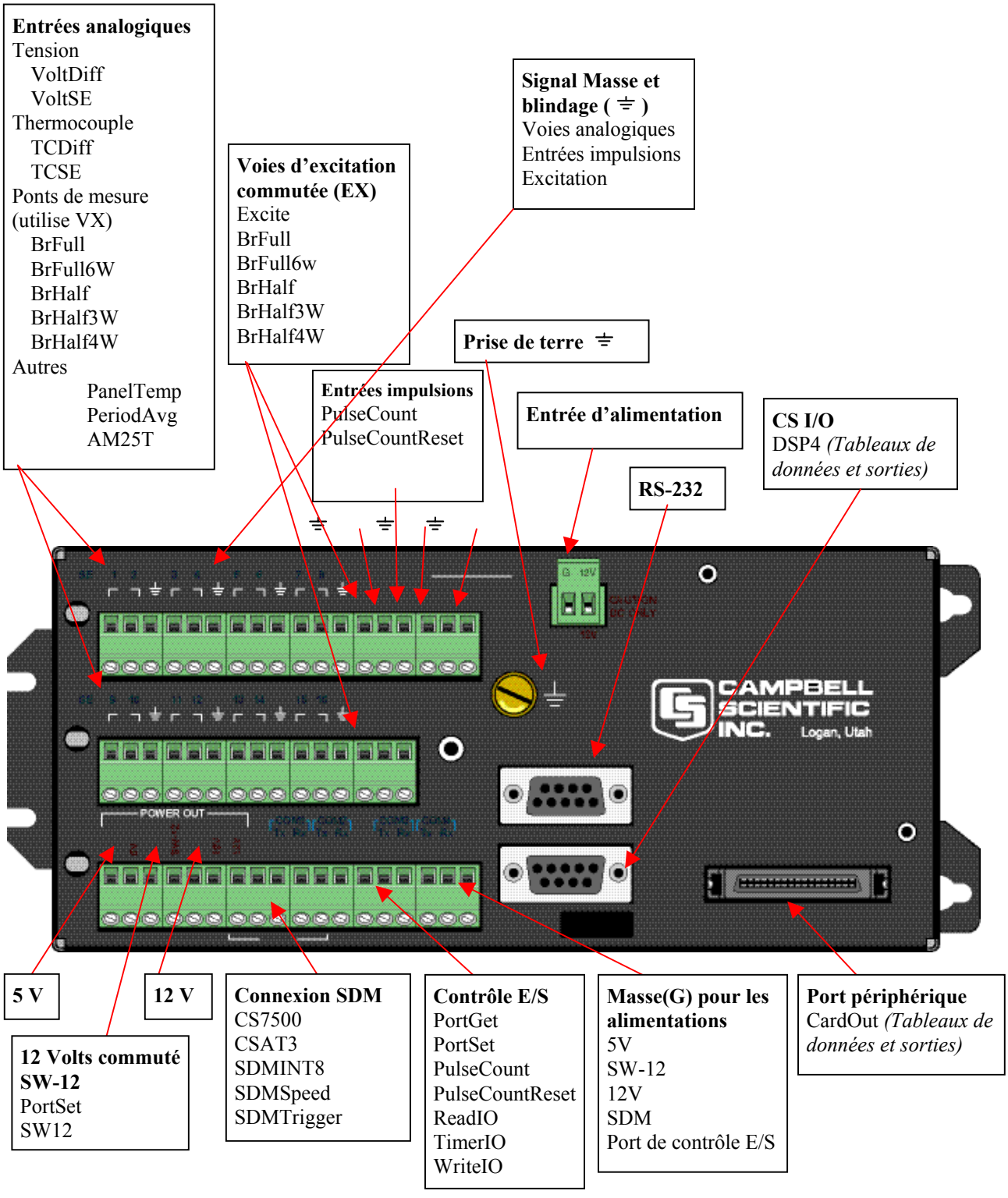


FIGURE OV1-2. Bornier de la CR1000 et instructions associées

OV1.1.2 Signal / masse et blindage (Signal / Shield Grounds) (⏚)

Les voies marquées ⏚ sont utilisées pour connecter les références de mise à la terre et les fils de blindage pour les voies analogiques unipolaires, les entrées d'impulsion (fréquence), les retours d'excitation. Il est préférable d'utiliser les bornes ⏚ à proximité des voies de comptage pour connecter le signal retour de ces entrées d'impulsion.

OV1.1.3 Masse d'alimentation / G (Power Ground / G)

Les voies G (masse d'alimentation) sont utilisées pour relier les masses des alimentations délivrées par les voies 5V, SW-12, 12V, et C1 à C8. L'utilisation de ces masses G de ces sorties ayant un large potentiel de courant minimisera l'intensité en courant sur la partie de mesure analogique, évitant ainsi d'entraîner des erreurs de mesure sur les voies unipolaires.

OV1.1.4 Prise de terre (Ground Lug) ⏚

La prise de terre est utilisée afin de relier un câble de section importante, à la terre. Une bonne connexion à la terre est nécessaire afin de fixer le potentiel de masse de la centrale de mesure, et pour transmettre à la terre les transitoires qui proviennent des voies du bornier, ou qui sont dirigées vers la masse lorsqu'ils sont déviés par les éclateurs à gaz, qui protègent les autres voies d'entrée.

OV1.1.5 Entrée d'alimentation (Power In)

Les voies G et 12V de la prise d'entrée d'alimentation, sont là afin de connecter à la CR1000 une alimentation provenant d'une batterie externe. C'est le seul emplacement où l'on peut amener de la tension en entrée ; les autres voies (12V et SW-12V) sont uniquement des sorties.

OV1.1.6 12V commuté (Switched 12 Volts SW-12)

La voie SW-12 fournit une alimentation 12V non régulée qui peut être commutée sous contrôle du programme.

OV1.1.7 Voies d'excitation commutée (Switched Voltage Excitation) EX

Trois sources d'excitation commutée permettent d'obtenir des tensions d'excitation précises et programmables jusqu'à ±2,5V afin d'effectuer des mesures de ponts de mesure. Chaque sortie analogique fournira jusqu'à 25mA à ±2,5 V.

OV1.1.8 Entrées / Sorties numériques (Digital I/O)

Il y a 8 voies d'entrées / sorties numériques (0V à l'état bas, 5V à l'état haut) permettant d'effectuer des mesures de fréquence, du comptage d'impulsion, du contrôle numérique et du changement d'état (triggering). En plus des fonctions associées à l'ensemble des voies d'entrée / sortie numérique, il y a certains groupes de voies qui ont des fonctions supplémentaires associées.

Les voies C1, C2 et C3, associées à 12V et G, permettent de connecter des capteurs ou des périphériques utilisant le protocole SDM (Synchronous Device for Measurement).

Les groupements COM peuvent être utilisés afin d'effectuer des communications série ou pour des entrées de capteurs intelligents.

OV1.1.9 Entrées d'impulsion (Pulse Inputs)

Deux voies d'entrée impulsion peuvent compter des impulsions à haut niveau (signal carré à 5V), des contacts sec ou du courant alternatif bas niveau.

OV1.2 Communication et stockage des données

OV1.2.1 Port pour périphérique (Peripheral Port)

Le port pour périphérique permet d'ajouter le support pour du stockage de donnée ou des périphériques de communication. Le CFM100 et le NL115 sont des modules qui se branchent sur le port périphérique; ils permettent alors d'ajouter des cartes Compact Flash de Type I ou II (voir paragraphe Section 2.1.2).

ATTENTION Le fait de retirer une carte présente dans un CFM100 ou un NL115 alors que la carte est active, peut engendrer des données erronées ou peuvent endommager la carte. Il faut toujours appuyer sur le bouton qui désactive la carte avant de couper l'alimentation de la CR1000.

OV1.2.2 CS I/O

Tous les périphériques de Campbell Scientific connectés aux ports de communication de la CR1000 utilisent un connecteur de type DB-9 (sub-D), étiqueté "CS I/O" (Figure OV1-3), qui se situe en face avant du bornier. Le tableau OV1-1 donne une rapide description de chaque broche (Pin).

TABLEAU OV1-1. Description des broches

ABR = Abréviation pour la fonction des broches.
PIN = Numéro de la broche.
S = Signal en sortie (**O**utput) de la CR1000 à un périphérique.
E = Signal en entrée (**I**nput) de la CR1000 provenant d'un périphérique.

PIN	ABR	I/O	Description
1	5 V	S	5V: Source 5 VCC, utilisée pour alimenter un périphérique.
2	SG		Signal Ground (masse) : Fournit un retour d'alimentation pour la broche 1 (5V), et s'utilise comme référence des niveaux de tension.
3	RING	E	Sonnerie: Activée par un périphérique pour mettre la CR1000 en mode télécommunication.
4	RXD	E	Receive Data (Réception des données) : Les données en série transmises par des périphériques sur la broche 4.
5	ME	S	Modem Enable (Activation du modem) : Activée lorsque la CR1000 détecte qu'un modem a activé la ligne d'appel.
6	SDE	S	Synchronous Device Enable (Activation d'appareil synchrone) : S'utilise pour adresser des appareils synchrones (Synchronous Devices (SD)), et peut aussi servir comme ligne de connexion à une imprimante.
7	CLK/HS	E/S	Clock/Handshake (Prise de main/liaison) : S'utilise avec les lignes SDE et TXD pour adresser des données aux SDs. Quand elle n'est pas utilisée comme horloge, la broche 7 peut servir de ligne de liaison (pendant une sauvegarde imprimante, l'état haut l'active, l'état bas l'a désactive).
8	+12 VCC	S	
9	TXD	S	Transmit Data (Transmission de données) : Les données en série sont transmises de la CR1000 aux périphériques par la broche 9; marquage logique bas (0V), écart logique haut (5V), ASCII standard asynchrone, 8 bits de données, pas de parité, 1 bit de début, 1 bit d'arrêt, 300, 1200, 2400, 4800, 9600, 19200, 38400, 115200 bauds (sélectionnable par l'utilisateur).

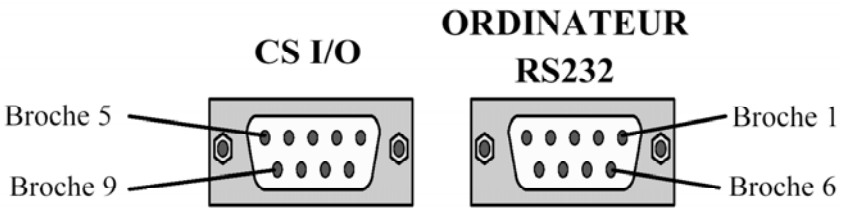


FIGURE OV1-3. Interfaces série de communication

OV1.2.3 RS-232 pour ordinateur (Computer RS-232)

Le port RS-232 de la CR1000 n'est pas isolé.

C'est un port RS-232 qui permet de se connecter directement à la plupart des ordinateurs (voir la figure OV1-3). Le tableau OV1-2 donne une brève description de chaque broche du connecteur "ordinateur" RS232.

Le port RS232 de la CR1000 est en DCE lorsqu'il est connecté à un PC avec un câble série. Il est à la fois en DTE, lorsqu'il est connecté à un modem par un câble null-modem. (la fonction DTR est sur la broche I, Ring est une entrée).

Entrée maximum = ± 25V
Sortie minimum = ± 5V
Sortie typique = ± 7V

NOTE Les câbles de communication série d'une longueur supérieure à 15m ne doivent pas être utilisés.

TABLEAU OV1-2. Broches RS-232

ABR = Abréviation pour la fonction des broches.
PIN = Numéro de la broche.
S = Signal en sortie (**O**utput) de la CR1000 à un appareil RS-232.
E = Signal en entrée (**I**nput) de la CR1000 provenant d'un appareil RS-232.

PIN	ABR	E/S	Description
1	DTR	S	data terminal ready
2	TX	S	asynchronous transmit
3	RX	E	asynchronous receive
4			Non connecté
5	GND		masse
6		S	connecté
7	CTS	E	clear to send
8	RTS	S	request to send
9	RING	E	ring

La CR1000 est livrée avec un câble série 9 broches de 2 mètres environ, afin de faciliter la connexion à un port RS-232 d'un PC.

OV1.3 Alimentation et adaptateur secteur

La CR1000 a besoin d'une alimentation 12V pour fonctionner. La PS100E-LA est une alimentation par batterie 7 Ah, avec un régulateur de charge intégré. Des adaptateurs secteur sont disponibles en option. L'énergie nécessaire à charger la batterie peut aussi provenir d'une entrée CC entre 17 et 28V, comme celle d'un panneau solaire par exemple.

OV2. Concept de la mémoire et de fonctionnement

OV2.1 Mémoire

La CR1000 a 2MB de Flash EEPROM qui est utilisée pour stocker le système d'exploitation. 128 K de mémoire Flash sont utilisés afin de stocker les paramètres de configuration. Un minimum de 2 MB de SRAM (ou 4MB en option) est disponible pour le stockage du programme (16k), l'utilisation du système d'exploitation, et le stockage des données. La taille mémoire disponible peut être affichée par le fichier d'état. De la mémoire additionnelle peut être disponible avec l'utilisation du CFM100 ou NL115, modules optionnels pour cartes Compact Flash.

OV2.2 Programmation

Le programme de la CR1000 détermine quand et comment les capteurs sont mesurés, et les données stockées. Le programme est créé sur ordinateur puis envoyé à la CR1000. La CR1000 peut stocker plusieurs programmes en mémoire. Campbell Scientific dispose de deux logiciels permettant de créer des programmes pour la CR1000 : ShortCut et l'éditeur CRBasic.

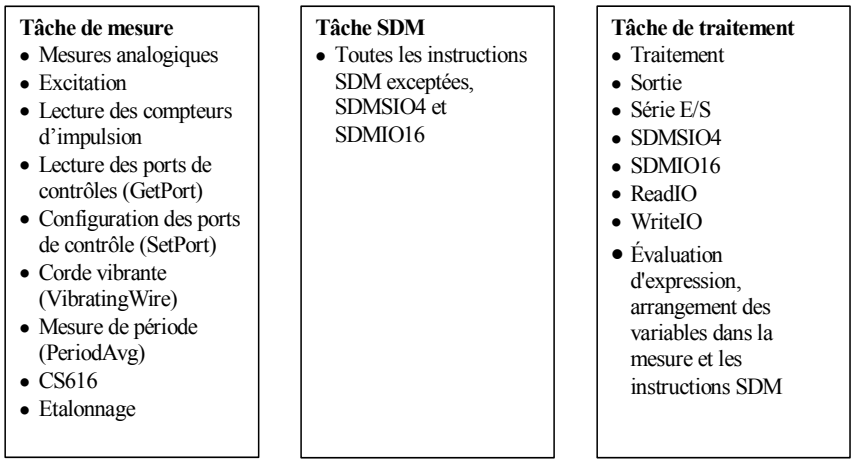
Pour un bon nombre d'applications, il est préférable de débuter la programmation avec ShortCut (ou SCWIN). Avec ShortCut vous êtes amené à sélectionner le capteur que vous souhaitez mesurer, l'unité de mesure dans laquelle vous souhaitez effectuer la mesure, puis le type de sauvegarde à y associer. ShortCut permet de programmer la plupart des capteurs commercialisés par Campbell Scientific, ainsi que des capteurs génériques. Les programmes pour CR1000 qui sont créés par ShortCut sont généralement clairs et fournissent un bon exemple de code en CRBasic, pour ceux qui souhaiteraient écrire le programme par eux même.

Pour les utilisateurs qui ont besoin ou souhaitent effectuer des programmes plus complexes, l'éditeur CRBasic sera utilisé afin de créer et d'éditer des programmes en CRBasic, que la CR1000 pourra alors faire fonctionner. Le paragraphe 4 est une introduction à la programmation en CRBasic. L'éditeur de CRBasic a une syntaxe, des couleurs et un menu d'aide en ligne pour le jeu d'instructions de la CR1000, qui sont décrits au paragraphe 5-12.

ShortCut est inclus aux logiciels PC200W, PC400 et LoggerNet, mais peut aussi être téléchargé gratuitement sur le site Internet de Campbell Scientific. L'éditeur CRBasic n'est présent qu'avec PC400 et LoggerNet.

OV2.3 Exécution des instructions dans la centrale de mesure

L'exécution des instructions dans une centrale de mesure est accomplie en utilisant 3 types de tâches différentes : la mesure, SDM, et le traitement. Comme son nom l'indique, la tâche de mesure a pour finalité de mesurer un signal reçu via le bornier de la centrale de mesure ; elle comprend aussi l'envoi de signaux de sortie pour la commande d'autres dispositifs. Les composants de mesure et de contrôle sont pilotés par une séquence synchronisée de façon stricte. La tâche SDM gère la mesure et le contrôle de la plupart des appareils SDM. La tâche de traitement convertit les valeurs brutes lues par la centrale de mesure en nombres représentant des grandeurs physique, elle effectue des calculs, stocke les données, prend les décisions pour actualiser les contrôles, et effectue les communications séries E/S.



La centrale de mesure peut exécuter ces tâches en mode « pipeline » ou séquentiel. Lorsqu'un programme est compilé la centrale de mesure évalue le programme et détermine quel mode sera utilisé. Cette information est incluse dans un message envoyé par la centrale de mesure et affiché par le logiciel. Le pré compilateur du CRBasic envoie aussi un message similaire. Un programme peut être configuré pour fonctionner en mode séquentiel, en utilisant l'instruction SequentialMode dans la section des déclarations du programme.

OV2.3.1 Mode « Pipeline »

Dans le mode « Pipeline », la tâche de mesure, la tâche SDM, et la tâche de traitement sont trois fonctions différentes. Dans ce mode ces trois tâches peuvent opérer simultanément. La tâche de mesure est planifiée pour prendre place selon un moment (« timing ») très précis et avec une priorité absolue lorsque la centrale de mesure démarre chaque scrutation (échantillonnage). Ce résultat selon un horodatage plus précis sur la mesure, améliore le traitement et optimise la consommation en énergie. Cependant, cette pré-planification des mesures oblige les instructions de mesure à être exécutées à chaque scrutation, et puisque les différentes tâches s'exécutent en même temps, la séquence dans laquelle les instructions son réellement exécutées ne sera peut être pas exactement le même que celle qui apparaît dans le programme. C'est pour cette raison, que les mesures conditionnelles ne sont pas permises dans le mode « pipeline ». Il est à noter aussi que de part le temps « timing » d'exécution des instructions de mesure, le traitement des mesures de l'échantillonnage en cours (incluant la mise à jour des variables « public » et des sorties pour les tableaux de données) est retardé jusqu'à ce que toutes les mesures soient effectuée.

OV2.3.2 Mode séquentiel

Dans le mode séquentiel les instructions sont exécutées dans la séquence décrite dans le programme. Le mode séquentiel peut être plus lent que le mode pipeline, car il ne traite qu'une partie du programme à la fois. Après qu'une mesure ait été faite il convertit la valeur déterminée par le traitement inclut dans l'instruction et ensuite la centrale de mesure passe à la prochaine instruction. Parce que l'exécution des ces instructions se fait étape par étape, les mesures conditionnelles sont permises en mode séquentiel. Le temps exact de chaque mesure peut varier si d'autres mesures ou traitements sont effectués sous condition, si il y a une importante activité de communication ou d'autres interruptions (comme par exemple l'insertion de la carte Compact Flash).

OV2.3.3 Scrutations à séquence lente (Slow Sequence Scans)

La centrale de mesure a la possibilité d'effectuer une ou plusieurs scrutations (échantillonnages) qui seront écrites et exécutées en dehors des instructions placées entre « Scan/NextScan ». Ces scrutations, répertoriées comme des scrutations à séquence lente, fonctionnent typiquement à un niveau plus lent que la scrutation principale. Jusqu'à quatre séquences lentes peuvent être définies dans un programme (les séquences lentes sont déclarées avec l'instruction SlowSequence).

Les instructions dans une scrutation à séquence lente sont exécutées lorsque la scrutation principale n'est pas active. Lorsque le mode pipeline fonctionne, les mesures de séquence lente seront découpées et exécutées à la suite des mesures du programme principal, en fonction du temps restant de disponible. En raison de ce mode de découpage, les mesures d'une séquence lente peuvent être réparties et exécutées au cours de plusieurs scrutations principales différentes. En mode séquentiel, toutes les instructions dans les séquences lentes sont exécutées dans l'ordre où elles sont écrites dans le programme (voir la tâche prioritaire, ci-dessous).

OV2.3.4 Tâche prioritaire (Task Priority)

Considérant les informations ci-dessus au sujet des modes « pipeline » et séquentiel, vous devez aussi considérer que certaines séquences dans le programme doivent avoir des priorités plus grandes que d'autres séquences du programme, et que généralement les tâches de mesure prennent le pas sur toutes les autres. De plus, les priorités de séquence sont différentes pour le mode « pipeline » et le mode séquentiel.

Lorsque le programme fonctionne en mode « pipeline », les tâches de mesure ont la priorité sur toutes les autres tâches. Les mesures dans le programme principal ont la priorité la plus élevée, puis l'étalonnage de fond, suivie par les mesures en séquences lentes si elles sont définies. L'exécution des tâches de traitement est gérée par un séquenceur de tâches, et toutes les tâches ont la même priorité. Lorsqu'une condition est validée pour démarrer une tâche, elle sera mise en file d'attente (cette condition validée peut être basée sur le temps, le déclenchement de WaitDigTrig, l'expiration de l'instruction Delay, ou un appel sur le port COM de la communication). Parce que les tâches ont la même priorité, la tâche est mise à la fin de la file d'attente. Toutes les 10 msec (ou plus rapidement si une nouvelle tâche est déclenchée) la tâche en cours d'exécution est suspendue et mise à la fin de la file d'attente, et la tâche suivante dans la file d'attente commence à s'exécuter. De cette manière, toutes les tâches ont un temps de traitement équivalent par la centrale de mesure. La seule exception à ce mode opératoire sera lorsqu'une tâche de mesures est déclenchée selon un basculement d'état (triggered). Si la centrale de mesure n'est pas capable d'accomplir une mesure lorsque le séquenceur de tâche s'exécute, la tâche sera suspendue jusqu'à ce que la mesure soit effectuée.

Lorsque le mode séquentiel fonctionne, la centrale de mesure utilise un système de file d'attente pour les tâches de traitement qui est similaires à celui utilisé dans le mode « pipeline ». La seule différence lorsque le programme fonctionne en mode séquentiel, est qu'il n'y a pas de pré planification du temps de mesure ; au lieu de cela, toutes les instructions sont exécutées dans l'ordre dans lequel elles sont inscrites dans le programme. Un arrangement prioritaire est utilisé pour éviter un conflit avec les composants destinés à la mesure. Dans cet arrangement la scrutation principale possède la priorité la plus élevée et empêche les autres séquences d'utiliser les composants de mesure tant que le programme principal n'est pas terminé (incluant le traitement). Les autres tâches, telles que le traitement d'autres séquences et les communications, peuvent se réaliser pendant que la séquence principale se déroule. Une fois que la scrutation principale est terminée les autres séquences ont accès aux composants de mesure ; la séquence d'étalonnage de fond s'effectuera alors prioritairement face aux séquences lentes, qui s'exécuteront alors avec l'ordre déclaré dans le programme. A noter que les tâches de mesure ont la priorité sur les autres tâches telles le traitement et la communication, afin de permettre une synchronisation précise du temps, dont ont besoin la plupart des instructions de mesure (par exemple les intégrations).

OV2.4 Tableaux de données (Data Tables)

La CR1000 peut stocker des mesures individuelles, ou bien elle peut utiliser ses capacités de traitement afin de calculer des moyennes, minimum ou maximum etc., à des intervalles de temps périodique ou selon des conditions particulières. Les données sont stockées dans des tableaux tels que celui qui est montré dans le tableau OV2-1. Les valeurs à enregistrer sont sélectionnées lorsqu'on crée le programme de la centrale de mesure par ShortCut ou CRBasic directement.

TABLEAU OV2-1. Exemple typique de tableau de données

TOA5	Fritz	CR1000	1079	CR1000.Std.1.0	CPU:TCTemp.CR1	51399	Temp	
TIMESTAMP	RECORD	RefT_Avg	TC_Avg(1)	TC_Avg(2)	TC_Avg(3)	TC_Avg(4)	TC_Avg(5)	TC_Avg(6)
TS	RN	DegC	DegC	DegC	DegC	DegC	DegC	DegC
		Avg	Avg	Avg	Avg	Avg	Avg	Avg
10/28/2004 12:10	119	23.52	23.49	23.49	23.5	23.49	23.5	23.5
10/28/2004 12:20	120	23.55	23.51	23.51	23.51	23.51	23.51	23.52
10/28/2004 12:30	121	23.58	23.52	23.53	23.53	23.53	23.53	23.53
10/28/2004 12:40	122	23.58	23.53	23.54	23.54	23.54	23.54	23.54

OV2.5 Communication PakBus avec la CR1000

La CR1000 utilise le protocole de communication réseau PakBus. PakBus accroît le nombre d'options de communication et de mise en réseau disponibles pour la centrale de mesure. En plus de la communication via les ports RS-232 et/ou CS I/O, la CR1000 peut aussi communiquer au travers des ports COM numériques E/S.

Certains des avantages de PakBus sont :

- Le routage – La CR1000 peut agir en tant que routeur et faire passer un message destiné à une autre centrale de mesure. PakBus permet de détecter et de sélectionner automatiquement un chemin de routage.
- La mise en réseau sur faible distance, sans matériel additionnel – Une CR1000 peut dialoguer avec une autre CR1000 sur une distance d'environ 10 mètres (30 pieds) en connectant 3 fils entre les centrales : transmission (TX), réception (RX), et masse. La communication provenant d'un PC sur une de ces centrales (par exemple via un modem téléphonique ou radio connecté au port CS I/O), peut être dirigée par la centrale qui reçoit la communication, vers l'autre centrale de mesure.
- Les communications de centrale de mesure à centrale de mesure – Des instructions PakBus spéciales permettent de simplifier les transferts de données entre les centrales de mesure afin d'obtenir une information ou d'effectuer des contrôles.

Tous les appareils qui envoient ou reçoivent des messages sur le réseau PakBus, doivent avoir une adresse PakBus unique. L'adresse par défaut de la CR1000 est « 1 ». Dans un réseau PakBus on devra donner à chaque centrale de mesure une adresse PakBus unique avant de l'installer sur le terrain. Pour qu'un logiciel PC (tel LoggerNet ou PC400) puisse communiquer avec une CR1000, celui-ci doit connaître le n° PakBus de la CR1000.

OV2.6 Configuration : Utilisation de “Device Configuration Utility” ou de clavier-écran

Lorsque vous recevez une CR1000 provenant de Campbell Scientific, elle devrait avoir l’adresse PakBus « 1 » par défaut. Si vous n’avez qu’une seule centrale PakBus ou que vous ne communiquez avec cette centrale de mesure que par lien direct (RS-232 ou téléphone), il n’y a pas de raison de changer cette adresse.

Cependant si cette centrale a été utilisée ou que quelqu’un l’a empruntée, il peut être nécessaire de vérifier quelle est son adresse ou quels sont ses autres paramètres de configuration. Il existe plusieurs moyens pour effectuer ces vérifications, mais les deux moyens les plus simples sont d’utiliser l’utilitaire de configuration « Device Configuration Utility » ou le clavier-écran CR1000KD.

OV3. Configuration des appareils (Device Configurator)

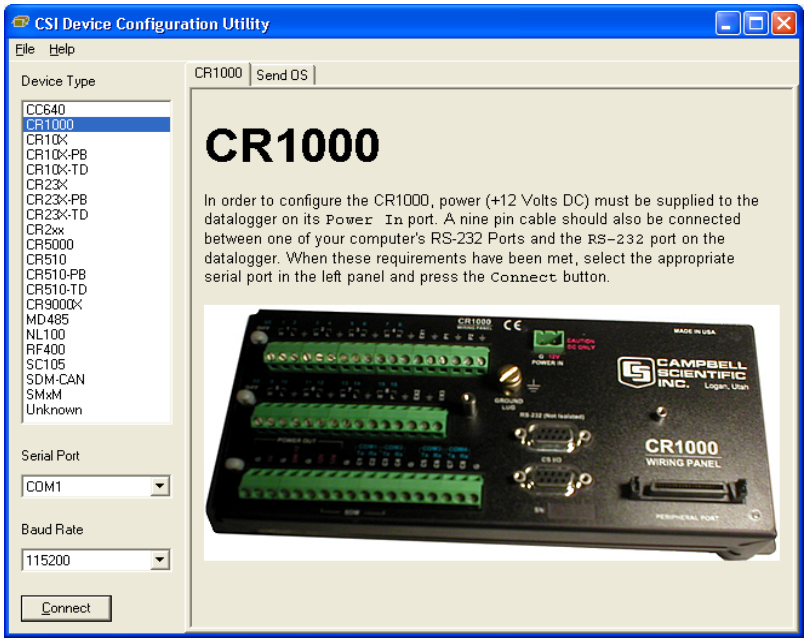
L’utilitaire de configuration des appareils (DevConfig) permet de configurer les centrales de mesure et les périphériques intelligents avant qu’ils ne soient installés dans les champs et avant que ces appareils soient intégrés à des réseaux définis dans les logiciels de Campbell Scientific tels LoggerNet ou PC400. Quelques éléments clé de DevConfig sont :

- DevConfig ne fonctionne que par connexion série directe entre le PC et les appareils.
- DevConfig sert à mettre à jour des systèmes d’exploitation (operating systems) uniquement pour les appareils connus, mais il peut aussi servir à mettre à jour l’horloge ou à envoyer un fichier programme.
- DevConfig vous permet de connaître le type de système d’exploitation et sa version.
- DevConfig permet d’effectuer un rapport de configuration pour un appareil, rapport qui peut être affiché à l’écran ou imprimé. Ce rapport peut aussi être sauvegardé dans un fichier, puis ré-utilisé afin de re-configurer un appareil ou afin d’appliquer à un éventuel appareil de remplacement, une configuration identique.
- Certains appareils peuvent ne pas supporter le protocole de configuration de DevConfig, mais pourront tout de même être configurés via l’écran d’émulateur de terminal.
- L’aide pour DevConfig est donnée à l’aide de lignes de conseils et d’explications directement sur l’écran principal de DevConfig. L’aide au sujet des configurations appropriées pour un appareil particulier, pourra alors être trouvée dans le manuel de l’appareil en question.
- Des mises à jour sont disponibles pour DevConfig à partir du site web de Campbell Scientific. Ces mises à jour peuvent être effectuées alors qu’une version précédente est déjà installée.

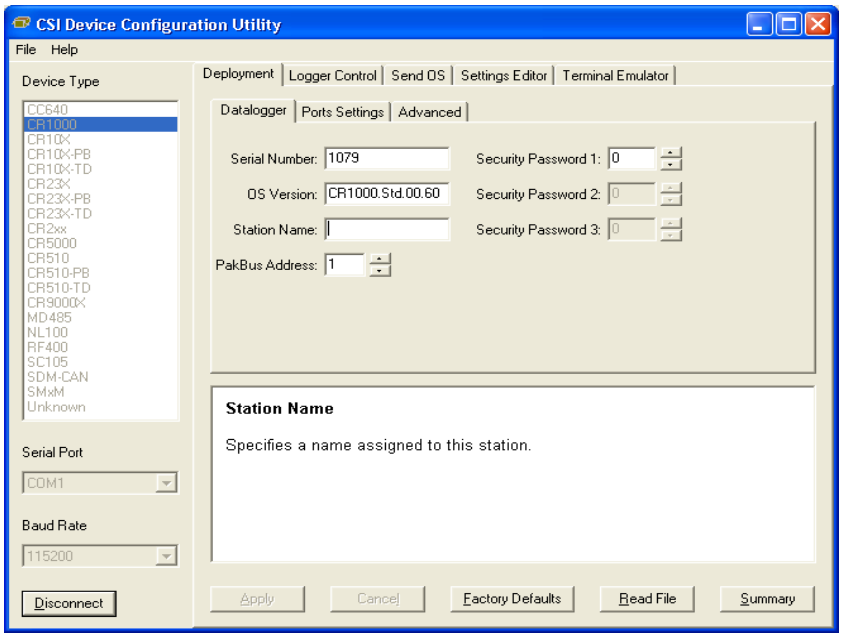
Note : Avant d’utiliser DeviceConfig, assurez-vous qu’aucun autre logiciel de votre ordinateur n’utilise les ports COM (PC200W, PC400 ou LoggerNet par exemple).

OV3.1 Ecran principal de DevConfig

La fenêtre de DevConfig est divisée en deux parties principales : la liste d'appareils à sélectionner sur la gauche, et les onglets de configuration sur la droite. Après avoir choisi un appareil sur la partie gauche de la fenêtre, vous aurez une liste de ports COM (COM1, COM2, etc.) installés sur votre PC. On vous proposera une liste de vitesses en baud dans le cas où l'appareil sélectionné supporte plusieurs vitesses en baud afin d'être configuré, et uniquement dans ce cas. La page dédiée à chaque appareil présente les instructions à propos de comment configurer l'appareil afin qu'il communique avec DevConfig. Différent types d'appareils auront sur la droite un ou plusieurs onglets.



Lorsque l'utilisateur appuie sur le bouton « Connect », le type d'appareil, le port série « Serial Port » et l'utilitaire de sélection de vitesse en baud « Baud Rate » sont désactivés et, si DevConfig est capable de se connecter à la CR1000, le bouton changera et passera de « Connect » (Connexion) à « Disconnect » (Déconnexion). L'affichage changera pour devenir :



OV3.2 Onglet “Deployment”

L’onglet « Deployment » permet à l’utilisateur de configurer sa centrale de mesure avant de l’installer sur le terrain.

OV3.2.1 Centrale de mesure (Datalogger)

Numéro de série (Serial Number) affiche le n° de série de la CR1000. Ce n° est donné en usine et ne peut pas être édité.

Version de système d’exploitation (OS Version) affiche la version du système d’exploitation qui est présent dans la CR1000.

Le nom de la station (Station Name) est alors affiché à cet emplacement.

PakBus Address vous permet de donner une adresse PakBus à la centrale de mesure. Les adresses possibles sont entre 1 et 4094. Chaque appareil PakBus devrait avoir une adresse unique. Les adresses supérieures à 3999 font en sorte que les autres appareils PakBus du réseau sont forcés de répondre, quelle que soit leur configuration. Merci de consulter le « PakBus Networking Guide » pour de plus amples informations.

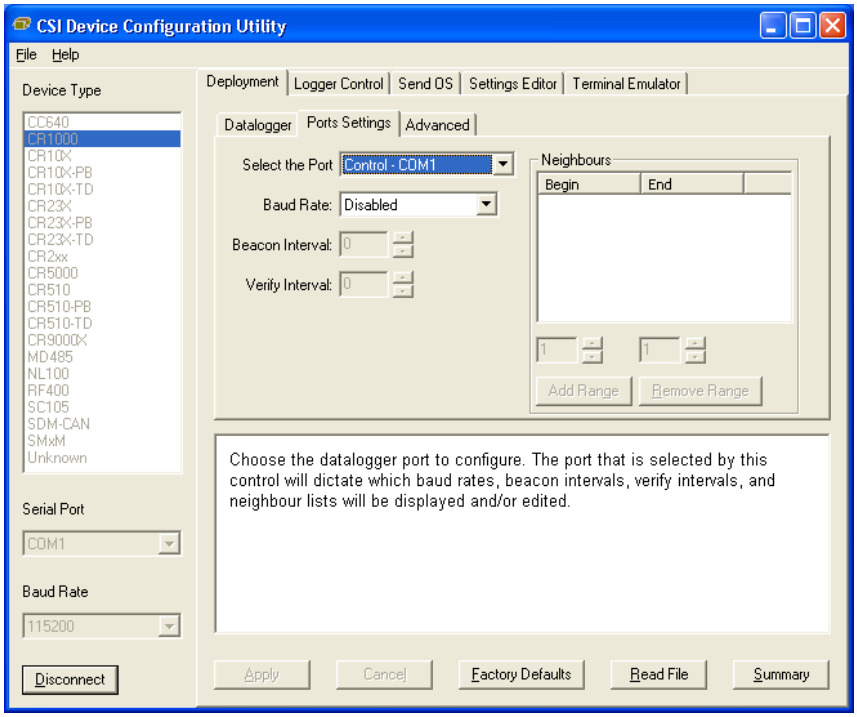
Sécurité (Security) :
Il est possible de mettre en place jusqu’à 3 niveaux de sécurité sur la CR1000. Le niveau 1 doit être mis en place avant que le niveau 2 ne puisse l’être, et le niveau 2 doit être actif si l’on veut mettre en place le niveau 3 de sécurité. Si un code de sécurité est fixé à 0 pour un certain niveau de sécurité, alors tous les niveaux supérieurs seront fixés à 0 (si par exemple on donne 0 comme code sécurité au niveau 2, et bien le code du niveau 3 sera aussi 0). Les codes sécurités valides sont compris entre 1 et 65535 (le 0 étant réservé à « pas de sécurité »). Chaque niveau de sécurité doit avoir son propre code. Les fonctions affectées par la mise en place des niveaux de sécurité sont :

Niveau 1 (Security Password 1) Lorsque ce niveau est activé, il est possible de collecter les données, de mettre à jour l’horloge de la centrale, et de donner des valeurs aux variables « Public » de la centrale, tout cela sans avoir à fournir de n° de sécurité. Si l’utilisateur entre le code de sécurité de niveau 1, il peut alors modifier le programme, le rapatrier sur son ordinateur, ou encore modifier les valeurs présentes dans le tableau d’état.

Niveau 2 (Security Password 2) Lorsque ce niveau de sécurité est activé, il est toujours possible de rapatrier des données sans avoir à donner de mot de passe. Si l’utilisateur donne le code de sécurité de niveau 2, l’horloge de la centrale de mesure pourra être mise à jour, et les variables du tableau « Public » pourront être modifiées. Si l’utilisateur entre le code de sécurité de niveau 1, il pourra alors modifier les valeurs qui ne sont pas en lecture seule dans le tableau d’état, et il pourra changer ou rapatrier le programme.

Niveau 3 (Security Password 3) Lorsque ce niveau de sécurité est activé, aucune communication n’est possible sans qu’au moins un des codes de sécurité ne soit entré. Si l’utilisateur entre le code de sécurité de niveau 3 il pourra collecter les données de la centrale de mesure. Si il entre le code de sécurité de niveau 2, il pourra collecter les données, mettre à jour les variables « Public » et mettre à jour l’horloge. S’il entre le code de sécurité de niveau 1 il a alors un accès intégral aux fonctionnalités de la centrale de mesure.

OV3.2.2 Configuration des Ports (Ports Settings)



Selected Port permet de spécifier le port série de la centrale de mesure sur lequel les données seront envoyées les valeurs de configuration des paramètres de « Beacon Interval » et « hello ».

Beacon Interval permet de définir l'intervalle (en secondes) auquel la centrale de mesure enverra sur le réseau une balise (beacon) à partir du port spécifié (sur « Select the Port »).

L'intervalle de vérification **(Verify Interval)** permet de définir l'intervalle (en secondes) auquel la centrale de mesure s'attendra à recevoir des paquets de données « packets » en provenance des voisins, sur le port spécifié à « Selected Port ». Une valeur de zéro (par défaut) indique que la centrale de mesure n'a pas de liste de voisins pour ce port.

Liste de voisins – centrales de mesure voisines (Neighbors List), ou bien d'une façon plus appropriée la « liste des voisins attendus », permet d'afficher la liste des adresses que cette centrale de mesure s'attend à trouver en tant que « voisins », sur le port spécifié au paramètre « Selected Port ». Lorsque vous sélectionnez des éléments sur cette liste, les valeurs de l'étendue de contrôle de début et de fin (**Begin** and **End** range controls) changeront afin de refléter l'étendue sélectionnée. Vous pouvez ajouter de multiples listes de voisins sur un même port.

Begin and End Range sont utilisés afin d'entrer une étendue d'adresses qui peut être ajoutée ou supprimée de la liste des « voisins », pour le port spécifié sur « Selected Port ». Lorsque vous manipulez ces paramètres de contrôle, les boutons « Add range » et « Remove Range » seront activés ou désactivés, selon que la valeur relative indiquée dans le paramètre de contrôle sera présente ou exempte de la liste d'adresses déjà configurées. Ces contrôles seront désactivés si l'intervalle de vérification (**Verify Interval**) est mis à la valeur zéro.

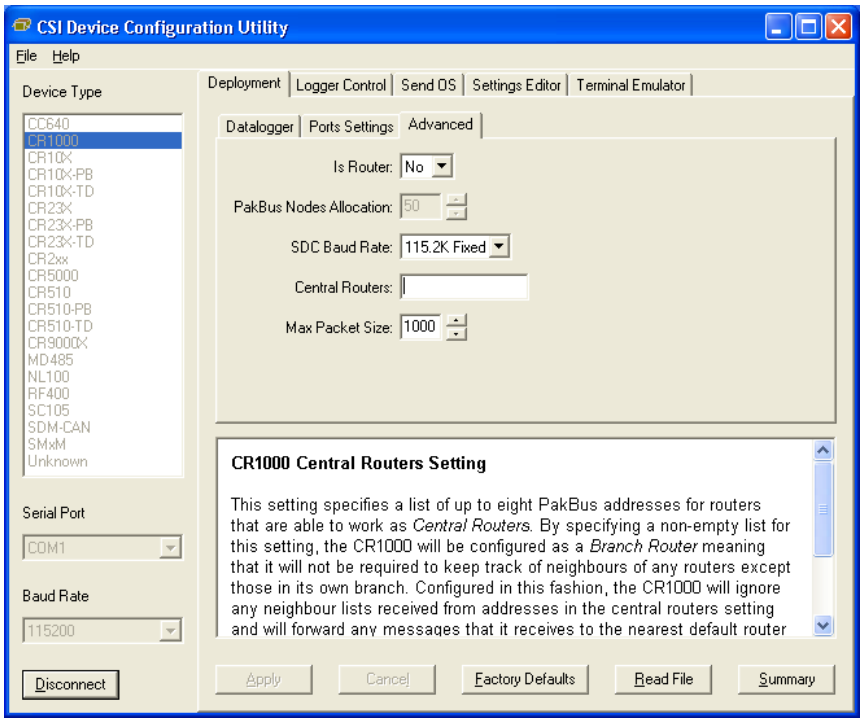
Add Range permettra d'ajouter à la liste de voisins et sur le port sélectionné dans « Selected Port », les numéros qui seront contenus dans l'étendue débutant par **Begin** et terminant par **End.** Ce contrôle sera désactivé si la valeur de l'intervalle de vérification est égale à zéro, ou si la valeur de « end » est inférieure à celle de « begin ».

Remove Range permet de supprimer une étendue de valeurs pour les voisins présents entre « begin » et « end », sur un port particulier. Ce contrôle est désactivé si l'étendue spécifiée n'est pas présente dans la liste, ou si l'intervalle de vérification est égal à zéro.

L'aide est affichée sur la partie inférieure de l'onglet « Deployment ». Une fois que vous avez terminé la configuration vous devez appuyer sur « **Apply** » (Appliquer) afin d'envoyer les configurations à la centrale. La fenêtre récapitulative (Summary window) sera affichée, et vous pourrez sauvegarder (**Save)** ou imprimer (**Print**) les configurations afin de les archiver ou de les ré-utiliser en tant qu'exemple sur d'autres centrales de mesure.

Annuler (Cancel) permet à la centrale de mesure d'ignorer les modifications qui ont été effectuées. **Read File** sert à charger un fichier de configuration qui aurait été sauvegardé auparavant pour cette centrale de mesure ou une autre du même type. Si vous chargez les paramètres à partir d'un fichier, les changements effectués ne seront réellement pris en compte qu'au moment où vous appuierez sur le bouton « **Apply** » (Appliquer).

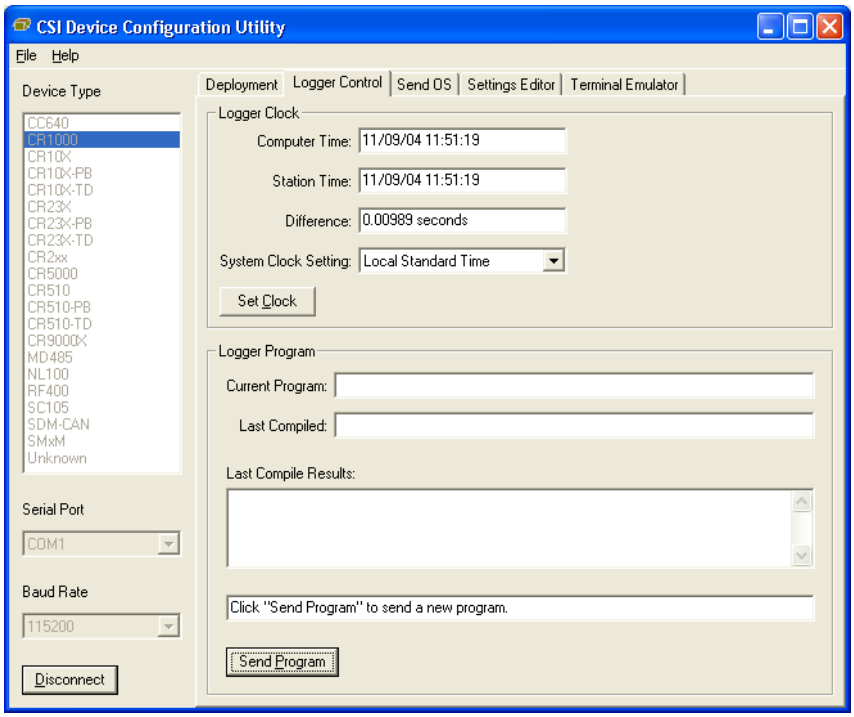
OV3.2.3 Avancé (Advanced)



Is Router vous permet de définir si la centrale de mesure agira ou non en tant que routeur PakBus.

PakBus Nodes Allocation permet de spécifier la quantité de mémoire que la CR1000 alloue afin de garder en mémoire les informations PakBus nécessaires au routage (PakBus Routing information). Cette valeur représente à peu près le nombre maximum de nœuds PakBus (PakBus Nodes) que la CR1000 sera capable de suivre dans son tableau de routage.

OV3.3 Onglet “Logger Control”



L’horloge du PC et celle de la centrale de mesure seront vérifiées chaque seconde, et la différence entre les deux sera affichée. Le système de mise à jour de l’heure (**System Clock Setting**) vous permet de configurer le type d’offset, s’il y en a, à ajouter à l’heure standard (Heure locale ou GMT). La valeur sélectionnée pour ce paramètre, sera mémorisée entre les sessions. Si on clique sur le bouton « **Set Clock Button** », l’heure de la centrale sera synchronisée avec celle du PC.

Current Program affiche le nom du programme qui fonctionne actuellement dans la centrale de mesure. Cette valeur sera vide si la centrale n’a pas de programme en fonctionnement

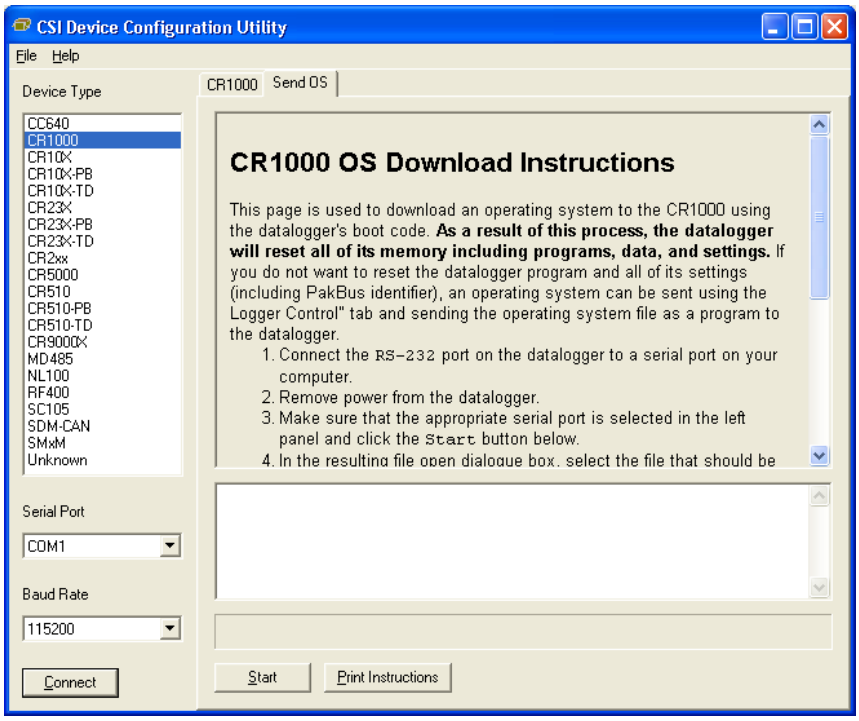
Le champ « **Last Compiled** » affiche l’heure à laquelle le programme qui est actuellement en cours d’exécution, a été compilé pour la dernière fois par la centrale de mesure. De même que pour le champ « Current Program », cette valeur ne sera affichée que si elle est disponible sur la centrale de mesure.

« **Last Compile Results** » affiche un rapport de la chaîne de résultats de compilation effectuée par la centrale de mesure.

Le bouton « **Send Program**» vous dirige vers une boite de dialogue permettant de sélectionner un programme à envoyer à la centrale de mesure. Le champs au dessus du bouton sera mis à jour une fois que l’envoi du fichier aura été effectué. Une fois qu’un programme a été envoyé, le nom du programme sera identique dans les champs « Current Program », « Last Compiled », et « Last Compile Results ».

OV3.4 Onglet “Send OS” – Télécharger un Système d’exploitation

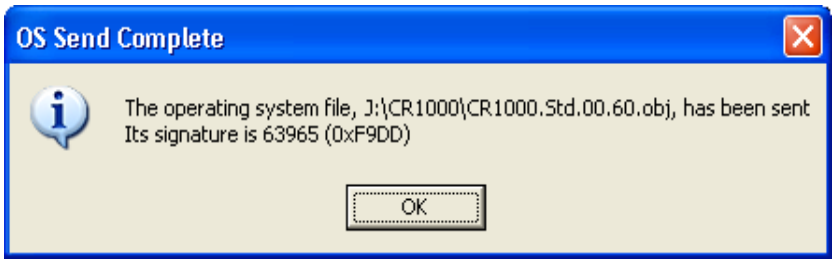
DevConfig permet d’envoyer un système d’exploitation vers tous les appareils de Campbell Scientific, qui sont munis d’un système d’exploitation sur mémoire flash. Un exemple est donné ci-dessous pour la CR1000 :



Le texte à droite donne les instructions afin de télécharger le système d’exploitation (OS). Il faut alors suivre ces instructions.

Lorsque vous cliquez sur le bouton « **Start** », DevConfig ouvre une fenêtre de dialogue afin de pointer sur un fichier de système d’exploitation (fichier *.obj). Lorsque la CR1000 et mise sous tension à nouveau, DevConfig débute l’envoi du système d’exploitation :

Lorsque le système d’exploitation a été envoyé, un message de dialogue similaire à celui présent ci-dessous, sera affiché :

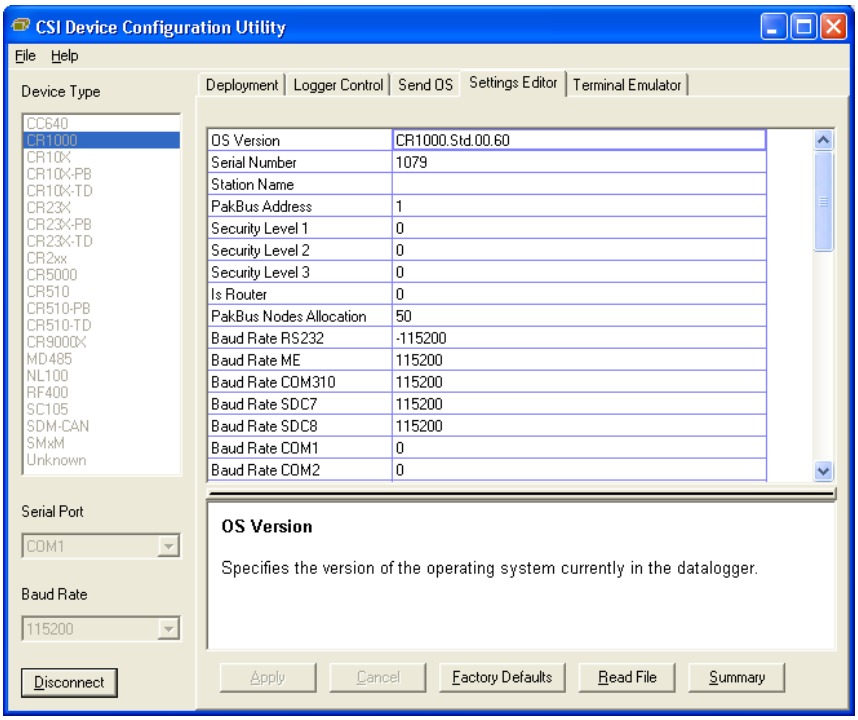


Les informations de la boite de dialogue permet de faire le lien avec la signature du système d’exploitation qu’on a envoyé. Pour des appareils tels que la CR10X (surtout celles qui ont une extension de mémoire), le temps de ré-initialisation après une mise à jour du système d’exploitation peut être long, et une fenêtre d’avertissement est alors affichée afin de rappeler qu’il ne faut pas interrompre le test effectué sur la mémoire.

OV3.5 Onglet “Settings Editor” – Editeur de configurations

La CR1000 possède plusieurs propriétés que l’on appelle des « configurations »; certaines sont spécifiques au protocole PakBus. La description de PakBus est d’ailleurs détaillée sur le site www.campbellsci.com , dans le document PakBus Networking Guide .

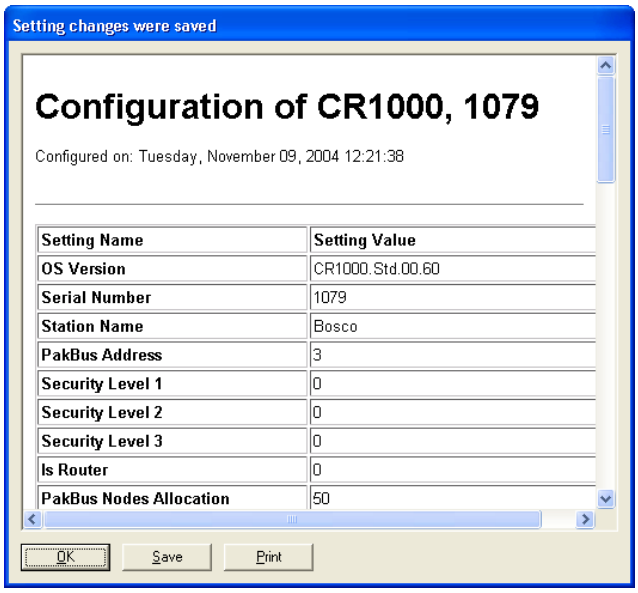
L’onglet « **Settings Editor »** donne accès à la plupart des configurations PakBus. L’onglet « **Deployment »** rend l’application de certaines configurations, un peu plus faciles à mettre en place.



La partie supérieure de l’onglet « Settings Editor » est une grille qui permet à l’utilisateur de voir et d’éditer les configurations pour l’appareil. La grille est divisée en deux colonnes, avec à gauche le nom de la configuration, et à droite sa valeur associée. Vous pouvez changer la cellule sélectionnée en utilisant la souris ou les flèches du clavier. Si vous cliquez sur le nom d’une variable sur la colonne de gauche, la valeur associée sur la colonne de droite deviendra automatiquement active. Vous pouvez éditer une configuration en sélectionnant la valeur, en pressant sur F2 ou en double-cliquant sur la valeur d’une cellule avec la souris. La grille ne permettra pas de modifier les valeurs qui sont en lecture seule.

La partie inférieure de l’onglet « Settings Editor » affiche l’aide au sujet des paramètres de configuration sélectionnés sur la partie supérieure.

Une fois que vous avez changé les configurations, vous pouvez appuyer sur le bouton « **Apply »** (Appliquer) afin de les prendre en compte, ou sur « **Cancel »** pour annuler. Ces boutons ne seront actifs qu’une fois qu’un paramètre aura été modifié. Si l’appareil accepte les configurations, une fenêtre de sommaire de configuration sera affichée et vous donnera la possibilité de sauvegarder et/ou d’imprimer la configuration de l’appareil :

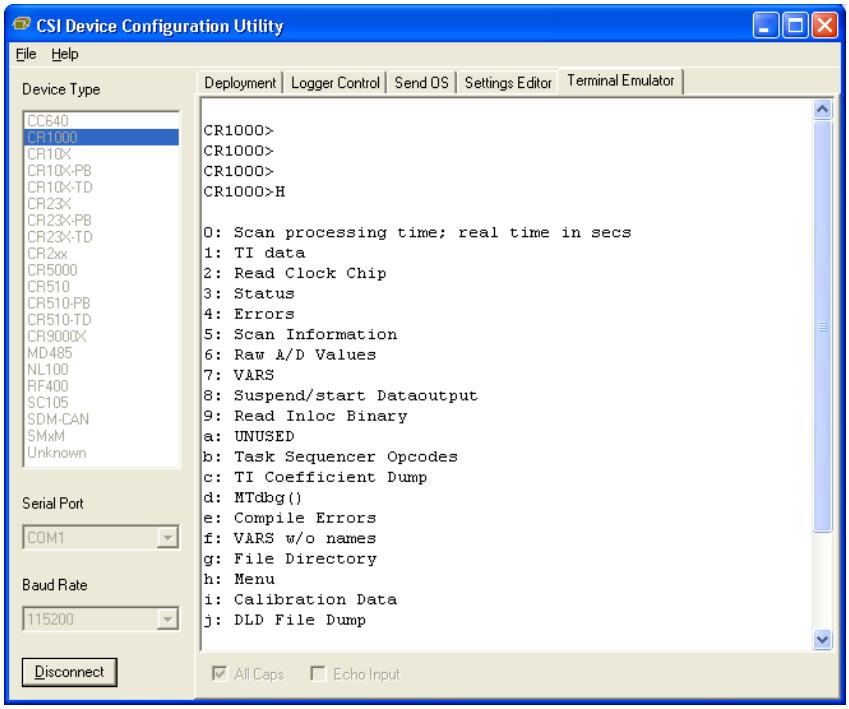


Le fait de cliquer sur le bouton « **Factory Defaults** » (Configuration usine par défaut) du « Settings Editor », enverra la commande permettant à l'appareil de récupérer ses configurations d'usine par défaut. Les valeurs d'usine ne seront pas prises en compte tant que l'on n'aura pas demandé de les prendre en compte (Apply - Appliquer). Ce bouton ne sera pas actif si l'appareil à configurer ne comprend pas les messages du protocole utilisé par « DevConfig ».

Si, après avoir effectué des changements de configuration ou après avoir cliqué sur le bouton « **Summary** » (Sommaire), vous avez cliqué sur le bouton « **Save** » (Sauvegarde) de l'écran du sommaire qui aura été affiché, vous aurez la possibilité d'utiliser le bouton « **Read File** » afin d'importer ces configurations. Les configurations contenues dans ce fichier sont immédiatement envoyées à l'appareil, et s'ils sont acceptés, vous pourrez les appliquer (**Apply**).

OV3.6 Onglet "Terminal"

L'onglet « Terminal » donne accès à l'émulateur de terminal que l'on peut utiliser sur la CR1000. Presser « Entrée » plusieurs fois jusqu'à ce que l'émulateur renvoie le prompt : "CR1000>". Les commandes du mode terminal sont constituées d'une liste de caractères uniques, suivis de « Entrée ». Si par exemple pour entrez « H » puis « Entrée », vous obtiendrez la liste des commandes disponibles.



OV4. Didacticiel de démarrage rapide (Quick Start Tutorial)

OV4.1 Logiciels pour la CR1000

Le logiciel de démarrage PC200W est capable d'effectuer une liaison directe entre un PC et une CR1000; ce logiciel comprend « Short Cut for Windows » (Short Cut) afin de créer des programmes pour CR1000. PC200W dispose d'outils de base afin de configurer l'heure de la centrale de mesure, d'envoyer un programme, de visualiser les valeurs issues des capteurs, ou encore de collecter manuellement et de visualiser les données. La CR1000 est présente dans PC200W à partir de la version 3.0. PC200W est un logiciel que l'on peut télécharger gratuitement à partir du site Internet de Campbell Scientific.

Le logiciel PC400 (le logiciel de milieu de gamme) permet de gérer de multiples options de télécommunication, il dispose de la collecte manuelle des données et de l'affichage des données en temps réel. PC400 comprend Short Cut ainsi que l'éditeur de programme CRBasic, adapté à la programmation de la CR1000. PC400 ne permet pas de combiner des options de télécommunication (par exemple d'un téléphone à un modem radio), de faire du routage PakBus®, ni d'effectuer des appels automatisés.

Le logiciel de support pour centrales de mesures, LoggerNet, permet de faire des appels via des moyens de communication combinés, il permet de faire des affichages de données et des appels automatisés. Ce logiciel comprend Short Cut ainsi que l'éditeur de programme CRBasic, adapté à la programmation de la CR1000, ainsi que des outils de configuration, de surveillance et de contrôle du réseau de centrales de mesures.

OV4.1.1 Options afin de créer des programmes pour CR1000

1. *Short Cut* est un générateur de programmes qui crée un programme pour centrale de mesure en quatre étapes simples, puis génère un diagramme associé pour le branchement des capteurs. Short Cut est compatible avec la plupart des capteurs commercialisés par Campbell Scientific, et c'est un outil recommandé lorsqu'on souhaite faire un programme simple et rapide afin de mesurer et d'enregistrer des données.
2. L'éditeur *CRBasic* est un éditeur de programmes utilisé afin de créer des programmes plus complexes pour la CR1000. Les programmes générés par Short Cut peuvent être importés dans l'éditeur CRBasic afin d'y ajouter des instructions ou des fonctionnalités qui ne seraient pas comprises dans Short Cut.

Pour les utilisateurs de CR10X qui souhaitent passer à la CR1000, il existe l'utilitaire « Transformer » qui permet de convertir les fichiers programme de CR10X en fichiers programme pour CR1000 (ou CR800), afin de les modifier avec CRBasic. Le langage de programmation et les instructions étant un peu différentes, certains programmes ne peuvent pas être entièrement convertis par « Transformer ». L'utilitaire est compris avec les logiciels PC400 et LoggerNet.

OV4.2 Connections à la CR1000

Les alimentations de Campbell Scientific sont décrites au chapitre 1.3. Lorsqu'on connecte une source d'alimentation à une CR1000, on retire tout d'abord le connecteur vert présent sur le bornier de la CR1000. On insère le fil positif 12V sur la borne « 12V », la borne de masse sur la borne « G ». On vérifie une dernière fois que l'on n'a pas inversé les polarités, puis on enfonce le connecteur vert câblé, au bornier de la CR1000.

Connecter le câble série entre le port étiqueté « RS232 » présent sur la CR1000, et le port série de l'ordinateur. Pour les ordinateurs qui n'ont qu'un port USB, un adaptateur USB-série est nécessaire.

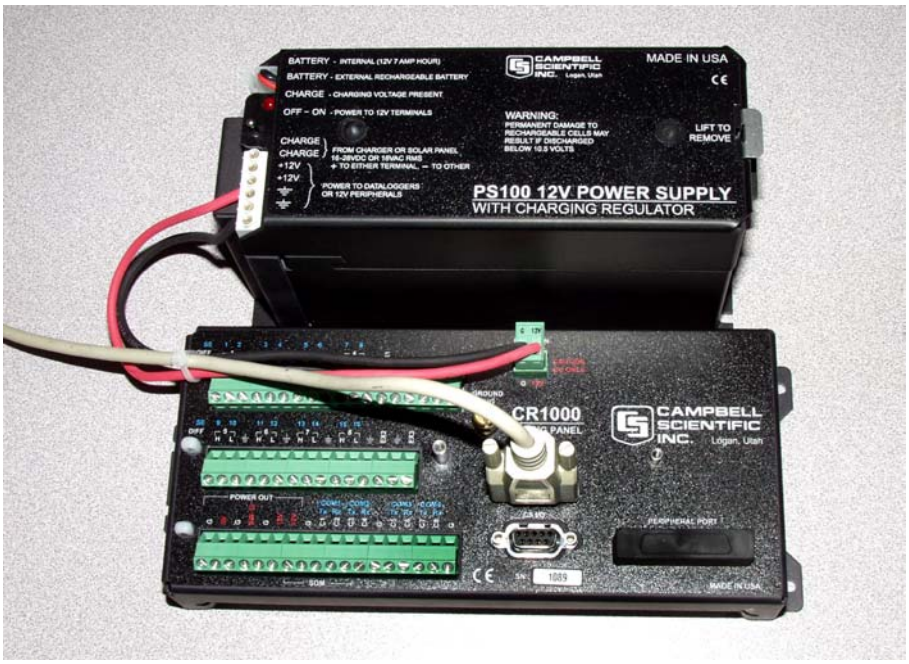


FIGURE OV4-1. Connexion de l'alimentation et de la RS232

OV4.3 Configuration de l'adresse PakBus de la CR1000

L'adresse PakBus par défaut de la CR1000 est l'adresse 1 (voir OV2.5). A moins d'utiliser la CR1000 en réseau, il n'y a pas de raison de changer d'adresse PakBus, ni aucun de ses autres paramètres de configuration par défaut. Pour changer les configurations par défaut il faut utiliser l'utilitaire DevConfig, comme cela est décrit au paragraphe 0V3.

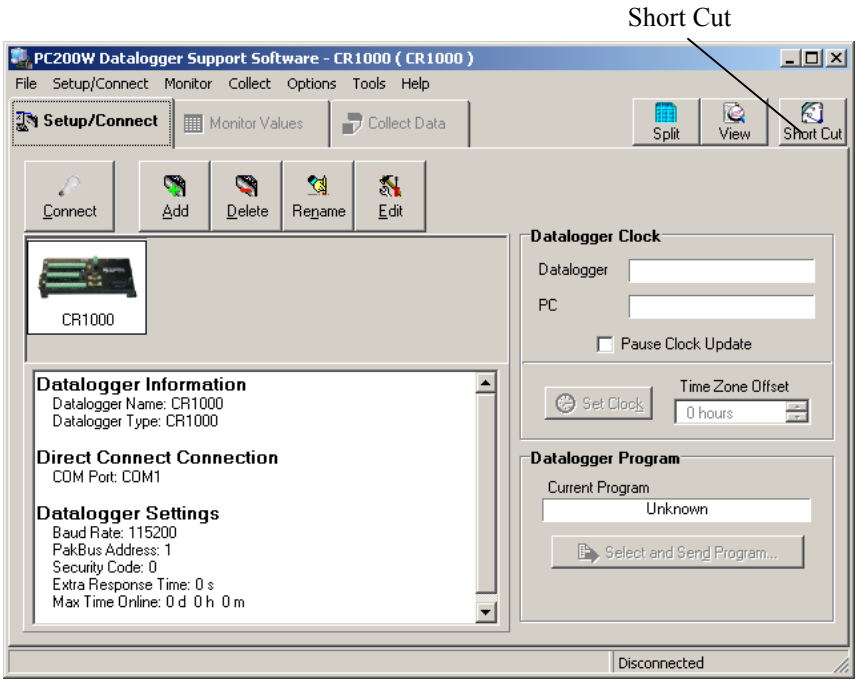
OV4.4 Logiciel PC200W

Ce petit didacticiel dirige l'utilisateur à travers un processus de création d'un programme pour CR1000, afin de visualiser des mesures effectuées par des capteurs, puis pour collecter les données et les visualiser sous forme de données avec le logiciel PC200W.

Lorsque PC200W est lancé pour la première fois, l'assistant de configuration *EZSetup* est aussi exécuté. On peut alors cliquer sur le bouton « **Next »** et suivre les étapes afin de sélectionner la CR1000, le port **COM** que l'ordinateur utilisera afin de communiquer, la vitesse en baud à utiliser (**115200** bauds), ainsi que **l'adresse PakBus** de 1. Lorsqu'on vous propose de tester la communication (**Test Communications**) vous pouvez cliquer sur le bouton « **Finish** » (Fin).

Pour changer un paramètre dans la configuration de la centrale de mesure, on peut sélectionner la centrale de mesure à partir de la fenêtre principale du logiciel et cliquer sur « **Edit** ». Si une centrale de mesure n'a pas encore été définie, on cliquera sur « **Add** » afin d'utiliser l'assistant de configuration EZSetup à nouveau.

Une fois que vous êtes sorti de l'assistant EZSetup, l'onglet « **Setup/Connect »** sera visible, tel que cela est montré ci-dessous. Le profil actuel de la CR1000, son horloge et les détails à propos du programme associé à la CR1000, sont intégrés à cet onglet. Les onglets sur la droite sont utilisés afin de visualiser les données (**Monitor Values**) et afin de les collecter (**Collect Data**). Les boutons sur la droite de ces onglets, sont utilisés afin de faire fonctionner les applications « **Split** », « **View** », et « **Short Cut** ».

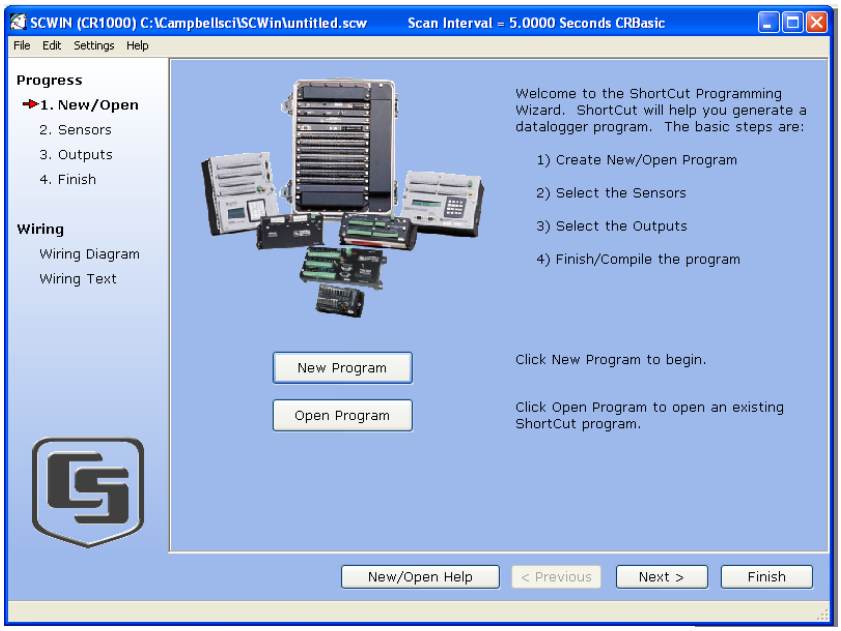


OV4.4.1 Création d'un programme pour CR1000 à partir de « Short Cut »

Objectif : une fois par seconde, mesurer la température de l'air en °C avec un thermocouple de type T, et stocker les moyennes effectuées sur une minute pour la tension d'alimentation, la température du bornier et la température du thermocouple.

NOTE Un thermocouple de type T est inclus avec la CR1000, ainsi qu'un tournevis. Le thermocouple est une paire de fils métalliques recouverts d'un isolant bleu / rouge, et soudés ensemble à une extrémité.

Cliquez sur le bouton de **Short Cut**, ce qui vous conduira à l'écran d'accueil tel que celui présenté ci-dessous.



En cliquant sur l'onglet Help (Aide), vous pouvez avoir accès à l'aide de ce logiciel. On peut utiliser l'aide en plus des descriptions données ci-dessous :

Etape 1 (Step 1) : Créer un nouveau fichier

L'étape 1 permet d'ouvrir un nouveau fichier ou bien un fichier déjà existant. A partir de la page d'accueil, on clique alors sur le bouton « **New** ». On utilise alors le menu déroulant afin de sélectionner la CR1000. On entre la valeur de 1 (en secondes) pour la grandeur de l'intervalle de scrutation (Scan Interval) puis on clique sur « **OK** » afin de terminer cette première étape.

Etape 2 (Step 2) : Sélectionner les capteurs

Un thermocouple de type T est un ensemble de deux fils métallique dissemblables (cuivre et constantan) soudés ensemble à une de leurs extrémités. La partie soudée est la partie de jonction de mesure ; la jonction qui est créée au niveau du branchement sur la CR1000, est la jonction de référence.

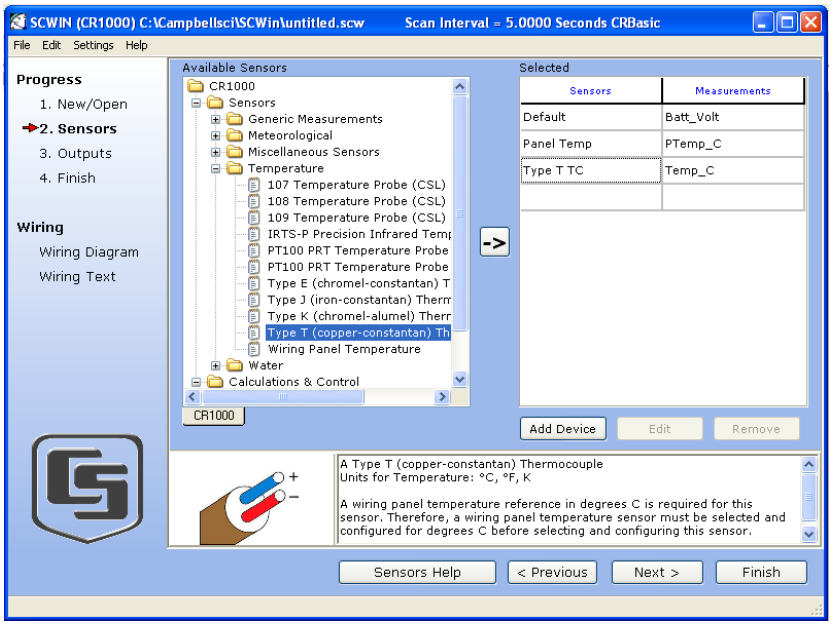
Lorsque les deux jonctions sont à des températures différentes, une tension proportionnelle à la différence de température, est induite dans les fils. La mesure de température du thermocouple nécessite de connaître la température de jonction de référence afin de déterminer la température à l'extrémité du thermocouple (à la soudure).

L'étape 2 permet de sélectionner le capteur que l'on veut mesurer. A partir de la page d'accueil, on clique sur le bouton « **Sensors** » (Capteurs). La partie réservée au choix des capteurs est divisée en 2 groupes : le premier groupe à gauche est une arborescence de capteurs, le second groupe à droite est le tableau des capteurs sélectionnées.

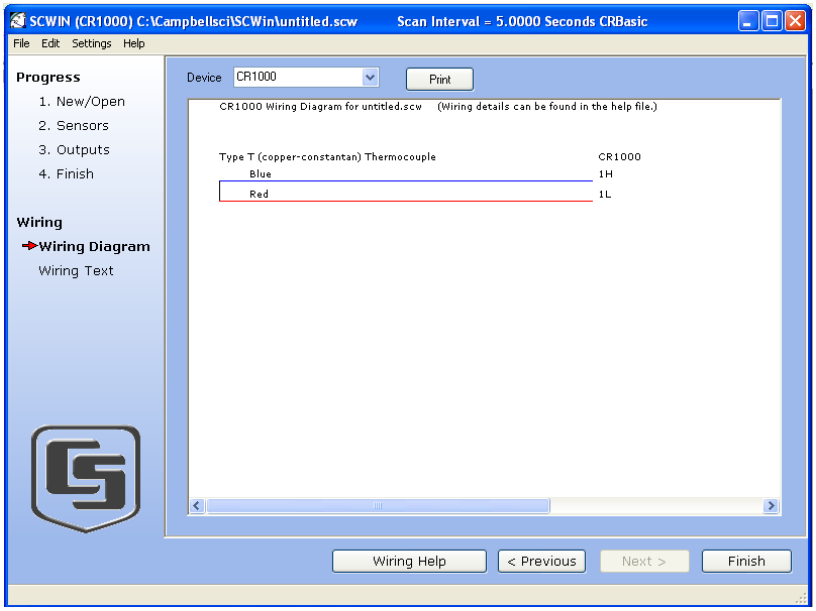
Les capteurs que vous allez programmer seront choisis à partir de la liste définie dans l'arborescence des capteurs.

En double-cliquant sur le groupe appelé « **Temperature** », vous affichez le détail des choix disponibles pour ce groupe de capteurs. Double-cliquez sur le capteur « **Wiring Panel Temperature** » afin de l'ajouter à la liste des capteurs sélectionnés. Cliquer sur OK afin d'accepter l'étiquette proposée pour le nom de la variable (PTemp_C).

Double cliquez sur « **Type T thermocouple** », changez la valeur proposée par « 1 », et cliquez sur « **OK** ». Sur l'écran suivant, assurez-vous que la variable « PTemp_C » (celle définie pour la température du bornier de la CR1000) est sélectionnée en tant que température de référence, et cliquez sur **OK** pour accepter l'étiquetage de la variable en « Temp_C ».



Cliquez sur votre gauche sur : « **Wiring Diagram** » afin de voir le schéma de câblage, tel que celui ci-dessous. Il faut alors câbler le thermocouple de type T (fourni avec la centrale) comme cela est spécifié. Cliquez sur votre gauche sur : « **Sensors** » (Capteurs) pour revenir à la page précédente et continuer avec l'étape 3.



Etape 3 (Step 3) : Enregistrements / sauvegardes (Output Processing)

L'étape 3 permet de définir les instructions de sauvegarde pour les capteurs mesurés. Sur votre gauche, il faut cliquer sur le bouton « **Output** » (Sortie).

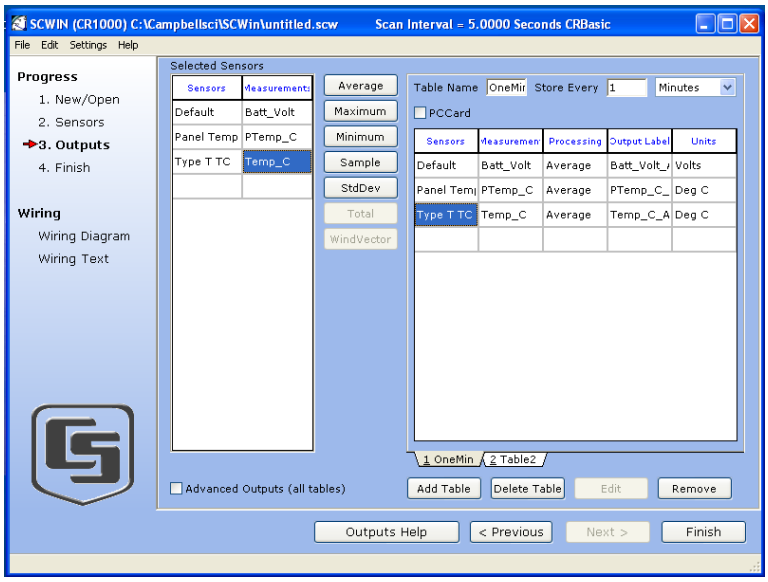
L'écran qui sert à définir les sauvegardes comprend une liste de capteurs sélectionnés sur la partie gauche, et des tableaux de sauvegarde sur la droite. Il y a par défaut 2 tableaux de définis, Table1 et Table2. Les deux tableaux ont un champs « **Store Every** » et un menu déroulant afin de déterminer à quel intervalle de temps les données vont être stockées.

L'énoncée de cet exercice demande une sauvegarde par minute. Pour supprimer la Table2 il faut cliquer sur l'onglet la définissant afin de la rendre active, puis cliquer sur le bouton « **Delete Table** ».

Le champ « **Table Name** » est le nom qui sera utilisé afin de nommer le tableau qui contiendra les données stockées. On se propose alors de changer le nom par défaut, de « Table1 » à « OneMin », et de changer la grandeur de l'intervalle pour la fixer à 1.

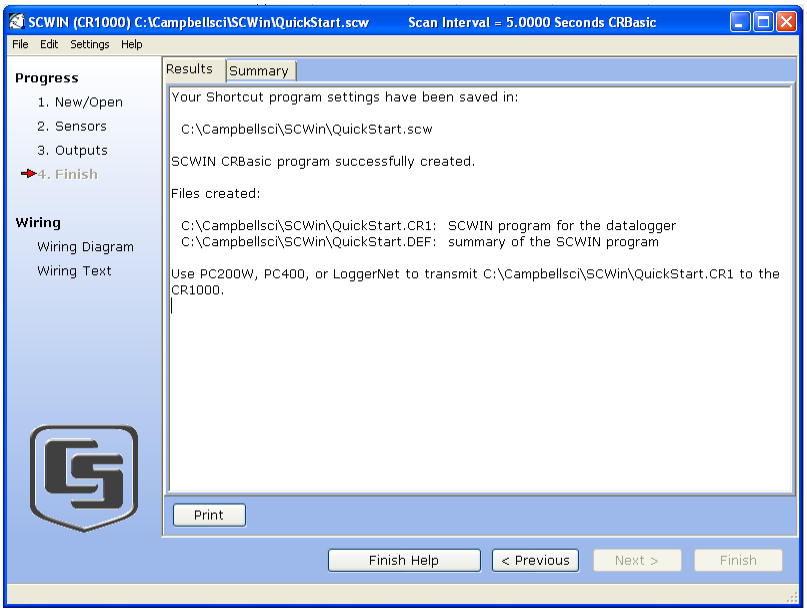
La liste des capteurs sélectionnés se trouve sur la partie gauche de l'écran. Pour ajouter une mesure de capteur à l'intérieur du tableau de sauvegarde, il faut surligner une mesure et cliquer sur un des boutons qui déterminent le type de calcul que l'on veut faire (par exemple la moyenne « Average »). On sélectionnera alors les capteurs appelés « Default », « Panel Temp », et « Type T TC » suivi du bouton « **Average** » (Moyenne) afin de les ajouter au tableau appelé «OneMin».

Cliquer sur votre gauche sur « **Finish** » afin de passer à l'étape 4.



Etape 4 (Step 4): Terminer (Finish)

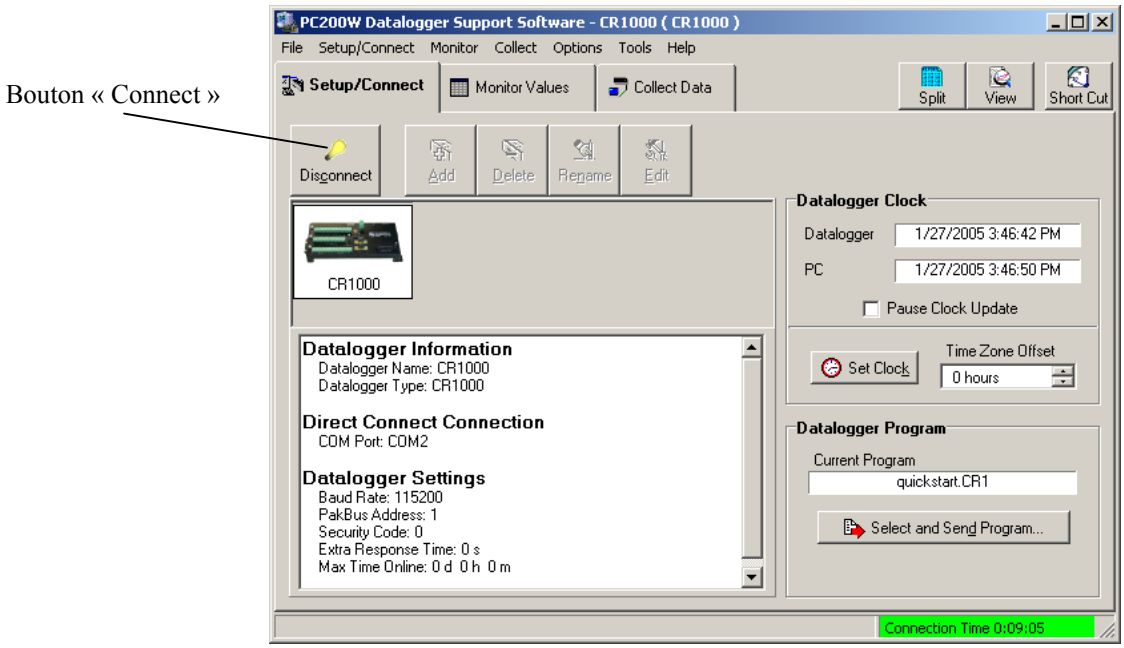
L'étape 4 sert à terminer le programme. Sur la partie gauche de l'écran SCWin vous cliquez sur le bouton « **Finish** ». On donne alors le nom QuickStart pour le nom du fichier. Si le compilateur détecte des erreurs, celles-ci seront affichées, de même que le nom des fichiers créés. Le fichier QuickStart.CR1 est le fichier programme qui sera envoyé à la CR1000, QuickStart.def est le résumé du câblage des capteurs et des noms d'étiquettes pour les variables utilisées (cliquer sur « **Summary** » ou « **Print** » afin de visualiser ou d'imprimer le fichier).



Pour fermer Short Cut il faut alors cliquer sur le « X » en haut à droite de la fenêtre, sur « Alt+F4 » au clavier, ou aller dans le menu « File / Exit ».

OV4.4.2 Configuration de l'onglet "Setup"

A partir de l'écran « **Setup/Connect** », cliquer sur le bouton « **Connect** » afin d'établir la communication avec la CR1000. Lorsque la communication est établie, le texte présent sur le bouton deviendra « **Disconnect** ».



OV4.4.3 Synchronisation des horloges

Cliquer sur le bouton « **Set Clock** » afin de synchroniser l'heure de la centrale de mesure avec celle de l'ordinateur.

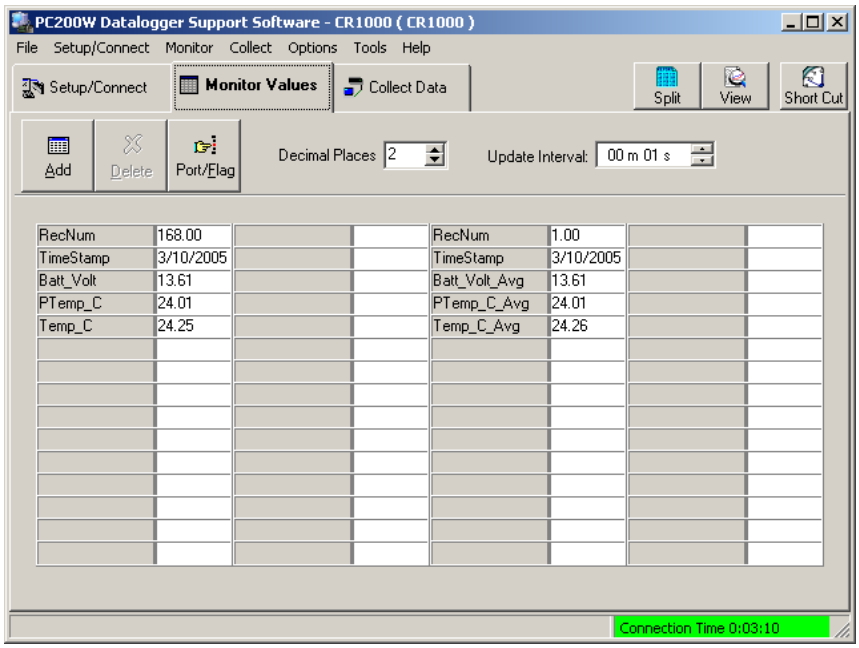
OV4.4.4 Envoyer le programme

Cliquer sur le bouton « **Select and Send Program** ». Naviguer afin de pointer sur le répertoire C:\CampbellSci\SCWin et y sélectionner le fichier appelé QuickStart.CR1 puis cliquer sur le bouton « **Open** » (Ouvrir). Une barre de processus est alors affichée, suivie d'un message confirmant que le programme a été correctement envoyé à la station.

OV4.4.5 Visualiser les données scrutées

L'onglet « **Monitor Values** » est utilisé afin d'afficher les valeurs actuelles qui ont été scrutées sur les capteurs, pour les variables de type « Public » définies dans les programmes, et les dernières valeurs présentes dans le tableau « OneMin ».

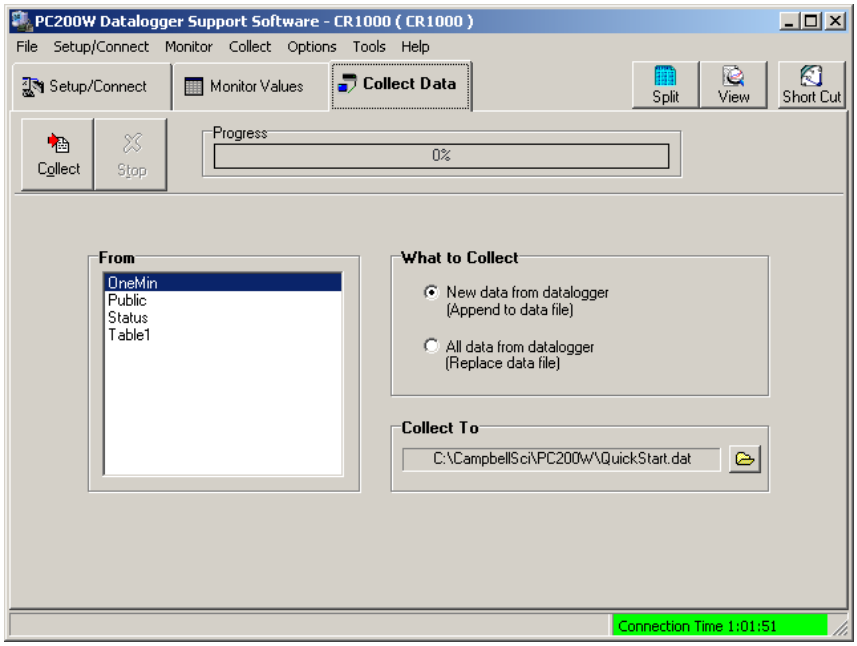
Cliquer sur l'onglet « **Monitor Values** ». Le contenu des variables « Public » est automatiquement sélectionnée et affichée. Pour visualiser les valeurs du tableau « OneMin » il faut cliquer sur le bouton « **Add** », sélectionner le tableau « OneMin » puis cliquer sur le bouton « **Paste** ».



OV4.4.6 Récupération des données (Collect Data)

Cliquer sur l'onglet « **Collect Data** ». A partir de cet onglet vous pouvez choisir quelles données vous souhaitez récupérer, et où sur votre ordinateur, si vous souhaitez les enregistrer.

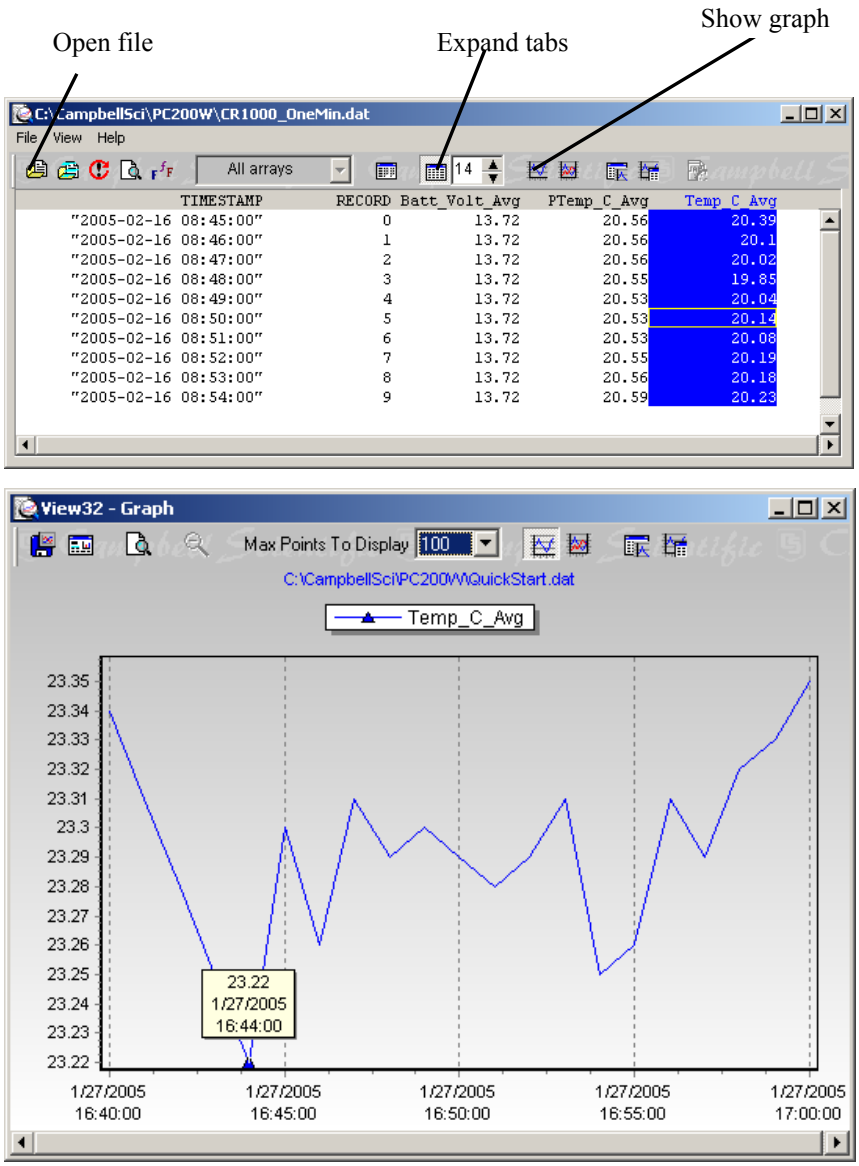
Cliquer sur le tableau appelé « **OneMin** », avec l'option « **New data from datalogger** » de sélectionnée. Cliquer sur le bouton « **Collect** » et une boite de dialogue apparaît, vous demandant un nom de fichier à donner. En cliquant sur « Save » vous utiliserez le nom par défaut prévu pour le fichier, soit CR1000_OneMin.dat. Une barre de progression s'affichera, avec à la fin le message « **Collection Complete** ».



OV4.4.7 Visualiser les données (View Data)

Pour visualiser les données collectées, il faut cliquer sur le bouton « **View** » (situé dans la partie supérieure droite de l'écran principal de PC200W. On accède aux options en utilisant les menus ou en cliquant sur l'icône associé à l'option. Si vous placez le pointeur de votre souris au dessus d'une de ces icônes durant quelques secondes, vous obtiendrez un petit descriptif de la fonction disponible sur cette icône.

Pour ouvrir un fichier de données on clique sur l'icône « **Open file** » et on double-clique sur le fichier CR1000_OneMin.dat présent dans le répertoire de PC200W. Cliquer sur « **Expand Tabs** » afin d'afficher les données en colonnes avec des en-tête de colonnes. Si vous souhaitez afficher les données issues du thermocouple, sous forme graphique, il faut cliquer sur la colonne qui porte le nom « Temp_C », puis cliquer sur l'icône « **Show Graph, 1 Y axis** » qui est présente dans le barre d'outils.



On peut alors fermer le graphique, View et PC200W.

OV4.5 Programmation de la CR1000 par l'éditeur CRBasic

Les utilisateurs qui avaient l'habitude de l'éditeur de programme Edlog, et qui passeraient à CRBasic pour programmer la CR1000, pourront trouver que Short Cut est un très bon outil pour apprendre à programmer en CRBasic. Vous pouvez tout d'abord créer un programme à l'aide de Short Cut, puis ouvrir ce fichier avec CRBasic afin de voir comment Short Cut a créé le programme. Le fichier ci-dessous est issu du fichier QuickStart.CR1 que nous avons crée avec le didacticiel (Short Cut) décrit dans le paragraphe précédent. Il a ensuite été importé dans l'éditeur CRBasic.

Voir le paragraphe 4 pour plus de détails au sujet de la programmation CRBasic.

```
'CR1000

'Declare Variables and Units – Déclaration des variables et des unités

Public Batt_Volt
Public PTemp_C
Public Temp_C

Units Batt_Volt=Volts
Units PTemp_C=Deg C
Units Temp_C=Deg C

'Define Data Tables – Définitions des tableaux de données
DataTable(OneMin,True,-1)
    DataInterval(0,1,Min,0)
    Average(1,Batt_Volt,FP2,False)
    Average(1,PTemp_C,FP2,False)
    Average(1,Temp_C,FP2,False)
EndTable

DataTable(Table1,True,-1)
    DataInterval(0,1440,Min,0)
    Minimum(1,Batt_Volt,FP2,False,False)
EndTable

'Main Program – Programme principal
BeginProg
    Scan(5,Sec,1,0)
        'Default Datalogger Battery Voltage measurement Batt_Volt:
        'Mesure de la tension de la batterie de la centrale de mesure Batt_Volt :
        Battery(Batt_Volt)
        'Wiring Panel Temperature measurement PTemp_C:
        'Mesure de la température de compensation du bornier Ptemp_C :
        PanelTemp(PTemp_C,_60Hz)
        'Type T (copper-constantan) Thermocouple measurements Temp_C:
        TCDiff(Temp_C,1,mV2_5C,1,TypeT,PTemp_C,True,0,_60Hz,1,0)
        'Call Data Tables and Store Data
        'Appel des tableaux de données et de la mémorisation des données
        CallTable(OneMin)
        CallTable(Table1)
    NextScan
EndProg
```

OV5. Clavier écran (Keyboard Display)

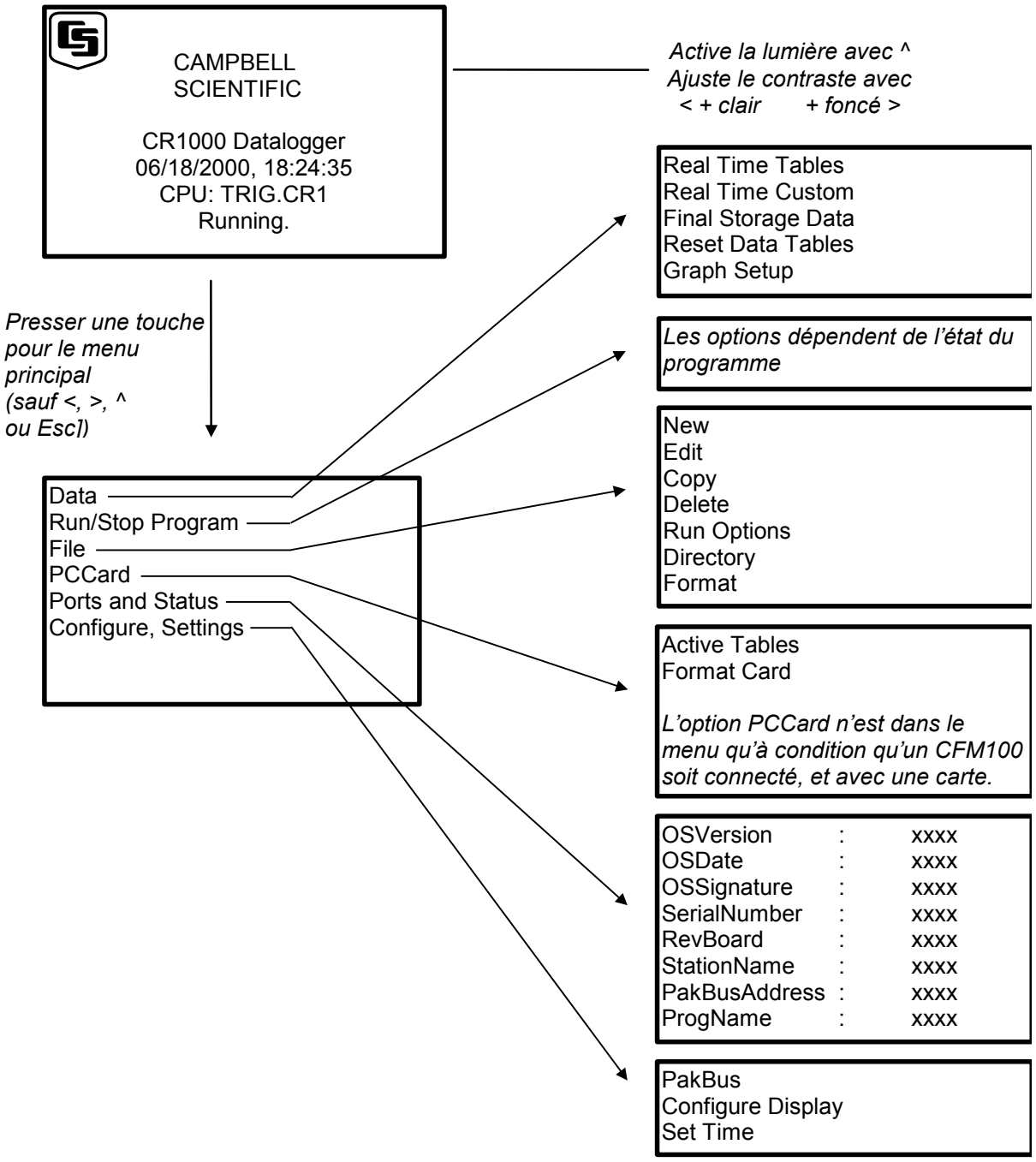
La CR1000 dispose d'un clavier / écran disponible en option. Ce paragraphe en décrit l'utilisation.

Le CR1000KD dispose de certaines touches, qui sont associées à des fonctions listées ci-dessous.

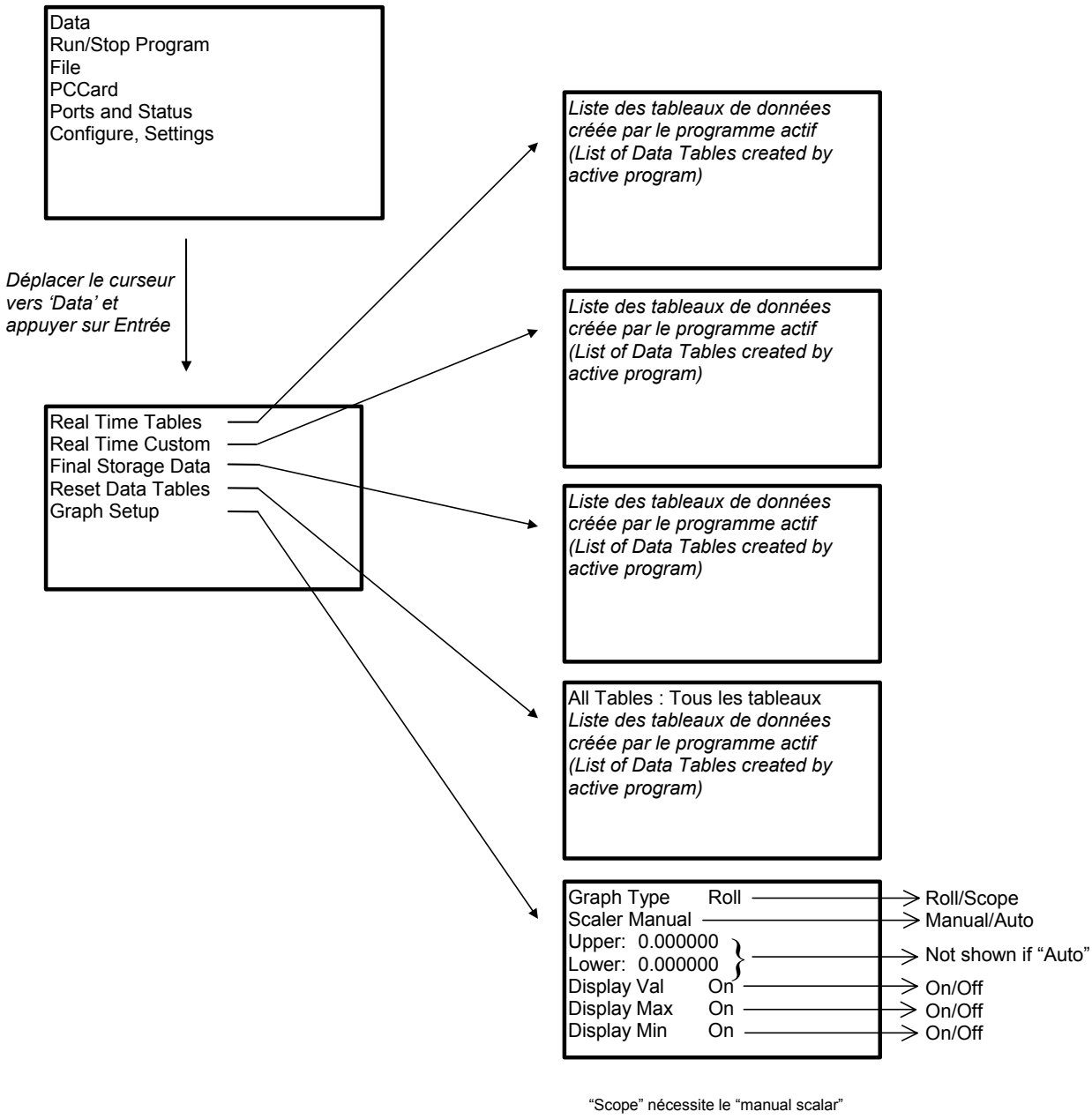
Touche		**Usage**
	[2] et [8]	Naviguer ligne par ligne (vers le haut ou vers le bas) à travers le menu.
	[Enter]	Sélectionne la ligne ou fait changer d'état l'option sur laquelle le curseur est positionné
[Esc]		Retourne au menu précédent du menu
[Home]		Déplace le curseur jusqu'au début de la liste
	[End]	Déplace le curseur jusqu'à la fin de la liste
	[Pg Up]	Déplace le curseur vers l'écran suivant situé en bas
[Pg Dn]		Déplace le curseur vers l'écran précédent situé en haut
	[BkSpc]	Supprime le caractère situé à la gauche du curseur
	[Shift]	Change le caractère alphabétique sélectionné
	[Num Lock]	Change l'entrée numérique
	[Del]	Supprime
	[Ins]	Insère/change la configuration de l'affichage
[Graph]		Mode graphique

Affichage de la CR1000

Ecran de mise sous tension



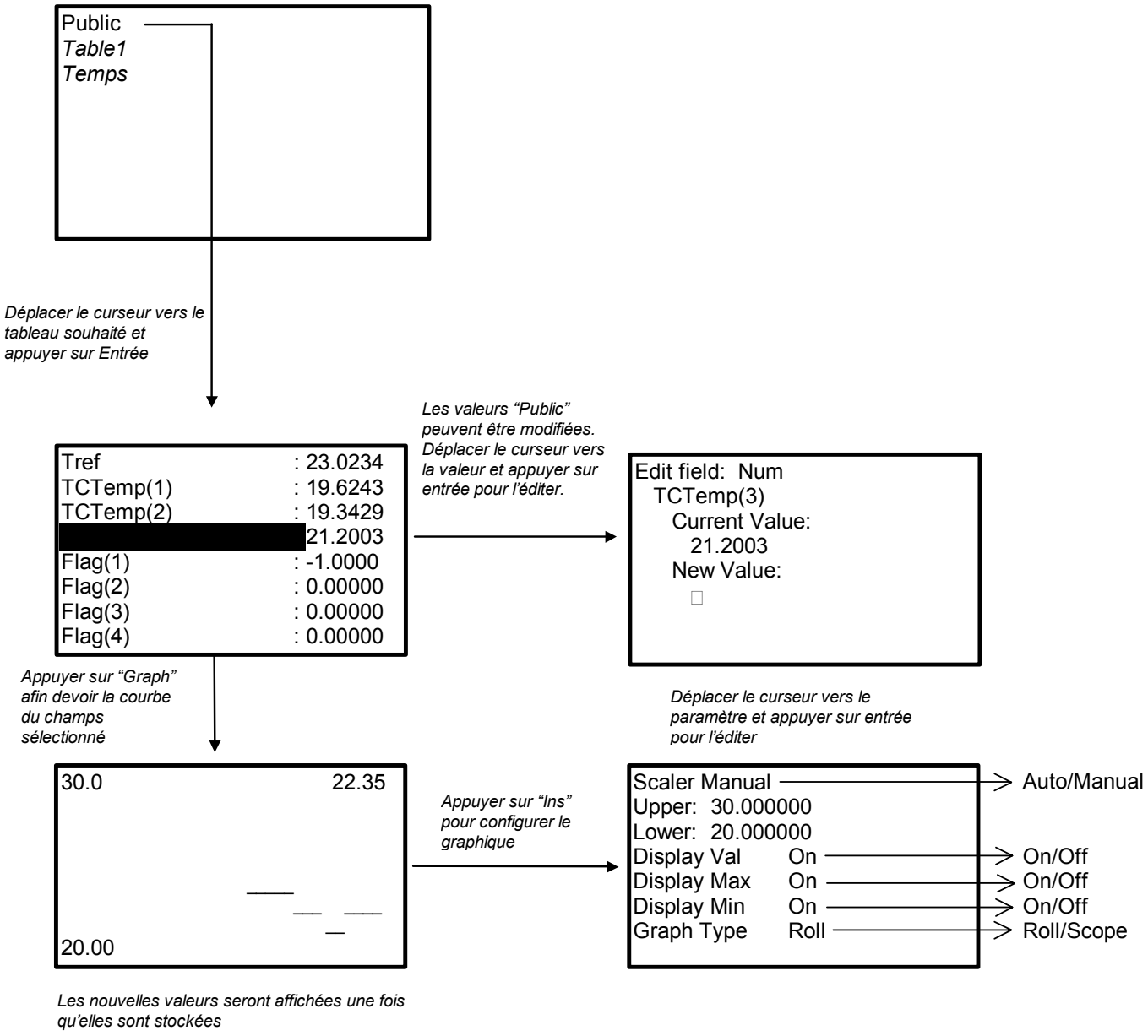
OV5.1 Affichage de données (Data Display)



"Scope" nécessite le "manual scalar"

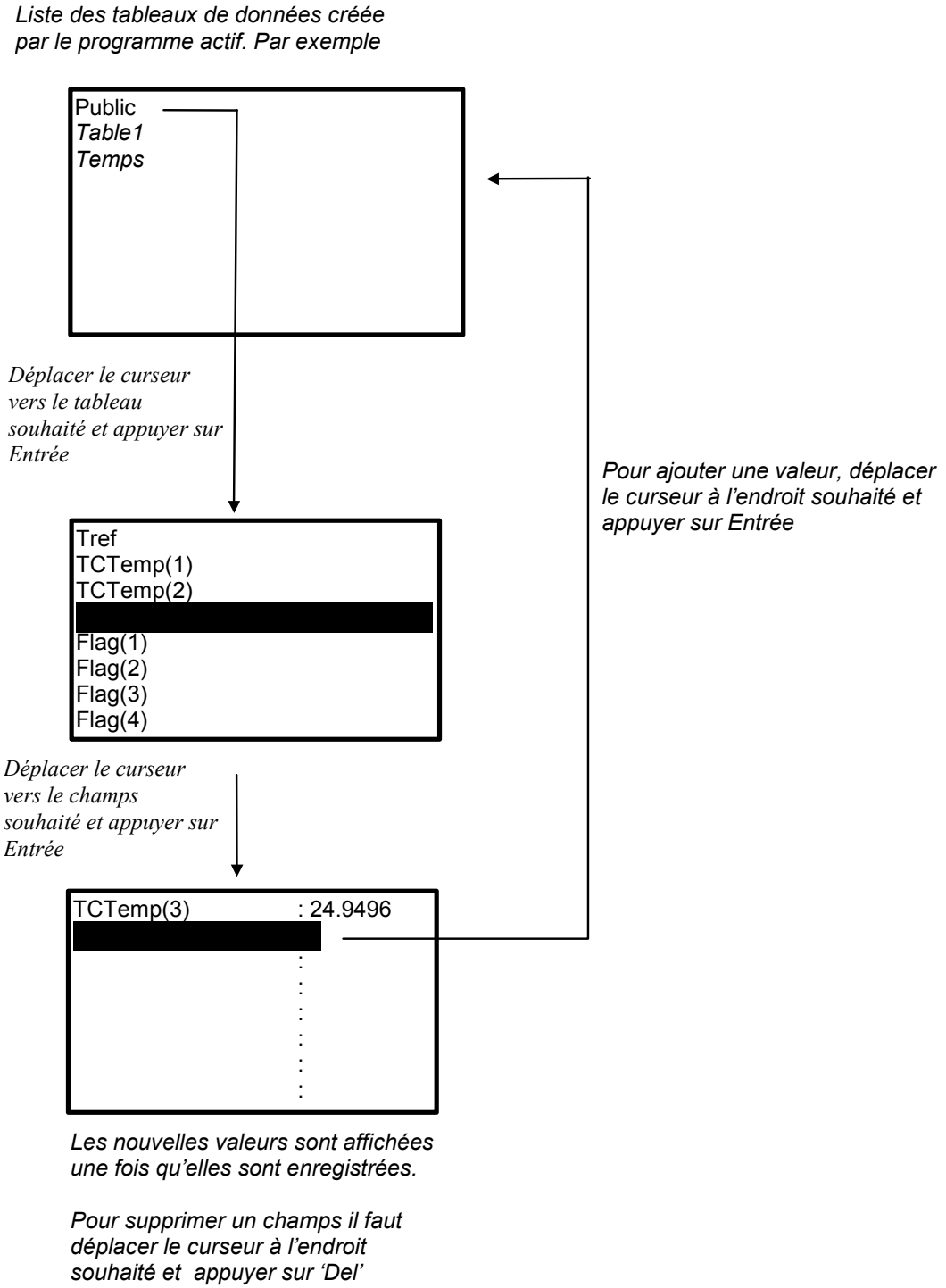
OV5.1.1 Tableaux de données scrutées (Real Time Tables)

Liste des tableaux de données créée par le programme actif. Par exemple



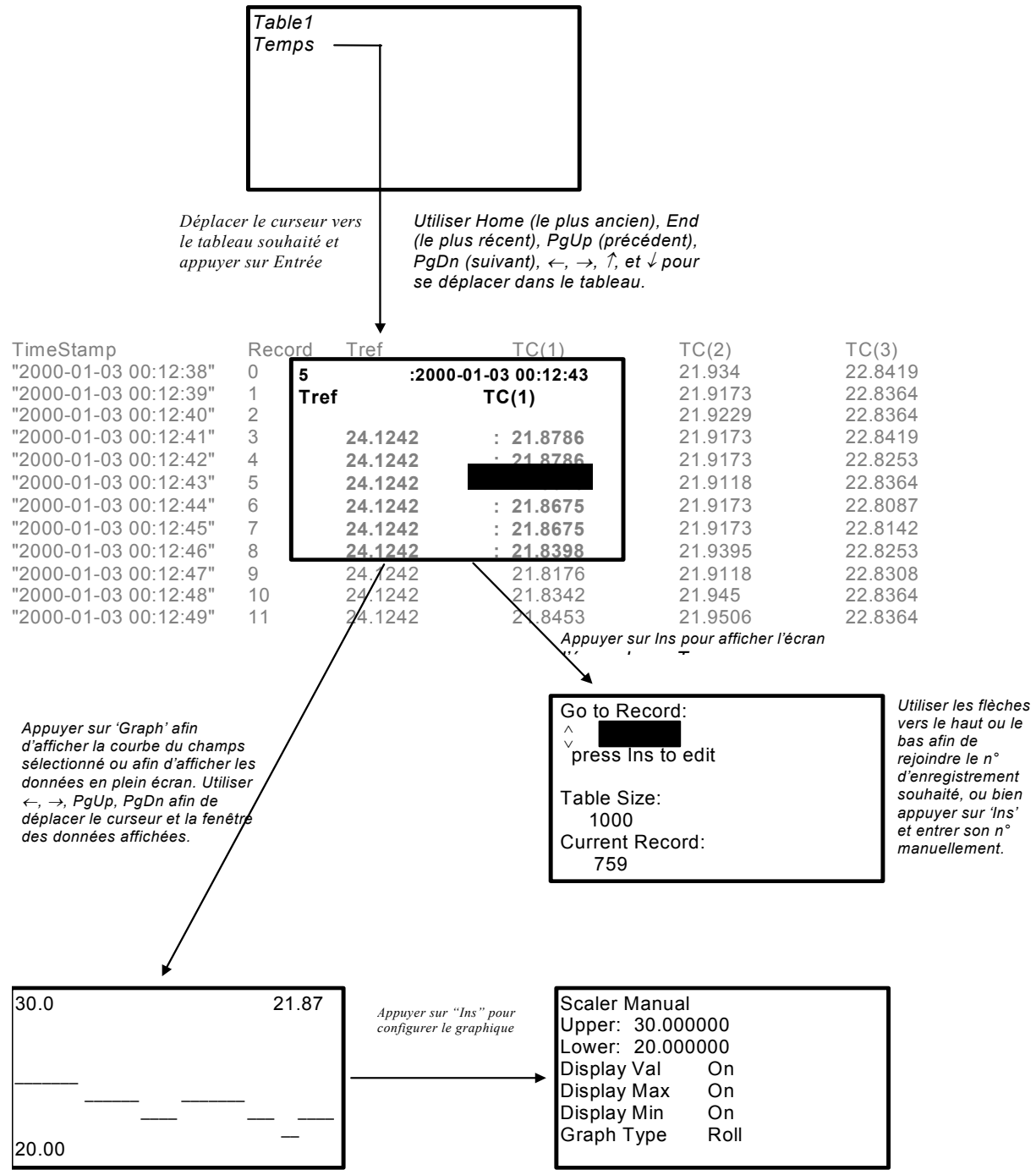
OV5.1.2 Personnalisation de l'affichage (Real Time Custom)

La première fois que vous naviguez sur « Real Time Custom » vous aurez besoin de configurer l'affichage. La CR1000 conservera la configuration tant que le même programme est en fonctionnement.

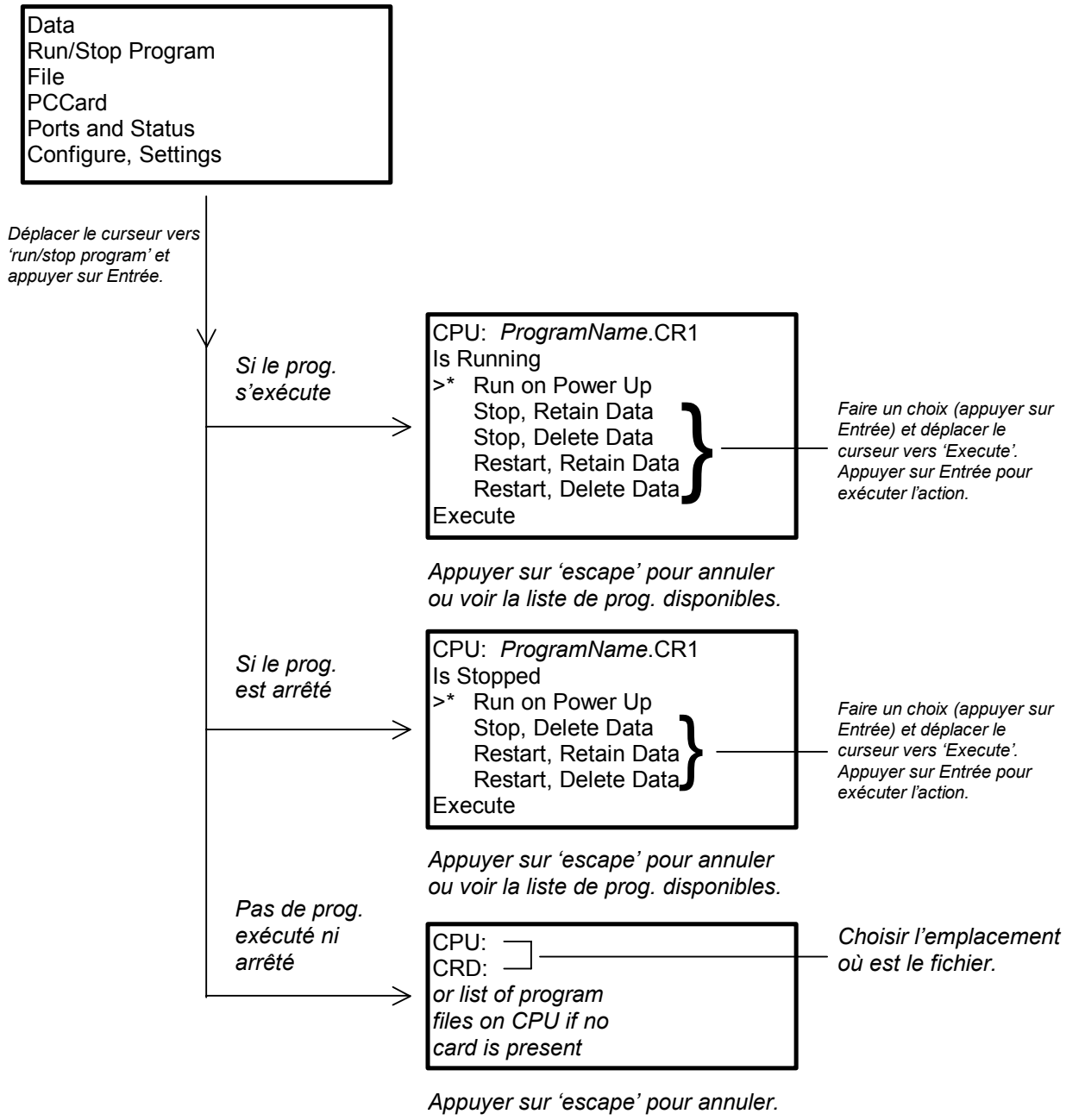


OV5.1.3 Tableaux de mémoires finales (Final Storage Tables)

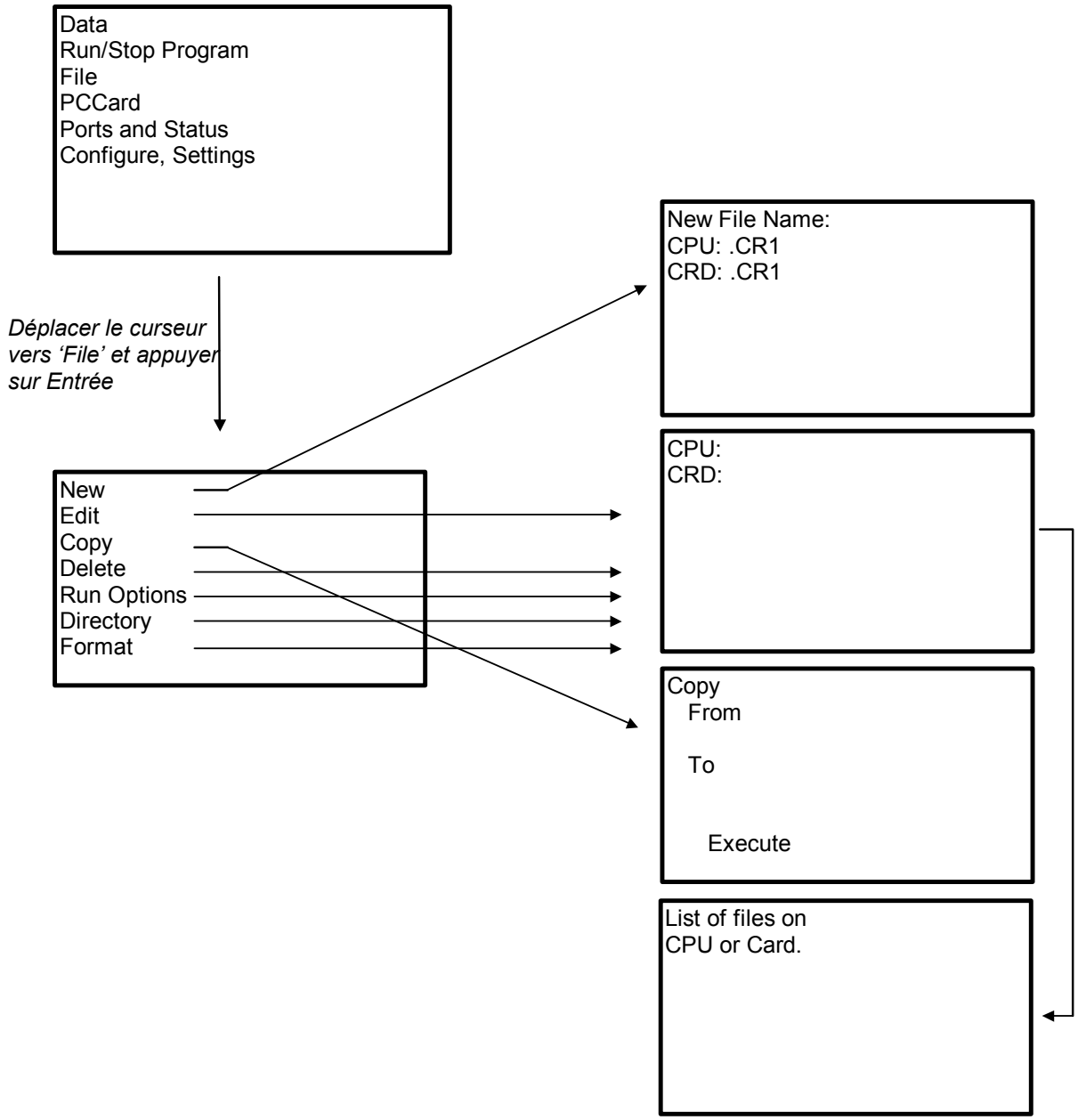
Liste des tableaux de données créée par le programme actif. Par exemple



OV5.2 Démarrer / Arrêter le programme (Run/Stop Program)

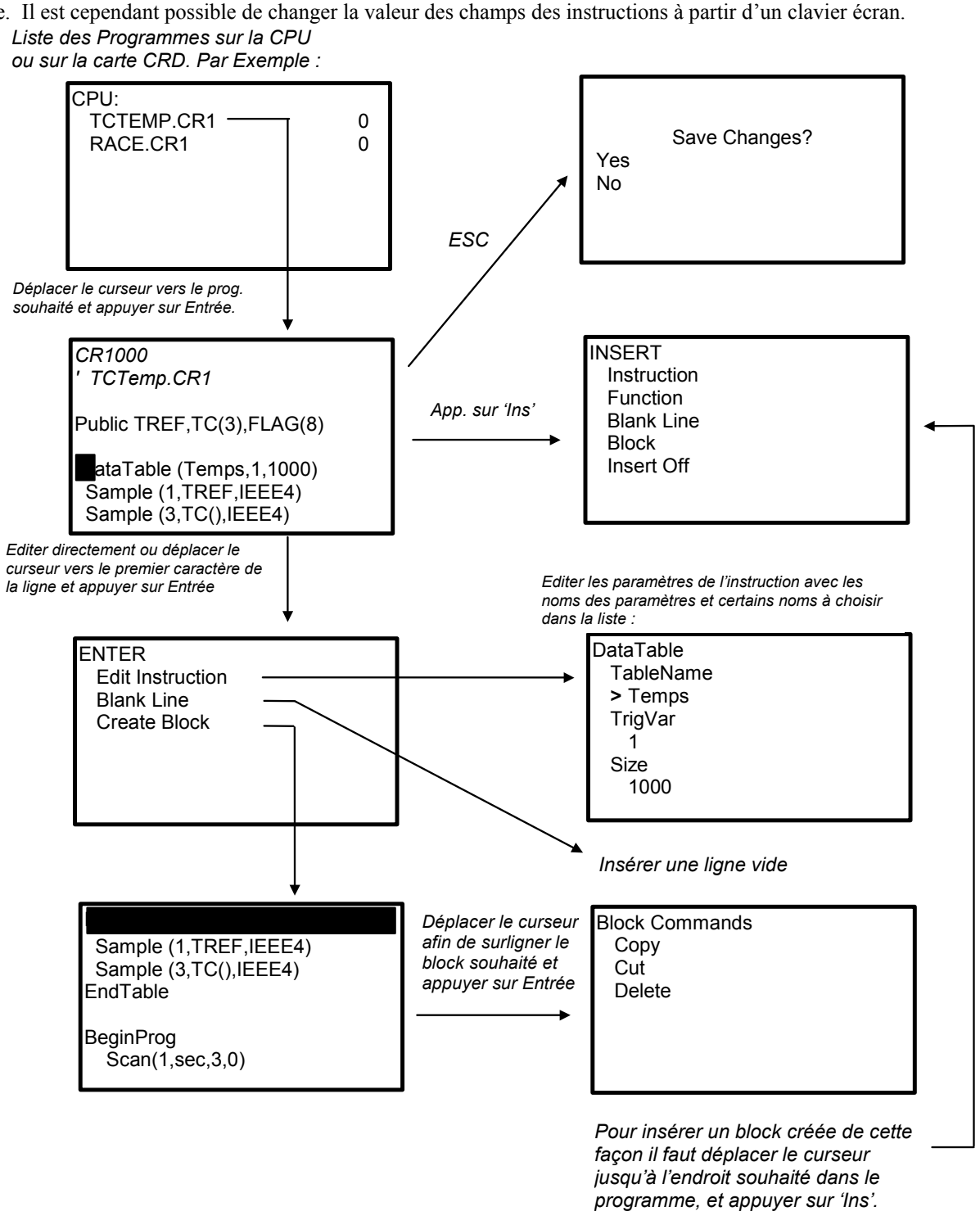


OV5.3 Afficher le fichier (File Display)

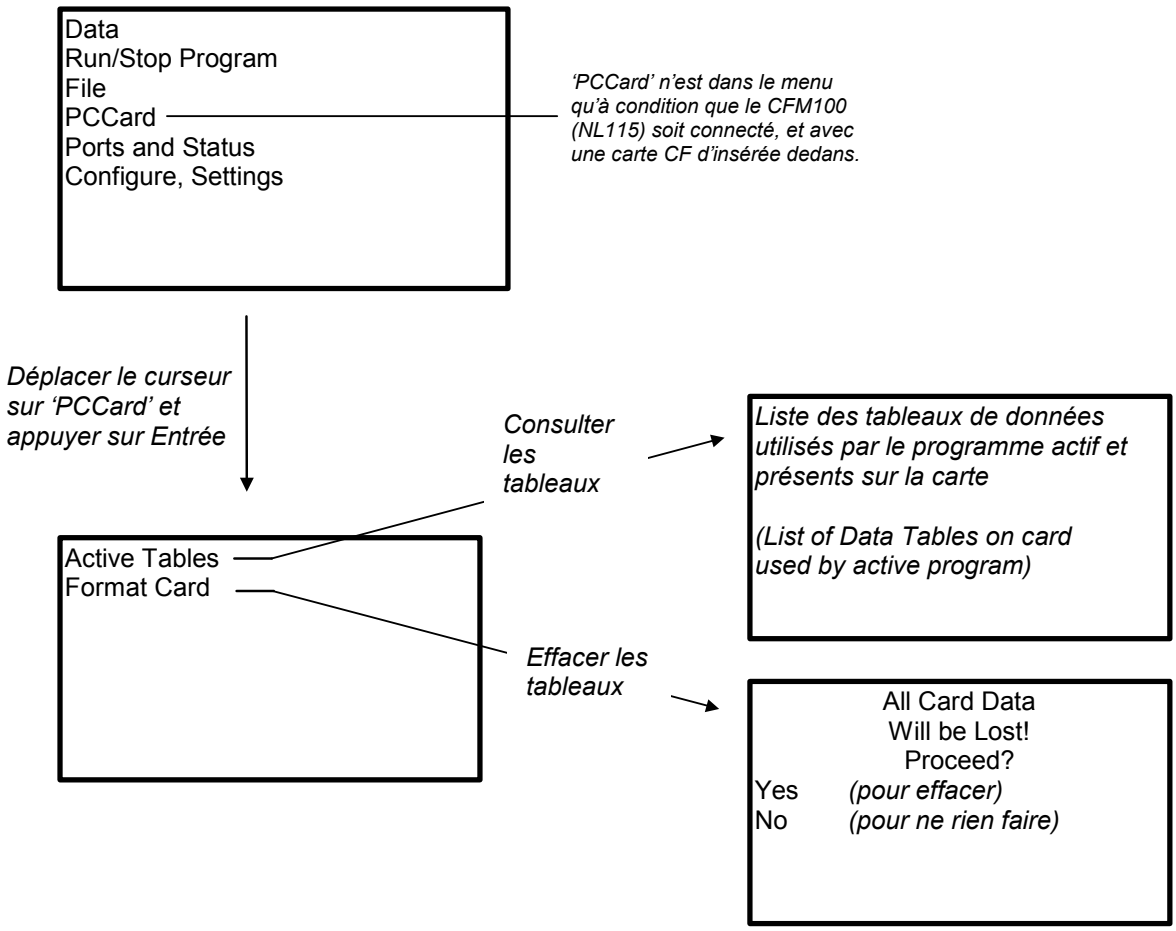


OV5.3.1 Editer le fichier (File: Edit)

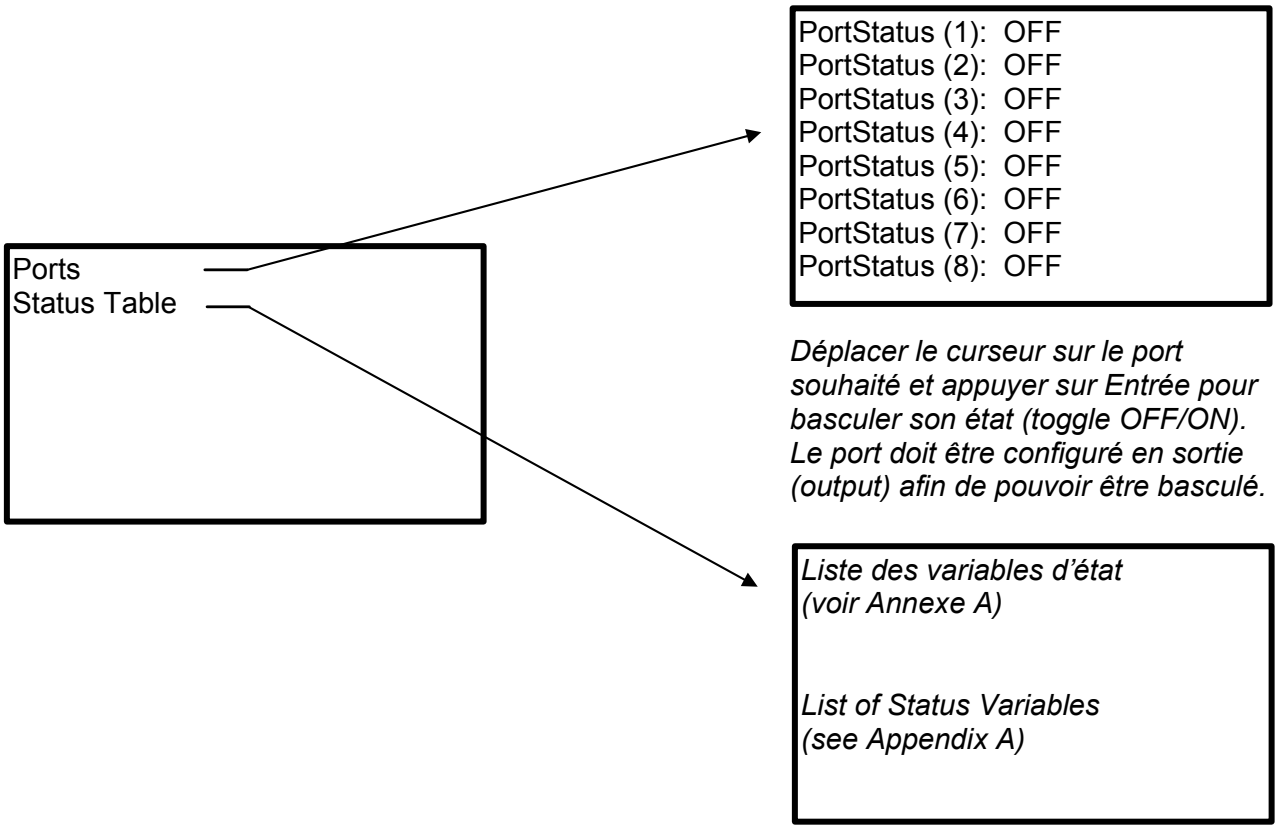
Il est conseillé d'utiliser l'éditeur de programmes CRBasic afin d'écrire et d'éditer les programmes des centrales de mesure. Il est cependant possible de changer la valeur des champs des instructions à partir d'un clavier écran.



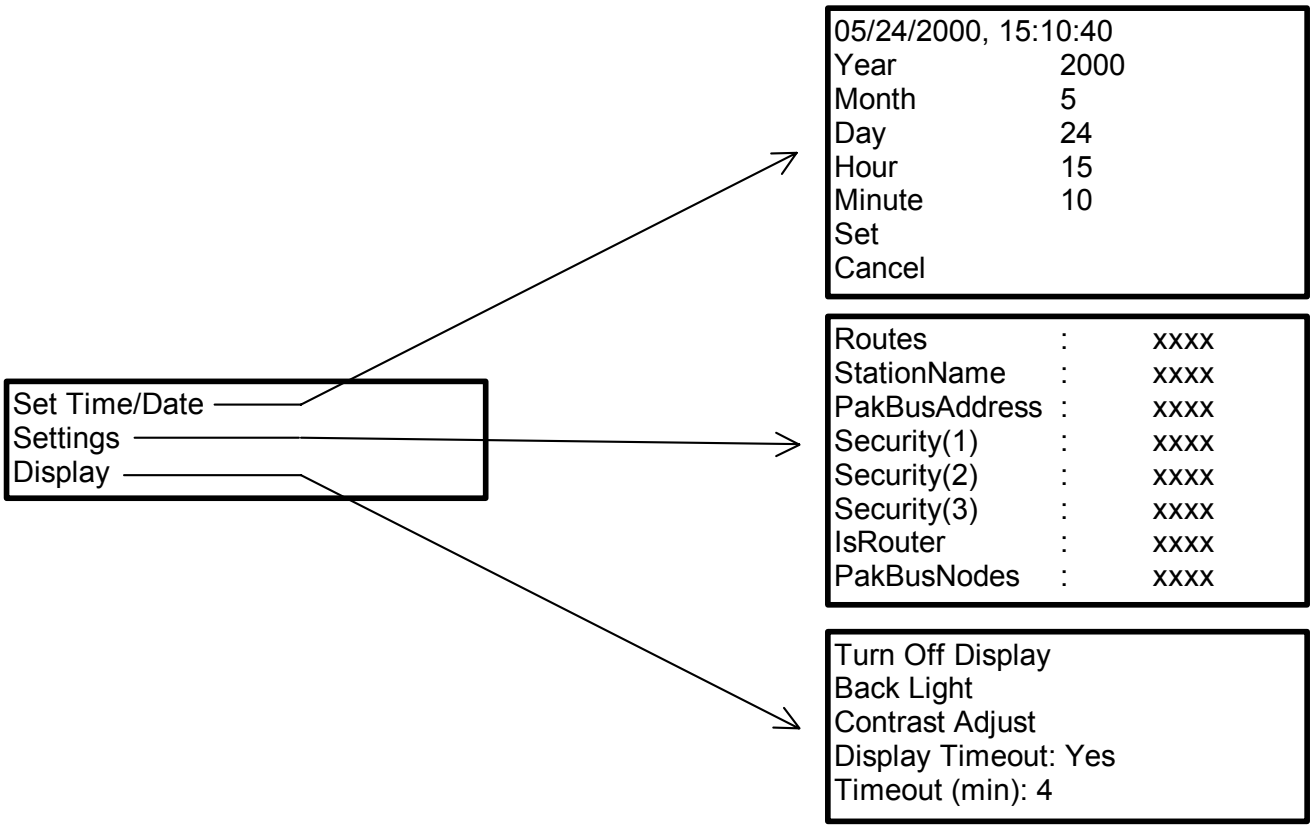
OV5.4 Affichage de la Carte PC (PCCard Display)



OV5.5 Etat de la centrale et des ports (Ports and Status)



OV5.6 Configurations (Settings)



Déplacer le curseur vers l'élément 'Time' et appuyer sur Entrée pour le modifier.

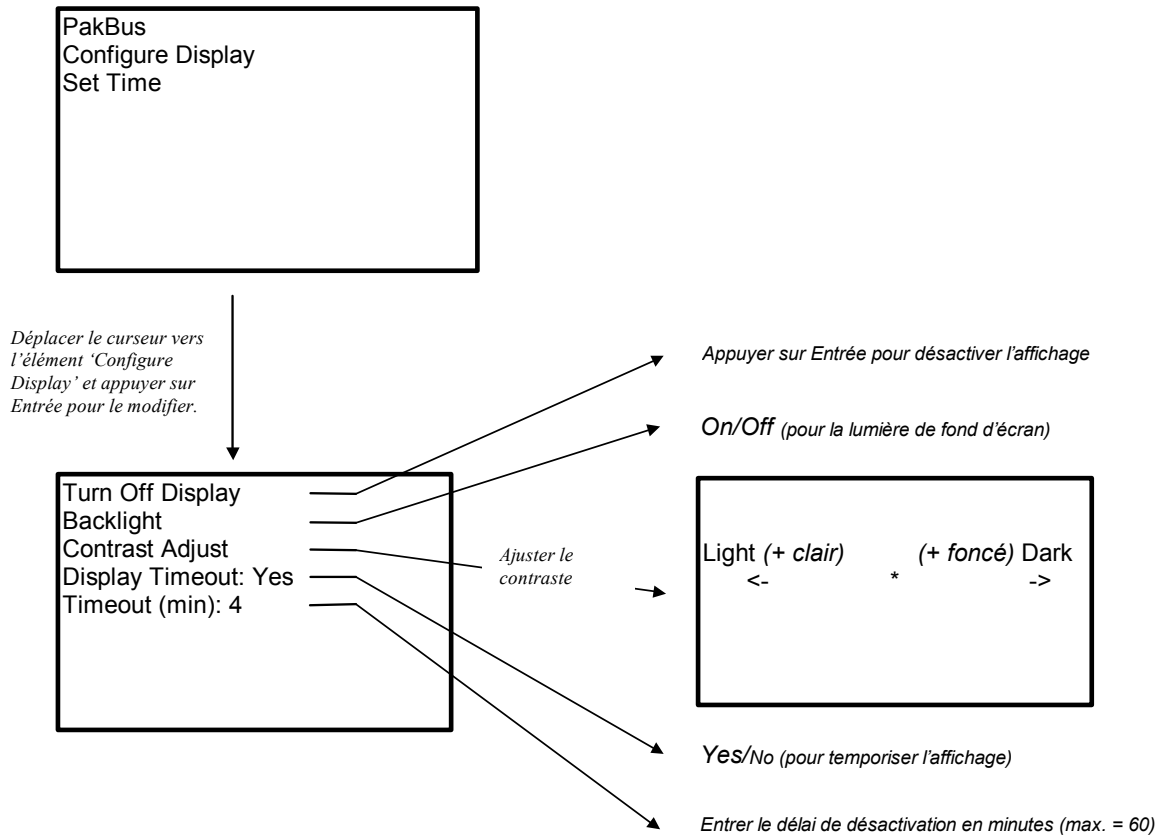
OV5.6.1 Configurations de l'horloge (Set Time/Date)

Déplacer le curseur jusqu'à l'élément que vous voulez modifier et appuyer sur 'Enter' pour le modifier. Puis déplacer le curseur jusqu'à '**Set'** et appuyer sur 'Enter' pour appliquer la modification.

OV5.6.2 Configurations PakBus (PakBus Settings)

Dans le menu de configuration « Settings », déplacer le curseur à l'élément PakBus et appuyer sur 'Enter' pour le modifier. Après la modification, appuyer sur 'Enter' pour appliquer la modification.

OV5.6.3 Configuration de l'affichage (Configure Display)



OV6. Caractéristiques (Spécifications)

Garanties pour une température comprise entre -25°C et +50°C sauf spécification spéciale. (Veuillez nous contacter pour des températures de fonctionnement étendues) Afin de garantir les caractéristiques techniques, Campbell Scientific recommande de ré-étalonner la CR1000 une fois tous les 2 ans.

TEMPS DE SCRUTATION DU PROGRAMME
10 ms à 30 min. @ incréments de 10 ms.

ENTREES ANALOGIQUES
Nombre de voies : 16 entrées unipolaires (SE) ou 8 différentielles (DF) avec configuration individuelle programmable par voie. Possibilité d'augmenter le nombre de voies en rajoutant les multiplexeurs AM16/32 et AM25T.

PLAGES DE MESURE, RESOLUTION ET BRUIT EN ENTREE : la résolution basique (Basic Res) est la résolution A/D d'une seule conversion. **La résolution des mesures DF avec une entrée réversible est la moitié de la « Basic Res ».** Les valeurs de bruits sont donnés pour les mesures DF en entrée réversible ; le bruit est plus important pour une mesure de type unipolaire SE.

Plage de Mesure (mV)	Basic Res (µV)	Int. 250 µs (µV RMS)	50/60 Hz (µV RMS)
±5000	1330	385	192
±2500	667	192	95,9
±250	66,7	19,2	19,2
±25	6,7	2,3	1,9
±7,5	2	0,62	0,58
±2,5	0,67	0,34	0,19

Offset pour DF avec entrée réversible = (Basic Res + 1,0 µV)
Offset pour DF sans entrée réversible = 2 Basic Res + 2,0 µV
Offset pour SE = 2 Basic Res + 3,0 µV

PRECISION [1]
±(0,06% de la lecture + offset), de 0°C à 40°C
±(0,12% de la lecture + offset), de -25°C à 50°C
±(0,18%)de la lecture + offset), de -55°C à 85°C
(Seulement pour la version -XT)

Offset pour les voies SE = 3BasicRes+3,0µV

TEMPS MINIMUM ENTRE LES MESURES DE TENSION : Inclut le temps de mesure et de conversion en unités de mesure. Pour les mesures de tension, la CR1000 intègre le signal d'entrée en 0,25 ms, en 16,66 ms ou en 20 ms pour un cycle de filtrage à 50/60 Hz. Les mesures DF avec entrée réversible incorporent deux intégrations avec une inversion de polarité pour réduire l'offset thermique et les erreurs en mode commun et seront ainsi deux fois plus longue.

Intégration analogique 250 µs : ~1 ms SE
Intégration analogique 1/60Hz : ~20 ms SE
Intégration analogique 1/50Hz : ~25 ms SE

MESURE EN MODE COMMUN : ±5V

FILTRAGE CC EN MODE COMMUN : > 100dB

FILTRAGE EN MODE NORMAL: 70dB @ 60 Hz
Lorsque le « 60 Hz réjection » est utilisé

TENSION D'ALIMENTATION MAXI. SANS DOMMAGE : ±16 V CC

COURANT D'ALIMENTATION : ±1nA typique ; ±6nA max. @50°C ; 90nA @ 85°C

RESISTANCE D'ENTREE : 20 GOhm typique

PRECISION DE LA THERMISTANCE POUR
la compensation de soudure froide (pour la mesure des thermocouples) :
±0,3°C, -25°C à 50°C
±0,8°C, -55°C à 85°C (-XT seulement)

SORTIES ANALOGIQUES
3 tensions commutées, actives l'une après l'autre uniquement pendant la mesure.

PLAGE ET RESOLUTION : Sorties tension programmable entre ±2,5V avec une résolution de 0,67mV.

PRECISION : ±(0,06% de base +0,8mV),0°C à 40°C
±(0.12% de base + 0,8 mV), -25°C à +50°C
±(0.18% de base + 0,8 mV), -55°C à 85°C (-XT seulement)

SOURCE D'INTENSITE : ±25 mA

LES MESURES DE RESISTANCES

TYPES DE MESURE : La CR1000 permet de mesurer le ratio de tension pour des ponts complets 4 et 6 fils, et des demi-ponts 2, 3, et 4 fils.

Les 3 sorties d'excitation sont à utiliser en double polarité pour éliminer les erreurs CC.

PRECISION DU RATIO DE TENSION[1] : En considérant une tension d'excitation de 1000mV, en n'incluant pas les erreurs des résistances du pont de mesure.

±(0,04° de la lecture + Offset/Vex)

La valeur de l'offset est divisé par un facteur de 2, lorsque vous utilisez une excitation inverse.

Offset DF avec entrée réversible= Basic Res + 1,0 µV
Offset DF sans entrée réversible= 2Basic Res +2,0 µV
Offset pour SE = 2 Basic Res + 3,0 µV

Offset pour les voies SE = 3BasicRes+3,0µV

LES MESURES DE PERIODE MOYENNE
La période moyenne d'un seul cycle est déterminée suite à la mesure moyenne d'un nombre spécifié de cycles. La résolution de la période est de 192 ns divisée par le nombre de cycles spécifié mesuré ; la précision de la période est ±(0,01% de lecture + résolution).[1]

N'importe quel des 16 voies unipolaires analogiques (SE) peuvent être utilisées pour mesurer une période moyenne. Certaines limitations sont à appliquer pour les voies SE analogique.

ETENDUE DE MESURE DE FREQUENCE :

E.M. EN ENTREE (MV)	*SIGNAL (PIC A PIC)[2] MIN*	*MAX*	*IMPULSION MINI*	*FREQ.[3] MAX*
±2500	500mV	10 V	2,5 µs	200 kHz
±250	10mV	2 V	10 µs	50 kHz
±25	5mV	2 V	62 µs	8 kHz
±2,5	2mV	2 V	100 µs	5 kHz

[2] les signaux sont centrés par rapport à la masse de la centrale d'acquisition.
[3] Fréquence maximum = 1/(largeur de deux fois le minimum d'impulsion) pour 50% signaux du cycle d'utilisation.

COMPTEURS D'IMPULSIONS
Deux entrés 24 bits configurable pour la mesure de contact sec, impulsions hautes fréquences ou CA bas niveaux.

COMPTAGE MAXIMUM PAR SCRUTATION : 16,7 x 106

MODE CONTACT SEC :
Temps minimum du contact fermé : 5 ms.
Temps minimum du contact ouvert : 6 ms.
Temps maximum de rebond : 1 ms ouvert sans être compté.

MODE D'IMPULSION HAUTE FREQUENCE :
Fréquence d'entrée maximum 250 kHz
Entrée tension maximum : ± 20V
Seuil de tension : Comptage au dessus d'une transition en dessous de 0,9V à plus de 2,2 V après le filtre d'entrée avec un temps constant de 1,2 µs.

MODE CA BAS NIVEAU
Le couplage interne CA corrige l'offset CC jusqu'à ± 0,5V
Hystérésis en entrée : 16mV à 1 Hz
Entrée tension CA maximum : ± 20V
Entrée tension CA minimum :

Signal sinusoïdal (mV RMS)	*E. M. (Hz)*
20	1,0 à 20
200	0,5 à 200
2000	0,3 à 10 000
5000	0,3 à 20 000

PORTS D'E/S NUMERIQUES

8 ports configurables par logiciel comme entrées binaires ou sorties de contrôle. Les ports C1 à C8 peuvent convenir à la synchronisation , routine, interruption, contact sec pour le comptage, comptage d'impulsion haute fréquence, communication asynchrone (UART), communications SDI-12 et communications SDM.

HAUTE FREQUENCE MAXIMUM: 400 KHz

FREQUENCE DE CONTACT SEC MAXI: 150 Hz

TENSION D'ENTREE (sans charge): haut 5,0V ±0,1V; bas < 0,1

RESISTANCE DE SORTIE : 330 Ohms

ETAT D'ENTREE: haut 3,8 à 5,3 V ; bas -0,3 à 1,2V

HYSTERISIS D'ENTREE: 1,4 V

RESISTANCE D'ENTREE: 100 Kohms

SUPPORT D'INTERFACE SDI-12

Les ports de contrôle 1,3,5 et 7 peuvent être configurés pour les communications asynchrone SDI-12. Jusqu'à 10 capteurs SDI-12 peuvent être connectés par port. Le standard SDI-12 version 1.3 pour le mode centrale d'acquisition est accepté.

COMPATIBILITE ELECTROMAGNETIQUE

CE standard, déclaré conforme à la norme :
BS EN61326 :2002

CPU ET INTERFACE

PROCESSEUR : Hitachi H8S 2322 (CPU 16-bit avec une architecture interne de 32 bit)

MEMOIRE : SRAM (protégée par pile) ; 2 Mbytes, 16 kbytes pour le stockage du programme ; 4 Mbytes en option.

INTERFACES SERIES : COM1 (CS I/O utilisé pour interfacer les périphériques de Campbell Scientific), COM2 (port de communication standard RS232)

INTERFACE PARALLELE : interface de 40 pin pour la connexion d'une carte de stockage de données ou un périphérique de communication tel que le module CFM100.

VITESSE DE TRANSMISSION : Sélection possible entre 300 à 115,2 kbps. Protocole ASCII avec 1 bit de début, 1 bit de fin, 8 bits de données et pas de parité.

PRECISION DE L'HORLOGE : ±3 min. par an (-30°C à 85°C) ; ±15 min. par an (-55°C à + 85°C, -XT seulement)

ALIMENTATION NECESSAIRE

TENSION : de 9,6 à 16V CC.
CONSOMMATION EN COURANT:
Au repos : ~0,6mA

Echantillon d'1Hz (8 mesures DF, 60Hz rej, 2 mesures d'impulsion)
Avec la communication RS232 : 19mA
Sans la communication RS232 : 4,2 mA

Echantillon d'1Hz (8 mesures DF., intégration 250 µs., 2 mesures d'impulsion)
Avec la communication RS232 : 16,7mA
Sans la communication RS232 : 1 mA

Echantillon de 100Hz (4 mesures DF, intégration 250µs)
Avec la communication RS232 : 27,6mA
Sans la communication RS232 : 16,2 mA

BATTERIES EXTERNES : 12 V CC nominale, protégé de l'inversion de polarité.

DIMENSIONS

TAILLE DU MODULE DE MESURE ET DE CONTROLE : 21,6 x 9,9 x 2,2 cm

TAILLE DU BORNIER CR1000WP : 23,9 x 10,2 x 6,1 cm. Vous devez ajouter à cela les dimensions des connecteurs du port série et des câbles des capteurs.

POIDS : 1Kg

GARANTIE

Trois ans pièces et main d'œuvre en usine**.**

Légendes des abréviations :

[1] Bruit du capteur et de la mesure non inclus.
DF = Voie différentielle
SE = Voie unipolaire
-XT = Version avec l'extension en température (en option).
Basic Res = Résolution de base
E.M. = Etendue de mesure

Chapitre 1. Installation et Entretien

1.1 Protection contre l'environnement

Les variables environnementales qui sont à surveiller sont la température et l'humidité. La CR1000 standard est faite pour fonctionner de façon optimum entre -25 et +50°C (-40 à +85°C en option), en atmosphère sans condensation. Lorsque l'humidité devient trop forte, les composants électroniques peuvent être endommager, le microprocesseur peut faillir, ou bien on peut perdre de la précision sur les mesures à cause de la condensation sur des composants du circuit imprimé. Le contrôle du niveau d'humidité supporté par les centrales, est de la responsabilité de l'utilisateur.

L'humidité à l'intérieur du module est éliminée grâce à un paquet de silicagel placé à l'intérieur de celui-ci au moment de l'assemblage en usine. Ce sachet est remplacé lorsque la CR1000 est réparée par Campbell Scientific. Le module de la CR1000 ne devra pas être ouvert, à moins de le faire afin de remplacer la pile au lithium qui protège la SRAM. Si on assemble / désassemble la CR1000 plusieurs fois, cela dégradera la jointure et pourra conduire à des problèmes de moisissure. Il sera par ailleurs nécessaire de placer des sachets de dessiccateurs dans le coffret afin de prévenir des risques de corrosion sur le bornier et les connexions entre la CR1000 et le bornier.

Campbell Scientific commercialise des coffrets environnementaux afin de contenir la CR1000 et ses périphériques. Ces coffrets en fibre de verre sont classifiés en tant que NEMA 4X (étanche, résistant à la poussière et à la corrosion, fait pour l'utilisation en extérieur et en intérieur). Une entrée/sortie de 1,25" de diamètre est présente sur la partie basse du coffret afin de permettre le passage des câbles. La porte du coffret peut être close à l'aide du fermoir présent, laissant un accès facile à l'intérieur, ou encore en utilisant les vis fournies si l'application nécessite moins d'ouverture pour le coffret (dans ce cas il faut retirer les petites protections en plastique blanc et y insérer les vis). Les coffrets sont de couleur blanche afin de réfléchir le rayonnement solaire et de réduire les gradients de température à l'intérieur du coffret.

1.2 Besoins en énergie

La CR1000 fonctionne en 12V CC nominal. En dessous de 9,6V ou au dessus de 16V, la CR1000 ne fonctionne pas correctement.

La tension d'entrée de la CR1000 est protégée par une diode contre l'inversion accidentelle de polarité en provenance de la batterie. Des tensions d'alimentation supérieures à 18V peuvent endommager la CR1000 et/ou l'alimentation. Un transzorb fournit une protection contre les transitoires en limitant la tension à approximativement 20V.

Le temps de fonctionnement du système avec batteries, peut être déterminé en divisant la capacité de la batterie (Ampères- Heure) par la consommation moyenne en courant du système. La CR1000 a une consommation moyenne en courant d'environ 0,5mA en état de veille (affichage inactif), de 0,6 mA pour un échantillonnage à 1Hz et de >10mA pour un échantillonnage à 100Hz.

ATTENTION

Les bornes 12V et 12V commuté de la centrale ne sont pas régulés par la CR1000 ; ils obtiennent leur tension directement à partir de la borne d'alimentation. Si vous utilisez la CR1000 pour alimenter des appareils, il faut vous assurer que votre alimentation régule correctement la tension ou que ces appareils soient compatibles avec les tensions supportées par la CR1000. La tension maximum que fournit une alimentation de Campbell Scientific en sortie, est d'approximativement 16V (tension de charge à –40°C).

1.3 Les alimentations de Campbell Scientific

Les alimentations disponibles auprès de Campbell Scientific, pour la CR1000, sont de type piles alcaline (référence BPALK) ou batterie acide / plomb (référence PS100).

Le CH100 comprend le même circuit que dans la PS100. Il est utilisé pour recharger une batterie externe 12V CC type Yuasa à l'aide d'un adaptateur secteur ou d'un panneau solaire. Le CH100 ne comprend pas de batteries. D'autres options d'alimentation relient une batterie 12V directement à la CR1000 (voir le paragraphe 1.5), ou bien fournissent de l'énergie à partir d'un véhicule (voir le paragraphe 1.6).

Chacune des alimentations dispose d'un fusible thermique dans le circuit d'alimentation afin de limiter l'intensité du courant fourni. Si il y a trop de demande en intensité, le fusible devient chaud, augmente la résistivité, et limite l'intensité. Lorsque le problème, qui a causé cette demande supérieure à la normale, est réglé, le fusible se refroidit, baisse en résistivité, et permet le cas échéant, le passage du courant. Si une demande importante en intensité est due à un court circuit sur les fils destinés à être reliés au bornier d'alimentation de la centrale de mesure, il faut laisser refroidir le fusible 10 à 15 secondes avant de re-connecter l'alimentation.

1.3.1 Alimentation par piles alcalines BPALK

La BPALK utilise 8 piles alcalines de type D (LR20) et est fourni avec un pack pour piles de type AA (LR06) afin de fournir du courant lors de la phase de remplacement des piles LR20. Les piles LR06 ne sont pas fournies.

Pour remplacer les piles sans interrompre l'exécution du programme de la CR1000 il vous faudra :

1) Connecter la batterie externe au à l'entrée étiquetée « temporary »
2) Enlever les anciennes piles
3) Mettre de nouvelles piles LR20 en place
4) Retirer la batterie externe.

Un jeu de piles LR20 toutes neuves fournira une tension d'environ 12,4 V et un rapport nominal d'environ 7,5 Ampère Heure à 20°C. Le rapport AH diminue avec la température tel que cela est indiqué au tableau 1.3-1. On peut utiliser l'instruction de mesure de tension batterie (Battery) afin de surveiller la tension des piles. Il faut alors remplacer les piles avant que la tension ne passe au dessous de 9,6V.

TABLEAU 1.3-1. Capacité typique des piles Alcalines en fonction de la Température	
Température (°C)	**% de la capacité à 20°C**
20 - 50	100
15	98
10	94
5	90
0	86
-10	70
-20	50
-30	30

NOTE

Ces chiffres sont donnés sur la base de piles LR20 sous des conditions de consommation de 50mA à 30 ohms de charge. Quand l'intensité de courant diminue, le pourcentage de capacité de service augmente pour la température donnée.

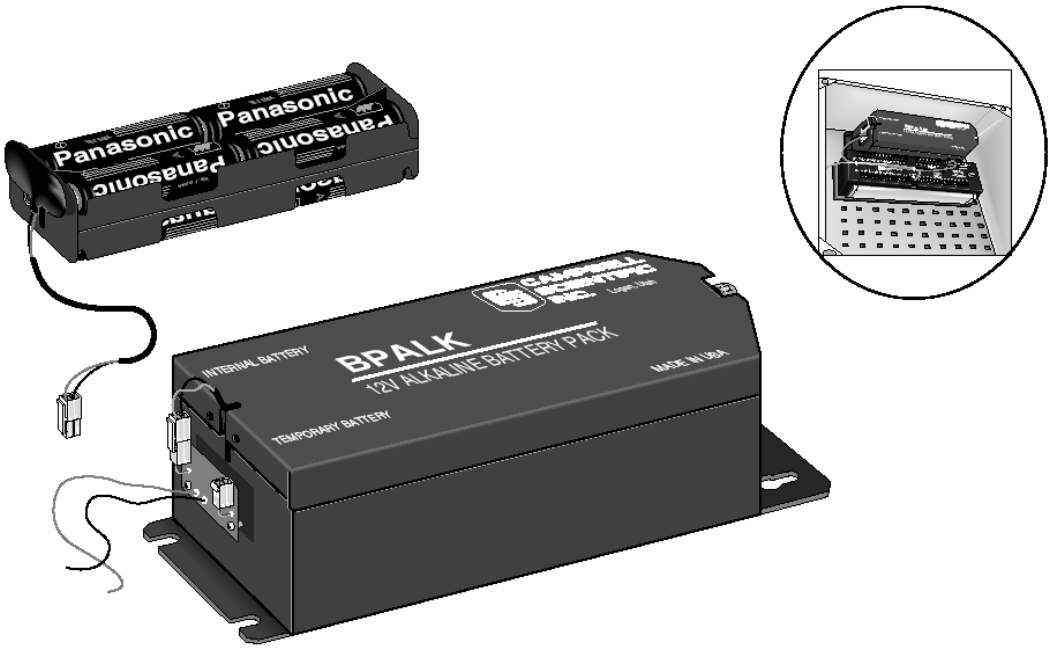


FIGURE 1.3-1. Alimentation BPALK

1.3.2 Alimentation Acide Plomb PS100

L'alimentation PS100 comprend une batterie acide plomb 7 Ampères Heure, un transformateur CA (18V) et un circuit de charge compensé en température avec une diode indiquant la charge. Il est recommandé de laisser un adaptateur secteur ou un panneau solaire connecté à la PS100 en permanence. La source qui sert à recharger la batterie servira aussi à alimenter la CR1000 en énergie. La batterie interne n'alimentera la centrale d'acquisition, qu'à condition que la source destinée à la recharge de la batterie soit interrompue ou insuffisante. Les caractéristiques de la PS100 sont données dans le tableau 1.3.2.

Les deux fils du circuit de charge peuvent être insérés dans n'importe laquelle des bornes étiquetées CHG, la polarité importe peu. Un transzorb fournit une protection contre les transitoires sur le circuit de charge. Une tension d'entrée soutenue supérieure à 40V conduira le transzorb à limiter la tension.

La lumière rouge (LED) de la PS100 est allumée lorsqu'une source de rechargement est connectée aux bornes CHG. L'interrupteur permet de mettre en fonctionnement ou de désactiver la fourniture de tension 12V (« on -*marche*» ou « off - *arrêt* »). La mise en charge de la batterie continue à se produire lorsque l'interrupteur est en position « off ».

ATTENTION

Il est recommandé de mettre l'interrupteur sur « off » avant de déconnecter ou de connecter les fils destinés à être reliés aux bornes d'alimentation de la CR1000. Le bornier de la CR1000 et la PS100 sont tous deux reliés à la même masse. Si le 12V est court-circuité à l'une de ces masses, il y aura beaucoup de courant demandé jusqu'à ce que le fusible thermique de la PS100 ne se mette en fonctionnement.

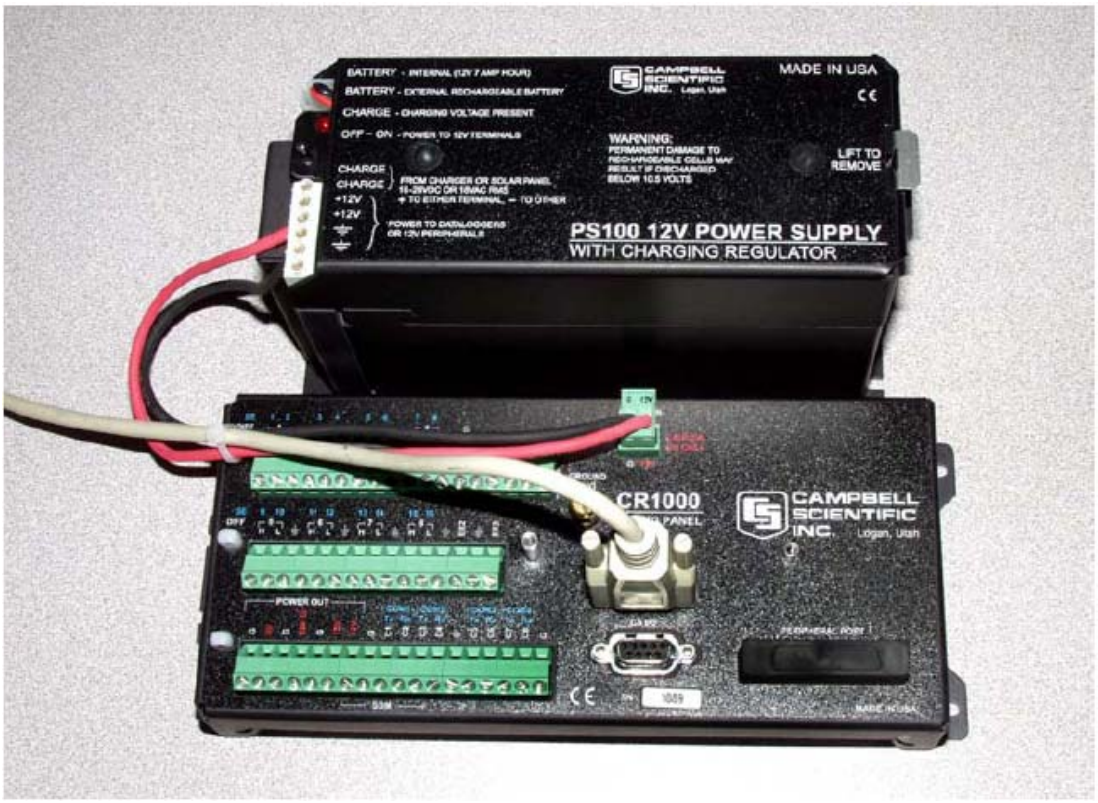


FIGURE 1.3-2. PS100

Il est conseillé de surveiller et de garder trace de la tension d'alimentation à l'aide de l'instruction « Battery » insérée dans votre programme de CR1000. Si la tension d'alimentation du système décroît de façon constante à travers le temps, c'est qu'un ou des éléments du circuit de charge sont en défaut. L'instruction « Battery » mesure la tension aux bornes de la voie « Power In » et non pas la tension réelle de la batterie. Pour mesurer la tension réelle de la batterie qui est incorporée à la PS100, il faudrait déconnecter le circuit de charge et la CR1000.

TABLEAU 1.3-2. Caractéristiques des PS100, Batterie, et transformateur CA

Tension en entrée (bornes CHG)	De 15 à 28V CC ou 18V CA RMS
Branchements de la batterie	
Tension de charge en sortie :	Charge flottante pour batterie 12V, compensée en température
Etendue de compensation en température :	De -40 à +60ºC
Limite de courant de charge :	Typiquement 1,2 A
En sortie (bornes +12)	
Tension :	12V non régulé provenant de la batterie
Courant limité par un fusible thermique 3A :	> 3 A @ < 20°C 3 A @ 20°C 2.1A @ 50°C 1.8 A @ 60°C
Batteries	
Etendue de fonctionnement en température :	-40 to +60ºC
Capacité :	
PS100E-LA	7 Ampères Heure
BP17	17 Ampères Heure
BP24	24 Ampères Heure
AC Transformer: CSI Model No. 9591	
Tension d'entrée:	120 VAC
Tension de sortie:	18 VAC RMS
Courant de sortie (max):	1.2 Amps RMS
Protection (Initialisation automatique):	85°C thermal reset breaker
Approuvé UL:	UL-1950
AC Transformer: CSI Model No. 14014	
Tension d'entrée:	90 - 264 VAC; 47 - 63 Hz
Tension de sortie:	18 VDC
Courant de sortie (max):	1.3 Amps
Approuvé UL, Fichier No.:	E137895

Il y a des dangers possibles lorsqu'on utilise des batteries acide plomb. En utilisation normale, elles génèrent une petite quantité d'hydrogène gazeux. Ce produit gazeux est généralement insignifiant car l'hydrogène disparaît de façon naturelle avant d'atteindre un niveau explosif (4%). Il est cependant possible, si les batteries sont court-circuitées ou qu'elles soient trop chargées, que la quantité d'hydrogène produite soit suffisante pour être dangereuse. Campbell Scientific effectue les recommandations suivantes :

1. Dans des applications nécessitant un niveau de sécurité intrinsèque il ne faut JAMAIS utiliser une CR1000 avec des batteries acide plomb.
2. Il ne faut pas mettre une batterie acide plomb dans un coffret hermétique aux gaz.

1.3.3 Adaptateur Null Modem A100

L'adaptateur pour Null Modem, l'A100, est utilisé lorsqu'il est nécessaire de disposer de 5V afin d'alimenter des modems externes à partir d'une PS100 ou d'un CH100. L'A100 fournit alors du 5V sur la broche 1 du port 9 broches Null Modem. L'A100 est souvent utilisé dans les applications nécessitant de la télémétrie radio.

La courant maximum que peut fournir le 5V de l'A100 est de 150 mA.

1.4 Panneaux Solaires

L'énergie photovoltaïque peut être utilisée afin de maintenir la charge des batteries acide plomb.

Lorsqu'on choisit un panneau solaire, il y a une règle de bon sens qui veut qu'on fasse en sorte que ce panneau soit capable de fournir autant d'énergie, que le système n'en consomme lors d'une journée peu ensoleillée (soit par exemple 10% du rayonnement solaire moyen annuel, en kW/m²). Les informations spécifiques au site, si elles sont disponibles, pourront grandement influencer le choix du panneau solaire. Les effets locaux tels que les ombres des montagnes, les brouillards provenant des vallées, la glace, les feuilles d'arbres ou la présence d'oiseaux etc., peuvent diminuer la surface efficace du panneau solaire et devront être prises en compte.

1.5 Connexion directe de la batterie, au bornier de la CR1000

N'importe quelle source d'énergie non bruitée et fournie par batterie, de 11 à 16V, peut être connecté aux bornes « Power In » du bornier de la CR1000. Lorsque vous reliez une batterie externe à la CR1000 il vous faudra tout d'abord déconnecter le connecteur vert destiné à l'alimentation. Insérez ensuite la borne positive de la batterie à l'emplacement marqué « +12 », puis la borne négative à l'emplacement marqué « G ». Vérifiez une dernière fois la polarité, et mettez en place le connecteur vert maintenant relié à la batterie et à la CR1000.

1.6 Connexion d'alimentation sur véhicule

Si la CR1000 est alimentée par une batterie 12V provenant d'un véhicule à moteur, il est nécessaire de disposer d'une seconde source de 12V. Lorsqu'un véhicule à moteur est mis en route et que son alimentation 12V est sollicitée, la tension chute en dessous de 11V et peut alors causer des interruptions de mesure à chaque fois que le véhicule est démarré. La seconde alimentation 12V évitera ce souci. La figure 1.6-3 montre comment relier les deux sources d'énergie à la CR1000. Les diodes permettent au véhicule d'alimenter la CR1000 sans que la seconde alimentation essaye d'alimenter le véhicule.

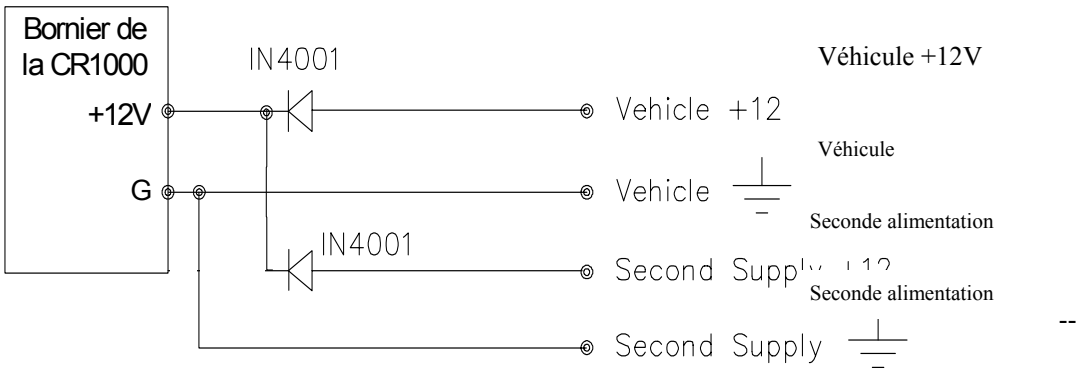


FIGURE 1.6-3. Connexion de la CR1000 à une alimentation pour véhicule

1.7 Mise à la terre de la CR1000

La mise à la terre de la CR1000, des périphériques et des capteurs, est un point important pour toutes les applications. La mise à la terre permettra d'assurer la protection ESD (décharge électrostatique – Electro Static Discharge) et une meilleure précision dans les mesures.

1.7.1 Protection ESD (electrostatic discharge)

Une ESD (décharge électrostatique) peut avoir plusieurs sources. Cependant la source la plus commune, et bien souvent la plus destructrice, sont les décharges foudroyantes directes ou secondaires. Les décharges directes (primaires) touchent la centrale de mesure ou les capteurs directement. Les décharges secondaires induisent des tensions sur les fils d'alimentation ou les câbles des capteurs.

Les éléments premiers pour la protection contre les ESD, sont les éclateurs à gaz (GDT, Gas-Discharge Tubes). Toutes les entrées et sorties de la CR1000 sont protégées avec des éclateurs à gaz ou des diodes de suppression de tension transitoire. Les GDT éclatent à 150V afin de permettre au courant d'être dévié vers la masse générale. Afin qu'ils soient efficaces, il faut que la masse de la centrale d'acquisition soit reliée au châssis de votre système de support. Comme cela est indiqué à la figure 1.7-1, la masse d'alimentation et la masse des capteurs, sont des lignes indépendantes, jusqu'à ce qu'elles se rejoignent dans la CR1000.

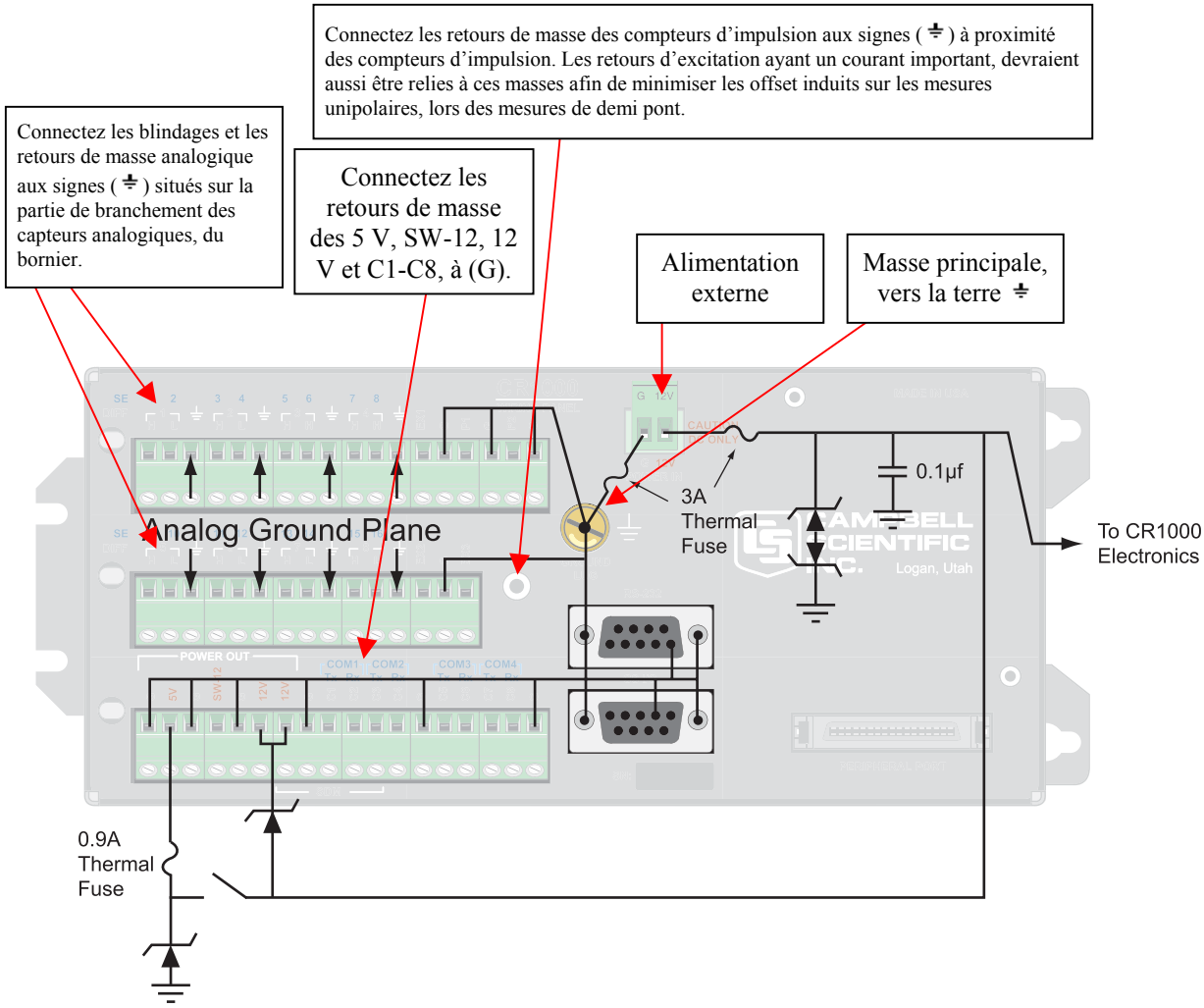


FIGURE 1.7-1. Schéma des masses de la CR1000

Le port 9 broches I/O de la CR1000 est encore une autre voie possible pour que les transitoires arrivent jusqu'à la CR1000 et la détériore. Les appareils de communication tels que les modems ou les modems courte distance, doivent être protégés par des protections à éclateur à gaz. Ce type de protection sont souvent proposées en option avec ces produits et doivent toujours être demandés lors de la commande. Les protections foudre de ces appareils devront alors être connectés à la masse principale de la CR1000, à la masse du coffret, ou à la masse du châssis.

Une bonne prise de terre (châssis) permettra de minimiser les dommages sur la centrale de mesure et les capteurs, en créant un chemin à un point à faible potentiel. Campbell Scientific recommande de connecter chaque centrale de mesure à la terre (châssis). Chaque composant du système (centrales de mesure, capteurs, alimentation externe, support, boîtiers etc.) devra être relié à une seule terre (châssis).

Lorsque le matériel est installé sur le terrain, le minimum conseillé est d'avoir un piquet en cuivre de 2 à 3m de long, enfoncé dans la terre et relié à la borne de mise à la terre de la CR1000 via un fil de cuivre de diamètre 12 AWG. Dans des substrats à faible conductivité, tels que le sable, les sols très secs, la glace ou les rochers, un seul piquet de terre ne devrait pas fournir de point de masse suffisant. Dans ce type d'installations, consultez des documents relatifs à la protection foudre, ou contactez un spécialiste dans ce domaine. Une très bonne source d'information au sujet des protections foudre, peut être trouvée sur Internet à : http://www.polyphaser.com/ .

Dans les applications automobiles, le fil de masse doit être solidement relié au châssis du véhicule via un câble 12 AWG ou plus épais.

Pour des applications en laboratoire, il n'est pas toujours facile de trouver un point de masse stable. Dans les bâtiments un peu anciens, des plaques en cuivre récentes reliées à des prises de courant anciennes, peuvent indiquer qu'une prise de terre de bonne qualité existe, alors qu'en fait la prise n'y est pas reliée. Si une prise de terre sûre n'existe pas, il est de bon usage de vérifier qu'elle ne fait transiter aucun courant. Si l'intégrité de l'installation CA est à mettre en doute, mettez alors le système à la terre via le bâtiment, par le système de canalisation ou un autre moyen de connexion à la terre.

1.7.2 Effet de la mise à la masse sur les mesures : Etendue de mesure en mode commun (Common Mode Range)

L'étendue de mesure en mode commun est l'étendue de tension par rapport à la masse de la CR1000, dans laquelle les deux entrées d'une mesure différentielle doivent être contenues afin que la mesure différentielle soit effectuée correctement. Le mode commun pour la CR1000 est de ±5,0V. Si par exemple la borne positive de la mesure différentielle est à 2V, et la borne négative est à 0,5V par rapport à la masse de la CR1000, alors une mesure effectuée avec l'étendue de mesure ±5,0V indiquerait un signal de 1,5V. Si par contre l'entrée positive était à 6V, le mode commun sera dépassé et la mesure sera sans doute entachée d'erreur.

L'étendue de mode commun peut être dépassée lorsque la CR1000 mesure les sorties d'un capteur qui a sa propre masse d'alimentation, et si la borne négative de la mesure est référencée à la masse de l'alimentation du capteur. Si la masse du capteur et la masse de la CR1000 ont des potentiels suffisamment différents, le signal pourra excéder l'étendue de mode commun. Pour résoudre ce problème, la masse d'alimentation du capteur et la masse de la CR1000 devraient être reliées ensemble afin de n'avoir qu'une seule masse pour le système.

Lors d'applications en laboratoire et si plus d'une source de CA sont utilisées pour alimenter plusieurs capteurs, il n'est pas très prudent de considérer que toutes les masses sont au même potentiel. Pour que l'installation soit plus sûre, les masses de chaque prise CA devraient être reliées ensemble par du fil 12 AWG.

1.7.3 Effet de la mise à la masse sur les mesures unipolaires

Les mesures de tension unipolaires de faible tension peuvent être problématiques à cause des fluctuations de potentiel de la masse. Le schéma de mise à la terre de la CR1000 a été dessiné afin d'éliminer les fluctuations de potentiel de masse dues aux changements des courants passant en retour des voies 12 V, SW-12, 5 V, ainsi que des ports de contrôle. Cela est effectué grâce à l'utilisation de voies séparées pour les masses de signaux (⏚) et les masses d'alimentation (G). Pour que cette configuration soit efficace il faut observer la règle de branchement suivant :

NOTE

Toujours relier la masse d'un appareil à proximité de la borne active associée à cette masse. Plusieurs fils de masse peuvent être reliés à la même borne de masse.

Exemples :

1. Connecter les masses des 5 Volt, 12 Volt, et ports de contrôle à la borne G.
2. Relier la masse associée à l'excitation à la borne ⏚ la plus proche, sur la partie dédiée aux excitations.
3. Connecter la masse de la mesure unipolaire à la borne ⏚ la plus proche, sur la partie dédiée aux mesures analogiques.
4. Connecter les fils de blindage à la borne ⏚ la plus proche sur la partie dédiée aux mesures analogiques.

Si des problèmes d'offset se produisent à cause de fils de blindage ou de fils de masse ayant un passage de courant important, le fait de connecter les fils causant le problème aux bornes ⏚ proches des voies d'excitation et de mesure d'impulsion, devrait aider. Les fils causant le problème peuvent aussi être connectés directement à la borne de masse principale afin de minimiser les offsets induits par les mesures unipolaires.

1.8 Alimentation des capteurs et des périphériques

La CR1000 peut devenir une source d'alimentation qui peut être utilisée pour des appareils nécessitant une alimentation continue ou semi-continue en 5V ou 12V CC. La CR1000 dispose de deux bornes pour fournir du 12V continu (12V), d'une borne pour du 12V commuté (SW-12) et d'une borne pour du 5V continu (5V). La tension sur les bornes 12V et SW-12 variera en fonction de la tension d'alimentation de la CR1000. La borne 5V est régulée et fournira, elle, toujours une tension proche de 5V (±4%) tant que la tension d'alimentation de la CR1000 sera supérieure à 9,6V. La borne 5V n'est pas recommandée pour l'excitation de ponts de mesure de résistances. Le tableau 1.8-1 montre les limite de courants que peuvent fournir les ports 5V et 12V. Le tableau 1.8-2 montre les courants nécessaires à l'alimentation de certains périphériques fabriqués par Campbell Scientific. Les autres appareils ont normalement leur consommation de listée dans leurs caractéristiques techniques. La consommation en courant de l'ensemble des périphériques et des capteurs ne devrait pas dépasser les limites de courant que la CR1000 peut fournir.

TABLEAU 1.8-1Q. Limites de courant que la CR1000 peut fournir

Bornes	Limite de courant à fournir
SW12	< 900 mA @ 20°C < 729 mA @ 40°C < 630 mA @ 50°C < 567 mA @ 60°C < 400 mA @ 80°C
12V + SW12	< 1,85 A @ 20°C < 1,50 A @ 40°C < 1,30 A @ 50°C < 1,17 A @ 60°C < 0,85 A @ 80°C
5V + CSI/O	< 200 Ma

Assurez-vous que la source primaire d'alimentation pour la CR1000 est capable de fournir le courant nécessaire durant la période de temps souhaitée. Au besoin contactez un ingénieur d'application chez Campbell Scientific afin de dimensionner votre besoin d'énergie nécessaire dans le cas, où vous approchez les limites de capacité de votre système d'alimentation. Soyez très vigilant lorsque vous utilisez des panneaux solaires ainsi que des modems téléphoniques GSM ou bien des radios, lorsque ces applications doivent être capables de fonctionner durant des périodes de temps longs et sans visites sur site ; soyez aussi vigilant si le matériel est utilisé à des températures extrêmes.

TABLEAU 1.8-2. Consommation typique pour certains périphériques

Périphérique	Consommation typique (mA) En veille	Actif
AM25T	0,5	1
COM210 Phone Modem	0,0012	140
SDM-INT8	0,4	6,5

1.9 Contrôle de l'alimentation de capteurs et de périphériques

Le contrôle de l'alimentation d'un appareil extérieur, est une utilisation habituelle de la CR1000.

Plusieurs appareils peuvent être contrôlés par le SW-12 (la tension 12V commutée) de la CR1000. Le tableau 1.9-1 donne les quantités de courant disponibles depuis le port SW-12.

Les applications qui nécessitent plus de port de contrôle ou une source d'alimentation plus importante, peuvent généralement être satisfaites avec l'utilisation du A21REL-12, du SDM-CD16AC ou en utilisant les ports de contrôle C1 à C8 comme cela est décrit au paragraphe 1.9.1.

1.9.1 Utilisation des ports de contrôle numériques E/S afin de commuter des relais

Chacun des huit ports de contrôle (C1 à C8) peuvent être configurés en tant qu'entrée, ou mis à l'état haut ou bas (0 ou 5V) par l'instruction PortSet ou WriteIO. On utilise souvent un port de contrôle numérique pour piloter un circuit externe de relais, puisque le port de contrôle lui même a une capacité de source de courant qui est limitée (2,0 mA minimum à 3,5V).

La figure 1.9-1 montre un circuit typique pour le pilotage d'un relais, en jonction avec un relais à bobine qui peut être utilisé pour piloter le circuit d'alimentation d'un autre appareil. Dans cet exemple, lorsque le port de contrôle est activé, le courant provenant de l'alimentation 12V passe au travers du relais à bobine, fermant ainsi le relais, ce qui ferme le circuit d'alimentation du moteur, qui s'allume donc.

Dans d'autres applications, il peut être nécessaire de commuter simplement le courant, sans passer par un relais. La figure 1.9-2 montre un circuit afin de commuter une source d'alimentation externe sans passer via un relais. Si le périphérique qui doit être alimenté, consomme plus de 75mA à température ambiante, (la limite moyenne du transistor d'alimentation 2N2907A), l'utilisation d'un relais (voir figure 1.9-1) devra être nécessaire.

D'autres circuits activés par des ports de contrôle peuvent être utilisés avec des applications demandant plus de courant que celles mentionnées sur les figures 1.9-1 et 1.9-2. Pour de plus amples informations, vous pouvez contacter Campbell Scientific.

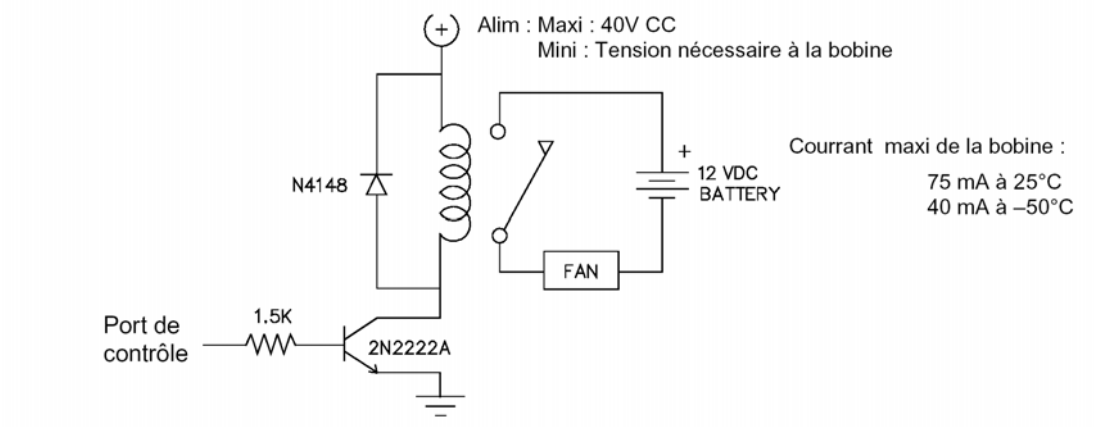


FIGURE 1.9-1. Circuit de pilotage avec un relais

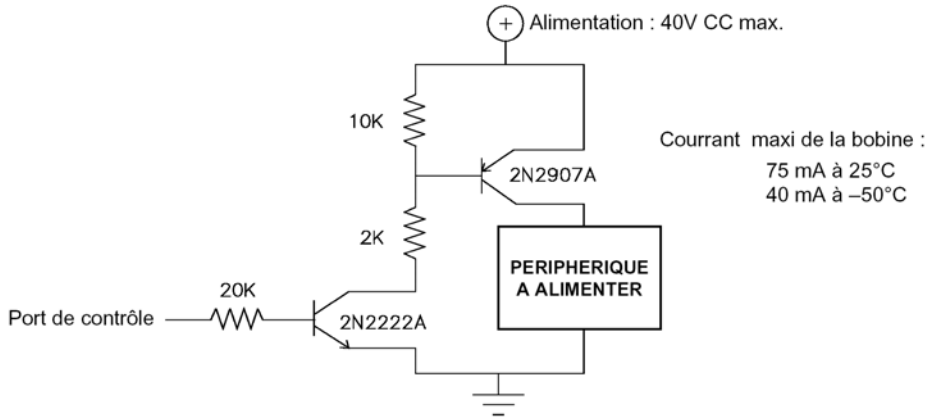


FIGURE 1.9-2. Circuit de pilotage sans relais

1.10 Entretien, maintenance

Les alimentations de la CR1000 nécessitent un minimum d'entretien de routine.

Lorsqu'elle n'est pas en cours d'utilisation, la batterie rechargeable doit être stockée dans un endroit frais et sec, avec un chargeur de courant alternatif connecté à elle (et en fonctionnement).

1.10.1 Dessiccateur

La CR1000 est fournie avec du dessiccateur afin de réduire l'humidité. Le sachet de dessiccateur devra être changé de façon périodique. Pour éviter d'avoir de la corrosion dans des atmosphères incontrôlées ou soumises à condensation, la CR1000 doit être mise en place dans un coffret environnemental avec du dessiccateur à l'intérieur. Ne mettez pas entièrement du joint sur votre coffret, si vous utilisez des batteries acide plomb. L'hydrogène généré par les batteries pourrait créer une atmosphère explosive.

1.10.2 Remplacer la pile interne

ATTENTION La mauvaise utilisation de la pile au lithium, ou le fait de la positionner incorrectement, peuvent conduire à de graves dommages : feu, explosion, ou brûlures sévères ! Il ne faut pas recharger, ouvrir ou chauffer la pile à plus de 100°C (donc ne pas souder directement sur la pile ni l'incinérer) ; ne pas exposer la pile à l'eau.

La CR1000 comprend une pile au lithium qui a pour rôle de conserver l'horloge et les données de la SRAM lorsque la CR1000 n'est pas alimentée. La CR1000 ne consomme aucun courant provenant de la pile au lithium lorsqu'elle est alimentée par une source de 12V CC. Avec une CR1000 stockée à température ambiante, la pile au lithium devrait servir durant environ 10 ans (un peu moins si elle subit des températures extrêmes). Lorsque la centrale de mesure est alimentée en continu ou bien la plupart du temps, la pile au lithium devrait fonctionner beaucoup plus longtemps encore.

Tant que la CR1000 est alimentée par une source externe, la CR1000 mesure la tension de la pile au lithium de façon journalière. Cette tension est affichée dans le tableau d'état («status table » - paragraphe 1.6). Une pile neuve aura une tension approximative de 3,6V. Dans le tableau d'état il y a un champ "Lithium Battery". Ce champs est en position "True" (la pile est en bon état) ou "False" (il faut remplacer la pile). Si on retire la pile au lithium ou si on laisse la pile se décharger au dessous de son seuil critique, la CR1000 continuera de fonctionner correctement lorsqu'elle est alimentée ! Sans la pile au lithium, l'horloge sera ré-initialisée et les données seront perdues dès que l'on coupera l'alimentation de la CR1000.

On peut commander une pile de remplacement auprès de Campbell Scientific (numéro de stock : 13519). Le tableau 1.10-1 liste les caractéristiques de la pile.

TABLEAU 1.10-1. Caractéristiques de la pile au lithium de la CR1000

Modèle	Tadiran	TL-59025 (3.6 V)
Capacité		1.2 Ah
Taux d'autodécharge		1%/an @ 20°C
Etendue de température de fonctionnement		-55°C to 85°C

La CR1000 doit être partiellement désassemblée afin de remplacer la pile au lithium.

Les figures 1.10-1 à 1.10-5 illustrent comment désassembler la CR1000. Il faudra alors effectuer ces opérations en sens inverse afin de ré-assembler la CR1000.

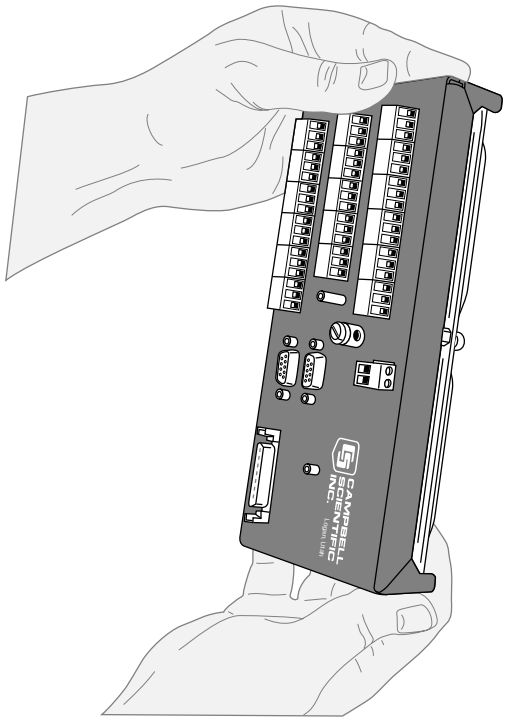


FIGURE 1.10-1. CR1000 avec son bornier.

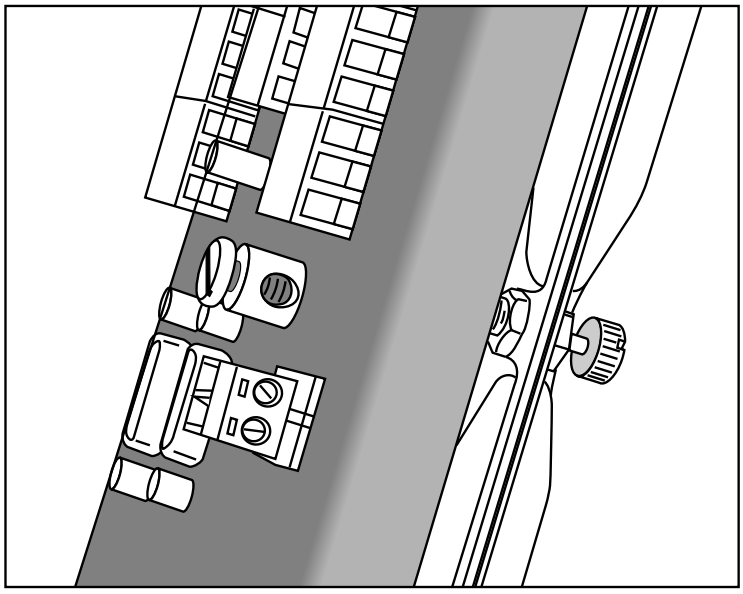


FIGURE 1.10-2. Dévisser les deux vis à bord striées afin de déboîter la partie métallique et le bornier.

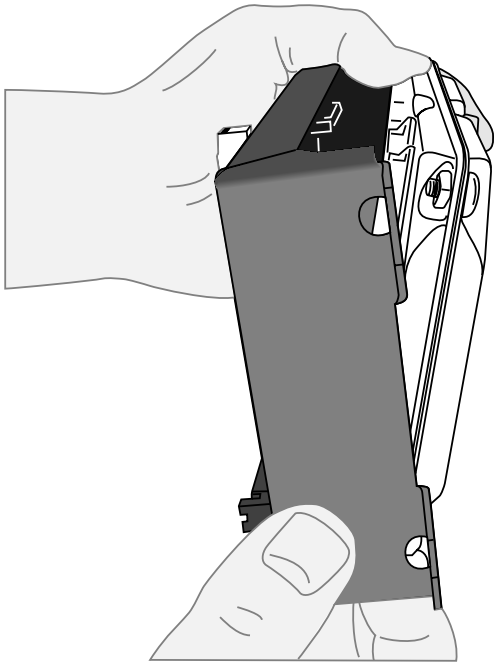


FIGURE 1.10-3. Pousser sur la partie où sont les vis striées pour retirer le bornier.

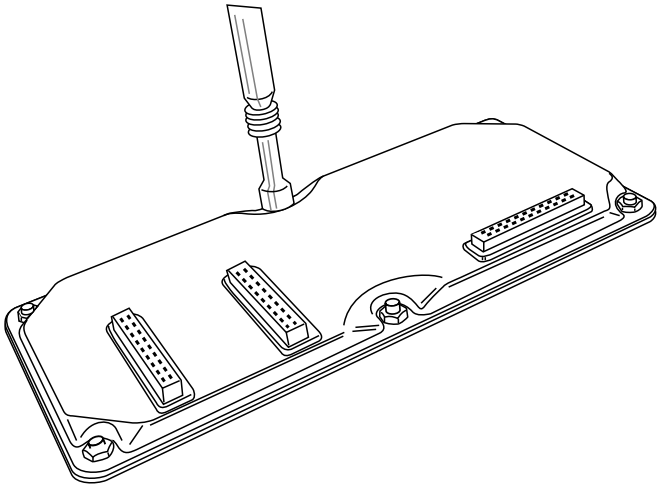


FIGURE 1.10-4. Retirer les écrous afin de désassembler la partie métallique de la centrale.

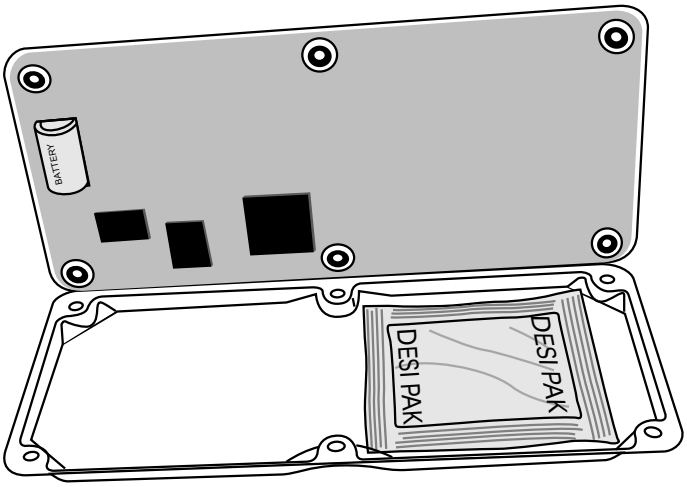


FIGURE 1.10-5. Retirer et remplacer la pile.

Chapitre 2. Stockage et récupération des données

La CR1000 peut enregistrer des données brutes individuelles, mais elle peut aussi être utilisée pour calculer des moyennes, des mini, maxi etc., sur des périodes de temps fixes ou conditionnelles. Les données sont stockées sous forme de tableau. Le nombre de tableaux et de valeurs qui peuvent être enregistrées dans chaque tableau, sont sélectionné lorsque l'on utilise le générateur de programmes Short Cut (voir l'Aperçu – début du manuel), ou lorsqu'on utilise un programme d'édition direct (voir paragraphes de 4 à 9).

2.1 Enregistrement de données sur la CR1000

Il existe deux zones où il est possible de stocker des données de CR1000 :

- La SRAM interne
- La carte compact flash qui est présente dans les modules optionnels CFM100 ou NL115.

La SRAM interne est utilisée uniquement en tant que mémoire de stockage pour les tableaux de données, ou encore en tant que mémoire tampon lorsque les données sont par la suite envoyées vers une carte compact flash (CF).

Lorsque la CR1000 reçoit une demande au sujet de données qui sont stockées sur une carte CF, la CR1000 ne regarde les données sur la CF qu'à condition que ce soit des données anciennes, ou si la donnée n'est pas disponible sur la RAM interne.

Dans l'éditeur CRBASIC, l'instruction DataTable permet de donner une taille aux tableaux de données ou à l'aire de mémoire tampon. Pour stocker un tableau sur la carte CF il faut ajouter à la déclaration du tableau, l'instruction **CardOut**.

2.1.1 SRAM interne

La SRAM interne est utilisée uniquement en tant que mémoire de stockage pour les tableaux de données, ou encore en tant que mémoire tampon lorsque les données sont par la suite envoyées vers une carte compact flash (CF). Le nombre maximum de tableaux de données qui peut être crée est de 30. La mémoire interne est sauvegardée par pile. Les données restent en mémoire lorsque la CR1000 n'est plus alimentée. Les données de la SRAM sont effacées lorsque un programme différent est mis en mémoire et activé.

Il y a 1 Mbyte de SRAM. Une partie de cette mémoire est utilisée pour le système d'exploitation et pour stocker le programme. Le reste est disponible afin de stocker les données. Lorsqu'un nouveau programme est compilé, la CR1000 vérifie qu'il y a assez d'espace dans la SRAM pour les tableaux ; si un programme demande plus de place que ce qui est disponible, il ne sera pas mis en fonctionnement.

2.1.2 CFM100 ou NL115

Le CFM100 et le NL115 sont des modules optionnels qui se connectent sur le port périphérique « peripheral port » de la CR1000, et qui dispose d'un slot pour cartes Compact Flash de type I ou II afin de permettre l'extension de mémoire de la CR1000. Un programme peut envoyer au maximum 30 tableaux sur une carte CF.

NOTE Pour incérer ou retirer le module CFM100 ou NL115, il faut couper l'alimentation de la CR1000.

Lorsqu'un tableau de données est envoyé vers la carte CF, le tableau de données de la SRAM est utilisé en tant que mémoire tampon (buffer) avant de transférer les données à la carte. Lorsque la carte est présente, le tableau d'état donnera des informations sur la taille du tableau présent sur la carte, pour le champ « table size ». Si la carte est absente ce sera la taille du tableau en SRAM qui sera indiqué.

Lorsqu'un nouveau programme est compilé et que ce programme envoie des données vers la carte CF, la CR1000 vérifie si la carte est présente et si la carte contient assez d'espace pour les tableaux de données. S'il y a assez de place pour les tableaux de données, la taille qui leur est nécessaire sera allouée sur la carte et la CR1000 commencera à y envoyer des données. S'il n'y a pas de carte présente ou si la place disponible est insuffisante, la CR1000 préviendra que la carte CF ne sera pas sollicitée et que le programme activé enregistrera les données sur la SRAM uniquement. Lorsqu'une carte avec assez d'espace disponible sera mise en place, la CR1000 créera les tableaux de données sur la carte et y stockera les données qui étaient accumulées en SRAM.

Les données stockées sur la carte peuvent être rapatriées via un lien de communication effectué avec la CR1000, ou bien en retirant la carte et en la lisant directement sur un ordinateur. Etant donné le succès des cartes CF due à leur utilisation avec les appareils photo numériques, il existe de multiples adaptateurs pour relier les cartes CF à un ordinateur. Il existe aussi des adaptateurs passifs qui permettent d'insérer ces cartes à l'intérieur des slots PCMCIA présents sur la plupart des ordinateurs portables. L'interface PCMCIA est beaucoup plus rapide que le lien de communication via la CR1000. Si la quantité de données à transférer est importante, il sera sans doute plus rapide d'installer la carte CF sur un ordinateur, que de passer via le lien de communication avec la CR1000.

La CR1000 utilise le formatage en FAT ou FAT32, pour les cartes CF. Celles-ci peuvent être formatées à partir du PC ou bien à partir de la CR1000.

ATTENTION

Pour éviter de perdre des données, nous vous suggérons de collecter les données avant d'envoyer ou de modifier un programme sur la centrale de mesure. Quand un programme est envoyé à la CR1000 en utilisant le bouton « Send » (Envoi) de l'écran Collect de LoggerNet ou PC400, un ordre est envoyé avec le programme qui commande à la centrale de mesure d'effacer toutes les données de la carte CF de l'ancien programme.

2.2 Format de stockage interne

TABLEAU 2.2-1. Types de données pour le CR1000

Type de donnée	Taille	Etendue de valeur	Résolution
LONG	4 bytes	-2 147 483 648 à +2 147 483 647	1 bit (1)
IEEE4	4 bytes	1,8 E -38 à 1,7 E 38	24 bits (environ 7 chiffres)
FP2	2 bytes	-7999 à +7999	13 bits (environ 4 chiffres)
Boolean	4 bytes	True/False	
String	1 byte par caractère + 1 byte, 16 bytes minimum		

Les données sont immédiatement stockées au format binaire. Les variables et les calculs sont effectués de façon interne en format IEEE à 4 byte et virgule flottante, avec certaines opérations effectués en double précision. Il y a six types de formats utilisés pour stocker les données. Le type de format est choisi par l'instruction qui définit la sauvegarde de la donnée. Dans la centrale de mesure le temps est stocké en tant que secondes et nano secondes entières à partir de la première seconde suivant minuit au commencement de 1990. Alors que le format IEEE à 4 byte et virgule flottante est utilisé pour les variables et les calculs internes, le format FP2 est approprié pour la plupart des enregistrements de données. Le format à 2 byte et virgule flottante de Campbell Scientific permet d'avoir comme résolution, 3 ou 4 chiffres significatifs après la virgule, et nécessite deux fois moins de place en mémoire que le format IEEE à 4 byte et virgule flottante (2 bytes contre 4).

TABLEAU 2.2-2. Résolution et valeurs limite du format FP2		
Zéro	Magnitude Minimum	Magnitude Maximum
0,000	±0,001	±7999

La résolution du format FP2 est réduite à 3 chiffres significatifs lorsque le premier (le plus à gauche) et supérieur ou égal à 8 (voir tableau 2.2-2). De ce fait, il peut être nécessaire d'utiliser un enregistrement en mémoire finale au format IEEE à 4 byte et virgule flottante, ou bien il peut être nécessaire d'utiliser un offset, afin de garder la résolution désirée pour la mesure. Un exemple peut être donné pour la hauteur d'eau et si on souhaite avoir une résolution de 0,01 m; dans ce cas il faut que la hauteur d'eau soit inférieure à 80 m si on garde la résolution standard FP2 et voir des incréments de 0,01 m. Si le niveau d'eau est exprimé entre 50 et 90 m, la donnée pourrait alors être sauvegardée en haute résolution (IEEE à 4 byte et virgule flottante) ou en décalant la mesure d'un offset de –20 m (ce qui transfèrera l'étendue de mesure entre 30 et 70 m, gardant l'incrément de 0,01 m en format FP2).

TABLEAU 2.2-3. Position de la virgule avec le format FP2	
Valeur absolue	**Position de la virgule**
0 - 7,999	X,XXX
8 - 79,99	XX,XX
80 - 799,9	XXX,X
800 - 7999.	XXXX.

2.3 Collecte des données

Les données peuvent être transférées sur un ordinateur à l'aide d'un lien de communication et d'un des logiciels de Campbell Scientific (soit PC200W, PC400 ou LoggerNet) ou en transportant une carte CF provenant d'une CR1000, à un ordinateur.

Lorsque la CR1000 est utilisée sans ordinateur et sur le terrain, ou lorsqu'un grand nombre de données sont collectées sur la carte CF, la carte CF peut être transportée (elle contiendra les données) jusqu'à l'ordinateur.

Le format des données sur la carte CF est différent du format des données qui est créé lorsqu'on collecte les données via un lien de communication. Les fichiers de données se lisent directement sur la carte CF nécessitent généralement d'être convertis à un autre format afin de pouvoir être utilisés.

2.3.1 Au travers d'un lien de communication

Voir le manuel et l'aide du logiciel de communication que vous utilisez.

2.3.2 Au travers d'une carte CF

Lorsque la CR1000 est utilisée sur le terrain sans lien de communication, ou lorsqu'un grand nombre de données sont collectées sur la carte CF, la carte CF peut être transportée (elle contiendra les données) jusqu'à l'ordinateur. Voir le manuel du CFM100 ou du NL115 pour plus de détails à ce sujet.

2.3.2.1 Insérer une carte CF

Une carte insérée dans le slot pour carte CF du CFM100 ou NL115, si elle n'est pas utilisée par le programme actif de la CR1000 ou que la CR1000 n'a pas de programme actif, ne causera alors aucune réponse de la CR1000. Lorsqu'un nouveau programme est compilé et qu'il envoie des données vers la carte CF, la CR1000 vérifie si une carte est présente et si elle dispose d'assez d'espace pour les tableaux définis. Si la carte comprend assez de place, les tableaux seront alloués et la CR1000 commencera à les remplir de données.

Lorsqu'une carte est insérée à une CR1000 programmée afin d'y envoyer ses données, la CR1000 détectera sa présence.

Si la carte ne comprend pas de tableaux avec les même noms que ceux définis dans le programme, la CR1000 vérifiera s'il y a assez de place sur la carte pour les tableaux. Si la carte contient assez de place, la place nécessaire à ces tableaux sera allouée et la CR1000 commencera à les remplir de données.

Si la carte a déjà des tableaux définis dont les noms sont identiques à ceux créés par le programme, et que le programme a déjà fonctionné avec cette carte, la CR1000 vérifiera que l'en-tête du tableau correspond exactement à celui du tableau ; si l'horodatage de la dernière fois que la CR1000 a envoyé des données sur la carte, correspond à une date présente sur cette carte, alors la CR1000 ajoutera les données à la suite dans le tableau présent sur la carte. Cette option permet donc de retirer une carte afin de la lire, et d'avoir un fichier qui est complété lorsqu'on remet la carte en place. Si vous utilisez cette possibilité offerte, soyez prévenu du fait que si vous retirez la carte pendant plus de temps que la CR1000 ne dispose de place dans sa mémoire tampon, vous aurez un trou dans les données sur la carte par la suite !

S'il n'y a pas assez de place sur la carte ou que les noms de tableaux ne correspondent pas, la CR1000 allumera la LED jaune afin d'indiquer que la carte n'est pas utilisée.

2.3.2.2 Retirer une carte CF de la CR1000

Pour retirer une carte il faut appuyer sur le bouton du CFM1000. La CR1000 enregistrera sur la carte les données présentes en mémoire tampon, puis coupera l'alimentation de la carte. Le LED d'état passera à la couleur verte lorsque l'on peut retirer la carte en toute sécurité. On doit alors retirer la carte. La carte sera ré-activée au bout de 20 secondes si on ne l'a pas retirée.

La LED d'état de la carte CF qui est juste au dessus de la porte de la carte CF est de couleur rouge lorsque des données sont écrites sur la carte.

ATTENTION

Le fait de retirer la carte ou de lui couper son alimentation lorsqu'elle est active peut induire l'enregistrement de données erronées et peut aussi endommager la carte. Avant de couper l'alimentation de la CR1000 il faut appuyer sur le bouton du CFM100 ou du NL115 et attendre que la LED devienne verte.

Lorsque la carte CF est insérée dans l'ordinateur, les fichiers de données peuvent être copies vers un autre emplacement ou bien utilisés directement, comme on utiliserait n'importe quel fichier de données présent sur un disque dur. Dans la plupart des cas cependant, il sera nécessaire de convertir le format du fichier avant d'utiliser les données.

2.3.2.3 Convertir le fichier de données

La CR1000 stocke les données sur la carte CF avec le format TOB3. Le format binaire TOB3 est un format qui incorpore des fonctionnalités afin d'augmenter la fiabilité des cartes CF. Le format TOB3 permet de déterminer de façon précise l'heure d'enregistrement de chaque évènement, sans pour autant utiliser tout l'espace nécessaire à l'enregistrement complet du datage à chaque enregistrement.

Lorsque les fichiers au format TOB3 sont convertis en un autre format, le nombre d'enregistrements pourra être supérieur ou inférieur à celui demandé dans la déclaration du tableau de données. Il y a toujours au moins deux champs additionnels qui sont alloués. Lorsque le fichier est converti il y aura toujours des enregistrements supplémentaires si aucune erreur ne s'est produite. S'il se produit plus d'erreur que ce à quoi on s'attendait, il pourrait y avoir moins d'enregistrements dans le fichier que ce qui a été alloué.

L'utilitaire CardConvert qui est compris dans PC200W, PC400, et LoggerNet, permet de convertir les données de la carte en un autre format de données.

2.4 Format des données sur l'ordinateur

Le format de fichier stocké sur le disque peut être de l'ASCII ou du binaire, selon ce que l'on choisit comme type de format avec le logiciel (PC200W, PC400, LoggerNet) dans l'option de collecte de données.

2.4.1 Information sur l'en-tête

Chaque fichier de données stocké sur le disque a un en-tête ASCII qui est présente au début du fichier. L'en-tête donne des informations sur le format, la centrale de mesure et le programme utilisé pour la prise de mesure. La figure 2.4 .1 est un exemple d'en-tête où le texte présent est un nom générique pour ce qui est contenu dans l'en-tête. Les valeurs sont décrites à la suite de la figure 2.4.1.

```
"File Format","Station","Logger","Serial No.","OS Ver","DLD File","DLD Sig","Table Name"
"TIMESTAMP","RECORD","Field Name","Field Name","Field Name"
"TS","RN","Field Units","Field Units","Field Units"
"","","Processing","Processing","Processing"
"Field Data Type","Field Data Type","Field Data Type","Field Data Type","Field Data Type"
timestamp,record number,field data,field data,field data,
```

FIGURE 2.4.1 Information sur l'en-tête

Format du fichier (File Format)

C'est le format du fichier présent sur le disque. TOA5 est un format ASCII. TOB1 est un format binaire.

Nom de la station (Station Name)

Le nom de la station donné à la centrale de mesure à partir de laquelle les données ont été collectées.

Modèle de centrale de mesure (Logger Model)

Le modèle de centrale de mesure de mesure à partir de laquelle les données ont été collectées.

Numéro de série de la centrale de mesure (Logger Serial Number)

Le numéro de série de la centrale de mesure à partir de laquelle des données ont été collectées. C'est un numéro de série du CPU (unité centrale) de la CR1000.

Version du système d'exploitation (Operating System Version)

Version du système d'exploitation de la centrale à partir de laquelle ont été collectées les données.

Fichier DLD (DLD File)

Le nom du fichier qui était en fonctionnement lorsque les données ont été créées.

Signature DLDL (DLD Signature)

Signature du programme qui a créé les données.

Nom du tableau de données (Table Name)
Nom qui est donné au tableau de données.

Nom des colonnes (Field Name)
Nom de la colonne (du champ) du tableau de données. Ce nom est crée par la CR1000 en ajoutant un tiret bas (_) et trois caractères mnémoniques décrivant le traitement des données (soit _AVG, _TOT, etc. pour une moyenne ou un total – Average / Total).

Unité de mesure de la colonne (Field Units)

Les unités des colonnes (champs) du tableau de mesure. Les unités sont données dans le programme, lors de la déclaration des unités de mesure.

Traitement de sauvegarde de la colonne (Field Processing)

Cela indique le type de traitement de sauvegarde qui a été utilisé lorsque la colonne a été enregistrée.

Smp = Sample, échantillon

Max = Maximum

Min = Minimum

Avg = Average, moyenne

Type de données de la colonne (Field Data Type)

La ligne d'en-tête est uniquement au format binaire TOB1, et identifie le type de données pour chacun des champs dans le tableau de donnée.

UINT4 = entier à 4 octets, non signé

IEEE4 = chiffre à virgule flottante à 4 octets

Marqueur horaire (Time Stamp)

Ce champ est le repère d'heure et de date de l'enregistrement. Il indique l'heure à laquelle les données ont été enregistrées, grâce à celle de la centrale de mesure.

Numéro d'enregistrement (Record Number)

Ce champ est le numéro d'enregistrement de cet enregistrement. Ce numéro augmentera jusqu'à 2^{E32}, et re-commencera à 0. Le numéro d'enregistrement sera aussi ré-initialisé (mis à 0) si le tableau est effacé.

Donnée du champs (Field Data)

Ceci est la donnée pour chacun des champs de l'enregistrement.

2.4.2 Format de fichier ASCII TOA5

Ce qui suit est un exemple de fichier au format TOA5.

```
"TOA5","Fritz","CR1000","1079","CR1000.Std.1.0","CPU:TCTemp.CR1","51399","Temp"
"TIMESTAMP","RECORD","RefTemp_Avg","TC_Avg(1)","TC_Avg(2)","TC_Avg(3)","TC_Avg(4)"
"TS","RN","degC","DegC","DegC","DegC","DegC"
"","","Avg","Avg","Avg","Avg","Avg"
"2004-10-27 16:20:00",0,24.1,24.03,24.04,24.05,24.04
"2004-10-27 16:30:00",1,24.01,24.01,24.01,24,23.99,
```

Ci-dessous est un exemple de ce à quoi ressemblera un fichier de données lorsqu'il sera importé dans un tableur.

TOA5	Fritz	CR1000	1079	CR1000.Std1.0	CPU:TCTemp.CR1	51399	Temp
TIMESTAMP	RECORD	RefT_Avg	TC_Avg(1)	TC_Avg(2)	TC_Avg(3)	TC_Avg(4)	
TS	RN	degC	DegC	DegC	DegC	DegC	
		Avg	Avg	Avg	Avg	Avg	
10/27/2004 16:20	0	24.1	24.03	24.04	24.05	24.04	
10/27/2004 16:30	1	24.01	24.01	24.01	24	23.99	

2.4.3 Format de fichier binaire TOB1

Ceci est un exemple d'en-tête du fichier binaire.

```
"TOB1","Fritz'","CR1000","1079","CR1000.Std.1.0","CPU:TCTemp.CR1","51399","Temp"
"SECONDS","NANOSECONDS","RECORD","RefTemp_Avg","TC_Avg(1)","TC_Avg(2)","TC_Avg(3)","TC_Avg(4)"
"SECONDS","NANOSECONDS","RN","degC","DegC","DegC","DegC","DegC"
"","","","Avg","Avg","Avg","Avg","Avg"
"ULONG","ULONG","ULONG","FP2","FP2","FP2","FP2","FP2"
```

(Les données sont en binaire et ne sont pas directement compréhensibles)

2.4.4 Format de fichier binaire TOB3

Le format binaire TOB3, a le même type d'en-tête que les autres formats. Les données TOB3 sont stockées dans des trames (frames) de taille fixe, qui contiennent habituellement un nombre particulier d'enregistrements. La taille des trames dépend du nombre d'enregistrements. Les trames ont un marquage de la date qui est associé, permettant ainsi aux enregistrements d'avoir aussi un temps qui leur est associé. S'il y a une perte de données entre des enregistrements qui sont normalement à intervalle de temps périodique, et que celui-ci n'apparaît pas dans le sommaire de la trame (frame boundary), une marque de temps supplémentaire est écrite dans la trame, et le moment auquel elle intervient est notée dans le sommaire de la trame. Cette marque supplémentaire prend de la place, qui pourrait autrement être utilisée pour contenir des données.

Quand les fichiers au format TOB3 sont convertis dans un autre format, le nombre d'enregistrement peut être supérieur ou inférieur au nombre demandé dans la déclaration du tableau de données. Il y a toujours au moins deux trames supplémentaires de données allouées. Lorsque le fichier est converti, cela créera des enregistrements supplémentaires si aucune perte de donnée ne s'est produite. Si il se produit plus de perte de données que ce qui a été prévu, il pourra y avoir moins de données dans le fichier, que ce qui avait été alloué.

Chapitre 3. Détails sur les mesures de la CR1000

3.1 Séquence de mesures de tension analogique

La CR1000 mesure des tensions analogiques avec soit une option d'intégration et attente, soit une option conversion analogique numérique (A/N). La conversion A/N est effectuée par une technique d'approximation successive sur 13 bits, qui donne une résolution de la tension égale à une portion de l'intervalle d'étendue de mesure divisé en 7500. Si on utilise la technique de mesure la plus rapide avec l'intervalle de scrutation le plus court possible

(10 µs / 100 Hz), la CR1000 peut effectuer des mesures et les stocker à cette vitesse sur les 8 entrées différentielles ou bien sur 13 entrées unipolaires. Le taux de conversion maximum est de 2700 par secondes pour les mesures effectuées en unipolaires.

La temporisation pour les mesures de la CR1000, est contrôlée de façon précise. L'organisation des mesures est déterminée lors de la compilation, puis chargé en mémoire. L'organisation définit des interruptions qui dirigent la tâche de mesure. **Le fait d'utiliser deux instructions de mesure différentielle avec la même étendue de mesure prendra le même temps d'exécution à la centrale, que le fait d'utiliser deux répétitions dans l'instruction de mesure.** (Ce qui n'est pas le cas avec les CR10X, 21X, CR23X ni CR7, pour lesquelles il y a toujours un temps de configuration pour chaque instruction.)
Il y a quatre paramètres dans l'instruction de mesure, qui peuvent faire varier l'ordre et la temporisation des mesures. Il existe des options pour mesurer et corriger l'offset entre chaque mesure unipolaire (**MeasOfs**), pour inverser les voies positives et négatives d'une voie différentielle (**RevDiff**), pour donner un moment au signal afin qu'il se stabilise entre l'instant où on sélectionne la voie de mesure et celui où on prend la mesure (**SettelingTime**), pour donner la durée sur laquelle on souhaite intégrer la mesure (**Integ**), et pour inverser la polarité d'une tension d'excitation **(RevEx)**.

3.1.1 Etendue de mesure en tension

Etendues de mesure déterminées

La CR1000 dispose de 6 étendues de mesure déterminées et d'une étendue de mesure automatique (autorange). Le convertisseur A/N a une résolution de

1 parmi 2^{13} (8192) parties. Pour permettre une certaine capacité de dépassement d'étendue de mesure, la conversion A/N est appliquée sur une étendue de mesure supérieur d'environ 9% à celle définie par l'étendue de mesure de pleine échelle, ce qui résulte alors en une résolution de 1 parmi 7500 parties. Par exemple, sur l'étendue de mesure ±2500mV, la pleine échelle est de 5000mV [2500 - (-2500)] et la résolution est de deux tiers de millivolts soit 5000 / 0.667 = 7500. Plus l'étendue de mesure est petite, plus la résolution absolue et bonne. En général, on utilise l'étendue de mesure la plus proche de celle du capteur à sa pleine échelle. Si la tension est supérieure à l'étendue de mesure que l'on programme, la CR1000 indiquera que l'on est hors gamme et donnera comme résultat de mesure la valeur « NAN » (Not-A-Number).

Etendue de mesure automatique (AutoRange)

Pour des signaux qui ne varient pas de façon trop rapide, l'« Autorange » de la CR1000 lui permet de choisir automatiquement l'étendue de mesure de tension à utiliser. L' « Autorange » de la CR1000 effectue deux mesures. La première détermine l'étendue de mesure à utiliser. Elle est effectuée sur l'étendue de mesure ±2500mV avec l'intégration 250µs. La seconde mesure est effectuée sur l'étendue de mesure appropriée avec le type d'intégration spécifié dans le paramètre de l'instruction de mesure. Les deux mesures utilisent le temps de stabilisation (settling time) qui est programmé dans l'instruction de mesure. L' « AutoRange » optimise la résolution mais prend plus de temps à effectuer la mesure que si l'on utilisait une étendue de mesure déterminée, car deux mesures sont nécessaires.

Une mesure en « AutoRange » donnera comme résultat Not-A-Number si la tension est supérieure à celle mesurée lors de la première de ses deux mesures. Pour éviter les problèmes avec les tensions situées aux environs d'une étendue de mesure, l'« AutoRange » choisira l'étendue de mesure suivant si la première mesure effectuée est supérieure à 90% d'une étendue de mesure.

L'« AutoRange » est une bonne solution lorsqu'un signal dépasse de façon occasionnelle une valeur limite. Cela peut être le cas d'un thermocouple de type J qui serait la plupart du temps à moins de 476°C (étendue ±25mV) mais qui passerait de façon occasionnelle la barre des 500 °C (étendue ±250mV, voir Tableau 3.4-2). L'« AutoRange » ne doit pas être utilisé pour des signaux qui fluctuent rapidement, surtout s'ils passent rapidement à travers plusieurs étendues de mesure, car cela pourrait se produire entre les deux mesures effectuées par l'instruction.

Détection de circuit ouvert / Ramener vers le mode commun (Open Circuit Detect / Pull into Common Mode)

Une autre option que l'ont peut sélectionner avec les codes d'étendues de mesure, est de vérifier s'il n'y a pas de circuit ouvert et vérifier au même moment la possibilité de ramener le signal vers l'étendue de mode commun. Les codes d'étendue de mesure pour cette option se terminent par un « C ». Par exemple l'étendue de mesure ±25mV dont le code d'étendue de mesure est "mV25", deviendra "mV25C" lorsqu'on y ajoute la détection de circuit ouvert.

La détection de circuit ouvert fonctionne de telle sorte que la CR1000 connecte brièvement (pendant 50 microsecondes) la voie d'entrée à une source de 300mV à l'intérieur de la centrale. Une voie différentielle aura la borne positive reliée aux 300mV et la borne négative reliée à la masse. Une fois déconnectée, la voie d'entrée peut se stabiliser, puis on prendre la mesure de tension. Si le capteur est ouvert (les entrées ne sont pas branchées et sont « flottantes »), la tension restera flottante autour du potentiel à laquelle elle a été ramenée; une mesure effectuée sur l'étendue de mesure ±2,5mV, ±7,5mV, ±25 mV, ou ±250mV donnera un résultat hors gamme et affichera Not-A-Number (NAN). Si le capteur est bon et bien branché, le signal provenant du capteur ramènera la tension de l'entrée à l'intérieur d'une valeur correcte.

L'étendu de mesure de détection automatique de circuit ouvert (AutorangeC) ne fonctionnera que jusqu'à ±250mV et ne pourra pas être utilisée avec des tensions supérieures. Si on permettait l'utilisation de l'option AutorangeC sur l'étendue de mesure ±2500mV, il ne serait pas possible d'y détecter les circuits ouverts.

Le fait de connecter brièvement les entrées à une tension interne de la CR1000 permet aussi de tirer une tension différentielle flottante dans l'étendue de mode commun de la CR1000 (voir chapitre 3.2). Cette étendue de tension devrait être utilisée pour effectuer des mesures analogiques différentielles de thermocouples (TCDiff) et pour d'autres capteurs ayant des entrées différentielles flottantes en sortie (tels que les capteurs de rayonnement solaire).

La détection de circuit ouvert ne fonctionne pas avec les étendues de mesure ±2500mV ou ±5000mV. Les options mV2500C et mV5000C peuvent cependant être utilisées afin de ramener des tensions différentielles flottantes à l'intérieur du mode commun. Il n'y a aucune raison d'utiliser ces options d'étendue de mesure pour des mesures unipolaires.

3.1.2 Excitation inverse ou Entrée différentielle (Reversing Excitation or the Differential Input)

L'inversion de polarité de l'excitation ou de l'entrée différentielle, sont des techniques afin d'annuler les offsets de tension qui ne font pas partie du signal. Par exemple, s'il y a un offset de +5µv dans le circuit de mesure, un signal de 5mV sera mesuré en tant que 5,005mV. Une fois la mesure inversée le signal sera de –4,995mV. Le fait de soustraire la seconde mesure à la première, et de diviser le tout par deux, donnera alors la mesure correcte à savoir : 5,005-(-4,995)=10, 10/2=5. La plupart des offsets sont des effets de thermocouple créés par des gradients de température dans les circuits de mesure ou les câblages.

Le fait d'inverser la polarité d'excitation annule les tensions d'offset dans le capteur, les câbles et le circuit de mesure. Une mesure est effectuée avec la tension d'excitation programmée, et une seconde mesure est effectuée avec la polarité inversée. La durée de l'excitation est strictement la même pour chaque polarité afin de s'assurer que les capteurs ioniques ne se polarisent pas au cours de mesures répétées.

L'inversion des entrées d'une mesure différentielle annules les offsets du circuit de mesure de la CR1000 et améliore la réjection en mode commun. Une mesure est effectuée avec l'entrée « High » par rapport à l'entrée « Low », et inversement.

3.1.3 Mesure de l'offset sur une mesure unipolaire

L'offset de mesure unipolaire est la tension d'offset qu'il y a sur une voie d'entrée unipolaire. On le mesure en connectant de façon interne l'entrée à la masse et en mesurant la tension qui en résulte. Lorsqu'on effectue une mesure unipolaire, cet offset est corrigé grâce à l'étalonnage. L'offset peut être mesuré automatiquement et en tâche de fond, ou bien mesuré en tant que partie intégrante de la séquence de mesure et à chaque fois que la mesure est effectuée (ce qui ajoutera du temps à l'exécution du programme). Lorsque l'offset est mesuré dans la séquence de mesures, l'offset est mesuré une fois puis toutes les répétitions de l'instruction sont effectuées.

Le paramètre **MeasOfs** de l'instruction qui effectue la mesure de tension unipolaire est utilisé afin de forcer la mesure de l'offset. Dans la plupart des applications l'étalonnage en tâche de fond sera adéquat pour la mesure. Il est possible de gagner de la précision supplémentaire en effectuant la mesure d'offset avec chaque instruction de mesure, lorsque l'offset change rapidement, comme cela peut être le cas lorsque la CR1000 est en train de subir des changements de température.

3.1.4 Stabilisation (SettlingTime)

Lorsque la CR1000 se connecte à une nouvelle voie de mesure ou qu'elle commute une excitation pour une mesure de pont, il y a un petit moment nécessaire pour que le signal atteigne sa valeur vraie. Le fait de donner un délai entre la configuration de la mesure (changer de voie, fixer la valeur de l'excitation) et la prise de mesure, permet au signal de se stabiliser à une valeur correcte. Les temps de stabilisation par défaut sont les temps minimum nécessaires à la CR1000 pour se stabiliser et d'être à l'intérieur de ses caractéristiques de précision. Il est préférable d'ajouter un délai lorsqu'on travaille avec des capteurs à forte résistance ou à longueur de câble importante (qui augmente leur pouvoir capacitif). Le fait d'utiliser un délai de stabilisation plus important augmentera le temps nécessaire à la mesure. Le paragraphe 3.3 donne plus de détails sur la façon de déterminer un temps de stabilisation adéquat.

Lorsque la CR1000 inverse l'entrée différentielle ou la polarité de l'excitation, elle laisse un délai de stabilisation identique avant la première mesure qu'après l'inversion. De ce fait il y a deux délais par voie d'utilisés dès qu'on utilise les instructions **RevDiff** ou **RevEx**. Si on sélectionne à la fois **RevDiff** et **RevEx**, il y aura quatre segments de mesure, une excitation positive et négative avec les entrées d'une certaine façon, puis excitation positive et négative avec les entrées inversées. La CR1000 se connecte à la voie :

Fixe la valeur de l'excitation, effectue un délai, **effectue la mesure**,
Inverse l'excitation, effectue un délai, **effectue la mesure,**
Inverse l'excitation, inverse l'entrée, effectue un délai, **effectue la mesure,**
Inverse l'excitation, effectue un délai, **effectue le mesure,**

De ce fait il y a quatre délais par voie mesurée. La CR1000 calcule ensuite la mesure effectuée par segments et ne retourne qu'une seule valeur pour la mesure effectuée.

3.1.5 Intégration

L'intégration est utilisée afin de réduire le bruit inclus à une mesure. La CR1000 peut utiliser une combinaison d'intégration analogique et numérique.

Avec l'intégration analogique, le signal d'entrée est intégré sur une période de temps précise. La valeur intégrée est gardée par le convertisseur A/N. Il existe trois possibilités de temps d'intégration qui sont de 20ms, 16,67ms et 250µs. Les temps de 20ms (1/50 second) et 16,667 ms (1/60 second) sont disponibles pour retirer les effets de bruit causes par les sources de CA à 50 ou 60 Hz.

Un temps d'intégration est spécifié à l'intérieur même de l'instruction de mesure. Un temps d'intégration de 250 sélectionnera l'intégration à 250µs, "_60 Hz" sélectionnera la réjection 60 Hz (16.667 ms), et "_50 Hz" sélectionnera la réjection 50 Hz (20 ms).

Les étendues de mesure de ±2500 et ±5000mV ne disposent pas des intégrations à 16,667 et 20ms. Sur ces étendues de mesure , les options "_50 Hz" et "_60 Hz" effectuent l'intégrations de rejet du bruit du 50 ou 60 Hz en moyennant deux mesures intégrées sur 250µs ,qui sont espacées d'exactement une demi période du cycle 50 ou 60 Hz. La moyenne de ces mesures est stockée en tant que résultat unique de cette mesure.

L'intégration spécifiée dans l'instruction de mesure est utilisée pour chaque segment de mesure. De ce fait si l'inversion de l'entrée différentielle ou l'inversion de l'excitation est spécifiée, il y aura deux intégrations par voie ; si les deux types d'inversions sont spécifiés, alors il y aura quatre intégrations.

3.2 Mesures de tension unipolaire et différentielles

Une mesure de tension unipolaire est effectuée sur une seule voie et par rapport à la masse. Une mesure différentielle est la différence de tension mesurée entre deux entrées.

NOTE

Il existe deux jeux de numéros de voies analogiques. Les voies différentielles (de 1 à 8) ont deux entrées : haut *high (H)* et bas *low (L)*. L'une des voies H ou L d'une entrée différentielle peut être utilisée pour effectuer une mesure unipolaire. Les voies unipolaires sont numérotées entre 1 et 16.

Du fait que les mesures unipolaires sont prises par référence à la masse de la CR1000, une quelconque différence de potentiel entre la masse du capteur et celle de la centrale de mesure induira une erreur de mesure. Par exemple si on mesure la jonction d'un thermocouple cuivre – constantan qui serait utilisé pour mesurer la température d'un sol, qu'elle n'est pas isolée et que le potentiel de la masse est supérieur de 1mV au niveau du capteur par rapport à la masse de la centrale de mesure, la tension mesurée sera supérieure de 1mV à la tension réelle, ce qui représente une sur-estimation d'environ 25°C. Un autre cas où une différence de potentiel de masse crée un souci, est quand un circuit de conditionnement externe est alimenté par la même source que la CR1000. Bien que ces appareils soient référencés à la même masse, les différences de consommation de courant et les résistances de câble peuvent conduire à des différences de potentiel de masse pour les deux instruments. Pour cette raison il sera préférable d'effectuer une mesure différentielle en sortie du conditionneur externe. Les mesures différentielles DOIVENT être utilisées lorsqu'on sait que les entrées ont des potentiels différents de celui de la masse, comme c'est le cas pour les ponts complets.

Etendue de mode commun

Pour faire des mesures différentielles, les entrées doivent être à l'intérieur du mode commun de la CR1000, soit ±5V. L'étendue de mode commun est l'étendue de tension, par rapport à la masse de la CR1000, dans laquelle les deux entrées d'une mesure différentielle doivent être comprises afin que la mesure différentielle puisse être effectuée. Par exemple si la tension du côté haut de la mesure est à 4V, et que le côté bas est à 3V par rapport à la masse, il n'y a pas de problème. Une mesure effectuée sur l'étendue ±5000mV donnera comme valeur 1000mV. Si par contre le côté haut de la mesure est de 5,8V et le côté bas est à 4,8V, la mesure ne pourra pas être effectuée car la tension du côté haut est supérieure aux ±5V de l'étendue de mode commun de la CR1000 (la centrale indiquera une valeur hors gamme et affichera Not-A-Number (NAN)).

Les capteurs qui ont une sortie flottante ou qui ne sont pas référencés à la masse via une connexion propre, peuvent nécessiter que l'on utilise l'option d'étendue de mesure qui tire le signal à l'intérieur de l'étendue de mode commun (paragraphe 3.1.1), ou bien nécessiter que l'on relie une des bornes de la voie différentielle qui soit connectée à la masse afin de s'assurer que le signal reste à l'intérieur de l'étendue de mode commun.

Des problèmes d'étendue de mesure de mode commun peuvent être rencontrés lorsque la CR1000 est utilisée pour mesurer le signal en sortie d'un circuit de conditionnement externe, s'il n'est pas relié correctement à la masse de la CR1000. Si on fonctionne avec une source de courant CA il n'est pas toujours bon de considérer que la mise à la terre de la prise CA est de bonne qualité. Si la CR1000 est utilisée afin de mesurer les signaux d'instruments de laboratoire (avec alimentation via CA et mise à la terre par la masse de la prise de courant), il est préférable de faire une liaison entre la CR1000 et le circuit externe. Même dans ces conditions les deux potentiels de masse ne seront peut être pas exactement identiques, c'est pourquoi il est préférable d'effectuer une mesure différentielle.

Une mesure différentielle dispose d'options d'inversion de polarité afin d'éliminer les offsets, comme décrit précédemment.

NOTE Des tensions au dessus de ±16V endommageront le circuit de la CR1000.

3.3 Temps de stabilisation du signal (Signal Settling Time)

Lorsqu'une entrée analogique est mise en lien avec le circuit de la CR1000, et avant la mesure de sa tension, une quantité de temps définie est nécessaire afin que le signal ne se stabilise à sa valeur correcte. La vitesse à laquelle le signal se stabilise est déterminée par la constante de stabilisation d'entrée, qui est une fonction dépendant de la résistance et de la capacité sur la voie en entrée.

La CR1000 effectue un délai après avoir commuté son circuit sur la voie programmée avant d'initialiser la mesure, afin de permettre à la voie en entrée de se stabiliser. Les délais par défaut dépendent de l'intégration demandée et l'étendue de mesure en tension. Le délai par défaut est 450µs lorsqu'on utilise l'intégration à 250µs, et de 3ms lorsqu'on utilise les options de réjection 50 ou 60 Hz. Cette durée de stabilisation est le minimum requis afin de permettre à la voie en entrée de se stabiliser et de répondre aux caractéristiques de résolution annoncées.

Le fait d'utiliser des longs câbles pour les capteurs peut ajouter un pouvoir capacitif du aux fils, et peut induire une augmentation de la durée de stabilisation du signal au point d'engendrer des erreurs. Il existe trois sources potentielles d'erreur qui doivent être stabilisées avant que la mesure ne soit effectuée :

1. Le signal doit croître et atteindre sa valeur correcte.
2. Un petit transitoire causé par la connexion du circuit électronique de la CR1000 à la voie analogique, doit se stabiliser.
3. Lorsqu'on effectue une mesure de pont de résistance en utilisant une voie d'excitation commutée, un transitoire plus important lié à la mise en fonctionnement de l'excitation, doit se stabiliser.

3.3.1 Réduire les erreurs de stabilisation (Minimizing Settling Errors)

Lorsqu'il est nécessaire d'utiliser de longs câbles, les pratiques mentionnées ci-dessous doivent être utilisées afin de réduire les risques d'erreur de mesure due à la stabilisation de la mesure :

1. NE PAS UTILISER DE CONDUCTEURS ISOLÉS PAR DU PVC. Le PVC a un fort pouvoir diélectrique, qui étend la durée de stabilisation en entrée.
2. Lorsque c'est possible, il est préférable de faire passer les câbles d'excitation et les câbles de mesure dans des gaines séparées afin de minimiser les effets de transitoire.
3. Lorsque la vitesse de mesure n'est pas une priorité, un délai supplémentaire peut être utilisé pour s'assurer que le temps de stabilisation est suffisant. Ce délai de stabilisation peut être mesuré par la CR1000.

3.3.2 Mesure du temps de stabilization nécessaire

La CR1000 peut mesurer le temps nécessaire pour la stabilisation de la mesure d'une certaine configuration de câble + capteur. Cela est effectué en faisant un nombre de mesures avec différents temps de stabilisations. Lorsqu'on regarde la série de mesure on peut voir la stabilisation du signal du capteur.

L'exemple suivant démontre la mesure du temps de stabilisation pour une tension différentielle. Si vous n'êtes pas encore familiarisé avec la programmation des CR1000, il est préférable de lire le chapitre 4 avant d'essayer de comprendre ce que fait le programme montré en exemple.

Les séries de mesures sur le capteur sont effectuées avec des mesures séparées pour chaque temps de stabilisation.

Avant que le programme destiné à mesurer le temps de stabilisation du signal ne soit mis en fonctionnement, le capteur avec la longueur de câble qui sera mise en place sur site doit être mis en condition de stabilité. Si la grandeur mesurée varie rapidement il sera difficile de séparer le temps de stabilisation et le changement de la valeur mesurée. Le programme qui suit mesure le temps de stabilisation sur le pont complet pour un transducteur de pression.

```
'CR1000 Series Datalogger
'Programme pour mesurer le temps de stabilisation du signal pour un capteur sur une voie diff.

Public PT(20)      'Variable qui contient la mesure

DataTable (Settle,True,100)
    Sample (20,PT(),IEEE4)
EndTable

BeginProg
    Scan (1,Sec,3,0)
        BrFull (PT(1),1,mV7_5,1,Vx1,1,2500,True ,True ,100,250,1.0,0)
        BrFull (PT(2),1,mV7_5,1,Vx1,1,2500,True ,True ,200,250,1.0,0)
        BrFull (PT(3),1,mV7_5,1,Vx1,1,2500,True ,True ,300,250,1.0,0)
        BrFull (PT(4),1,mV7_5,1,Vx1,1,2500,True ,True ,400,250,1.0,0)
        BrFull (PT(5),1,mV7_5,1,Vx1,1,2500,True ,True ,500,250,1.0,0)
        BrFull (PT(6),1,mV7_5,1,Vx1,1,2500,True ,True ,600,250,1.0,0)
        BrFull (PT(7),1,mV7_5,1,Vx1,1,2500,True ,True ,700,250,1.0,0)
        BrFull (PT(8),1,mV7_5,1,Vx1,1,2500,True ,True ,800,250,1.0,0)
        BrFull (PT(9),1,mV7_5,1,Vx1,1,2500,True ,True ,900,250,1.0,0)
        BrFull (PT(10),1,mV7_5,1,Vx1,1,2500,True ,True ,1000,250,1.0,0)
        BrFull (PT(11),1,mV7_5,1,Vx1,1,2500,True ,True ,1100,250,1.0,0)
        BrFull (PT(12),1,mV7_5,1,Vx1,1,2500,True ,True ,1200,250,1.0,0)
        BrFull (PT(13),1,mV7_5,1,Vx1,1,2500,True ,True ,1300,250,1.0,0)
        BrFull (PT(14),1,mV7_5,1,Vx1,1,2500,True ,True ,1400,250,1.0,0)
        BrFull (PT(15),1,mV7_5,1,Vx1,1,2500,True ,True ,1500,250,1.0,0)
        BrFull (PT(16),1,mV7_5,1,Vx1,1,2500,True ,True ,1600,250,1.0,0)
        BrFull (PT(17),1,mV7_5,1,Vx1,1,2500,True ,True ,1700,250,1.0,0)
        BrFull (PT(18),1,mV7_5,1,Vx1,1,2500,True ,True ,1800,250,1.0,0)
        BrFull (PT(19),1,mV7_5,1,Vx1,1,2500,True ,True ,1900,250,1.0,0)
        BrFull (PT(20),1,mV7_5,1,Vx1,1,2500,True ,True ,2000,250,1.0,0)
        CallTable Settle
    NextScan
EndProg
```

Le programme a été utilisé avec un capteur de pression Druck ayant un câble de 61m de long. Les six premières mesures sont indiquées dans le tableau 3.3-1. Les 20 valeurs sont mises en graphique sur la figure 3.3-1. Les lectures se stabilisent à partir de la 14ème mesure [PT(14)] ; on considèrera qu'une durée de stabilisation de 1400µs est adéquat.

TABLEAU 3.3-1. Six premières valeurs de données de temps de stabilisation

TOA5	Pepe'	CR1000	1079	CR1000.Std.01.00	CPU:Settlebridge.CR1	1455	Settle
TIMESTAMP	RECORD	PT(1)	PT(2)	PT(3)	PT(4)	PT(5)	PT(6)
TS	RN						
		Smp	Smp	Smp	Smp	Smp	Smp
1/3/2000 23:34	0	0.03638599	0.03901386	0.04022673	0.04042887	0.04103531	0.04123745
1/3/2000 23:34	1	0.03658813	0.03921601	0.04002459	0.04042887	0.04103531	0.0414396
1/3/2000 23:34	2	0.03638599	0.03941815	0.04002459	0.04063102	0.04042887	0.04123745
1/3/2000 23:34	3	0.03658813	0.03941815	0.03982244	0.04042887	0.04103531	0.04103531
1/3/2000 23:34	4	0.03679027	0.03921601	0.04022673	0.04063102	0.04063102	0.04083316

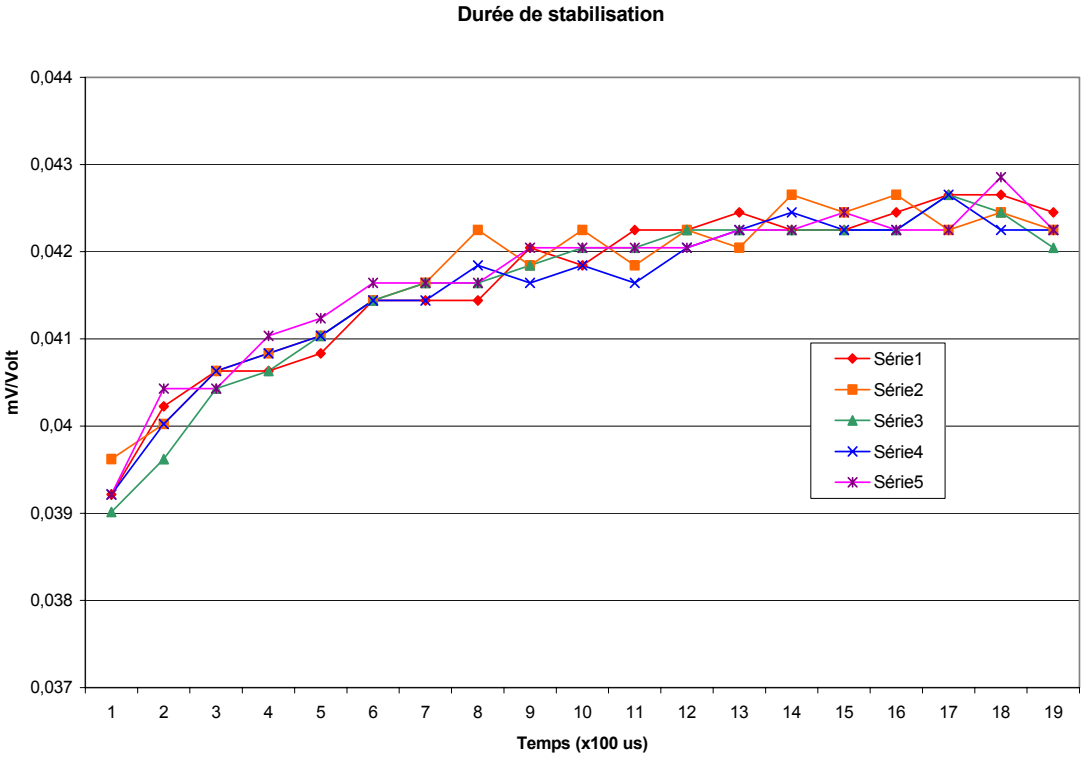


FIGURE 3.3-1. Temps de stabilisation pour un transducteur de pression

3.4 Mesures de Thermocouple

Un thermocouple est constitué de deux câbles, chacun d'un métal ou alliage différent, reliés ensemble à chaque extrémité. Si les deux jonctions sont à des températures différentes, une tension proportionnelle à la différence de température est induite dans les câbles. Quand un thermocouple sert à mesurer des températures, les câbles sont soudés ou collés ensemble à l'extrémité de la mesure. La deuxième jonction, qui devient la jonction de référence, est située à l'endroit où l'appareil de mesure est connecté aux autres extrémités des câbles. (Avec les connecteurs à la même température, la dissimilitude chimique entre le câble de thermocouple et le connecteur n'induit aucune tension). Quand la température de la jonction de référence est connue, la température de la jonction de mesure peut être déterminée en mesurant la tension du thermocouple et en ajoutant la différence de température correspondant à la température de référence.

La CR1000 détermine les températures du thermocouple en suivant la séquence indiquée ci-après. Premièrement, la température de la jonction de référence est mesurée et stockée en °C. Si la température de jonction de référence est celle du bornier de la CR1000, celle-ci est mesurée par la thermistance interne grâce à l'instruction *PanelTemp*. L'instruction de mesure du thermocouple (*TCDiff* ou *TCSE*) prend la tension de la voie analogique. L'instruction de mesure du thermocouple calcule la valeur de la tension qu'aurait un thermocouple du type spécifié à la température de jonction de référence, si sa jonction de référence était à 0°C, et ajoute cette tension à la tension mesurée pour le thermocouple. La température de la jonction de mesure est ensuite calculée d'après une approximation polynomiale des étalonnages du *NIST (National Institute of Standards and Technology (NIST M))*.

3.4.1 Analyse des erreurs

L'erreur de mesure de la température d'un thermocouple est la somme des erreurs de mesure de la températures de la jonction de référence, du signal de sortie du thermocouple (dérive des standards publiés par le *NIST Monograph 175*), de la mesure de la tension du thermocouple, et de l'erreur polynomiale (différence entre le standard du *NIST* et les approximations polynomiales de la CR1000). L'analyse suivante est limitée aux erreurs d'étalonnage et de mesure, et n'inclut pas les erreurs d'installation ou d'adaptation du capteur à l'environnement de mesure.

Température de la jonction de référence du bornier de la CR1000

La thermistance mesurant la température du bornier est située juste au dessous du bornier, au centre des 2 lignes de connecteurs pour entrées analogiques.

La thermistance (BetaTherm 10K3A1A) a une interchangeabilité de 0,1°C pour des températures allant de 0 à 70°C. En condition de gel ou à forte température cette caractéristique est dégradée. La totalité des erreurs combinées (circuit de mesure de la résistance et équation de Steinhart et Hart utilisée) aboutiront à une précision de +/- 0,3°C sur l'étendue de mesure de -25 à +50°C, et +/- 0,8°C sur l'étendue de mesure de –55 à +80°C.

L'erreur dans la mesure de la température de référence est une combinaison entre la température de la thermistance et la différence de température entre la thermistance et la jonction réelle du thermocouple (la voie à laquelle le thermocouple est connecté). Le couvercle pour le bornier a été dessiné afin de limiter les gradients de température le long du bornier. Il isole le bornier des fluctuations rapides de température ; il est aussi un conducteur de chaleur afin de réduire les gradients de température. Quand on utilise la centrale sur le terrain, avec le couvercle sur le bornier, et dans un des coffrets de Campbell Scientific qui n'est pas exposé à des changements violents en température, l'erreur sera généralement inférieure à 0,2°C.

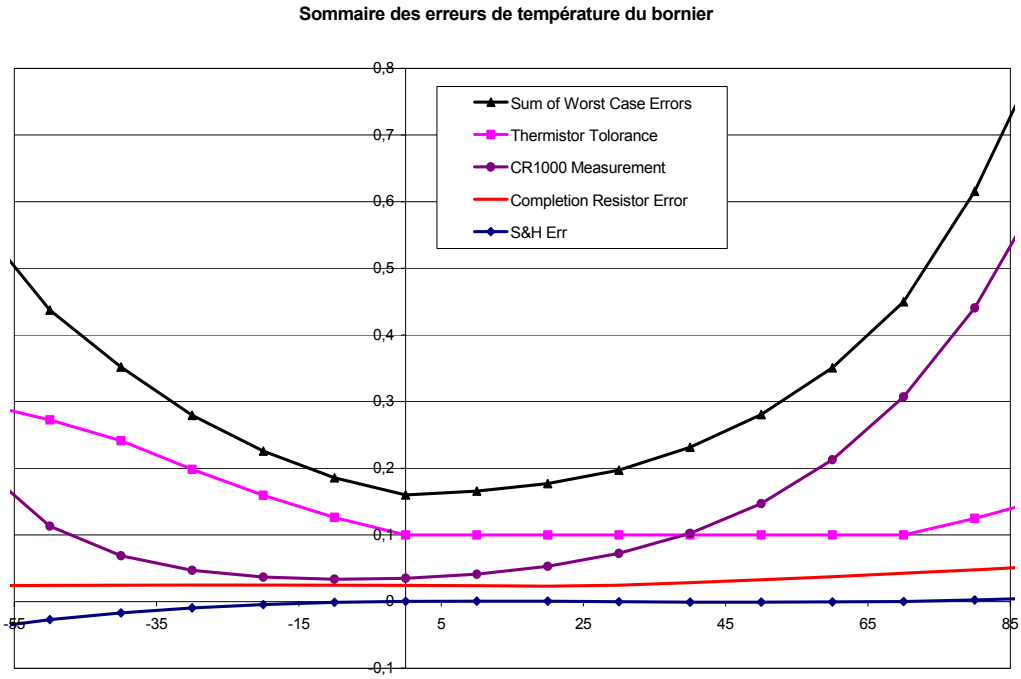


FIGURE 3.4-1 Erreur de température du bornier

Quand la CR1000 est sujet à de forts gradients de température ou des changements rapides de température, l'erreur de température de référence peut être bien plus importante. Par exemple, on place une CR1000 dans une chambre à température régulée. Des thermocouples placés sur les voies de côté et au milieu de chacun des borniers de mesure analogique prennent la température d'une barre d'aluminium isolée placée à l'extérieur de la chambre. La température de cette barre est aussi mesurée par une centrale de mesure. Les différences entre la température mesurée par un des thermocouples et la température réelle de la barre, sont dues à la différence de température entre la voie analogique sur laquelle le thermocouple est connecté et celle de la thermistance de référence (les chiffres sont corrigés vis à vis de l'erreur de la thermistance). La figure 3.4-2 montre les erreurs lorsque la chambre passe de –55 à +85°C en environ 15 minutes. La figure 3.4-3 montre les résultats du passage de 85 à 25°C. Au cours de ces changements rapides de température, la température de la thermistance du bornier aura tendance à être en retard par rapport à la température réelle de la chambre, car elle est placée à l'intérieur du bornier de la CR1000.

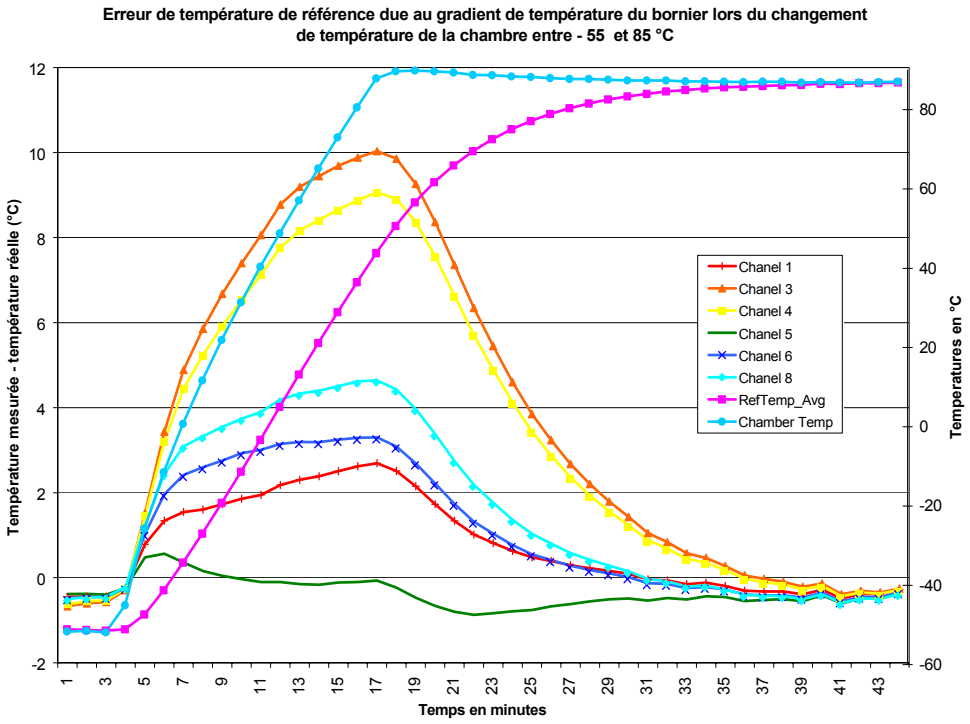


FIGURE 3.4-2. Gradients de température du bornier lors du changement de -55 à 80 °C

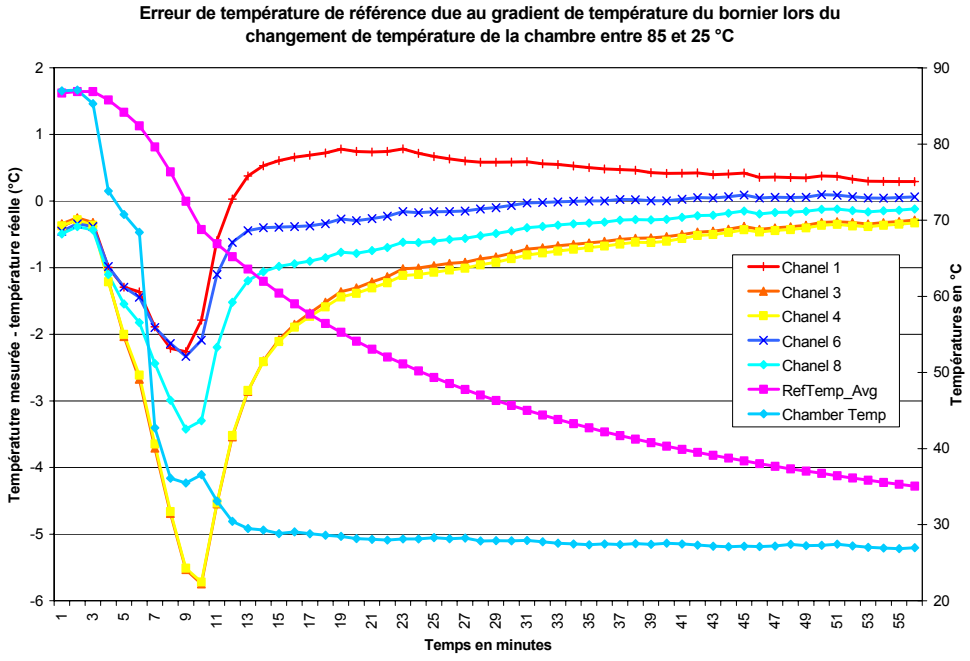


FIGURE 3.4-3. Gradients de température du bornier lors du changement de 80 to 25 °C

Limites d'erreur de thermocouple

La norme de référence qui donne la liste des tensions de sortie du thermocouple, en fonction de la température (jonction de référence à 0°C) est dans le Monograph 175 (1993) du *NIST*. L'institut national des normes des Etats-Unis a établi des limites d'erreur pour les câbles de thermocouple ;

celles-ci sont acceptées comme standard de l'industrie (ANSI MC 96.1, 1975). Le tableau 3.4-1 donne les limites d'erreur de l'ANSI pour les câbles de thermocouple standard ou de niveau spécial, pour les types de câble qui s'adaptent à la CR1000.

TABLEAU 3.4-1. Limite des erreurs de thermocouple (avec jonction de référence à 0°C)

Type de Thermocouple	Etendue de mesure en température (ºC)	Limite d'erreur (maximum)	
		Standard	Spéciale
T	-200 à 0	± 1.0ºC ou 1.5%	
	0 à 350	± 1.0ºC ou 0.75%	± 0.5ºC ou 0.4%
J	0 à 750	± 2.2ºC ou 0.75%	± 1.1ºC ou 0.4%
E	-200 à 0	± 1.7ºC ou 1.0%	
	0 à 900	± 1.7ºC ou 0.5%	± 1.0ºC ou 0.4%
K	-200 à 0	± 2.2ºC ou 2.0%	
	0 à 1250	± 2.2ºC ou 0.75%	± 1.1ºC ou 0.4%
R ou S	0 à 1450	± 1.5ºC ou 0.25%	± 0.6ºC ou 0.1%
B	800 à 1700	± 0.5%	Non établie

Quand les deux jonctions du thermocouple sont à la même température, aucune tension n'est produite (loi des métaux intermédiaires). Une des conséquences de cela est qu'un thermocouple ne peut pas avoir d'erreur d'offset. Toute dérive de la norme (en supposant que chaque câble est homogène, et qu'aucune jonction secondaire n'existe) est due à une dérive de la pente. A la lumière de cela, les limites fixes d'erreur de températures (par exemple +1,0°C pour un type T, en opposition à l'erreur de pente de 0,75% de la température) du tableau ci-avant, sont probablement supérieures à celles qu'on pourrait rencontrer dans l'étendue de mesure environnementale. En d'autres termes, la jonction de référence, à 0°C, est relativement proche de la température mesurée, donc l'erreur absolue (le produit de la différence de température et de l'erreur de pente) devrait être plus proche du pourcentage d'erreur que de l'erreur constante. Ainsi, comme l'erreur de étalonnage du thermocouple est une erreur de pente, la précision peut être améliorée quand la température de jonction de référence est proche de la température de mesure. Pour la même raison, les mesures différentielles de températures, sur un faible gradient de températures, peuvent être extrêmement précises.

Afin d'évaluer quantitativement l'erreur de thermocouple, quand la jonction de référence n'est pas fixée à 0°C, les limites d'erreur du coefficient de Seebeck (pente de tension du thermocouple vs. courbe de température) sont nécessaires pour la plupart des thermocouples. Sans cette information, une attitude raisonnable est d'appliquer le pourcentage d'erreur, en ajoutant peut-être 0,25%, à la différence de température mesurée par le thermocouple.

Précision de la mesure de tension du thermocouple

La précision de la mesure de tension différentielle de la CR1000 est spécifiée à +/- (0,075% de la tension mesurée + l'erreur d'offset en entrée qui est de 2 fois la résolution de base de l'étendue de mesure utilisée pour effectuer la mesure + 2µV). L'erreur d'offset en entrée réduit la résolution de base si la mesure différentielle est effectuée en utilisant l'option d'inversion de l'entrée différentielle.

Pour avoir une résolution optimum on utilise l'étendue de mesure +/-20mV à l'exception des fortes températures (voir Tableau 3.4-2). L'erreur d'offset en entrée est supérieure à l'erreur de mesure de tension pour les mesures environnementales. Une différence de température entre 45 et 64°C entre une mesure et la jonction de référence est nécessaire pour qu'un thermocouple fournisse 2,67mV en sortie ; à cette tension 0,075% de la lecture équivaut à 2µV. Si par exemple on mesure un thermocouple de type T pour mesurer une température de 45°C avec une température de référence à 25°C. La tension fournie par le thermocouple sera de 830,7µV. A 45°C un thermocouple de type T fournit 42,4µV par °C. L'erreur de pente possible dans la mesure est de 0,00075 * 830,7µV = 0,62µV ou 0,014 °C (0,62 / 42,4). La résolution de base sur l'étendue de mesure +/-20mV est de 0 ,67µV ou 0,01°C. L'erreur d'offset de 2µV représente 0,047°C. Ainsi l'erreur possible due à la mesure de tension est de 0,081°C sur une voie de mesure différentielle et sans inversion de tension en entrée, ou 0,024°C si on effectue une mesure inversée en entrée. Le fait d'utiliser l'inversion de tension en entrée afin d'augmenter la précision de la mesure vient ici d'être démontrée.

L'erreur de température due au manque de précision de la mesure de la tension du thermocouple est pire à chaque extrémité des températures, particulièrement lorsque la température et le type de thermocouple nécessitent l'utilisation de l'étendue de mesure de 200mV. Par exemple, supposez que des thermocouples de type K (chrome-alumel) soient utilisés pour mesurer des températures de 1300°C. La sortie TC est de l'ordre de 52mV, demandant une étendue de mesure en entrée de +/- 200mV. A 1300°C le thermocouple fournit 34,9µV/°C. L'erreur possible sur la mesure de tension est de 0,0075 * 52mV = 39µV soit 1,12°C (39/34,9). La caractéristique de précision de base sur l'étendue de mesure de 200mV est de 6,67µV soit 0,19°C. L'erreur due à la mesure de tension est donc de 1,56°C sur une mesure différentielle sans inversion en entrée, contre 1,31°C avec inversion en entrée.

TABLEAU 3.4-2. Etendues de mesure en tension et résolution maximum pour la mesure de thermocouple

Type de TC & ét. de mes. T° (°C)	Etendue de T° pour ±2,5mV d'ét. de mes.	Etendue de T° pour ±7,5mV d'ét. de mes.	Etendue de T° pour ±25mV d'ét. de mes.	Etendue de T° pour ±250mV d'ét. de mes.
T De -270 à 400	De -45 à 75	De -270 à 180	De -270 à 400	Non utilisé
E De -270 à 1000	De -20 à 60	De -120 à 130	De -270 à 365	>365
K De -270 à 1372	De -40 à 80	De -270 à 200	De -270 à 620	>620
J De -210 à 1200	De -30 à 65	De -145 à 155	De -210 à 475	>475
B De 0 à 1820	De 0 à 710	De 0 à 1265	De 0 à 1820	Non utilisé
R De -50 à 1768	De -50 à 320	De -50 à 770	De -50 to 1768	Non utilisé
S De -50 à 1768	De -50 à 330	De -50 à 820	De -50 à 1768	Non utilisé
N De -270 à 1300	De -80 à 105	De -270 à 260	De -270 à 725	>725

Lorsque la température de jonction du thermocouple est en contact électrique avec l'objet mesuré (ou qu'il a la possibilité d'être en contact), il est préférable d'effectuer une mesure différentielle.

Bruit sur les mesures de tension

Le bruit typique en entrée sur l'étendue de mesure +/-25mV et pour une mesure différentielle avec l'intégration de 250µs et l'inversion de mesure de tension en entrée, est de 1,2µV RMS. Sur un thermocouple de type T (environ 40µV/°C) cela donne 0,03°C. A noter que cette valeur est RMS (racine carrée – *Root Mean Square*) ; et que certaines valeurs seront plus importantes que cela. Avec l'intégration à 16,66µs le bruit est réduit à un niveau de 0,33µV RMS.

Polynômes de thermocouple - conversion de tension en température

Le Monograph 175 du *NIST* donne des polynômes d'ordres élevés pour calculer la tension de sortie d'un type de thermocouple donné, sur une large plage de température. Afin d'accélérer le traitement tout en se servant des capacités mathématiques et de stockage de la CR1000, 4 polynômes d'ordre 6, sont utilisés pour convertir les tensions en températures, sur l'étendue de mesures couverte par chaque type de thermocouple. Le tableau 3.4-3 donne les limites d'erreur pour les fonctions de linéarisation de ces polynômes.

Tableau 3.4-3 Limites d'erreur de la linéarisation de la sortie du thermocouple dans la CR1000 (par rapport au standard du NIST)

Type de TC	Plage °C	Limites d'erreur °C
T	-270 à 400	
	-270 à -200	+18 @ -270
	-200 à -100	± 0,08
	-100 à 400	± 0,001
	100 à 400	± 0,015
E	-240 à 1000	
	-240 à -130	± 0,40
	-130 à 200	± 0,005
	200 à 1000	± 0,02
K	-50 à 1372	
	-50 à 950	± 0,01
	950 à 1372	± 0,04
J	-150 à 760	± 0,008
	-100 à 300	± 0,002

Compensation de la jonction de référence - conversion de la température en tension

Les polynômes utilisés pour la compensation de la jonction de référence (conversion de la température de référence en une tension de sortie équivalente au TC) ne couvrent pas la totalité de l'étendue de mesure des thermocouples. Des erreurs conséquentes pourront se produire si la température de la jonction de référence est hors de l'étendue de mesure étalonnée. Les étendues de mesure couvertes par ces étalonnages incluent l'étendue de mesure ambiante de fonctionnement de la CR1000 et il n'y a donc aucun problème quand la CR1000 sert de jonction de référence. Des boîtes externes de jonction de référence, doivent par conséquent aussi se trouver dans ces étendues de mesure de température. Les mesures de différence de températures effectuées hors des étendues de mesure des températures de référence, doivent s'effectuer en prenant la température réelle référencée à une jonction dans l'étendue de mesure des températures de référence, puis en faisant une soustraction. Le tableau 3.4-3 donne les étendues de mesure de température de référence couvertes, ainsi que les limites d'erreur de linéarisation à l'intérieur de ces étendues de mesure.

Deux sources d'erreur apparaissent quand la température de référence est hors de l'étendue de mesure. L'erreur la plus significative réside dans la tension de compensation calculée, mais une erreur est aussi créée lors du calcul de la différence de température en sortie du thermocouple. Supposez par exemple que la température de référence pour une mesure avec un thermocouple de type T est 300°C. La tension de compensation calculée par la CR1000 correspond à une température de 272,6°C, donc avec une erreur de -27,4°C. Le thermocouple de type T, avec une jonction de mesure à 290°C et une jonction de référence à 300°C aurait une tension de sortie de -578,7µV. Avec la température référence de 272,6°C, la CR1000 calcule une différence de température de -10,2°C, donc avec une erreur de -0,2°C. La température calculée par la CR1000 serait de 262,4°C, soit 27,6°C de moins que la réalité.

Tableau 3.4-4 Etendue de mesure de compensation de la température de référence et erreur de linéarisation (par rapport au standard du NIST)

Type TC	Etendue de mesure en °C	Limites d'erreur en°C
T	-100 à 100	± 0,001
E	-150 à 206	± 0,005
K	-50 à 100	± 0,01
J	-150 à 296	± 0,005

Résumé des erreurs

L'amplitude des erreurs décrites dans les chapitres précédentes illustre bien que les sources majeures d'erreur, pour une mesure de température du thermocouple, sont dues le plus souvent aux limites d'erreur du câble du thermocouple et aux températures de référence déterminées par la thermistance située sous le bornier. Les erreurs du thermocouple et des polynômes de température de référence sont extrêmement faibles et les erreurs de mesures de tension sont négligeables.

Pour illustrer l'amplitude relative de ces erreurs, dans une étendue de mesure environnementale, nous prendrons la pire des situations, où toutes les erreurs sont maximales et s'additionnent. Une température de 45°C est mesurée par un thermocouple de type T (cuivre constantan), sur une étendue de mesure de ±2,5mV. La précision nominale de cette étendue de mesure est 1 µV (0,01% de 10mV), ce qui à 45°C, modifie la température de 0,012°C. La température de référence de l'appareil (*RTD*) est de 20°C et indique 20°C, mais la voie du bornier à laquelle le thermocouple est connecté est à 0,05 °C de moins que la RTD.

TABLEAU 3.4-5. Exemple d'erreurs de température de thermocouple

Source	**Error: ºC : % d'erreur totale**			
	Mes. Diff. Simple avec intégration 250 µs		**Mes. Diff. inversée avec intégration de réjection 50/60 Hz**	
	Erreur ANSI du TC (1ºC)	**Erreur de pente du TC de 1%**	**Erreur ANSI du TC (1ºC)**	**Erreur de pente du TC de 1%**
Temp. Référence	0,15º:11,7%	0,15º:31,1%	0,15º:12,4%	0,15º:36,4%
Sortie TC -	1,0º:78%	0,2º:41,5%	1,0º:82,5%	0,2º:48,6%
Mesure de tension	0,10º:7,8%	0,10º:20,8%	0,05º:4,1%	0,05º:12,1%
Bruit	0,03º:2,3%	0,03º:6,2%	0,01º:0,8%	0,01º:2,4%
Linéarisation de la réf.	0,001º:0,1%	0,001º:0,2%	0,001º:0,1%	0,001º:0,25%
Linéarisation en sortie	0,001º:0,1%	0,001º:0,2%	0,001º:0,1%	0,001º:0,25%
Erreurs totales	1,282º:100%	0,482º:100%	1,212º:100%	0,412º:100%

3.4.2 Utilisation d'une jonction de référence externe ou d'une boîte de jonction

Une boîte de jonction externe est souvent utilisée pour faciliter les connexions et réduire les dépenses de câble de thermocouple quand les mesures de température doivent s'effectuer à une certaine distance de la CR1000. Dans la plupart des cas, il est préférable de prendre la boîte comme jonction de référence, auquel cas sa température est mesurée et utilisée comme référence pour les thermocouples ; des câbles de cuivre relient la boîte à la CR1000. Sinon, la boîte de jonction peut servir à relier les thermocouples utilisés pour faire des mesures, par l'intermédiaire de câble d'extension ; le bornier de la CR1000 sert alors de jonction de référence. Un câble d'extension de thermocouple a une étendue de mesure de température plus petite que le câble de thermocouple standard, mais a les mêmes limites d'erreur dans cette étendue de mesure. Il peut être nécessaire d'utiliser un câble d'extension au lieu d'une jonction de mesure externe, quand la température de la boîte de jonction est hors de l'étendue de mesure de la compensation de la jonction de référence fournie par la CR1000. Ce n'est à prendre en compte que lorsqu'on utilise des thermocouples de type K, pour lequel la limite supérieure de la compensation de la linéarisation de référence est de 100°C, et la limite supérieure de température du câble d'extension est de 200°C. Avec les autres types de thermocouples, l'étendue de mesure de la compensation de référence est égale ou supérieure à l'étendue de mesure du câble d'extension. Dans tous les cas, des erreurs peuvent apparaître si des gradients de température existent à l'intérieur de la boîte de jonction.

La figure 3.4-4 illustre une boîte de jonction type. Les borniers seront d'un métal différent du câble de thermocouple. Ainsi, si un gradient de température existe entre A et A', ou B et B', la boîte de jonction agit comme un autre thermocouple en série, introduisant une erreur dans la tension mesurée par la CR1000. Cette tension de dérive thermoélectrique est un facteur indépendant du fait que la boîte de jonction soit ou non utilisée comme référence. Elle peut être minimisée avec une grande conduction thermique et une petite distance entre les deux points. La meilleure solution dans le cas où le câble d'extension est connecté au câble du thermocouple, serait d'utiliser des connecteurs qui mettent en contact les deux câbles.

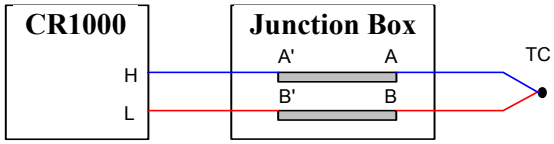


FIGURE 3.4-4. Diagramme de boîte de jonction

Une boîte de jonction de référence externe doit être construite pour que toute la surface du bornier soit le plus proche d'une même température. Ceci est nécessaire pour avoir une température de référence fiable, et pour éviter une tension de dérive thermoélectrique, qui peut être induite si les borniers sur lesquels les câbles du thermocouple sont connectés (points A et B de la figure 3.4-4), sont à des températures différentes. La boîte doit contenir des éléments à haute conductivité thermique, qui agiront rapidement pour enlever tout gradient thermique auquel la boîte serait sujette. Il n'est pas nécessaire de concevoir une boite à température constante. Il est souhaitable que la boîte réagisse lentement à de fortes variations externes de température.

Un abri contre les rayonnements, doit être mis en place quand une boîte de jonction est installée sur le terrain. Il faut également faire attention au gradient thermique qui pourrait être induit par conduction au travers des câbles entrants. La CR1000 peut servir à mesurer les gradients de température à l'intérieur de la boîte de jonction.

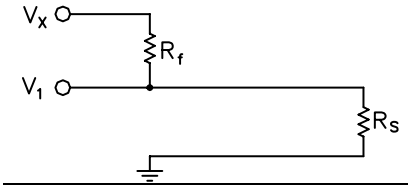
3.5 Mesures de résistance de pont

Six instructions de mesures de pont sont incluses dans le logiciel standard de la CR1000. La figure 3.5-1 montre les circuits qui pourraient généralement être mesurés par ces instructions. Sur les schémas, les résistances marquées R_s sont normalement les capteurs et celles marquées R_f sont des résistances fixes. Des circuits différents de ceux qui sont montrés peuvent être mesurés, à partir du moment où les types de mesure et d'excitation sont appropriés.

Toutes les instructions de mesures de pont ont l'option (**RevEx**) permettant d'effectuer les mesures avec l'excitation programmée, et un autre jeu de mesure avec la polarité d'excitation inversée. L'erreur dans les deux mesures, due à des tensions (fem) thermiques, peut alors être comptée dans le traitement de l'instruction de mesure. La voie d'excitation garde la tension ou l'intensité d'excitation jusqu'à ce que la conversion A/N soit effectuée. Lorsqu'il est nécessaire d'effectuer plus d'une mesure par capteur (demi pont 3 ou 4 fils, pont complet 6 fils), l'excitation est appliqué de façon séparée pour chaque mesure. Par exemple dans l'instruction de demi pont 4 fils, et lorsque l'excitation est inversée, la mesure différentielle de la chute de tension le long du capteur est effectuée avec l'excitation à chaque polarité, et l'excitation est ensuite de nouveau appliquée puis inversée pour la mesure de la chute de tension au travers de la résistance fixe.

Le calcul de la résistance réelle du capteur, qui est une des branches du pont résistif, nécessite généralement l'utilisation d'une ou deux instructions de traitement, en plus des instructions de mesure du pont. En plus des schémas typiques de ponts et de leur configuration, la figure 3.5-1 liste les calculs nécessaires afin de déterminer la résistivité d'une quelconque résistance pour autant que la valeur des autres résistances présentes dans le circuit du pont de mesure soit connues.

BrHalf (demi pont)



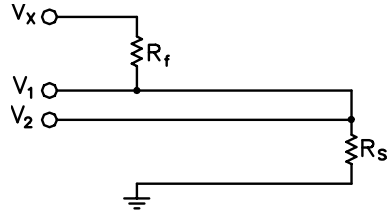
X = résultat avec mult =1, offset =0

$$X = \frac{V_1}{V_x} = \frac{R_s}{R_s + R_f}$$

$$R_s = R_f \frac{X}{1-X}$$

$$R_f = \frac{R_s(1-X)}{X}$$

BrHalf3W (demi pont 3 fils)



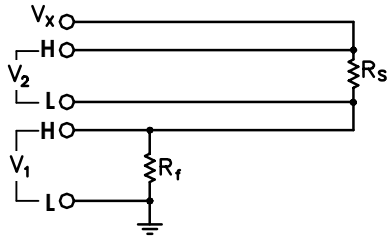
X = résultat avec mult =1, offset =0

$$X = \frac{2V_2 - V_1}{V_X - V_1} = \frac{R_s}{R_f}$$

$$R_s = R_f X$$

$$R_f = R_s / X$$

BrHalf4W (demi pont 4 fils)



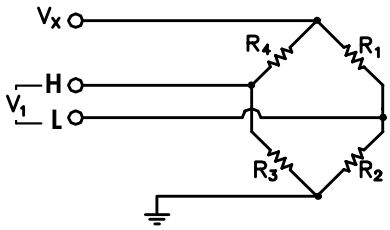
X = résultat avec mult =1, offset =0

$$X = \frac{V_2}{V_1} = \frac{R_s}{R_f}$$

$$R_s = R_f X$$

$$R_f = R_s / X$$

BrFull (pont complet)



X = résultat avec mult =1, offset =0

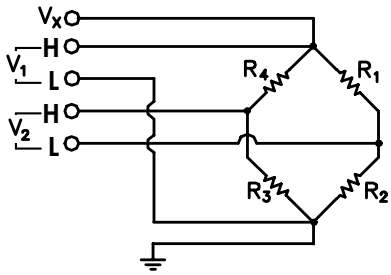
$$X = 1000\frac{V_1}{V_x} = 1000\left(\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2}\right)$$

$$X_1 = -X/1000 + R_3/(R_3 + R_4)$$

$$R_1 = \frac{R_2(1-X_1)}{X_1}$$

$$R_2 = \frac{R_1 X_1}{1-X_1}$$

BrFull6W (pont complet 6 fils)



X = résultat avec mult =1, offset =0

$$X = 1000\frac{V_2}{V_1} = 1000\left(\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2}\right)$$

$$X_2 = X/1000 + R_2/(R_1 + R_2)$$

$$R_3 = \frac{R_4 X_2}{1-X_2}$$

$$R_4 = \frac{R_3(1-X_2)}{X_2}$$

FIGURE 3.5-1. Circuits utilisés avec les instructions de mesure de pont

3.6 Mesure de résistance nécessitant une excitation CA

Certains capteurs résistifs nécessitent une excitation CA. Ce sont les capteurs à électrolyte, les blocs d'humidité de sol, les capteurs de conductivité de l'eau, et les grilles de lecture d'humidité. L'utilisation d'une excitation CC avec ces capteurs peut provoquer une polarisation, ce qui donnera des mesures erronées, et peut dévier l'étalonnage du capteur et/ou le mener à une dégradation rapide.

D'autres capteurs tels que les LVDTs (sans électronique intégrée) nécessitent une excitation CA car ils s'appuient sur un couplage inductif afin de fournir le signal. Une excitation CC ne produirait pas de signal en sortie.

Chacune des instructions de mesure de pont peuvent inverser l'excitation afin d'éviter la polarisation. La fréquence de l'excitation peut être déterminée par les temps de délai et d'intégration utilisés avec la mesure. La fréquence la plus élevée possible est de 5kHz, l'excitation est commutée puis inversée 100µs plus tard lorsque la première mesure est gardée puis l'excitation est désactivée après 100 autres µs lorsque la seconde mesure est gardée (c'est à dire inversion de l'excitation, 100µs de délai, pas d'intégration).

Influence de la boucle de masse sur les mesures

Quand on mesure un bloc d'humidité de sol ou la conductivité de l'eau, il est possible qu'une boucle de masse puisse affecter défavorablement la mesure. Cette boucle de masse apparaît parce que le sol et l'eau fournissent un chemin alternatif à l'excitation, pour revenir vers la masse de la CR1000.

La figure 3.6-2 en montre la représentation.

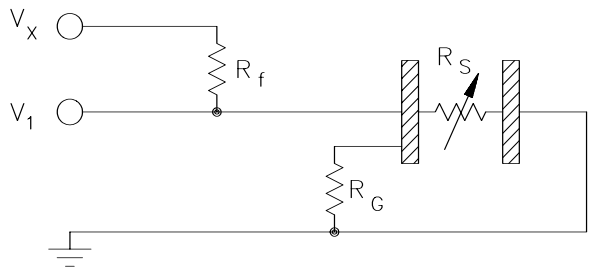


Figure 3.6-1 Modèle de capteur résistif avec boucle de masse

Dans la figure 3.6-1, V_X est la tension d'excitation, R_f est la résistance fixe, R_S est la résistance du capteur et R_G est la résistance entre l'électrode excitée et la prise de terre de la CR1000. Avec R_G dans le réseau, le signal mesuré est :

$$V1 = Vx \quad \frac{R_s}{(R_s+R_f) + R_sR_f/R_G} \qquad [3.6\text{-}1]$$

R_SR_f/R_G est la source d'erreur due au circuit de masse. Quand R_G est grand, l'équation est réduite idéalement. La géométrie des électrodes a un effet important sur l'amplitude de cette erreur. Le bloc de gypse Delmhorst utilisé dans la sonde 227 a deux électrodes cylindriques concentriques. L'électrode centrale sert à l'excitation. Comme elle est encerclée par l'électrode de masse, le chemin pour un circuit de masse par le sol est très réduit. Les blocs d'humidité qui sont constituées de deux électrodes plates et parallèles, sont particulièrement sensibles aux problèmes de circuit de masse. Des considérations similaires s'appliquent à la géométrie des électrodes pour les capteurs de conductivité de l'eau.

L'électrode de masse de la sonde de conductivité ou d'humidité de sol, et la prise de terre de la CR1000 forment une pile galvanique, la solution d'eau/terre agissant comme un électrolyte. Si du courant pouvait passer, l'oxydation ou la réduction résultante endommagerait rapidement l'électrode, comme si une excitation CC servait à effectuer la mesure. Les sondes de Campbell Scientific sont construites avec des condensateurs série dans les câbles, afin de bloquer ce courant CC. En plus de prévenir la détérioration du capteur, les condensateurs bloquent toute composante CC qui pourrait affecter la mesure.

3.7 Mesures de comptage d'impulsions

Plusieurs types de capteurs à signal de sortie en impulsion (comme les anémomètres ou les débit-mètre) sont calibrés en terme de fréquence (comptages par seconde). Pour ce type de mesures, la précision est liée directement à l'intervalle de temps entre lequel les impulsions sont accumulées. Les mesures variant en fonction de la fréquence, devraient être programmées avec l'instruction de mesure « PulseCount » mesure en fréquence. Si le nombre de comptage est d'un intérêt plus grand, l'instruction « PulseCount » doit être programmée afin de mesurer des comptages (par exemple le nombre de fois qu'une porte est ouverte ou qu'un auget a basculé sur un pluviomètre).

L'intervalle de la boucle de scrutation dans laquelle se trouve l'instruction PulseCount, n'est pas le seul facteur déterminant pour le calcul de la fréquence. Si le comptage d'impulsion est normalement lu à chaque intervalle de scrutation, et s'il se produit un retard du à un traitement de sauvegarde plus long que de coutume, alors le nombre d'impulsion ne sera pas lu jusqu'à ce que la synchronisation avec l'horloge soit restaurée. La CR1000 mesure alors le temps qui s'est écoulé depuis la dernière fois que le compteur a été lu, et en détermine la fréquence. S'il y a eu un dépassement de table (*overrun*), la fréquence fournie sera tout de même correcte.

La résolution de la mesure du comptage, est à plus ou moins un comptage. La résolution de la fréquence calculée dépend de l'intervalle de scrutation : résolution de la fréquence = 1/intervalle de scrutation (c'est à dire qu'un comptage d'impulsion avec une fréquence de scrutation d'une seconde, a une résolution de 1 Hz ; un intervalle de scrutation de 0,5 Hz a une résolution de 2Hz, et un intervalle de scrutation d'10ms a un résolution de 100Hz.). Les mesures résultantes vont tenir compte de la résolution. Par exemple si vous avez une scrutation chaque seconde, d'un signal à 2,5 Hz en entrée, vous aurez certains intervalles avec 2 comptages, et d'autres avec 3 comptages comme cela est explicité en figure 3.7-1. Si la mesure d'impulsion est moyennée, le résultat sera la valeur correcte.

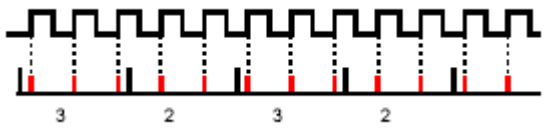


Figure 3.7-1 Nombre de comptage variant à l'intérieur d'un intervalle de scrutation.

La résolution devient de pire en pire lorsque l'intervalle de scrutation rétrécit, et que le signal a une fréquence plus importante. Par exemple, prenons un moteur fonctionnant sur le principe : Tours Par Minute (TPM) fournissant 30 impulsions par révolution. A 2000 TPM, le signal a une fréquence de 1000 Hz (2000 TPM x (1min / 60 sec) x 30 = 1000). Le multiplicateur utilisé afin de convertir la fréquence en TPM est de 2TPM/Hz (1TPM / (30impulsions / 60 sec) = 2). Avec une seconde d'intervalle de scrutation, la résolution est de 2TPM. Si l'intervalle de scrutation était de 10ms, la résolution serait de 200 TPM. Avec un tel intervalle de scrutation, si tout était parfait, à chaque intervalle il devrait y avoir 10 comptages. Si cependant il y avait une légère variation à l'intérieur de l'intervalle, cela pourrait donner 9 comptages dans un intervalle, et 11 comptages dans le suivant, avec un résultat variant entre 1800 et 2200 TPM !

3.8 Auto-étalonnage

La CR1000 effectue un auto-étalonnage des mesures de tension analogiques et des tensions d'excitation. Les étendues de gain et d'offset et les tensions d'excitation en sortie varieront en fonction de la température. L'auto étalonnage permet à la CR1000 de garder ses caractéristiques techniques tout au long de l'étendue de mesure en température.

Au lieu d'effectuer toutes les mesures nécessaires afin d'étalonner toutes les combinaisons d'étendues de mesure et d'intégrations disponibles sur la CR1000, l'étalonnage ne mesure que les combinaisons de l'étendues de mesure et d'intégrations qui sont présentes dans le programme. L'étalonnage peut se produire dans 3 modes différents.

1. Étalonnage au moment de la compilation. Ceci se produit avant que le programme ne soit exécuté, et ceci étalonne toutes les combinaisons d'étendue de mesure et d'intégration nécessaires. Pour l'intégration à 250µsec, de multiples mesures sont effectuées et moyennées afin de déterminer la valeur de gain à utiliser pour les mesures. Ce nombre est de 5 pour l'intégration à 250µV. Lorsque cet étalonnage est effectué, les valeurs présentes dans la table utilisée pour l'étalonnage, sont totalement ré-initialisées (c'est à dire qu'il n'y a pas de filtre d'utilisé).
2. Étalonnage en tâche de fond. Ceci s'effectue automatiquement en tâche de fond lorsque le programme de l'utilisateur est en fonctionnement. Il n'y a pas plusieurs mesures de moyennées, mais un filtre est appliqué aux nouvelles valeur de gain / offset obtenues. Le filtre est utilisé afin que la valeur d'étalonnage change doucement. Le filtre combine la valeur nouvellement mesurée et multipliée par 0,1, avec la valeur précédente multipliée par 0,9 afin d'arriver à la nouvelle valeur d'étalonnage. Une partie de l'étalonnage en tâche de fond est ajouté à chaque scrutation rapide présente dans le programme de l'utilisateur. Les mesures en tâche de fond seront répétées toutes les 4 secondes ou bien à l'intervalle de temps qui leur est nécessaire pour s'exécuter, ce qui est un intervalle de temps plus important. S'il n'y a pas assez de temps pour effectuer l'étalonnage en tâche de fond, la CR1000 affichera : "Warning when Fast Scan X is running background calibration will be disabled." Soit « Attention car lorsque la scrutation rapide X est effectuée, l'auto étalonnage en tâche de fond sera désactivé. » (X est le numéro du fast scan – scrutation rapide - où la première scrutation entrée dans le programme est 1, la seconde est 2 etc.).
3. Étalonnage sous contrôle du programme. Lorsque l'instruction d'étalonnage est comprise dans un programme, le processus d'étalonnage est identique à celui du mode « à la compilation ». La table d'étalonnage est entièrement remplacée par les valeurs nouvellement entrées. Les intégrations rapides ont le même type de moyenne qu'en mode d'étalonnage à la compilation. Lorsqu'une instruction d'étalonnage est présente dans une des scrutations (*scan*) du programme, l'étalonnage en tâche de fond sera désactivé (même si le « scan » n'est pas exécuté). L'instruction d'étalonnage est décrite au chapitre 7.

L'auto étalonnage ne s'effectue pas s'il n'y a pas assez de temps pour qu'il s'accomplisse, ou si l'instruction d'étalonnage est présente dans le programme de la CR1000 sans être jamais exécutée. Sans cet auto étalonnage, le biais dans la précision sera d'un facteur 10. Par exemple sur la gamme -40 à 85°C (pour une centrale testée en T°), les caractéristiques de précision sont données à 0.1% de la valeur lue. Si l'auto étalonnage est désactivé, la précision deviendra d'environ 1%. La température est le facteur principal causant le biais de précision, et nécessitant l'auto étalonnage. Si la CR1000 reste à température constante, il y aura peu de biais même si l'auto étalonnage est désactivé

La constante de temps pour l'étalonnage en tâche de fond (au taux de 4 secondes) est approximativement de 36 secondes. Cela permet à la CR1000 de rester étalonnée lorsque la température change assez rapidement. Dans le cas de changements de températures extrêmes, comme le passage d'une condition de -30°C à une condition de jour d'été ensoleillé en plein midi, il sera préférable d'outrepasser l'étalonnage en tâche de fond et de placer l'instruction d'étalonnage dans le « scan » qui contient les prises de mesure.

Un autre cas où le fait d'exécuter l'instruction d'étalonnage peut avoir un sens, est lorsqu'il n'y a pas assez de temps dans la boucle normale d'exécution du programme, pour lancer l'étalonnage, mais qu'à un autre pas de temps la centrale stoppe l'acquisition de mesures afin de faire l'étalonnage dans un « scan » séparé.

Chapitre 4. Langage de programmation – CRBasic

La CR1000 est programmée dans un langage qui a des similitudes avec du basic structuré. Il y a des instructions spéciales pour effectuer des mesures et pour créer des tableaux de sauvegarde de données. Le résultat de toutes les mesures est dans des variables assignées (auxquelles on attribue des noms). Des opérations mathématiques sont écrites presque de la même façon que si c'était une écriture algébrique. Ce chapitre décrit un programme, sa syntaxe, sa structure et sa séquence de programmation.

4.1 Introduction au format

4.1.1 Opérations mathématiques

Les opérations mathématiques sont écrites d'une façon algébrique. Par exemple pour convertir une température en Celsius en une température en Fahrenheit, on peut écrire :

TempF = TempC * 1.8 + 32

Avec la CR1000 il peut y avoir 2 à 20 mesures de température (ou autre type de mesure). Au lieu d'avoir 20 noms de variables différents, une ligne de variable, avec un nom et 20 éléments, pourrait être utilisée. Une température de thermistance pourrait être appelée Temp. Avec une ligne de 20 éléments, le nom de chacune des températures seraient Temp(1), Temp(2) et jusqu'à Temp(20). La notion de ligne permet de compacter le code afin d'effectuer des opérations sur toutes les variables. Par exemple, pour convertir dix températures d'une ligne variable, de °C en °F, on a :

```
For I=1 to 10
        Temp(I)=Temp(I)*1.8+32
Next I
```

4.1.2 Instructions de mesure et de traitement de sauvegarde

Les instructions de mesure sont des procédures qui configurent le matériel (hardware) afin de faire une mesure, et placent le résultat dans une variable ou une ligne de variable (aussi appelée « ligne de données »). Les instructions de traitement de sauvegarde sont des procédures qui stockent le résultat des mesures effectuées, ou calculent des valeurs avant de les stocker. Les instructions de traitement de sauvegarde comprennent le calcul de la moyenne, de la sauvegarde du minimum ou du maximum, l'écart type, la transformée de Fourier (FFT) etc.

Les instructions qui servent à faire des mesures ou qui servent à sauvegarder des données, ne sont pas basées sur un langage basic standard. Les instructions que Campbell Scientific a créé pour effectuer ces opérations, sont sous la forme de procédures. La procédure a un nom que l'on entre au clavier, et une série de paramètres qui contiennent les informations nécessaires pour effectuer la procédure. Par exemple, l'instructions de mesure de la tension batterie de la CR1000 :

PanelTemp (*Dest,Integ*)

PanelTemp est le nom que l'on entre au clavier (*keyword*), pour cette instruction. Les deux paramètres associés à cette instruction PanelTemp sont : la *Dest*ination, qui est le nom de la variable dans laquelle sera mise la tension, et l'*Integ*ration, la durée de temps laissée pour intégrer le signal. Si vous voulez mettre la valeur de la température interne dans la variable appelée BattVolt, vous devrez entrer le code suivant :

PanelTemp(BattVolt,250)

L'utilisation de ces instructions devrait devenir de plus en plus claire au fur et à mesure que l'on avance dans cette introduction.

4.1.3 Insertion de commentaires dans un programme

Des commentaires peuvent être insérés dans le programme en débutant la ligne de commentaire par une marque « ‘ ». Les commentaires peuvent être ajoutés en début de ligne, ou à la suite suivant le code de la CR1000. Quand le compilateur de la CR1000 voit un « ‘ », il ignore le reste de la ligne.

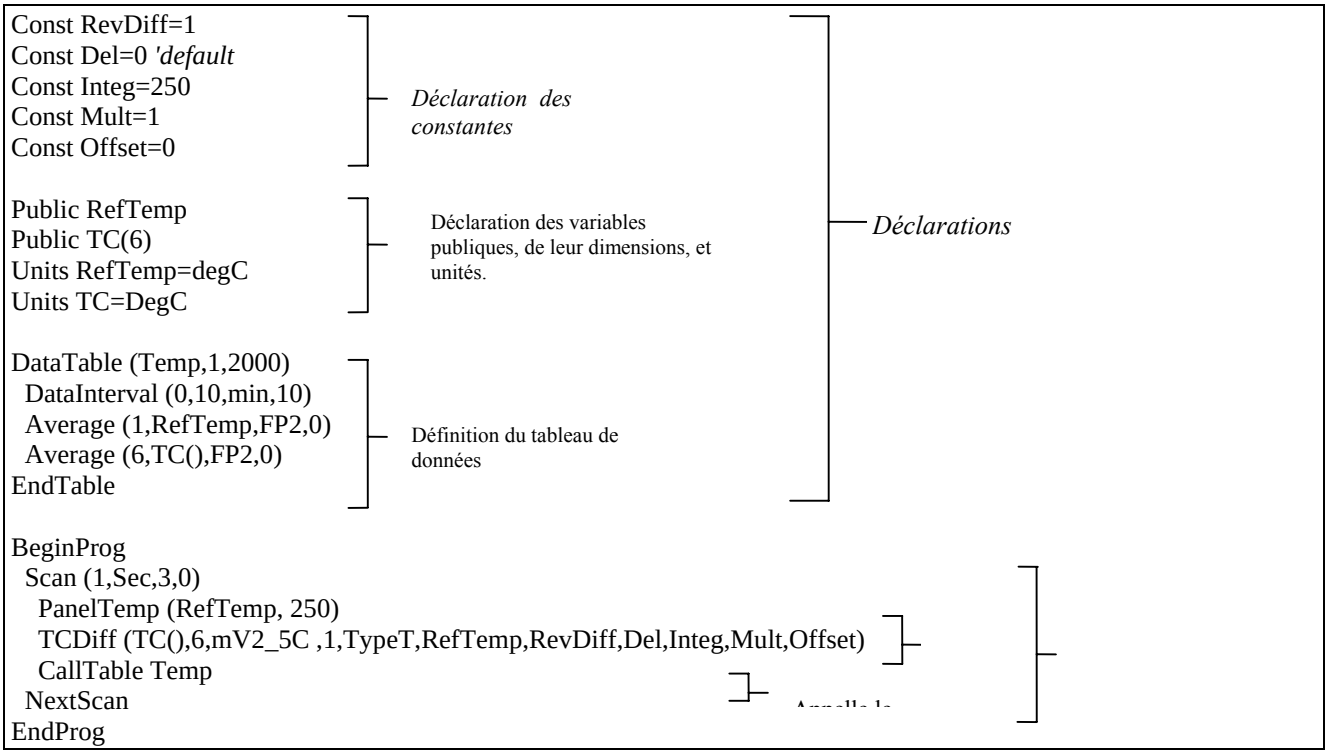
```
‘ La déclaration des variables débute ici
Public Start(6)                    ‘Déclare la ligne de début de temps.
```

4.2 Séquence de programmation

Le tableau suivant décrit la structure typique d’un programme de CR1000 :

Declarations	*Faire une liste de ce que l’on va mesurer et calculer*
Declare constants	*A l’intérieur de cette liste, insérez les constantes utilisées*
Declare Public variables	*Indiquez les variables que l’utilisateur peut visualiser lorsque le programme est en cours d’exécution,*
Dimension variables	*le nombre de chaque mesure qui sera effectuée,*
Define Aliases	*et un nom spécifique pour chacune des mesures effectuées (alias)*
Define data tables	*Décrit, en détail, les tableaux de données qui seront enregistrés pendant l’expérience.*
Process / store trigger	*Ceci est actif lorsque les données doivent être enregistrées. Sont-elles enregistrées lorsque certaines conditions sont atteintes ? Les données sont-elles enregistrées à intervalle de temps régulier ? Sont-elles enregistrées à intervalle régulier seulement lorsque certaines conditions sont atteintes ?*
Table size	*Ceci configure la taille du tableau dans la CR1000*
Other on-line storage device	*Les données doivent-elles aussi être envoyées à la carte CF ?*
Processing of Data	*Quels genre de données sont à enregistrer (les données brutes, moyennées, maximum, minimum etc. ?)*
Define Subroutines	*S’il y a des séries d’instructions qui sont répétées dans le programme, elles peuvent être incluses dans une « subroutine » et appelées lorsque cela est nécessaire sans avoir à re-taper le code en entier.*
Program	*Le paragraphe dédié au programme, définit les actions faites par la centrale de mesure*
Set scan interval	*L’intervalle de scrutation définit la périodicité de mesure*
Measurements	*Entrez les mesures à effectuer*
Processing	*Entrez les calculs additionnels à effectuer sur les mesures*
Call Data Table(s)	*Le tableau de données doit être appelé afin que les données soient enregistrées*
Indicate controls	*Vérifie les mesures et initie des contrôles si cela est nécessaire*
NextScan	*Termine la boucle (et attendre si cela est nécessaire) pour le prochain intervalle de scrutation.*
End Program	

4.3 Exemple de programme



4.3.1 Tableaux de données

Le stockage des données suit une structure fixe dans la CR1000, afin d'optimiser le temps et l'espace nécessaire. Les données sont enregistrées dans des tableaux tels que :

TOA5	Fritz	CR1000	1079	CR1000.Std.1.0	CPU:TCTemp.CR1	51399	Temp	
TIMESTAMP	RECORD	RefT_Avg	TC_Avg(1)	TC_Avg(2)	TC_Avg(3)	TC_Avg(4)	TC_Avg(5)	TC_Avg(6)
TS	RN	degC	DegC	DegC	DegC	DegC	DegC	DegC
		Avg	Avg	Avg	Avg	Avg	Avg	Avg
10/28/2004 12:10	119	23.52	23.49	23.49	23.5	23.49	23.5	23.5
10/28/2004 12:20	120	23.55	23.51	23.51	23.51	23.51	23.51	23.52
10/28/2004 12:30	121	23.58	23.52	23.53	23.53	23.53	23.53	23.53
10/28/2004 12:40	122	23.58	23.53	23.54	23.54	23.54	23.54	23.54

Le programme de l'utilisateur détermine les valeurs qui seront enregistrées, et l'ordre dans lequel ce sera fait. La CR1000 assigne automatiquement un nom à chaque champ dans le tableau de données. Dans le tableau précédent, TIMESTAMP, RECORD, RefTemp_Avg et TCAvg(1) sont des noms de champ. Les noms de champs sont une combinaison du nom de la variable (ou de l'alias s'il existe) et d'un mot mnémotechnique à 3 lettres identifiant le type de traitement sur la donnée à enregistrer. De façon alternative, l'instruction « FieldNames » peut être utilisée pour modifier le nom par défaut qui serait donné au champ.

L'en-tête du tableau de données a aussi une colonne qui liste les unités de mesure pour les données enregistrées. Les unités doivent être déclarées à la CR1000 afin de pouvoir être prises en compte dans cette colonne (par exemple : Unit RefTemp = degC). Les unités ne servent qu'à la documentation de l'utilisateur. La CR1000 ne fait aucun contrôle au sujet de leur justesse.

Le tableau ci-avant est le résultat de la description de tableau de l'exemple de programme suivant :

```
DataTable (Temp,1,2000)
    DataInterval(0,10,min,10)
    Average(1,RefTemp,fp2,0)
    Average(4,TC(),fp2,0)
EndTable
```

Toutes les descriptions de tableau de données débutent avec « **DataTable** » et finissent par « **EndTable** ». Entre ces descriptions se trouvent des instructions qui disent ce qu'il faut enregistrer, ou qui peuvent modifier les conditions sous lesquelles la sauvegarde se produit.

```
'DataTable(Name, Trigger,Size)
DataTable (Temps,1,2000)
```

L'instruction de **DataTable** a trois paramètres : un nom défini par l'utilisateur pour le tableau de données, une condition de basculement (*trigger condition*), et la taille que fera le tableau dans la mémoire RAM de la CR1000. La condition de basculement peut être une variable, une expression, ou une constante. La condition de basculement est vraie si elle n'est pas égale à zéro. Les données sont envoyées en mémoire finale si la condition de basculement est atteinte (vraie) et il n'y a aucune autre condition à atteindre. Il n'y a aucune sauvegarde d'effectuée si la condition de basculement est fausse (=0). L'exemple crée un tableau de données appelé Temp, effectue la sauvegarde à chaque fois que d'autres conditions sont atteintes, et garde 2000 enregistrements en mémoire RAM.

```
'DataInterval(TintoInt,Interval,Units,Lapses)
DataInterval(0,10,min,10)
```

L'instruction **DataInterval** est une instruction qui modifie la condition pour laquelle les données seront stockées. Les quatre paramètres sont le temps à l'intérieur de l'intervalle de temps, la durée de l'intervalle de temps auquel les données sont stockées, l'unité de temps de l'intervalle, et le nombre de trous de données (*lapses or gaps*) dans l'intervalle, pour lesquels il faut garder une trace. L'exemple donné enregistrera des valeurs à chaque valeur du temps « 0 » dans l'intervalle de temps faisant « 10 » minutes, avec la valeur du 0 calé par rapport à l'horloge interne de la centrale de mesure, et en gardant trace de 10 trous de données. L'instruction DataInterval permet de réduire la taille de la mémoire nécessaire pour un tableau de sauvegarde car l'heure de chaque enregistrement peut être calculé à partir de la grandeur de l'intervalle de sauvegarde et de l'heure de l'enregistrement le plus récent. D'autres modificateurs de condition d'enregistrement sont *WorstCase* et *FillandStop*.

Les instructions de sauvegarde comprises dans la définition du tableau de données, déterminent les valeurs qui seront enregistrées dans le tableau. Le tableau doit être appelé par le programme si l'on veut que les instructions de sauvegarde soient exécutées. Ceci se produira à chaque fois que de nouvelles mesures sont effectuées. Lorsque le tableau est appelé, les instructions de sauvegarde sont exécutées sur les valeurs courantes contenues dans la mémoire d'entrée. Si les conditions de basculement du tableau sont atteintes, alors les données calculées par les instructions de sauvegarde sont envoyées dans le tableau de données. Dans l'exemple ci-dessous, plusieurs données sont sauvegardées.

```
'Average(Reps, Source,DataType, DisableVar)
Average(1,RefTemp,fp2,0)
Average(6,TC(1),fp2,0)
```

L'instruction « Average » (moyenner), est une instruction de sauvegarde dont le résultat est de calculer la moyenne d'une variable, sur la durée de l'intervalle de sauvegarde. Les paramètres utilisés sont le nombre de répétitions (le nombre d'élément d'une ligne de données 'array ', pour lesquels on va calculer une moyenne), la variable source ou la ligne de donnée à moyenner, le format de données dans lequel on va effectuer la sauvegarde 'voir Tableau 4.3-1) et une variable de « passage outre », permettant de ne pas prendre en compte certaines valeurs pour le calcul de la moyenne, si certaines conditions sont atteintes. Une valeur ne sera pas incorporée à la moyenne, si la variable de « passage outre » est différente de 0. L'exemple précédent a « 0 » comme valeur pour le paramètre de « passage outre », ainsi toutes les valeurs seront prises en compte dans le calcul de la moyenne.

Tableau 4.3-1 Formats pour les données de sauvegarde

Code	Format de donnée	Taille	Etendue de mesure	Résolution
FP2	Format à virgule flottante de Campbell Scientific	2 bytes	±7999	13 bits (environ 4 digits)
IEEE4	Format IEEE à virgule flottante et 4 byte	4 bytes	De $1.8^{E\ -38}$ à $1.7^{E\ 38}$	24 bits (environ 7 digits)
LONG	Entier à 4 byte et signe	4 bytes	De -2,147,483,648 à +2,147,483,647	1 bit (1)
BOOLEAN	Entier à 4 byte et signe	4 bytes	0, -1	Vrai ou Faux (-1 ou 0)
STRING	Chaîne ASCII	Fixée par le programmeur		

4.3.2 Temps de scrutation – Temporisation pour la mesure et le calcul

Une fois que vous savez ce que vous voulez, que les mesures et les calculs ont été listés et que vos tableaux de sauvegarde ont été définis, le programme en lui même peut être relativement court. Le programme à exécuter débute alors par « **BeginProg** » et se termine par « **EndProg** ». Les mesures, les calculs et les appels afin de remplir les tableaux de sauvegarde, sont à l'intérieur des « crochets » définis par les instructions « **Scan** » et « **Nextcan** », qui déterminent la fréquence de scrutation de la centrale de mesure.

```
BeginProg
Scan(1,Sec,3,0)
    ModuleTemp(RefTemp, 250)
    TCDiff(TC(),6,mV2_5C,4,1,TypeT,RefTemp,RevDiff,Del,Integ,Mult,Offset)
    CallTable Temp
NextScan
EndProg
```

L'instruction « **Scan** », détermine à quelle fréquence les mesures comprises dans la boucle de scrutation (*scan*), sont effectuées :

```
'Scan(Interval,Units,BufferSize,Count)
Scan(1,Sec,3,0)
```

L'instruction « Scan » a quatre paramètres. L'*interval* est l'intervalle de temps entre deux scrutations. L'*Units*, est l'unité de temps à utiliser pour l'intervalle. L'intervalle des temps minimum est de 10 milisecondes et l'intervalle maximum que l'on peut donner est de 30 minutes. La *BufferSize* est la taille (en unité de nombre de scrutations) de mémoire tampon de la RAM, qui contiendra les valeurs brutes effectuées. Le fait d'utiliser un Buffer (une mémoire tampon) permet aux instructions de calcul de s'effectuer en décalé par rapport aux instructions de mesure, sans affecter l'acquisition des mesures à un temps donné (voir l'instruction *scan* au chapitre 9 pour plus de détails). Le paramètre *Count* est le nombre de scrutations qu'il faut effectuer avant d'effectuer les mesures qui suivent l'instruction **NextScan**. Si la valeur de *Count* est « 0 », le programme effectuera la boucle jusqu'à l'infini (ou jusqu'à ce qu'il y ait une instruction **ExitScan**). Dans l'exemple précédent la scrutation est effectuée toutes les une seconde, trois scrutations sont mises en mémoire tampon, et les mesures ainsi que les sauvegardes sont effectuées à l'infini.

4.4 Types de variables de données

La déclaration des variables (via la mention DIM ou PUBLIC) permet d'attribuer un descriptif supplémentaire (un « type »), à la suite de la variable, en utilisant le terme AS. Le type de donnée par défaut, si l'on n'utilise pas de descriptif, est le format IEEE à 4 bits et virgule flottante (FLOAT). Les types de données sont FLOAT, LONG, BOOLEAN et STRING.

4.4.1 FLOAT

Si on ajoute la mention "AS FLOAT" cela définira le format par défaut qui est l'IEEE4. Si aucun type de donnée n'est explicitement spécifié à la suite d'un descriptif « AS », c'est ce type FLOAT qui est attribué.

```
Public Z, RefTemp, TCTemp(3)
Public X AS FLOAT
```

4.4.2 LONG

"AS LONG" déterminera une variable en tant qu'entier de 32 bits de long compris entre -2 147 483 648 et +2 147 483 647 (31 bits plus le signe). Il y a deux raisons pour lesquelles un utilisateur pourrait choisir ce format : Tout d'abord une question de vitesse car le système d'exploitation de la CR1000 effectuera des calculs sur des entiers, plus rapidement que sur des chiffres à virgule flottante ; ensuite une question de résolution car le format LONG possède 31 bits alors que le format IEEE4 n'en possède que 24.

Exemple:

```
Dim I AS  LONG
Public LongCounter AS LONG
```

4.4.3 BOOLEAN

"AS BOOLEAN" permet de définir une variable en tant que booléen à 4 bits. Les variables de type booléen sont généralement utilisées pour tester l'état de drapeaux (*flags*) ou pour représenter une condition ou l'état d'un matériel qui ne supporte que deux états (par exemple On/Off). Une variable de type Booléen utilise le même format à 32 bits que le format de type LONG, mais ne peut être fixé qu'à deux valeurs : Vrai (*True*) qui est représenté par –1, et Faux (*False*) qui est représenté par 0. Le type de donnée Booléen permet au logiciel d'application de d'afficher cette variable en tant que ON/OFF, VRAI/FAUX, ROUGE/BLEU etc.

```
Public Switches(8) AS Boolean, FLAGS(16) AS Boolean
```

4.4.4 STRING

La mention « AS STRING * *taille* » déclarera la variable en tant que chaîne de caractères ASCII, terminés par le caractère NULL, et ayant la *taille* spécifiée qui sera la longueur maximum de caractères dans la chaîne de caractères. Les chaînes de caractères sont intéressantes pour gérer les capteurs série, les chaînes d'initialisation de modems, les messages texte etc.

Les chaînes de caractères ne peuvent avoir que 2 dimensions contrairement à 3 dimensions possibles pour les autres types de données. (Ceci est vrai parce que la dimension la moins significative est, en fait, utilisée pour déterminer la taille de la chaîne de caractères)

```
Public FirstName AS STRING * 20
Public LastName AS STRING * 20
```

4.4.5 Expressions numériques avec les types Float, Long et Boolean

Les types Float, Long et Boolean sont automatiquement convertis dans l'un ou l'autre des formats.

Conversion en type Boolean à partir du type Float ou Long

Lorsqu'un entier de type « Long » ou « Float » est converti en Boolean, la valeur zéro (0) sera Fausse, et tout autre valeur sera Vraie (-1).

```
Public X, Y
Public I AS Long, B AS Boolean
BeginProg
        X = 0
        Y = 0.125
        I = 126
        B = X           'Cela configurera B = False (0)
        B = Y           'Cela configurera B = True (-1)
        B = I           'Cela configurera B= True (-1)
EndProg
```

Conversion en type Float à partir du type Long ou Boolean

Lorsqu'un type Long ou Boolean est converti en format Float, la valeur entière est mise en mémoire dans la variable de type Float. Les Boolean seront converti en tant que –1 ou 0 selon que la valeur du booléen soit vraie ou fausse. A noter que les entiers supérieurs à 24 bits (16 777 215 ; qui est la taille de la mantisse font les données de type Float) perdront en résolution lorsqu'ils sont convertis en donnée de type Float.

Conversion en type Long à partir du type Float ou Boolean

Les booléens seront convertis en tant que –1 ou 0 selon que leur valeur soit vraie ou fausse. Lorsqu'une donnée de type Float est convertie en entier de type Long, elle est tronquée. La conversion est identique à celle effectuée par la fonction INT (voir chapitre 8). A noter que la conversion en entier, se fait vers l'entier égal ou inférieur à la valeur de la donnée de type Float. Cela peut ne pas être intuitif pour des nombres négatifs, comme par exemple :

```
Dim I as Float
BeginProg
        I = 4.6         'Cela configurera I en 4.
        I = -4.6        'Cela configurera I en –5.
EndProg
```

Si la donnée de type Float est supérieure à la valeur maximum de l'entier de type Long supportée, alors l'entier se verra attribuer la valeur maximum soit +2 147 483 647. Si la valeur de la donnée de type Float est inférieure à l'entier de type Long le plus petit supporté, alors l'entier se verra attribuer la valeur minimum soit -2 147 483 648.

Les expressions sont évaluées le plus souvent possible en tant qu'entiers

```
Public X, I AS Long
BeginProg
        I = 126
        X = (I+3) * 3.4
                'I+3 est évalué en tant qu'entier , puis converti en donnée
                'de type ' FLOAT' avant qu'elle ne soit multipliée par 3,4
EndProg
```

Les constantes seront converties en tant que données de type Long et/ou Float à la compilation

Si une constante (déclarée comme une valeur numérique ou bien avec la définition de CONST) peut être exprimée correctement en tant qu'entier, alors le compilateur utilisera le type de donnée qui est le plus efficace dans chaque expression. La version avec les entiers sera utilisée de préférence, c'est à dire si l'expression n'a pas été comparée auparavant à une donnée de type Float.

```
Public I AS Long,          'I est un entier
Public F1, F2              'F1 et F2 sont de type Floats
CONST ID = 10
BeginProg
    I = ID * 5             'ID ('10') et '5' sont chargés à la compilation
                           'en tant que données de type ' Floats'
    F1 = F2 + ID           'ID ('10') est chargé à la compilation en tant que
                           'donnée de type ' float' afin d'éviter une
                           'conversion lors de l'exécution du programme et
                           'ce à partir d'un entier, avant chaque addition.
EndProg
```

4.5 Entrées numériques

En plus du chiffrage en base 10, il y a 3 autres façons de représenter des nombres à l'intérieur d'un programme : la notation scientifique, binaire, et hexadécimale (voir tableau 4.5-1).

TABLEAU 4.5-1. Formats utilisables afin d'entrer des nombres en CRBasic

Format	Exemple	Valeur
Standard	6.832	6,832
Notation scientifique	5.67^{E-8}	$5{,}67 \times 10^{-8}$
Binaire	&B1101	13
Hexadécimale	&HFF	255

Le format binaire permet d'avoir une visualisation simple des opérations lorsque les 1 et les 0 se déplacent grâce à des commandes spécifiques. On peut par exemple affecter un nombre de bloc de ports de contrôle, avec l'état du port sous forme binaire (1 = *High* ; 0 = *Low*). Si on souhaite activer les ports 1, 3, 4 et 6 à l'état haut (*high*) et les ports 2,5, 7 et 8 à l'état bad (*low*), on utilisera le nombre &B00101101. Le bit le moins significatif est à droite et représente le port 1. Ceci est plus facile à visualiser que de décoder son équivalent décimal dont la valeur est 72.

4.6 Evaluation des expressions logiques

4.6.1 Qu'est-ce qui est vrai ?

Certains mots sont utilisés afin de décrire une condition ou le résultat d'un test. L'expression, X>5, est soit « vraie » soit « fausse ». Lorsqu'on parle de l'état d'un port de contrôle ou d'un drapeau, les mots « activé » et « désactivé » (*On/Off*), ou « haut » et « bas » (*High/Low*) sont plus facilement utilisés. En CRBasic il y a plusieurs tests conditionnels ou plusieurs paramètres d'instructions, qui peuvent être décrits par un ou plusieurs mots du tableau 4.6-1.
La CR1000 évalue le test ou le paramètre en tant que numéro ; 0 si le résultat est faux, différent de 0 si c'est vrai.

TABLEAU 4.6-1. Synonymes pour « vrai » ou « faux »		
Constante prédéfinie	Vrai (-1) / *True*	Faux (0) / *False*
Synonyme	Haut / *High*	Bas / *Low*
Synonyme	Activé / *On*	Désactivé / *Off*
Synonyme	Oui / *Yes*	Non / *No*
Synonyme	Basculement / *Trigger*	Pas de basculement / *Do not trigger*
Nombre	≠ 0	0
Port numérique	5 Volts	0 Volts

4.6.2 Evaluation de l'expression

Les test conditionnels nécessitent à la CR1000 d'évaluer une expression, et de suivre une voie si la condition est vraie, ou une aut re voie si elle est fausse. Par exemple :
IF X>=5 then Y=0
Donnera la valeur 0 à la variable Y si la variable X est supérieure ou égale à 5.

La CR1000 pourra aussi évaluer des expressions multiples liées par des « **and** » ou des « **or** ». Par exemple:
If X>=5 and Z=2 then Y=0
Donnera la valeur 0 à Y seulement si X est supérieur ou égal à 5 et si Z est égal à 2.
If X>=5 or Z=2 then Y=0
Donnera la valeur 0 à Y si l'une des deux conditions est vraie (X supérieur ou égal à 5 ou Z = 2). Voir le descriptif de « **And** » et « **Or** » au chapitre 9. Une condition peut prendre en compte plusieurs liens de type « **And** » ou « **Or** ».

4.6.3 Résultats numériques de l'évaluation de l'expression

La fonction d'évaluation de la CR1000 évalue une expression, et donne un chiffre en résultat. Une expression conditionnelle utilise le chiffre afin de décider quel chemin prendre. L'expression conditionnelle est fausse si le chiffre est égal à 0, et vraie si le chiffre est différent de 0.

Par exemple :
If 6 then Y=0,
Est une condition qui est toujours vraie ; Y sera toujours mis à 0 à chaque fois que l'expression conditionnelle sera exécutée.
If 0 then Y=0
Est toujours fausse; Y ne sera jamais mis à 0 par cette expression conditionnelle.

La fonction d'évaluation de la CR1000 évalue l'expression, X>=5, et donne le résultat –1, si l'expression est vraie, et 0, si l'expression est fausse.
W=(X>Y)
Donnera –1 à la variable W si X est supérieur à Y, ou donnera la valeur 0 à la variable W, si X est inférieur ou égal à Y.

La CR1000 utilise la valeur –1 plutôt qu'un autre chiffre différent de 0, parce que les opérateurs « and » et « or » sont les mêmes pour des états logiques et des comparaisons binaire de comparaison de bits (voir « and » et « or » au chapitre 9). Le chiffre –1 est exprimé en binaire, avec tous les bits égaux à 1, alors que le chiffre 0 a tous les bits égaux à 0. Lorsque –1 est ajouté (par l'expression « and ») à n'importe quel autre chiffre, le résultat est identique à l'autre chiffre, en s'assurant que si l'autre chiffre est différent de zéro (vrai), le résultat sera différent de zéro.

4.7 Les drapeaux (Flags)

Bien que n'importe quelle variable puisse être utilisée en tant que « drapeau », pour autant que des tests logiques soient utilisés avec le CRBasic, il est préférable d'utiliser des variables de type booléennes. Si la valeur de la variable est différente de zéro, alors l'état du drapeau est l'état haut. Si la valeur de la variable est zéro, alors le drapeau est à l'état bas (voir paragraphe 4.6). Les variables booléennes ne peuvent avoir qu'une des deux valeurs, vrai (-1) ou faux (0).

4.8 Les types de paramètre

Les paramètres des instructions permettent d'entrer différents types de choses en entrée. Ces différents types d'entrée sont listées ci-dessous, et sont repris dans les chapitres suivants, ou dans le menu d'aide à la programmation du CRBasic.

Constante
Variable
Variable ou ligne de données
Constante, variable ou expression
Constante, variable, ligne de données ou expression
Nom
Nom ou liste de noms
Variable ou expression
Variable, ligne de données ou expression

Le tableau 4.8-1 donne la liste des longueurs maximales et des caractères autorisés pour ce qui est du nom des variables, des lignes de constante etc.

TABLEAU 4.8-1. Règles pour l'établissement des noms

Nom pour	Longueur maximum (nombre de caractères)	Caractères admis
Variable ou Ligne de données	16	Les lettres de A à Z (majuscule ou minuscule), les tirets bas « _ », et les chiffres de 0 à 9. Le nom doit débuter par une lettre. Le CRBasic n'est pas « case sensitive » (pas de différence entre un nom avec majuscule ou sans majuscule).
Constante	16	
Alias	16	
Nom d'un tableau de sauvegarde	8	
Nom d'un champ (colonne)	16	

4.8.1 Expressions dans les paramètres

Plusieurs paramètres donnent la possibilité de mettre des expressions à l'intérieur du paramètre. Si l'expression est une comparaison, le résultat de la comparaison sera –1 si la comparaison est vraie, et 0 si elle est fausse (voir paragraphe 4.6.3). Un exemple d'utilisation de cela, est dans l'instruction **DataTable**, pour la condition de basculement (« *trigger* »), qui peut être une expression. Si l'on suppose que la variable TC(1) est une mesure de thermocouple :

```
' DataTable(Name, TrigVar, Size)
DataTable(Temp, TC(1)>100, 5000)
```

Le fait d'entrer une condition de basculement dans l'expression, TC(1)>100, va induire le fait que l'enregistrement n'aura lieu que lorsque la température de TC(1) sera supérieure à 100.

4.8.2 Lignes de données de multiplicateurs et d'offsets pour l'étalonnage de capteurs

Si l'on utilise des lignes de données variables lors de mesures effectuées avec des répétitions, alors l'instruction de mesure utilisera automatiquement les multiplicateurs et offset définis comme ligne de donnée, à mesure que la centrale fait les mesures sur les voies consécutives. Cela permet à une seule instruction de mesure d'effectuer la mesure de plusieurs capteurs préalablement étalonnés individuellement, en appliquant le coefficient adéquat pour chaque capteur. Si le multiplicateur et l'offset ne sont pas définis dans une ligne de donnée, le même multiplicateur et le même offset sont alors utilisés pour chacune des répétitions et donc pour chacun des capteurs.

```
Public Pressure(3), Mult(3), Offset(3)

DataTable (AvgPress,1,-1)
    DataInterval (0,60,Min,10)
    Average (3,Pressure(),IEEE4,0)
EndTable

BeginProg
    'Facteurs d'étalonnage :
    Mult(1)=0.123 : Offset(1)=0.23
    Mult(1)=0.115 : Offset(1)=0.234
    Mult(1)=0.114 : Offset(1)=0.224

    Scan (1,Sec,10,0)
        'L'instruction VoltSe utilise une ligne de données de
        'multiplicateur et d'offsets :
        VoltSe (Pressure(),3,mV5000,1,True,0,_60Hz,Mult(),Offset())
        CallTable AvgPress
    NextScan
EndProg
```

4.9 Accès du programme aux tableaux de données

Les données enregistrées dans les tableaux de sauvegarde, peuvent être accessibles depuis le programme. Le format utilisé est le suivant :

Tablename.Fieldname(fieldname index,records back)
Tablename est le nom du tableau de sauvegarde dans lequel les mesures que l'on souhaite lire, sont stockées. *Fieldname* est le nom du champ (colonne) dans le tableau. Le champ Fieldname est toujours une ligne de donnée, même s'il n'est constitué que d'une seule variable. Le *fieldname index* doit toujours être spécifié. Ce qui correspond à *Records back*, est le numéro d'enregistrement antérieur au dernier enregistrement présent dans le tableau de sauvegarde (1 est l'enregistrement le plus récent, 2 est l'enregistrement enregistré juste avant le plus récent etc.).
L'expression :

Tdiff=Temp.TC_Avg(1,1)-Temp.TC(1,2)

pourrait être utilisée dans l'exemple de programme (paragraphe 4.3) afin de calculer le changement de température moyenne sur 10ms pour le premier thermocouple, entre la moyenne la plus récente et celle qui a été enregistrée pour la seconde précédente (100 x 10 ms).

En plus du fait d'accéder au données réellement enregistrées dans un tableau de sauvegarde, il existe certains pseudo fichiers associés au tableau de sauvegarde, qui peuvent être récupérés en utilisant la même syntaxe :

Tablename.record(1,n) = le numéro d'enregistrement de l'enregistrement enregistré « n » enregistrement auparavant.

Tablename.output(1,1) = -1 si les données ont été enregistrées la dernière fois que le tableau de sauvegarde a été appelé, = 0 si aucune donnée n'a été enregistrée.

Tablename.timestamp(m,n) = élément « m » de l'enregistrement du temps (« timestamp ») enregistré « n » enregistrements auparavant, où :

timestamp(1,n) = microsecondes depuis 1990
timestamp(2,n) = microsecondes dans l'année à laquelle nous sommes
timestamp(3,n) = microsecondes dans le mois où nous sommes
timestamp(4,n) = microsecondes dans la journée présente
timestamp(5,n) = microsecondes dans l'heure présente
timestamp(6,n) = microsecondes dans la minute présente
timestamp(7,n) = microsecondes dans la seconde présente

Tablename.tablesize(1,1) = la taille du tableau en tant que nombre d'enregistrements.

Tablename.tablefull(1,1) = / ou 0 afin d'indiquer si un tableau en mode remplissage et arrêt (*fill and stop*) est rempli, ou si un tableau en mode de remplissage circulaire (*ring memory*) a commencé à écraser les données les plus anciennes. (0 indique que le tableau n'est pas plein.)

Tablename.eventend(1,1) n'est valide que pour un tableau de données qui utilise l'instruction **DataEvent**, *Tablename*.eventend(1,1) = -1 si le dernier enregistrement d'un évènement s'est produit lors de la dernière fois que le tableau a été appelé, = 0 si le tableau de données n'a pas effectué d'enregistrement ou si on se situe au milieu d'un événement.

Tablename.eventcount(1,1) = le nombre d'évènements d'enregistrement de données qui se sont produit dans un tableau de données qui utilise l'instruction **DataEvent**.

NOTE Les valeurs de Tablename.output(1,1) et Tablename.eventend (1,1) ne sont mises à jour qu'au moment ou les tableaux sont appelés.

L'exemple de **WorstCase** présent dans le paragraphe 6.2 illustre l'utilisation de cette syntaxe.

Chapitre 5. Déclarations dans un programme

Alias

Cette instruction est utilisée afin de donner un second nom à une variable.

La syntaxe est la suivante :
Alias *VariableA = VariableB*

Remarques :
Les Alias permettent de donner un second nom à une variable. A l'intérieur du programme de la centrale de mesure, l'un ou l'autre des deux noms peuvent être utilisés. Lorsqu'on regarde les valeurs « Public » (*PublicTable*), et si une variable a un Alias, c'est le nom de l'Alias qui est affiché en tant que nom de la variable. L'Alias est aussi utilisé afin d'être le nom prioritaire utilisé pour identifier les noms des colonnes dans les tableaux de sauvegarde.

Avec des Alias, le programme peut avoir l'efficacité de la mesure et du traitement des données appliquées aux lignes de données, couplée à des noms individualisés pour le descriptif des mesures effectuées.

Exemple de déclaration d'Alias
L'exemple suivant montre une façon d'utiliser la déclaration d'Alias.

```
Dim Temp(4)
Alias Temp(1)= CoolanT
Alias Temp(2)= ManifoldT
Alias Temp(3)= ExhaustT
Alias Temp(4)= CatConvT
```

AngleDegrees

Cette instruction est utilisée afin de préciser aux fonctions mathématiques du programme à exécuter, que les données en résultat aussi bien qu'en source seront en degrés en non en radians.

La syntaxe est la suivante :
AngleDegrees

Remarques :
L'instruction **AngleDegree** est placée dans la partie de déclaration des variables du programme, avant le code compris entre les instructions BeginPro/EndProg.

AngleDegree a une influence sur les instructions qui donnent comme résultat des radians de façon habituelle soit :

ATN, ARN2, ACOS, ASIN, RectPolar.

AngleDegree a une influence sur les instructions qui s'attendent à avoir une source de données en radians de façon habituelle soit :

CAS, COSH, TAN, TANH, SIN, SINH.

Les radians négatifs seront convertis en degrés négatifs.

Type AS

La déclaration des variables (via les instructions DIM ou PUBLIC) permet de définir le type de descriptif pour les données, après la déclaration AS. Le type de données par défaut et sans descriptif, est le format à virgule flottante IEEE4 (FLOAT). Les types de données sont FLOAT, LONG, BOOLEAN et STRING.

AS FLOAT déterminera le type de données par défaut, IEEE4. Si aucun type de donnée n'est spécifié, alors le descriptif FLOAT est assigné.

```
Public Z, RefTemp, TCTemp(3)
Public X AS FLOAT
```

AS LONG déterminera une variable en tant qu'entier de 32 bits de long compris entre -2 147 483 648 et +2 147 483 647 (31 bits plus le signe). Il y a deux raisons pour lesquelles un utilisateur pourrait choisir ce format : Tout d'abord une question de vitesse car le système d'exploitation de la CR1000 effectuera des calculs sur des entiers, plus rapidement que sur des chiffres à virgule flottante ; ensuite une question de résolution car le format LONG possède 31 bits alors que le format IEEE4 n'en possède que 24.

```
Dim I AS LONG
Public LongCounter AS LONG
```

AS BOOLEAN déterminera une variable en tant que booléen à 4 bits. Les variables de type booléen sont généralement utilisées pour tester l'état de drapeaux (*flags*) ou pour représenter une condition ou l'état d'un matériel qui ne supporte que deux états (par exemple On/Off). Une variable de type Booléen utilise le même format à 32 bits que le format de type LONG, mais ne peut être fixé qu'à deux valeurs : Vrai (*True*) qui est représenté par –1, et Faux (*False*) qui est représenté par 0. Le type de donnée Booléen permet au logiciel d'application d'afficher cette variable en tant que ON/OFF, VRAI/FAUX, ROUGE/BLEU etc.

```
Public Switches(8) AS Boolean, FLAGS(16) AS Boolean
```

AS STRING * *taille* déclarera la variable en tant que chaîne de caractères ASCII, terminés par le caractère NULL, et ayant la *taille* spécifiée qui sera la longueur maximum de caractères dans la chaîne de caractères. Les chaînes de caractères sont intéressantes pour gérer les capteurs série, les chaînes d'initialisation de modems, les messages texte etc.

Les chaînes de caractères ne peuvent avoir que 2 dimensions contrairement à 3 dimensions possibles pour les autres types de données. (Ceci est parce que la dimension la moins significative est, en fait, utilisée pour déterminer la taille de la chaîne de caractères)

```
Public FirstName AS STRING * 20
Public LastName AS STRING * 20
```

Const

Cette instruction est utilisée afin de déclarer des constantes symboliques que l'on utilise à la place d'entrées numériques.

La syntaxe est la suivante :
Const *nomdelaconstante* = *expression*

Remarques :
La fonction **Const** est constituée des parties suivantes:

Partie	**Description**
nomdelaconstante	Nom de la constante
expression	Expression que l'on assigne à la constante. Cela peut être une expression littérale (du type 1.0), une autre constante, ou un des opérateurs logiques arithmétiques.

NOTE Le fait d'utiliser des constantes, peut rendre votre programme auto-documenté et facile à modifier. Contrairement aux variables, les constantes ne peuvent pas être modifiées pendant que votre programme est en cours d'exécution.

ATTENTION Les constantes doivent être définies avant qu'on ne les appelle.

NOTE Il est conseillé d'utiliser des lettres majuscules pour le nom des constantes, afin de les reconnaître facilement dans votre programme.

Exemple de déclaration de constante

L'exemple suivant est utilisé afin de définir la constante symbolique PI.

```
Const PI = 3.141592654          'Définition de la constante
Dim Aire, Circonf, Rayon        'Déclaration des variables
Rayon = Volt(1)                 'Prise de la mesure
Circonf = 2 * PI * Radius       'Calcul de la circonférence
Aire = PI * ( Rayon ^ 2)        'Calcul de l'aire
```

Dim

Déclare les variables et leur alloue un espace mémoire. En CRBasic, **TOUTES** les variables **DOIVENT** être déclarées.

La syntaxe est la suivante :
Dim *varname*[(*subscripts*]) [, *varname*[([*subscripts*])]]

Remarques :
La fonction Dim est constituée des parties suivantes:

Partie	**Description**
varname	Nom de la constante
subscripts	Dimensions d'une variable de type ligne de variable. On peut déclarer plusieurs dimensions.

L'argument « *subscripts* » a la synthaxe suivante :

Taille [taille, taille]

En CRBasic le plus petit nombre de dimension est 1 et non 0.

```
' Crée la variable A en tant que ligne de données à 8 éléments.
Dim A( 8)
```

Le nombre maximum de dimensions pour une ligne de données avec une instruction **Dim** est de 3. Si un programme utilise un paramètre de « subscript » qui est supérieur à la valeur dimensionnée, une erreur (*subscript out of bounds error*) est enregistrée.

Quand les variables sont initialisées, elles sont mises à la valeur 0.

NOTE On met la déclaration des variables **Dim** en début de programme.

PipelineMode

L'instruction PipelineMode est utilisée afin de configurer la centrale de mesure pour qu'elle effectue toutes les instructions décrites dans le programme d'acquisition, de façon organisée et temporisée (mode *pipeline*).

La syntaxe est la suivante :
PipelineMode

Remarques :
La centrale de mesure dispose de deux modes de fonctionnement : les modes séquentiels et « pipeline ». En mode séquentiel les instructions sont effectuées par la centrale de mesure de façon séquentielle, dans l'ordre où elles sont écrites dans le programme. En mode « pipeline » les tâches de mesure et les tâches de traitement sont effectuées de façon séparée et exécutées en même temps.

Le mode par défaut est le mode Pipeline. Cependant lorsque le programme de la centrale de mesure est compilé, la centrale de mesure analyse les instructions du programme et passe automatiquement du mode pipeline au mode séquentiel si le code le nécessite. On peut forcer la centrale de mesure à fonctionner en mode pipeline ou séquentiel en plaçant l'instruction appropriée au début du programme, avant l'instruction BeginProg.

Voir le paragraphe OV2.3 pour une plus ample description des modes pipeline et séquentiel.

Public

Déclare une variable en tant que variable publique, ce qui la rend accessible à la consultation dans le tableau « Public » de la CR1000.

La syntaxe est la suivante :
Public (liste de variables [dimensionnées] qui constituent le tableau Public)

Remarques
On peut utiliser plusieurs fois la déclaration « Public ».

Exemple de déclaration de variable publique

L'exemple montre l'utilisation de la déclaration « Public » :

```
Dim x( 3 ), y, z( 2, 3, 4 )
Public x, y, z
Public Dim x( 3 ), y, z( 2, 3, 4 )      'Dim est optionnel
Public x( 3 ),y, z( 2, 3, 4 )
Public w
```

SequentialMode

L'instruction SequentialMode est utilisée afin de configurer la centrale de mesure pour qu'elle effectue toutes les instructions décrites dans le programme d'acquisition, de façon séquentielle.

La syntaxe est la suivante :
SequentialMode

Remarques :
La centrale de mesure dispose de deux modes de fonctionnement : les modes séquentiels et « pipeline ». En mode séquentiel les instructions sont effectuées par la centrale de mesure de façon séquentielle, dans l'ordre où elles sont écrites dans le programme. En mode « pipeline » les tâches de mesure et les tâches de traitement sont effectuées de façon séparée et exécutées en même temps.

Le mode par défaut est le mode Pipeline. Cependant lorsque le programme de la centrale de mesure est compilé, la centrale de mesure analyse les instructions du programme et passe automatiquement du mode pipeline au mode séquentiel si le code le nécessite. On peut forcer la centrale de mesure à fonctionner en mode pipeline ou séquentiel en plaçant l'instruction appropriée au début du programme, avant l'instruction BeginProg.

Voir le paragraphe OV2.3 pour une plus ample description des modes pipeline et séquentiel.

Station Name

Donne un nom à la station.

La syntaxe est la suivante :

StationName *StaName*

Remarques :

StationName est utilisé afin de donner un nom de station à la centrale de mesure à partir du programme. Le nom de la station est affiché par LoggerNet et stocké dans les en-têtes de tableaux de sauvegarde (voir chapitre 2.4).

Sub, Exit Sub, End Sub

Ces instructions déclarent le nom et le code nécessaires à l'écriture d'un sous-programme (*subroutine*).

La syntaxe est la suivante :
Sub *SubName[(VariableList)]*
    [*statementblock*]
    [Exit Sub]
    [*statementblock*]
End Sub

La fonction Sub est constituée de ces parties :

Partie	Description
Sub	Marque le début du sous-programme
SubName	Donne un nom au sous-programme. Le nom du sous-programme (S*ubName*), ne peut pas être le même que celui d'une autre variable globalement reconnue dans le programme.
VariableList	Liste de variables qui sont passées au sous-programme quand il est appelé. La liste des variables de sous-programme à passer est optionnelle. Les sous-programmes peuvent fonctionner à partir des variables globales déclarées par les instructions Dim et Public. L'avantage de passer des variables est de faire en sorte que le sous-programme puisse fonctionner à partir de n'importe quelles variables du programme qui lui sont passées (voir exemple). Si la liste de variables de sous-programme est utilisée, alors les noms de variables utilisées pour cette liste ne devront pas être les même que les noms de variables, d'alias ou de constantes déclarées autre-part. Plusieurs variables sont séparées par des virgules. Lorsque le sous-programme est appelé, l'instruction call doit lister les variables du programme ou les valeurs à passer au sous-programme. Le nombre et l'ordre des variables / valeurs du programme présentes dans l'instruction call, doit concorder avec le nombre et l'ordre de la liste des variables de la déclaration de l'instruction Sub. Le fait de changer la valeur de l'une des variables de la liste à l'intérieur du sous-programme, change la valeur de la variable passée dans le sous-programme lors de la procédure d'appel. L'appel (call) peut passer des constantes ou des expressions qui évaluent des constantes (c'est à dire qui ne contiennent pas de variables), vers certaines des variables. Si une constante est passée, la « variable » vers laquelle elle est envoyée devient une constante et ne peut plus être modifiée par le sous-programme. Si des constantes sont passées, le sous-programme devra être écrit afin de ne pas chercher à modifier la valeur des « variables » vers lesquelles elles auront été passées.
Statementblock	N'importe quel groupement d'instructions qui est exécuté à l'intérieur du corps du sous-programme.
Exit Sub	Cela permet de sortir immédiatement du sous-programme. Le programme poursuit son chemin, à la suite de l'instruction qui a appelé le sous-programme. N'importe quel nombre d'instructions **Exit Sub** peuvent être utilisées dans un sous-programme.
End Sub	Cela marque la fin du sous-programme.

Un sous-programme est une procédure qui peut prendre des variables, faire des séries d'instructions, et changer les valeurs des variables. Cependant, un sous-programme ne peut pas être utilisé à l'intérieur d'une expression. Vous pouvez appeler un sous-programme en utilisant simplement son nom, suivi de la liste des variables. Reportez-vous à l'instruction « Call » pour voir comment appeler un sous-programme.

ATTENTION Les sous-programmes peuvent être récursifs ; cela implique qu'ils peuvent s'appeler les uns les autres afin d'effectuer une tâche précise. Mais cela peut conduire à des résultats surprenants.

Exemple de sous-programme (Subroutine)

```
'CR1000
'Déclare les variables utilisées dans le programme :
Public RefT, TC_C(4), TC_F(4), I

'Données enregistrées en °C :
DataTable (TempsC,1,-1)
   DataInterval (0,5,Min,10)
   Average (1,RefT,FP2,0)
   Average (4,TC_C(),FP2,0)
EndTable

'Même données enregistrées en °F :
DataTable (TempsF,1,-1)
   DataInterval (0,5,Min,10)
   Average (1,RefT,FP2,0)
   Average (4,TC_F(),FP2,0)
EndTable

'Sous-programme afin de convertir les températures de °C à °F
Sub ConvertCtoF (TmpC, TmpF)
   TmpF = TmpC*1.8 +32
EndSub

BeginProg
   Scan (1,Sec,3,0)
      'Mesure les températures (bornier + 4 thermocouples) en °C
      PanelTemp (RefT,250)
      TCDiff (TC_C(),4,mV2_5C,1,TypeT,RefT,True ,0,250,1.0,0)
      'Appel du tableau de sauvegarde pour les °C
      CallTable TempsC

      'Conversion des températures en °F en utilisant le sous-programme :

      'Le sous-programme est appelé à l'aide de l'instruction Call,
      'RefT est utilisée en tant que source et destination.
      Call ConvertCtoF(RefT, RefT)
      For I = 1 to 4
         'Appel du sous-programme sans l'instruction Call :
         ConvertCtoF(TC_C(I),TC_F(I))
      Next I

      CallTable TempsF
   NextScan
EndProg
```

Units

Ceci associe un nom d'unité de mesure, à un champ associé à une variable.

La syntaxe est la suivante :
Units *Variable = UnitName*

Remarques :
L'emploi de l'instruction « **Units** » permet d'assigner un nom d'unité de mesure, à une colonne. L'unité de mesure apparaît dans l'en-tête du fichier de données sauvegardées. Le nom de l'unité est contenu dans un champ texte, ce qui permet à l'utilisateur d'étiqueter les données. Quand l'utilisateur modifie les unités, le texte donné n'est pas vérifié par l'éditeur de programme de la CR1000.

Exemple

```
Dim TCTemp( 1 )
Units TCTemp( 1 ) = Deg_C
```

Chapitre 6. Déclarations du tableau de sauvegarde et instructions de traitement de sauvegarde

6.1 Déclaration du tableau de sauvegarde

DataTable (Name, TrigVar, Size)

output trigger modifier

export data destinations

output processing instructions

EndTable

L'instruction DataTable est utilisée pour définir / déclarer un tableau de sauvegarde. Le nom du tableau, la condition de basculement (*trigger*) et la taille du tableau occupé dans la RAM, sont fixés dans l'instruction DataTable. La déclaration du tableau doit être placée au début du code, avant l'instruction BeginProg. La déclaration du tableau débute par l'instruction **DataTable**, et se termine par l'instruction **EndTable**. A l'intérieur de la déclaration on donne des conditions de basculement pour la sauvegarde (ceci est optionnel avec par exemple DataEvent, DataInterval ou WorstCase), l'appareil de sauvegarde vers lequel on envoie les données (en option, soit par exemple DCP4 ou CardOut), et les instructions de sauvegarde qui décrivent le type de données du tableau de sauvegarde.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Name
Nom</td><td>Le nom du tableau de données. Le nom du tableau est limité à huit caractères.</td></tr>
<tr><td>TrigVar

Constante, Variable ou Expression</td><td>Le nom de la variable à tester pour la condition de basculement. Les modificateurs de condition de basculement ajoutent des conditions supplémentaires.

<table><tr><th>Valeur</th><th>Résultat</th></tr><tr><td>0
≠ 0</td><td>Ne pas effectuer de basculement
Basculer</td></tr></table></td></tr>
<tr><td>Size
Constante</td><td>La taille que l'on donne pour faire le tableau de données. Le nombre de jeu de données (enregistrements) pour lequel on prévoit de la place dans la mémoire. A chaque fois qu'une variable ou une condition de basculement prédéfinie est atteinte, une ligne (ou une colonne) de données est envoyée en sortie, avec le nombre de valeurs déterminées par les instructions de sauvegarde à l'intérieur du tableau. Ce type de données est appelé un enregistrement. Le nombre total d'enregistrements stockés est égal à la taille.

Note :
Si vous entrez une valeur négative, la taille attribuée au tableau pour cette valeur négative, sera la taille restante par rapport aux tableaux ayant une taille définie positivement (qui doivent être définis auparavant), ou bien sera répartie entre touts les tableaux ayant une valeur négative pour leur taille. L'algorithme de partitionnement essaye de faire en sorte que les tableaux soient remplis au même moment.</td></tr>
</table>

Exemple de Tableau de Données - voir au paragraphe 4.3 .

EndTable

Utilisé afin de marquer la fin du tableau de données.

Voir l'instruction **DataTable**.

6.2 Modifications des conditions de basculement (Trigger modifiers)

DataInterval (TintoInt, Interval, Units, Lapses)

Cette instruction est utilisée afin de fixer un intervalle de temps pour un tableau de sauvegarde de données. Cette instruction est insérée à l'intérieur de la déclaration du tableau de sauvegarde, à la suite de l'instruction DataTable, afin de définir un intervalle de temps fixe. Les tableaux à intervalle de temps fixé occupent moins de place en mémoire que les tableaux conditionnels car l'horodatage n'est pas sauvegardé à chaque enregistrement. Le temps de chaque enregistrement est calculé à partir du temps où a eu lieu le dernier enregistrement, et de l'intervalle de sauvegarde. L'instruction DataInterval n'influence pas la condition de basculement dans l'instruction DataTable. Si la condition de basculement n'est pas mise à une valeur positive de façon à ce qu'elle soit toujours vraie, cela devient une condition qui doit être atteinte en plus de la condition de temps définie, afin que les données soient enregistrées.

L'intervalle (**Interval**) détermine la fréquence avec laquelle les données sont enregistrées dans le tableau de sauvegarde. L'intervalle est synchronisé avec l'heure de la centrale de mesure. L'heure est conservée par la centrale en tant que durée écoulée depuis le 1er janvier 1990 à 0h00min00sec. Quand l'intervalle de temps est un multiple du temps écoulé, cela devient l'heure pour effectuer la sauvegarde (le MOD du temps écoulé devient = à 0). Si on entre la valeur 0 dans l'intervalle, il prend alors la valeur de l'intervalle d'exécution (*scan Interval*).

Le paramètre **TintoInt** (temps à l'intérieur de l'intervalle) permet à l'utilisateur de fixer le moment à l'intérieur de l'intervalle ou l'offset par rapport à l'heure de la centrale, auquel la sauvegarde s'effectuera ([temps écoulé + temps à l'intérieur de l'intervalle] MOD intervalle = 0). Si par exemple vous avez un temps à l'intérieur de l'intervalle qui est égal à 360 (**TintoInt**), et un intervalle (**Int**) égal à 720 en minutes (pour l'Unité - **Units**), alors la sauvegarde devrait se produire à 6h et 18h, avec un intervalle de 720 minutes (12 heures) calé sur minuit (00h00). On donne la valeur 0 au paramètre **TintoInt** si l'on souhaite garder la synchronisation par rapport à l'heure de la centrale de mesure.

Les données enregistrées à intervalle de temps constant permettent une gestion plus efficace de la mémoire car il n'est pas nécessaire d'horodater les mesures à chaque enregistrement. La CR1000 continue à enregistrer un horodatage, mais d'une façon espacée, environ à chaque 1K de mémoire utilisée pour le tableau. A chaque nouvel enregistrement l'horodatage est vérifié afin de s'assurer que l'intervalle de temps est correct. La centrale de mesure garde une trace des trous (*lapses*) et des discontinuités dans les données. Si un trou se produit, la CR1000 ajoute un marqueur de temps (*time stamp*) dans les données. Lorsque les données sont collectées, un marqueur de temps pourra donc être calculé et associé à chaque enregistrement.

Le marqueur de temps pour un trou de données occupe de la mémoire qui serait autrement utilisée pour les données. Bien que la CR1000 alloue un peu plus de place que ce qui est normalement nécessaire pour un tableau de données, il ne sera pas possible d'enregistrer la quantité d'enregistrement souhaitée s'il se produit beaucoup de trous. Le paramètre de « *Lapses* » indiqué dans l'instruction **DataInterval** permettra au programmateur d'allouer de la place supplémentaire pour le nombre de trous spécifiés. Ceci est utilisé dans les cas où on sait que le programme créé engendrera des trous. Si par exemple l'enregistrement des données est conditionné par une condition de basculement (trigger) présent dans l'instruction DataTable en plus de l'instruction DataInterval, les trous se produiront à chaque fois que la condition de basculement aura été fausse sur une durée de temps supérieure à celle de l'intervalle.

Pour que la mémoire soit utilisée de façon plus efficace, il faut toujours entrer un nombre de lapses supérieur ou égal à 1 même si on ne s'attend pas à avoir de trous. Le fait d'entrer « 0 » fera en sorte que chaque enregistrement sera associé à un marqueur de temps.

Le fait d'entrer une valeur négative dit à la CR1000 de ne pas garder trace des trous. Seul le marqueur de temps périodique (à peu près tous les 1K) est inséré.

<table>
<tr><th>Paramètres
& type de donnée</th><th>Entrée</th></tr>
<tr><td>TintoInt
Constante</td><td>Le temps à l'intérieur de l'intervalle (offset par rapport à l'intervalle), auquel seront enregistrées les données à sauvegarder. L'unité de temps est la même que l'unité de l'intervalle de temps.</td></tr>
<tr><td>Interval
Constante</td><td>Entrez la durée de l'intervalle, à laquelle les données doivent être enregistrées dans le tableau. L'intervalle est à définir en µs, ms, sec ou min, dans le paramètre d'Unités. On entre la valeur 0 si on souhaite que la sauvegarde s'effectue à chaque intervalle de scrutations.</td></tr>
<tr><td>Units (Unités)

Constante</td><td>Les unités pour le paramètre de temps sont indiquées ci-dessous. Seul l'instruction « PowerOff » utilise des heures ou des jours.
<table>
<tr><th>Code alphanumérique</th><th>Unité</th></tr>
<tr><td>USEC</td><td>Microseconde</td></tr>
<tr><td>MSEC</td><td>Milliseconde</td></tr>
<tr><td>SEC</td><td>Seconde</td></tr>
<tr><td>MIN</td><td>Minutes</td></tr>
</table></td></tr>
<tr><td>Lapses
Constante</td><td>A chaque nouvel enregistrement de donnée, l'horloge est interrogée afin de vérifier que l'intervalle entre 2 enregistrements est conforme à celui défini dans le programme. La centrale garde trace des trous ou des discontinuités dans les données.</td></tr>
</table>

OpenInterval

Lorsque l'instruction DataInterval est comprise dans un tableau de données, la CR1000 n'utilise que les valeurs présentes dans cet intervalle pour effectuer les calculs (par exemple moyenner, prendre le minimum ou le maximum). Lorsque les données sont sauvegardées à chaque intervalle, l'instruction de sauvegarde est ré-initialisée à chaque fois qu'une sauvegarde se produit. Pour s'assurer que les données d'un intervalle précédent ne sont pas prises en compte dans le calcul sauvegardé, les calculs sont ré-initialisés à chaque fois qu'un intervalle de sauvegarde est dépassé. (Un intervalle peut être dépassé parce que le tableau n'a pas été appelé ou qu'une condition de basculement n'a pas été atteinte.) La CR1000 ré-initialise les calculs la prochaine fois que le tableau est appelé après qu'un intervalle de sauvegarde ait été dépassé. Si ce prochain appel est basé sur le temps, il n'y aura pas de sauvegarde. La sauvegarde sera visible sur le prochain intervalle. (Si *Sample* est le seul type d'instruction de sauvegarde d'un tableau, les données seront présentes en sortie à chaque fois que le tableau est appelé et quel que soit l'intervalle car cette instruction échantillonne la valeur actuelle de la variable et n'effectue aucun calcul).

OpenInterval est utilisée afin de modifier un tableau basé sur un intervalle de temps, afin que les calculs effectué lors de la sauvegarde prennent en compte toutes les valeurs qui ont été présentes en mémoire d'entrée depuis la dernière fois au le tableau a donné lieu à un enregistrement. Les données seront sauvegardée lorsque le tableau sera appelé lors de son intervalle de sauvegarde (pour autant que les autres conditions sont respectées), qu'une sauvegarde ait eu lieu lors du précédent intervalle ou non.

Exemple avec OpenInterval :
Dans l'exemple suivant on mesure 5 thermocouples toutes les 500 msec. Chaque 10 Secondes et si le drapeau *Flag(1)* est en position vraie, la moyenne des températures de référence et celle des thermocouples sont sauvegardée. L'utilisateur peut faire basculer l'état du *Flag(1)* afin de déclencher ou non la sauvegarde. Si l'on ne met pas l'instruction OpenInterval, les premières moyennes enregistrées après que le Flag(1) soit activé, n'inclurai que les données présentes depuis les 10 dernières secondes d'intervalle de temps. C'est qui se produit par défaut, et ce que la plupart des utilisateurs souhaitent avoir. Avec l'instruction OpenInterval présente dans le programme (si on supprime l'apostrophe qui la fait considérer comme un commentaire), toutes les mesures prises pendant que le *Flag(1)* a été désactivé, seront prises en compte lors du premier enregistrement déclenché par l'activation du drapeau.

```
Const RevDiff 1         'Inversion en entrée pour annuler les offsets
Const Del 0             'On utilise le délai par défaut
Const Integ 250         'On utilise l'intégration à 250 µs
Public RefTemp          'On déclare la variable utilisée pour la température de référence
Public TC(5)            'On déclare la variable utilisée pour la mesure des thermocouples
Public Flag(8)
Units RefTemp=degC
Units TC=degC

DataTable (AvgTemp,Flag(1),1000)      'Sauvegarde lorsque le Flag(1)=vrai
        DataInterval(0,10,sec,10)     'Sauvegarde toutes les 10 sec(tant que  Flag(1)=vrai)
        'OpenInterval        'Si on enlève l'apostrophe cela prendra en compte les toutes les données de
l'intervalle où Flag(1) = Faux, dans le prochain enregistrement
        Average(1,RefTemp,IEEE4,0)
        Average(5,TC,IEEE4,0)
EndTable

BeginProg
        Scan(500,mSec,0,0)
                PanelTemp (RefTemp,250)
                TCDiff (TC(),5,mV2_5C,9,TypeT,RefTemp,RevDiff,Del,Integ,1,0)
                CallTable AvgTemp
        NextScan
EndProg
```

DataEvent (RecsBefore, StartTrig, StopTrig, RecsAfter)

Cette instruction est utilisée afin de fixer une condition de basculement pour débuter l'enregistrement de données et une autre condition de basculement pour arrêter d'effectuer des enregistrements dans un tableau. Le nombre d'enregistrements avant la condition de basculement de début et le nombre d'enregistrement après la condition de basculement d'arrêt peuvent aussi être fixées. Un marqueur de fichier (*filemark*, voir chapitre 8) est automatiquement enregistré dans le tableau et entre chaque événement.

<table>
<tr><th>Paramètre
& type de donnée</th><th>Entrée</th></tr>
<tr><td>RecsBefore
Constante</td><td>Le nombre d'enregistrements à enregistrer avant la condition de basculement de début (Start Trigger).</td></tr>
<tr><td>StartTrig
Variable ou Expression</td><td>La variable ou l'expression de test pour effectuer le basculement puis copier les enregistrements précédent le basculement vers le tableau de données et débuter l'enregistrement de chaque nouvel enregistrement.
Valeur | Résultat
0 | Ne pas basculer
≠ 0 | Basculer</td></tr>
<tr><td>StopTrig
Variable, Expression ou Constante</td><td>La variable, l'expression ou la constante à tester afin d'arrêter l'enregistrement dans le tableau. La CR1000 ne teste la condition de basculement pour l'arrêt qu'une fois que la condition de basculement pour le début est atteinte. Une contante différente de 0 (vraie) peut être utilisée afin d'enregistrer un nombre fixe d'enregistrements lorsque la condition de basculement pour le début est atteinte (nombre total d'enregistrements = PreTrigRecs+ 1 enregistrement pour le basculement +PostTrigRecs.). Zéro (faux) peut être entré si on souhaite enregistrer les données de façon continue une fois que la condition de basculement de début est atteinte.
Valeur | Résultat
0 | Ne pas basculer
≠ 0 | Basculer</td></tr>
<tr><td>RecsAfter
Constante</td><td>Le nombre d'enregistrements à enregistrer après que la condition de basculement d'arrêt se soit produite.</td></tr>
</table>

Exemple avec DataEvent :
Dans cet exemple 5 thermocouples de type T sont mesurés. La condition de basculement pour le début est pour la variable TCTemp(1) supérieure à 30 °C. La condition de basculement pour l'arrêt est lorsque la variable TCTemp(1) est inférieure à 29 °C. L'événement à enregistrer est constitué de 20 enregistrements précédent la condition de basculement de début, et 10 enregistrement suivant la condition de basculement d'arrêt.

```
Const RevDiff 1             'Inversion en entrée pour annuler les offsets
Const Del 0                 'On utilise le délai par défaut
Const Integ 0               'On n'utilise pas d'intégration
Public RefTemp              'On déclare la variable utilisée pour la température de référence
Public TC(5)                'On déclare la variable utilisée pour la mesure des thermocouples
Public Flag(8)
Units RefTemp=degC
Units TC=degC

DataTable (Event,1,1000)
        DataInterval(0,00,msec,10)          'On fixe l'intervalle d'échantillonnage identique à
                                            ' l'intervalle de scrutation
        DataEvent(20,TC(1)>30,TC(1)<29,10)      '20 enregistrements avant que  TC(1)>30,
                                                'après que TC(1)<29 en enregistrer 10 de plus
        Sample(1,RefTemp,IEEE4)             'Echantillonner la T° de référence
        Sample(5,TC,IEEE4)                  'Echantillonner les 5 températures de thermocouple
EndTable

BeginProg
        Scan(500,mSec,0,0)
                PanelTemp (RefTemp,250)
                TCDiff (TC(),5,mV2_5C,1,TypeT,RefTemp,RevDiff,Del,Integ,1,0)
                CallTable Event
        NextScan
EndProg
```

FillStop

Les tableaux de données sont, en standard, configurés pour mémoriser en boucle (*ring memory*) où une fois remplie, les données les plus récentes écrasent les plus anciennes. Si **FillStop** est incéré dans une déclaration de tableau, cela configure le tableau en mode remplissage et arrêt (*fill and stop*). Une fois que le tableau de données est plein, plus aucune donnée n'est enregistrée jusqu'à ce que le tableau soit ré-initialisé. Le tableau peut être ré-initialisé (toutes les données sont effacées) à partir de l'instruction **ResetTable** exécutée par le programme.

Exemple avec FillStop :

```
DataTable (Temp,1,2000)
        DataInterval(0,100,msec,10)
        FillStop            ' Le tableau arrêtera de stocker des enregistrements après le 2000ème.
        Average(1,RefTemp,fp2,0)
        Average(6,TC(1),fp2,0)
EndTable
```

WorstCase (TableName, NumCases, MaxMin, Change, RankVar)

Permet de sauvegarder les évènements les plus significatifs ou du "pire des cas" (worst-case) dans des tableaux séparés.

Un tableau de données dont la taille est destinée à enregistrer un évènement, est créé. Ce tableau agit en tant que mémoire tampon d'événement. Chaque événement qui se produit est enregistré dans ce tableau. Ce tableau peut utiliser l'instruction DataEvent ou une autre condition afin de déterminer à quel moment un événement est enregistré. L'importance significative d'un événement est déterminée par un algorithme présent dans le programme, et un rang numérique correspondant à l'événement, est enregistré dans une variable.

WorstCase crée autant de clones du tableau spécifié, que le nombre de cas pour lequel l'instruction doit garder les données. Quand l'instruction WorstCase est exécuté, elle vérifie la variable de rang; si la valeur de la variable est un « pire des cas » qui soit nouveau, la donnée présente dans le tableau de l'évènement remplacera la donnée du tableau clone qui contient l'évènement le moins signifiant actuellement stocké.

Un tableau additionnel est crée, *nom*WC (par exemple evenementWC); il contient la valeur du rang de la variable pour chacun des tableaux de "pire des cas", ainsi que l'heure à laquelle ce tableau a été enregistré.

WorstCase doit être utilisée avec un enregistrement du tableau de données effectué sur le SRAM de la CR1000. Cela ne fonctionnera pas si le tableau d'événement est enregistré sur la carte CF.

Bien que WorstCase agisse en tant que modificateur de condition de basculement (*Trigger Modifier*) et en tant que déclaration de tableau (en créant un tableau de données "clone"), l'instruction est écrite à l'intérieur du programme afin d'appeler le tableau de "pire des cas" (voir l'exemple).

<table>
<tr><th>Paramètre
& Type de donnée</th><th>Entrée</th></tr>
<tr><td>TableName
Nom</td><td>Le nom du tableau de donnée à cloner. La longueur de ce nom devrait être de 6 caractères ou moins, afin que le nom complet des tableaux « worst case » soient retenus lorsqu'ils sont collectés (voir NumCases).</td></tr>
<tr><td>NumCases</td><td>Le nombre de « pire » des cas à stocker. C'est le nombre de clones du tableau de donnée qu'il faut créer. Les tableaux clonés utilisent le nom du tableau à cloner (jusqu'à 6 caractères) plus un nombre à 2 digits (par exemple Evnt01, Evnt02, Evnt03, …). Les nombres donnent aux tableaux un nom unique; ils n'ont pas de lien avec l'ordonnancement des évènements.</td></tr>
<tr><td>MaxMin
Constante</td><td>Un code qui spécifie si les évènements à sauvegarder sont les mini. ou maxi.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>Min, sauvegarde les évènements associés avec le rang minimum. Par exemple on garde trace de la variable RankVar associée avec chaque évènement stocké. Si une nouvelle valeur de RankVar est inférieure au maximum précédent alors on copie l'évènement à la place de l'évènement maximum précédent.</td></tr>
<tr><td>1</td><td>Max, sauvegarde les évènements associés avec le rang maximum. Par exemple on garde trace de la variable RankVar associée avec chaque évènement stocké. Si une nouvelle valeur de RankVar est supérieure au minimum précédent alors on copie l'évènement à la place de l'évènement minimum précédent.</td></tr>
</table></td></tr>
<tr><td>Change
Constante</td><td>Le changement minimum qui doit se produire sur la variable RankVariable avant qu'un nouveau « pire des cas » ne soit sauvegardé.</td></tr>
<tr><td>RankVar
Variable</td><td>La variable par rapport à laquelle on va ordonner les évènements.</td></tr>
</table>

Exemple avec WorstCase :
Ce programme illustre l'instruction **WorstCase**. Cinq thermocouples de type T sont mesurés. L'événement est similaire à celui de l'exemple de l'instruction DataEvent; la condition de basculement pour le début de l'évènement de données est valable pour la variable TC(1) supérieure à 30° C. Dans cet exemple cependant, la condition de basculement pour l'arrêt est tout de suite activée. Cela est effectué afin de donner une taille fixe à l'événement qui peut être dupliqué dans les tableaux de "pire des cas". Pour utiliser l'instruction de pire des cas avec des évènements de durée variable, la taille du tableau d'évènement doit être sélectionnée afin d'être au plus proche de la durée maximum à laquelle on s'attend (ou dont on a besoin). L'événement est constitué de 20 enregistrements avant la condition de basculement de début et continue avec 100 enregistrements suivant la condition de basculement de début.

Le critère de rang est le nombre de lectures qui suivent la condition de basculement pour laquelle TC(1) reste au dessus de 30°C. Plus ne nombre est grand, plus l'événement a duré longtemps (est « pire »).

```
'CR1000 Series Datalogger

Const NumCases = 5      'Nombre de "pire des cas" à sauvegarder
Const Max = 1           'Une constante afin d'indiquer le rang des valeurs maxi. Dans le pire des cas.

Public RefTemp                  'Déclare la variable utilisée pour la température de référence
Public TC(5)                    'Déclare la variable utilisée pour les mesures de thermocouple
Public I, NumAbove30            'Déclare l'index et la variable de rang

Units RefTemp = degC
Units TC = degC

DataTable (Evnt,1,125)
    DataInterval(0,00,msec,10)          'Fixe l'intervalle d'échantillonnage égal à la scrutation
    DataEvent(20,TC(1)>30,-1,100)       '20 enregistrements avant que TC(1)>30,
                                        '100 enregistrements après que TC(1)>30
    Sample(1,RefTemp,IEEE4)             'Echantillonner la température de référence
    Sample(5,TC,IEEE4)                  'Echantillonner les 5 températures de thermocouples
EndTable

BeginProg
    Scan(500,mSec,10,0)
        PanelTemp (RefTemp,250)
        TCDiff(TC(),5,mV2_5C,1,TypeT,RefTemp,True,0,250,1,0)
        CallTable Evnt
        IF Evnt.EventEnd(1,1) then                  'Vérifie si un évènement vient de se terminer
            I=100                                   'Initialise l'index
            NumAbove30=0                            'Met à zéro la variable de rang
            Do  'Loop through the Event table
                NumAbove30=NumAbove30+1             'compte le  # de fois où TC(1)>30
                I=I-1
            Loop While I>0 and Evnt.TC(1,I)>=30     'Sort de la boucle à la fin ou quand TC(1)<30
            WorstCase(Evnt,NumCases,Max,0,NumAbove30)       'Vérifie le "pire des cas"
        EndIf
    NextScan
EndProg
```

6.3 Instructions d'export de données

CardOut (StopRing, Size)

Cette instruction est utilisée pour envoyer les données de sauvegarde au module pour cartes Compact Flash, le CFM100 ou le NL115. Cette instruction crée un tableau de données sur la carte CF. CardOut doit être entrée dans chaque déclaration de tableau qui doit sauvegarder les données sur la carte CF.

Paramètre *& Type de donnée*	**Entrée**
StopRing *Constante*	Un code pour spécifier si le tableau de données sur la carte CF est en mode remplissage et arrêt ou mémoire en boucle (*fill and stop / ring mode*). **Valeur** / **Résultat** 0 — *Ring* — Mémoire en boucle (données les plus anciennes écrasées) 1 — *Fill and Stop* — Mémoire en remplissage et arrêt
Size *Constante*	La taille (*size*) que doit avoir le tableau. Le nombre de jeux de données (enregistrements) pour lesquels on alloue de la taille mémoire sur la carte. A chaque fois qu'une variable ou un intervalle de basculement se produisent, une ligne (ou un jeu) de données est sauvegardé avec le nombre de valeurs déterminées par les instructions de sauvegarde présentes dans le tableau. Cette donnée est appelée un enregistrement. **Note** Si on entre un nombre négatif, toute la place disponible restante (après avoir créé les tableaux à taille fixée) sera allouée à ce tableau, ou sera partitionnée entre les tableaux ayant des valeurs négatives pour leur taille. L'algorithme de partitionnement essaye d'avoir les tableaux remplis à la même vitesse.

DSP4 (FlagVar, Rate)

Envoyer les données au DSP4. Si l'instruction apparaît à l'intérieur d'un tableau de données DataTable, le DSP4 peut afficher le champ de ce tableau, sinon les variables Public sont utilisées par le DSP4. L'instruction ne peut être utilisée qu'une seule fois dans le programme; de ce fait seules les variables Public ou un seul tableau de données pourront être affiché sur le DSP4.

Paramètre *& Type de donnée*	**Entrée**
FlagVar *Ligne de données*	La variable en ligne de données à utiliser, afin que les 8 drapeaux (*flags*) puissent être affichés et changés d'état par le DSP4. Une valeur de 0 = bas (*low*); ≠0 = haut (*high*). Si la ligne de données est dimensionnée à moins de 8 éléments, le DSP4 ne fonctionnera qu'avec les drapeaux déclarés dans la ligne de donnée. La ligne de données utilisée pour les drapeaux dans les affichages en temps réel est Flag ().
Rate *Constante*	Fréquence (en millisecondes) à laquelle envoyer les nouvelles valeurs au DSP4.

Exemple avec DSP4 :

```
DSP4 (Flag( ), 200)
```

On utilise Flag( ) afin d'utiliser les boutons, et rafraîchit l'affichage du DSP4 toutes les 200 msec (5 fois par seconde).

GOESData (Dest, Table, TableOption, BufferControl, DataFormat)

L'instruction GOESData est utilisée afin de transmettre les données vers le transmetteur de données satellite SAT HDR GOES. L'instruction GOESData n'est pas incorporée à la déclaration du tableau de données, elle est incluse au programme, typiquement dans la boucle de scrutation (*scan*).

Le transfert de données vers le transmetteur peut advenir au travers du port CS I/O de la centrale de mesure uniquement. L'instruction GOESData a les paramètres suivants :

NOTE Lorsque la centrale de mesure envoie une commande, les tâches de traitement suivantes ne seront effectuées qu'une fois qu'une réponse sera reçue du transmetteur HDR GOES.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Variable ou
Ligne de données</td><td>La variable qui contient un code de résultat pour la transmission. Les codes sont :
<table>
<tr><th>Code de Résultat</th><th>Description</th></tr>
<tr><td>0</td><td>Commande exécutée avec succès</td></tr>
<tr><td>2</td><td>Temporisation dépassée pour attendre le caractère STX provenant du transmetteur, après l’adressage SDC</td></tr>
<tr><td>3</td><td>Mauvais caractère reçu après l’adressage SDC</td></tr>
<tr><td>4</td><td>Autre chose que “ACK” a été retourné quand la commande “select data buffer” (sélectionner la mémoire tampon de données) a été exécutée</td></tr>
<tr><td>5</td><td>Temporisation dépassée pour attendre le caractère ACK</td></tr>
<tr><td>6</td><td>Le port CS I/O n’est pas disponible</td></tr>
<tr><td>7</td><td>Message aléatoire d’erreur de transmission (peut être qu’il n’y a pas de données dans la mémoire tampon)</td></tr>
</table></td></tr>
<tr><td>Table
Nom du tableau</td><td>Le tableau de données à partir duquel les enregistrements devraient être transmis.</td></tr>
<tr><td>TableOption
Constante</td><td>TableOption indique quels enregistrements devraient être envoyés à partir du tableau de données.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Envoyer tous les enregistrements depuis la dernière exécution</td></tr>
<tr><td>1</td><td>Envoyer uniquement les enregistrements sauvegardés le plus récemment dans le tableau</td></tr>
</table></td></tr>
<tr><td>BufferControl
Constante</td><td>Le paramètre BufferControl spécifie quelle mémoire tampon devrait être utilisée (aléatoire ou auto-temporisé - random or self-timed) ainsi que si les données devraient être écrasées ou ajoutées aux données existantes. Les données enregistrées dans la mémoire tampon auto-temporisée ne sont transmises que lorsqu’il y a une fenêtre de temps prédéterminée. Les données sont effacées de la mémoire tampon du transmetteur après chaque transmission. Les données de la mémoire tampon aléatoire sont transmises immédiatement après que le seuil soit dépassé. La transmission est répétée de façon aléatoire afin de s’assurer qu’elle soit bien reçue.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Ajouter à la mémoire tampon auto-temporisée</td></tr>
<tr><td>1</td><td>Ecraser la mémoire tampon auto-temporisée</td></tr>
<tr><td>2</td><td>Ajouter à la mémoire tampon aléatoire</td></tr>
<tr><td>3</td><td>Ecraser la mémoire tampon aléatoire</td></tr>
<tr><td>9</td><td>Nettoyer la mémoire tampon aléatoire</td></tr>
</table></td></tr>
<tr><td>DataFormat
Constant</td><td>Le paramètre DataFormat spécifie le format de la donnée envoyée au transmetteur
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Format FP2 de CSI; 3 octets (bytes) par point de donnée</td></tr>
<tr><td>1</td><td>ASCII à virgule flottante; 7 octets (bytes) par point de donnée</td></tr>
<tr><td>2</td><td>Entier binaire à 18-bit; 3 octets (bytes) par point de donnée, les nombre à la droite de la décimale sont tronqués</td></tr>
<tr><td>3</td><td>RAWS7; 7 points de donnée :
<table>
<tr><th>Points de donnée</th><th>Description</th></tr>
<tr><td>1</td><td>Pluviométrie totale en “inches”, format = xx.xxx</td></tr>
<tr><td>2</td><td>Vitesse du vent en “MPH”, format = xxx</td></tr>
<tr><td>3</td><td>Vecteur moyen de la direction du vent en degrés, format = xxx</td></tr>
<tr><td>4</td><td>Température de l’air en °F, format = xxx</td></tr>
<tr><td>5</td><td>Pourcentage d’HR, format = xxx</td></tr>
<tr><td>6</td><td>Température du combustible en °F , format = xxx</td></tr>
<tr><td>7</td><td>Tension batterie en V CC, format = xx.x</td></tr>
</table></td></tr>
<tr><td>4</td><td>ASCII à décimale pré-positionnée de type xxx.x</td></tr>
<tr><td>5</td><td>ASCII à décimale pré-positionnée de type xx.xx</td></tr>
<tr><td>6</td><td>ASCII à décimale pré-positionnée de type x.xxx</td></tr>
<tr><td>7</td><td>ASCII à décimale pré-positionnée de type xxx</td></tr>
<tr><td>8</td><td>ASCII à décimale pré-positionnée de type xxxxx</td></tr>
</table></td></tr>
</table>

GOESGPS (GoesArray1(6), GoesArray2(7))

L'instruction GOESGPS est utilisée afin de stocker dans des lignes de données variables, les données du GPS provenant du satellite.

La syntaxe est :
GOESGPS (GoesArray1(6), GoesArray2(7))

Remarques :
L'instruction GOESGPS donne pour réponse deux lignes de données. La première ligne de donnée, qui doit avoir une dimension de 6 éléments, contient le code de résultat indiquant si l'instruction a été exécutée avec succès, suivie par les informations de positionnement global.

Les codes de résultat sont ceux qui suivent :

Code	Description
0	Commande exécutée avec succès
2	Temporisation dépassée pour attendre le caractère STX provenant du transmetteur, après l'adressage SDC
3	Mauvais caractère reçu après l'adressage SDC
4	Autre chose que "ACK" a été retourné quand la commande "select data buffer" (sélectionner la mémoire tampon de données) a été exécutée
5	Temporisation dépassée pour attendre le caractère ACK
6	Le port CS I/O n'est pas disponible; GOES n'est pas connecté
7	ACK n'est pas renvoyé après la commande d'ajout ou re remplacement des données

Les valeurs des données du GPS sont de la forme suivante :

Valeur	Description
Temps - *Time*	Secondes depuis le 1er janvier 2000
Latitude	Fraction de degrés; résolution de 100 nanodegrés
Longitude	Fraction de degrés; résolution de 100 nanodegrés
Altitude - *Elevation*	Nombre signé à 32-bit, en centimètres
Variation magnétique - *Magnetic Variation*	Fraction de degrés; résolution de 1 millidegré

La seconde ligne de données, qui doit avoir une taille de 7 éléments, contient les valeurs de temps suivantes : année, mois, jour, heure (GMT), minutes, secondes, microsecondes.

GOESSetup (ResultCode, PlatformID, MsgWindow, STChannel, STBaud, RChannel, RBaud, STInterval, STOffset, RInterval)

L'instruction GOESSetup est utilisée afin de programmer le transmetteur GOES afin qu'il communiqué avec le satellite.

La Syntaxe est :
GOESSetup (ResultCode, PlatformID, MsgWindow, STChannel, STBaud, RChannel, RBaud, STInterval, STOffset, RInterval)

Remarques :
Etant donné que le but de cette instruction est de configurer le transmetteur pour la communication, elle ne devra être exécutée qu'une seule fois à l'intérieur du programme de la centrale de mesure. Les informations pour tous les paramètres de cette instruction sont fournies par NESDIS.

Reportez-vous à l'éditeur CRBasic ou au manuel de SATHDRGOES afin d'avoir plus de détails sur cette instruction.

GOESStatus (Dest, StatusCommand)

L'instruction GOESStatus est utilisée afin de demander les informations d'état et de diagnostique en provenance du transmetteur pour satellite SAT HDR GOES.

NOTE Lorsque la centrale de mesure envoie une commande, les tâches de traitement supplémentaires ne seront effectuées qu'après que le transmetteur HDR GOES ait envoyé une réponse.

<table>
<tr><th>Paramètre
& Type de donnée</th><th colspan="3">Entrée</th></tr>
<tr><td>Dest
Ligne de données</td><td colspan="3">Une ligne de données qui contiendra les codes de résultat renvoyés par le transmetteur. La taille de la ligne de données est déterminée par l'option choisie dans StatusCommand.
Code | Description
0 | Commande exécutée avec succès
1 | Erreur de Checksum dans la réponse
2 | Temporisation dépassée pour attendre le caractère STX provenant du transmetteur, après l'adressage SDC
3 | Mauvais caractère (différent de STX) reçu après l'adressage SDC
4 | En train de recevoir un « NAK »
5 | Temporisation dépassée pour attendre le caractère ACK
6 | Le port CS I/O n'est pas disponible
7 | Message aléatoire d'erreur de transmission (peut être qu'il n'y a pas de données)
9 | Commande invalide</td></tr>
<tr><td>StatusCommand
Constante</td><td colspan="3">StatusCommand spécifie le type d'information demandée par le transmetteur.
Code | Description | Taille de la ligne de donnée nécessaire
0 | Temps réel - Read time | 4
1 | Etat - Status | 13
2 | Etat du dernier message | 14
3 | Transmet un message aléatoire | 1
4 | Lire le registre des erreurs | 10
5 | Ré-initialise le registre des erreurs | 1
6 | Remet le transmetteur en mode "en ligne" | 1</td></tr>
</table>

6.4 Instructions de sauvegarde

Average (Reps, Source, DisableVar)

Cette instruction permet de stocker la valeur moyenne à chaque intervalle de sauvegarde spécifié, pour la variable source ou chaque élément mentionné dans la ligne de données spécifiée.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Reps
Constante</td><td>Le nombre de moyennes à calculer. Quand le nombre de répétitions est supérieur à « 1 », la source doit être une ligne de données.</td></tr>
<tr><td>Source
Variable</td><td>Comprend le nom de la variable qui doit être moyennée.</td></tr>
<tr><td>DataType
Constante</td><td>Un code afin de spécifier le format d’enregistrement des données.
<table>
<tr><th>Code</th><th>Format des données</th></tr>
<tr><td>IEEE4</td><td>IEEE à 4 byte et virgule flottante</td></tr>
<tr><td>FP2</td><td>Format de Campbell Scientific à 2 byte et virgule flottante</td></tr>
</table></td></tr>
<tr><td>DisableVar
Constante, Variable, ou Expression</td><td>Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Par exemple, avec une instruction de moyenne, et si la variable de désactivation est « différent de 0 » (≠0), la valeur actuelle de la variable ne sera pas inclus dans la moyenne. La moyenne qui sera éventuellement sauvegardée, sera la moyenne des valeurs en entrée qui se sont produites lorsque la variable de désactivation portait la valeur 0.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On traite l’espace mémoire du moment</td></tr>
<tr><td>≠ 0</td><td>On ne traite pas l’espace mémoire du moment</td></tr>
</table></td></tr>
</table>

Covariance (NumVals, Source, DataType, DisableVar, NumCov)

Calcule la covariance au cours du temps, des valeurs présentes dans une ligne de données. La covariance de X et Y est calculée selon :

$$Cov(X,Y) = \frac{\sum_{i=1}^{n}\left(X_i \cdot Y_i\right)}{n} - \frac{\sum_{i=1}^{n} X_i \cdot \sum_{i=1}^{n} Y_i}{n^2}$$

où n est le nombre de valeurs prises en compte dans le calcul, sur l’intervalle de sauvegarde, et où X_i et Y_i sont des valeurs individuelles de X et Y.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>NumVals
Constante</td><td>Le nombre d’éléments de la ligne de sauvegarde à inclure dans les calculs de covariance.</td></tr>
<tr><td>Source
Variable, Ligne de donnée</td><td>La ligne de donnée variable qui contient les valeurs à partir desquelles on calcule les covariances. Si les calculs de covariances doivent débuter à un certain endroit de la ligne de données qui ne soit pas le premier élément, assurez-vous que vous indiquez l’élément de départ dans la variable de source (par exemple X(3)).</td></tr>
<tr><td>DataType
Constante</td><td>Un code pour sélectionner le format d’enregistrement des données.<table><tr><th>Code Alphanumérique</th><th>Format de donnée</th></tr><tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr><tr><td>FP2</td><td>Format Campbell Scientific à 2 byte et virgule flottante</td></tr></table></td></tr>
<tr><td>DisableVar
Constante Variable, or, Expression</td><td>Une valeur différente de zéro désactivera le traitement de sauvegarde intermédiaire. Quand cette variable est ≠0, la valeur actuelle n’est pas comprise dans la covariance.<table><tr><th>Valeur</th><th>Résultat</th></tr><tr><td>0</td><td>On traite l’espace mémoire du moment</td></tr><tr><td>≠ 0</td><td>On ne traite pas l’espace mémoire du moment</td></tr></table></td></tr>
<tr><td>NumCov
Constante</td><td>Le nombre de covariances à calculer. Le nombre maximum de covariances est de Z/2*(Z+1). Où Z= NumVals. Si X(1) est le premier élément spécifié pour la ligne de données, les covariances sont calculées et sauvegardées dans l’ordre suivant : X_Cov(1)…X_Cov(Z/2*(Z+1)) = Cov[X(1),X(1)], Cov[X(1),X(2)], Cov[X(1),X(3)], … Cov[X(1),X(Z)], Cov[X(2),X(2)], Cov[X(2),X(3)], … Cov[X(2),X(Z)], … Cov[X(Z),X(Z)]. Les premiers “NumCov” de ces covariances possibles sont sauvegardés.</td></tr>
</table>

FFT (Source, DataType, N, Tau, Units, Option)

L’instruction FFT effectue une Transformée de Fourier (Fast Fourier Transform) sur une série de mesures espacées dans le temps et sauvegardées dans une ligne de données. Elle peut aussi effectuer une transformée inverse de Fourrier, pour générer une série temporelle à partir du résultat d’une FFT. En fonction de l’option choisie pour le résultat de l’instruction, celui-ci peut être : 0) La partie réelle et imaginaire de la FFT; 1) Spectre d’amplitude ; 2) Spectre d’amplitude et de phase; 3) Spectre de puissance; 4) La densité de la puissance spectrale (*Power Spectral Density1, PSD*); ou 5) FFT inverse.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Source
Variable</td><td>Le nom de la ligne de donnée variable qui contient les données en entrée pour le FFT.</td></tr>
<tr><td>DataType
Constante</td><td>Un code pour sélectionner le format d’enregistrement des données.
<table>
<tr><th>Code Alphanum.</th><th>Code Num.</th><th>Format de donnée</th></tr>
<tr><td>IEEE4</td><td>24</td><td>IEEE à 4 octets et virgule flottante</td></tr>
<tr><td>FP2</td><td>7</td><td>Campbell Scientific 2 byte floating point</td></tr>
</table></td></tr>
<tr><td>N
Constante</td><td>Nombre de points de la série temporelle originale. Le nombre de points doit être une puissance de 2 (par exemple 512, 1024, 2048, etc.).</td></tr>
<tr><td>Tau
Constante</td><td>L’intervalle d’échantillonnage de la série temporelle.</td></tr>
<tr><td>Units
Constante</td><td>Les unités pour le paramètre Tau.
<table>
<tr><th>Code Alphanumérique</th><th>Unités</th></tr>
<tr><td>USEC</td><td>microsecondes</td></tr>
<tr><td>MSEC</td><td>millisecondes</td></tr>
<tr><td>SEC</td><td>secondes</td></tr>
<tr><td>MIN</td><td>minutes</td></tr>
</table></td></tr>
<tr><td>Options
Constante</td><td>Un code pour déterminer quelles valeurs calculer en résultat final.
<table>
<tr><th>Code</th><th>Résultat</th></tr>
<tr><td>0</td><td>FFT. Le résultat est de N/2 points de mesure complexes, c’est à dire la partie réelle et imaginaire de la FFT. La première paire est le composant DC et la seconde est le composant Niquist. La première paire est une exception car elle les composants DC et Niquist n’ont pas de partie imaginaire.</td></tr>
<tr><td>1</td><td>Spectre d’amplitude. Le résultat est N/2 magnitudes. Avec Acos(wt); A est la magnitude.</td></tr>
<tr><td>2</td><td>Spectre d’amplitude et de phase. Le résultat est N/2 paires de magnitude et de phase; avec Acos(wt - ϕ); A est l’amplitude, ϕ est la phase ($-\pi$,π).</td></tr>
<tr><td>3</td><td>Spectre de puissance. Le résultat est N/2 valeurs normalisées afin de donner un spectre de puissance. Avec Acos(wt - ϕ), la puissance est A^2 / 2. La somme des N/2 valeurs fournit la puissance totale du signal de la série temporelle.</td></tr>
<tr><td>4</td><td>La densité de la puissance spectrale (PSD). Le résultat est N/2 valeurs normalisées afin de donner une densité de puissance spectrale (puissance par hertz). La puissance spectrale multipliée par T = N*tau fournit la PSD. L’intégrale de la PSD sur une largeur de bande (bandwidth) fournit le total de la puissance sur cette bande. A noter que la largeur de bande de chaque valeur est de 1/T hertz.</td></tr>
<tr><td>5</td><td>FFT inverse. L’entrée est N/2 nombres complexes, organisés de la même façon que les résultats de l’option 0, que l’on considère comme la transformée d’une série temporelle réelle. Le résultat est une série temporelle dont la FFT aurait pour résultat les valeurs de la ligne de données fournie en entrée.</td></tr>
</table></td></tr>
</table>

T = N*tau : la longueur, en secondes, de la série temporelle.

Champs de calcul (*Processing field*) : "FFT,N,tau,option". Les marques sur l'axe des X sont espacées de 1/(N*tau) Hertz. N/2 valeurs, ou paires de valeurs, sont calculées en sortie, selon le code choisi en option.

Détails normalisés :

Résultat complexe d'une FFT i, i = 1 .. N/2: ai*cos(wi*t) + bi*sin(wi*t).

$wi = 2\pi(i-1)/T$.
$\phi i = atan2(bi,ai)$ (4 quadrant arctan)
$Puissance(1) = (a1^2 + b1^2)/N^2$ (DC)
$Puissance(i) = 2*(ai^2 + bi^2)/N^2$ (i = 2..N/2, AC)
$PSD(i) = Puissance(i) * T = Puissance(i) * N * tau$
$A1 = sqrt(a1^2 + b1^2)/N$ (DC)
$Ai = 2*sqrt(ai^2 + bi^2)/N$ (AC)

Notes :

- La puissance est indépendante du temps d'échantillonnage (1/tau) et du nombre d'échantillons (N).
- Le PSD est proportionnel à la longueur de la période d'échantillonnage (T=N*tau), tant que la largeur ("*width*") de chaque intervalle (*bin*) est 1/T.
- La somme des intervalles CA(en excluant le CC) du spectre de puissance, est la Variance (puissance CA) de la série temporelle. Le facteur de 2 dans le calcul de Puissance(i) est du aux séries de puissances qui sont minorées à propos de la fréquence Niquist N/(2*T); seule la moitié de la puissance est représentée dans les intervalles de la FFT en dessous de N/2, à l'exception du CC. De ce fait le CC n'a pas ce facteur de 2.
- L'option de FFT inverse considère que la ligne de données fournie en entrée est le résultat d'une transformation de série temporelle. Le filtrage est effectué en prenant une FFT sur le jeu de données, en mettant certains intervalles de fréquence à la valeur zéro, puis en prenant la FFT inverse. L'interpolation est effectuée en prenant une FFT, en remplissant les résultats de zéros, puis en prenant la FFT inverse de la ligne de données la plus large. La résolution dans le domaine temporel est accrue par le rapport de la taille de la FFT remplie de zéros, sur la FFT non remplie de zéros. Ceci peut être utilisé afin d'augmenter la résolution d'un minimum ou d'un maximum, tant que l'utilisation des alias est évitée.

Exemple de FFT :

```
Const SIZE_FFT 16
CONST PI 3.141592654
Const CYCLESperT 2
Const AMPLITUDE 3
Const DC 7
Const OPT_FFT 0
CONST PI 3.141592654

dim i
public x(SIZE_FFT),y(SIZE_FFT)

DataTable(Amp,1,1)
  fft(x,fp2,SIZE_FFT,10 msec,1)
EndTable
DataTable(AmpPhase,1,1)
  fft(x,fp2,SIZE_FFT,10 msec,2)
EndTable
DataTable(power,1,1)
  fft(x,fp2,SIZE_FFT,10 msec,3)
EndTable
DataTable(PSD,1,1)
  fft(x,fp2,SIZE_FFT,10 msec,4)
EndTable
DataTable(FFT,1,1)
  fft(x,IEEE4,SIZE_FFT,10 msec,0)
EndTable
DataTable(IFFT,1,1)              'FFT inverse
  fft(y,IEEE4,SIZE_FFT,10 msec,5)
EndTable

BeginProg

Scan(10, msec,0,SIZE_FFT)
  i=i+1
  X(i) = DC + Sin(PI/8+2*PI*CYCLESperT*i/SIZE_FFT) * AMPLITUDE + Sin(PI/2+PI*i)
Next Scan

CallTable(Amp)
CallTable(AmpPhase)
CallTable(Power)
CallTable(PSD)
CallTable(FFT)
for i = 1 to SIZE_FFT          ' renvoyer le résultat vers y()
  y(i) = FFT.x_fft(i,1)
next
CallTable(IFFT)              ' on inverse, le résultat est le même que x()

EndProg
```

FieldNames ("Fieldname1:Description1,Fieldname2:Description2…")

L'instruction **Fieldname** doit être positionnée immédiatement après l'instruction de sauvegarde pour laquelle les noms sont créés. Les noms de champs sont entrés sous la forme « Fieldname :Description ». Les noms de champs et les descriptions doivent être séparés par « : », et la chaîne entière doit être comprise entre des guillemets.

Le nom des champs (*fieldname*) est le nom qui sera utilisé pour le champ. Il est limité à 19 caractères. Les éléments de description (qui sont optionnels) donnent la possibilité à l'utilisateur de donner plus d'information au sujet du champ. Dans le fichier de données, la description est comprise dans l'en-tête du fichier, en dessous du nom de champs, et à côté de la description du paramètre de calcul. Le nombre maximum de caractères de la Description dépend du descriptif qui est automatiquement généré par le type d'instruction de calcul. Le nombre maximum pour ces deux descriptifs est de 65 caractères (guillemets, espaces et autres caractères compris). Par exemple, la description du type de calculs et de la description du champ pour une instruction « Sample » pourrait être :

Smp,:"Ceci est un echantillon de temperature de l'air"

Le terme « Smp,: » ainsi que les guillemets ouvrant et fermant la description utilisent déjà 7 caractères. Il ne reste donc plus que 58 caractères de disponibles.

Si une instruction de sauvegarde génère plusieurs champs, des noms individualisés peuvent être entrés pour chacun, ou une ligne de données peut être utilisée. Des noms individuels devront être séparés par des virgules. Si une ligne de données est utilisée, son nom et sa dimension doivent être spécifiés (par exemple « Temp(4) » pour une ligne de données avec 4 noms de champs allant de Temp(1) à Temp(4)). A noter qu'une expression qui a pour résultat une constante peut aussi être utilisée pour définir la dimension d'une ligne de données. Lorsque le programme est compilé, la centrale de mesure déterminera combien de champs sont créés. Si la liste de noms est supérieure aux nombres de champs, les noms supplémentaires sont ignorés. Si le nombre de champs est supérieur au nombre de descriptifs texte entrés par l'instruction *Fieldnames*, les noms par défaut seront utilisés pour les champs restant.

Exemples

```
Sample(4, Temp(1),IEEE4)
Fieldnames "IntakeT, CoolerT, PlenumT, ExhaustT"
```

Les 4 variables de la ligne de variable de température, sont stockées dans le tableau de sauvegarde, avec les noms IntakeT, CoolerT, PlenumT et ExhaustT.

```
Sample(4, Temp(1),IEEE4)
Fieldnames "IntakeT, CoolerT"
```

Les 4 variables de la ligne de variable de température, sont stockées dans le tableau de sauvegarde, avec deux noms individuels et deux noms par défaut : IntakeT, CoolerT, Temp(3) et Temp(4).

```
Sample(4, Temp(1),IEEE4)
Fieldnames « IntakeT(2) »
```

Les 4 variables de la ligne de variable de température, sont stockées dans le tableau de sauvegarde, avec deux noms individuels de type ligne de donnée (IntakeT) et deux noms par défaut : IntakeT(1), IntakeT(2), Temp(3) et Temp(4).

Histogram (BinSelect, DataType, DisableVar, Bins, Form, WtVal, LoLim, UpLim)

Cette instruction fait des calculs sur les données en entrée afin d'effectuer un histogramme standard (distribution en fréquence) ou un histogramme à valeur pondérée.

L'histogramme standard compte le temps pendant lequel la variable sélectionnée est comprise dans la fraction de l'intervalle de sauvegarde d'une étendue totale spécifiée. Un compteur associé à chaque sous étendue, est incrémenté à chaque fois qu'une valeur de la variable spécifiée, est comprise dans cette sous-étendue. La valeur qui est sauvegardée en mémoire finale peut être le total d'itération pour chaque sous étendue ou la fraction de temps relatif à ce sous intervalle calculé en divisant le total d'itération par sous étendue, par le total de scrutations. Ce genre de sauvegarde est qualifié de distribution de fréquence.

L'histogramme à valeur pondérée n'ajoute pas une constante à une tranche, mais ajoute à la place la valeur actuelle de la variable. Le nom de cette variable est appelé la variable pondérée. A chaque fois que l'instruction est exécutée, la valeur pondérée est ajoutée à une tranche. La sous étendue dans laquelle se trouve la valeur de classement, détermine dans quelle tranche la valeur pondérée est ajoutée. Quand la sauvegarde est effectuée, la valeur accumulée dans chaque tranche peut être sauvegardée, ou bien elle peut être divisée par le nombre total de scrutations, puis sauvegardées dans le tableau de sauvegarde. Ces valeurs sont la contribution de chaque sous étendue, à la valeur pondérée globale. Une utilisation commune de l'histogramme de valeur pondérée à intervalle fermé, est la rose des vents. Les valeurs de vitesse de vent (la valeur pondérée en entrée) sont accumulées dans des secteurs de direction de vent correspondants (tranches sélectionnées).

Pour obtenir la moyenne des valeurs pondérées qui ont eu lieu dans une sous étendue particulière, la valeur sauvegardée en mémoire finale doit être divisée par la fraction de temps pendant laquelle la valeur de classement a été dans la sous étendue souhaitée (c'est à dire qu'il faut aussi sauvegarder un histogramme standard pour la valeur de classement ; pour chaque sous étendue, la valeur pondérée sauvegardée doit être divisée par la fréquence de distribution sauvegardée).

L'histogramme standard (fréquence de distribution) est défini en entrant une constante pour le paramètre « WtVal ». On entre « 1 » pour définir la sauvegarde en tant que fraction du temps total durant lequel la variable a été dans l'intervalle de la sous étendue spécifiée. On entre « 100 » afin d'avoir la sauvegarde en tant que pourcentage de temps. On entre un nom de variable pour avoir un histogramme en valeur pondérée.

Selon les souhaits, l'histogramme peut être ouvert ou fermé. La forme ouverte prend en compte toutes les valeurs comprises en dessous et au dessus des limites de l'étendue. La forme fermée exclue toute valeur non comprise dans l'étendue spécifiée pour l'histogramme.

La différence entre la forme ouverte et fermée, est montrée dans l'exemple suivant, pour des valeurs de température :

Limite inférieure de la plage	10°C
Limite supérieure de la plage	30°C
Nombre de sous-intervalles	10

	Forme fermée	**Forme ouverte**
Plage de la première tranche	10 à 11,99 °C	<12 °C
Plage de la dernière tranche	28 à 29,99 °C	>28°C

Paramètre *& Type de donnée*	**Entrée**
BinSelect *Variable ou Ligne de donnée*	La variable qui est testée afin de déterminer quel sous intervalle est sélectionné. L'instruction Histogram nécessite d'avoir une ligne de donnée dont la dimension comprend au moins autant d'éléments que les dimensions de l'histogramme.
DataType *Constante*	Un code pour sélectionner le format de stockage des données. **Code Alphanumérique** : **Format de données** IEEE4 : IEEE à 4 octets et virgule flottante FP2 : Campbell Scientific à 2 octets et virgule flottante
DisableVar *Constante, Variable, ou Expression*	Une valeur différente de zéro désactivera le traitement intermédiaire. On entre généralement « 0 » afin que toutes les entrées soient prises en compte. Par exemple lorsque DisableVar est ≠0, l'entrée actuelle n'est pas incluse dans l'histogramme. L'histogramme qui sera éventuellement stocké, inclura les entrées qui se seront produites lorsque DisableVar avait pour valeur « 0 ». **Valeur** : **Résultat** 0 : Prendre en compte la valeur actuellement présente en entrée ≠ 0 : Ne pas prendre en compte la valeur actuellement présente en entrée
Bins *Constante*	Le nombre de sous intervalles à inclure dans l'étendue définie pour l'histogramme. La largeur de chaque fenêtre est égale à l'étendue sélectionnée (UpLim - LowLim) divisée par le nombre d'intervalles.
Form *Constante*	L'argument du paramètre « Form », comprend 3 chiffres type - **ABC** **Code** : **Form** A = 0 : Ré-initialise l'histogramme après chaque sauvegarde. A = 1 : Ne ré-initialise pas l'histogramme. B = 0 : Divise les intervalles par le nombre total de comptages. B = 1 : Enregistre le nombre total d'itération par intervalle. C = 0 : Forme ouverte. Comprend les valeurs hors des bornes de l'intervalle. C = 1 : Forme fermée. Exclu les valeurs en dehors des bornes de l'intervalle. 101 signifie : Ne pas ré-initialiser. Diviser / le nombre de comptages. Forme fermée.
WtVal *Constante ou Variable*	Le nom de la variable pondérée. On entre une constante si on souhaite avoir une distribution en fréquence, ou bien on entre la valeur de BinSelect.
LoLim *Constante*	La limite basse de l'étendue couverte par la sélection de l'intervalle.
UpLim *Constante*	La limite haute de l'étendue couverte par la sélection de l'intervalle.

Histogram4D (BinSelect, Source, DataType, DisableVar, Bins1, Bins2, Bins3, Bins4, Form, WtVal, LoLim1, UpLim1, LoLim2, UpLim2, LoLim3, UpLim3, LoLim4, UpLim4)

Effectue des calculs sur les données en entrée et sous forme d'histogramme standard (distribution en fréquence), ou pondérée jusqu'à 4 dimensions.

La description de l'instruction "Histogram" s'applique aussi à l'instruction "Histogram4D". La différence est que l'instruction Histogram4D permet d'avoir en entrée jusqu'à quatre variables (dimensions). Les variables sélectionnées sont spécifiées en tant que ligne de donnée de variables. Chacune des variables sélectionnées a son intervalle de sélection et son nombre de sous intervalle propre. Le nombre total de sous intervalles est le produit des sous intervalles pour chaque dimension à savoir (Sous intervalle1 x Sous intervalle2 x Sous intervalle3 x Sous intervalle4).

Exemple de sauvegarde avec Histogram4D

L'exemple de programme qui suit est un exemple d'utilisation de l'instruction Histogram4D afin de calculer un histogramme à 2 dimensions pour une distribution RPM vs rapport de vitesse (RPM distribution vs Gear).

```
'\\\\\\\\\\\\\\\\\\\\ VARIABLES et CONSTANTES ////////////////////

Public RPM, Gear, Port(4)
Dim Bin(2)

'\\\\\\\\\\\\\\\\\\\\\\\\ SECTION SAUVEGRDE (OUTPUT) ////////////////////////

DataTable (RPMvsG,1,100)
    DataInterval(0,60,Min,100)
    Histogram4D(Bin(), FP2, 0,4,8, 0, 0,000,1,0.5, 4.5, 0,8000, 0, 0, 0, 0)
    '4 intervalles pour la variable gear (vitesse), entre 0,5 to 4,5 ; 8 intervalles pourRPM,
    'avec une  étendue allant de 0  à 8000
    'On utilise la forme ouverte de façon à ce que RPM >8000 soit inclus dans l'intervalle
    ' de 7000 à 8000
EndTable

'\\\\\\\\\\\\\\\\\\\\\\\\\\ PROGRAM ////////////////////////////

BeginProg
    Scan (100,mSec,3,0)
    PulseCount (RPM,1,1 ,1,1,0.4225,0)     'RPM provenant d'un Volant à 142 dents
                                           '60 rpm/142 Hz = 0.42253 ...
    Portget (Port(1),1)          'Il y a des entrées numériques vers les ports 1 à 4
    Portget (Port(2),2)          'Si C1 est à l'état haut alors on sera sur la première vitesse
    Portget (Port(3),3)          'C2 indiquera qu'on est à la deuxième vitesse etc.
    Portget (Port(4),4)

    IF Port(1) then Gear = 1
    If Port(2) Then Gear = 2
    If Port(3) Then Gear = 3
    If Port(4) Then Gear = 4

    Bin(1) = Gear
    Bin(2) = RPM
    CallTable RPMvsG
    Next Scan

EndProg
```

LevelCrossing (Source, DataType, DisableVar, NumLevels, 2ndDim, CrossingArray, 2ndArray, Hysteresis, Option)

<table>
<tr><td>Paramètre
& Type de donnée</td><td colspan="2">Entrée</td></tr>
<tr><td>Source
Variable ou ligne de donnée</td><td colspan="2">La variable qui est testée afin de déterminer si elle dépasse les niveaux spécifiés. Si un dépassement à deux dimensions est sélectionné, la source doit être une ligne de données. Le second élément de la ligne de données (ou l'élément suivant, consécutive à celui spécifié pour la source) est la variable qui est testée afin de déterminer la seconde dimension de l'histogramme.</td></tr>
<tr><td>DataType
Constante</td><td colspan="2">Un code afin de sélectionner le format de stockage des données.<table><tr><th>Code Alphanumérique</th><th>Format de données</th></tr><tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr><tr><td>FP2</td><td>Campbell Scientific à 2 octets et virgule flottante</td></tr></table></td></tr>
<tr><td>DisableVar
Constante, Variable, ou Expression</td><td colspan="2">Une valeur différente de zéro désactivera le traitement intermédiaire. On entre généralement 0 afin que toutes les données soient traitées. Par exemple, lorsque DisableVar est ≠ 0, la valeur actuellement en entrée n'est pas incluse dans l'histogramme. L'histogramme qui est éventuellement stocké comprend les entrées qui se sont produites pendent que la variable de désactivation (DisableVar) était à 0.<table><tr><th>Valeur</th><th>Résultat</th></tr><tr><td>0</td><td>Prendre en compte la valeur actuellement présente en entrée</td></tr><tr><td>≠ 0</td><td>Ne pas prendre en compte la valeur actuellement présente en entrée</td></tr></table></td></tr>
<tr><td>NumLevels
Constante</td><td colspan="2">Le nombre de niveaux sur lequel effectuer le comptage des dépassements. C'est le nombre d'intervalles dans lesquels on stocke le nombre de dépassements pour le niveau associé. Les niveaux réels sont entrés dans la ligne de donnée de dépassement «Crossing Array». Un comptage est ajouté à un intervalle lorsque la source a une valeur inférieure à celle du niveau associé, à une valeur plus importante que celle de l'intervalle associé (front montant ou polarité positive). Si le front descendant ou la polarité négative sont choisi, le comptage se produit si la valeur de la source passe d'un intervalle supérieur à un intervalle inférieur.</td></tr>
<tr><td>2ndDim
Constante</td><td colspan="2">La seconde dimension de l'histogramme. Le nombre total d'intervalles sauvegardés = NumLevels*2ndDim. On entre 1 pour un histogramme à une dimension qui ne consiste qu'en un nombre de niveaux de dépassement. Si 2ndDim a une valeur supérieure à 1, l'élément de le la ligne de donnée source qui suit celle qui est testée pour le dépassement de niveau, est utilisée afin de déterminer la seconde dimension.</td></tr>
<tr><td>Crossing Array
Ligne de donnée</td><td colspan="2">Le nom de la ligne de donnée qui contient les niveaux de dépassement à vérifier. Du fait que cela n'ait pas de sens de changer les niveaux alors que le programme est en cours d'exécution, le programme devrait être écrit afin de charger les valeurs dans la ligne de données une fois, avant d'entrer dans la scrutation.</td></tr>
<tr><td>2ndArray
Ligne de donnée</td><td colspan="2">Le nom de la ligne de données qui contient le niveau qui détermine la seconde dimension. Du fait que cela n'ait pas de sens de changer les niveaux alors que le programme est en cours d'exécution, le programme devrait être écrit afin de charger les valeurs dans la ligne de données une fois, avant d'entrer dans la scrutation.</td></tr>
<tr><td>Hysteresis
Constante</td><td colspan="2">Le changement minimum dans la source qui doit se produire pour qu'un dépassement soit comptabilisé.</td></tr>
<tr><td>Option
Constante</td><td></td><td>Le code d'option est à 3 chiffres - ABC<table><tr><th>Code</th><th>Forme</th></tr><tr><td>A = 0</td><td>Comptage sur front descendant (la source passe de > au niveau à < niveau</td></tr><tr><td>A = 1</td><td>Comptage sur front montant (la source passe de < au niveau à > au niveau</td></tr><tr><td>B = 0</td><td>Ré-initialisation à 0 du comptage de l'histogramme après chaque sauvegarde.</td></tr><tr><td>B = 1</td><td>Ne pas ré-initialiser l'histogramme; continuer à accumuler les comptages.</td></tr><tr><td>C = 0</td><td>Diviser le comptage de chaque intervalle par le nombre total de comptages dans tous les intervalles.</td></tr><tr><td>C = 1</td><td>Sauvegarder le nombre total de comptages dans chaque intervalle.</td></tr></table>101 signifie : Comptage sur front montant, ré-initialiser à 0 à chaque sauvegarde, sauvegarder les comptages.</td></tr>
</table>

On traite les données avec l'algorithme de comptages "Level Crossing". La sauvegarde est un histogramme de dépassement de niveau à deux dimensions (Level Crossing Histogram). Une dimension est celle des niveaux dépassés ; la seconde dimension, si elle est utilisée, est la valeur de la seconde entrée au moment où les dépassements ont été détectés. Le nombre total d'intervalles sauvegardés = NumLevels*2ndDim. Pour avoir un histogramme de dépassement de niveau à une dimension, il faut entrer 1 pour la variable 2ndDim.

La valeur source peut être le résultat d'une mesure ou d'un calcul. A chaque fois que le tableau de données avec l'instruction « Level Crossing » est appelée, la source est vérifiée afin de voir si sa valeur a changé à partir de la valeur précédente et si en cas de changement, elles ont dépassé un quelconque niveau de dépassement spécifié. L'instruction peut être programmée afin de comptabiliser les dépassements de front montant ou descendant.

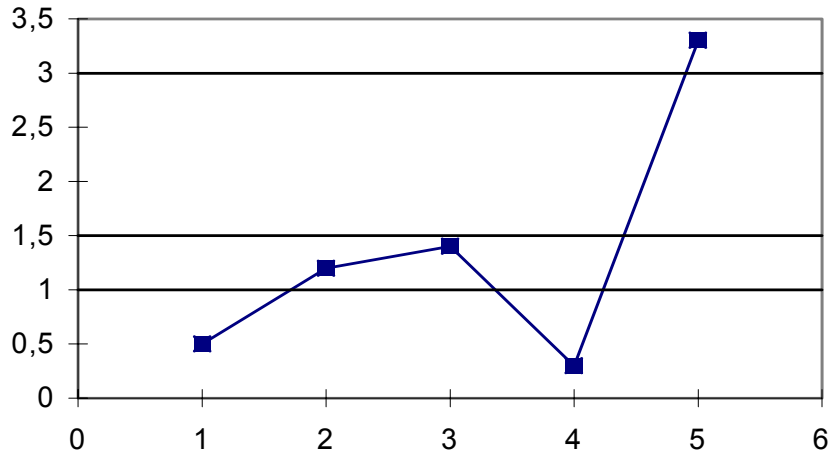


FIGURE 6.4-1. Exemple de données de dépassement

Comme exemple d'algorithme de dépassement de niveau nous considèrerons que nous avons un histogramme à une dimension et 3 intervalles de niveau de dépassement (la seconde dimension est égale à 1) et que l'on effectue les comptages sur le front montant. Les niveaux de dépassements sont 1 / 1,5 et 3. La figure 6.4-1 montre un exemple de données. Si on va dans les données point par point on a :

Point	Source	Action	Intervalle 1 (niveau = 1)	Intervalle 2 (niveau = 1,5)	Intervalle 3 (niveau = 3)
1	0.5	Première valeur, pas de comptage	0	0	0
2	1.2	On ajoute un comptage au premier intervalle, le signal dépasse la valeur 1	1	0	0
3	1.4	Pas de niveau dépassé, pas de comptage	1	0	0
4	0.3	Un niveau est dépassé mais sur front descendant d'où pas de comptage	1	0	0
5	3.3	On ajoute un comptage au premier, second et troisième intervalle car le signal a dépassé les valeurs 1 / 1,5 et 3.	2	1	1

La seconde dimension, lorsqu'elle est supérieure à 1, est déterminée par la valeur de l'élément présent dans la ligne de donnée source qui suit l'élément vérifié pour le dépassement. C'est la valeur de cette variable au moment où les dépassements sont détectés, qui déterminent la seconde dimension.

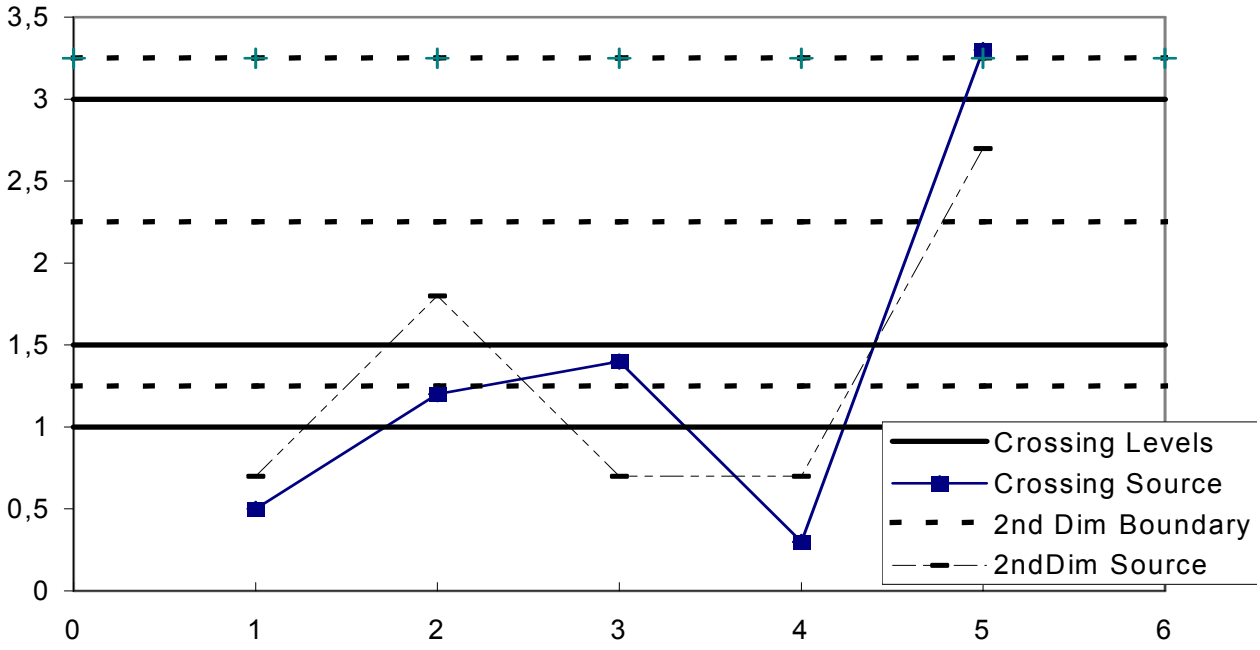


FIGURE 6.4-2. Donnée de dépassement avec valeur de seconde dimension

Point	Source de dépassement	2ème source de Dim.	Action
1	0,5	0,7	Première valeur, pas de comptages
2	1,2	1,8	Ajouter un comptage au premier dépassement, second intervalle 2D, le signal a dépassé 1

Histogramme :

	2D < 1,25	1,25<2D<2,25	2,25<2D<3,25
Dépassement 1	0	1	0
Dépassement 1, 5	0	0	0
Dépassement 3	0	0	0

3	1,4	0,7	Pas de niveaux dépassés, pas de comptages
4	0,3	0,7	Dépassement d'un niveau mais sur front descendant, pas de comptages
5	3,3	2,7	On ajoute un comptage au premier, second et troisième intervalle de dépassement dans les troisièmes intervalle 2D, le signal ayant dépassé 1 / 1,5 et 3.

Histogramme :

	2D < 1,25	**1,25<2D<2,25**	**2,25<2D<3,25**
Dépassement 1	0	1	1
Dépassement 1,5	0	0	1
Dépassement 3	0	0	1

A noter que le premier intervalle de la seconde dimension est toujours de forme « ouverte ». Toutes les valeurs inférieures à la valeur seuil définie, sont incorporées à cet intervalle. Le dernier intervalle de la seconde dimension est toujours de forme « fermée ». Il comprend uniquement les valeurs qui sont inférieures à la valeur seuil supérieure définie, et supérieure ou égale à la valeur seuil supérieure de l'intervalle précédent. Si vous souhaitez avoir un histogramme « ouvert » sur les deux bornes de l'intervalle de la seconde dimension, il faut entrer un seuil limite supérieur pour le dernier intervalle de la seconde dimension, qui soit supérieur à une quelconque valeur que vous seriez susceptible d'avoir en valeur source de la seconde dimension.

Les niveaux de dépassement et les seuils de la seconde dimension, ne sont spécifies dans l'instruction LevelCrossing mais sont contenus dans des variables de ligne de donnée. Cela permet aux niveaux d'être espacés de la façon dont le programmeur en a envie. Les lignes de données nécessitent d'avoir au minimum la même dimension que celles de l'histogramme. Si on sélectionne un histogramme à dépassement de niveau à une dimension (1 entré pour la seconde dimension), le nom de la ligne de données de dépassement peut aussi être donné pour la deuxième ligne de données, afin d'éviter de déclarer une ligne de donnée non utilisée. Le programme doit charger les valeurs dans ces lignes de données.

La ligne de donnée qui spécifie les seuils de la seconde dimension, est chargée avec les limites supérieures de chaque intervalle. Par exemple on considère que la seconde dimension est 3, et que les limites supérieures de la ligne de données qui contient les seuils de seconde dimension sont 1 / 3 et 6.

La valeur de chaque élément (intervalle) de l'histogramme, peut être le nombre de fois que le signal a dépassé le niveau associé avec cet intervalle, ou bien cela peut être la fraction du nombre total de dépassements comptabilisés et qui ont été associés avec cet intervalle (à savoir le nombre de comptages dans l'intervalle, divisé par le nombre total de comptages dans tous les intervalles).

L'hystérésis détermine le minimum de changement qui doit se produire sur l'entrée avant que le dépassement ne soit comptabilisé. Si la valeur est trop petite, les "dépassements" seront comptabilisés mais ne seront en fait que du bruit. Si par exemple on suppose que le niveau de dépassement est la valeur 5. Si la valeur en entrée ne change pas vraiment mais varie entre 4,999 et 5,001 alors une hystérésis de 0 permettrait à tous ces dépassements d'être comptabilisés. Le fait de fixer l'hystérésis à 0,1 évitera à des bruits, de générer des comptages.

Maximum (Reps, Source, DataType, DisableVar, Time)

L'instruction de « maximum » enregistre le maximum de la valeur indiquée dans la variable source, durant l'intervalle de sauvegarde écoulé. L'heure associée à la valeur maximale, peut être enregistrée de façon optionnelle si l'on donne le paramètre approprié dans l'instruction de mesure, pour le paramètre « Time ».

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Reps
Constante</td><td>Le nombre de maximum à calculer. Quand le nombre de répétitions est supérieur à 1, la source doit être une ligne de données.</td></tr>
<tr><td>Source
Variable</td><td>Comprend le nom de la variable qui est la donnée source.</td></tr>
<tr><td>DataType
Constante</td><td>Un code afin de sélectionner le format de stockage des données.
<table>
<tr><th>Code Alphanumérique</th><th>Format de données</th></tr>
<tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr>
<tr><td>FP2</td><td>Campbell Scientific à 2 octets et virgule flottante</td></tr>
</table></td></tr>
<tr><td>DisableVar
Constante, Variable, ou Expression</td><td>Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Si la variable de désactivation est différente de 0, la valeur de la variable du moment ne sera pas pris en compte pour l’obtention du nouveau maximum. Le maximum qui sera sauvegardé sera le maximum des valeurs en entrée, qui se seront produit lorsque la variable de désactivation portait la valeur 0.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On prend en compte la valeur du moment</td></tr>
<tr><td>≠ 0</td><td>On ne prend pas en compte la valeur du moment</td></tr>
</table></td></tr>
<tr><td>Time
Constante</td><td>Option afin d’enregistrer le moment où a eu lieu le maximum. Quand le temps est sauvegardé, les maximums de chaque répétition sont enregistrés, suivis par le moment auxquels ils se sont déroulés.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On n’enregistre pas l’heure</td></tr>
<tr><td>1</td><td>On enregistre l’heure</td></tr>
</table></td></tr>
</table>

Minimum (Reps, Source, DataType, DisableVar, Time)

L’instruction « minimum » enregistre le minimum de la valeur indiquée dans la variable source, durant l’intervalle de sauvegarde spécifié. L’heure associée à la valeur minimale, peut être enregistrée de façon optionnelle si l’on donne le paramètre approprié dans l’instruction de mesure, pour le paramètre « Time ».

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Reps
Constante</td><td>Le nombre de minimum à calculer. Quand le nombre de répétitions est supérieur à 1, la source doit être une ligne de données.</td></tr>
<tr><td>Source
Variable</td><td>Comprend le nom de la variable qui est la donnée source.</td></tr>
<tr><td>DataType
Constante</td><td>Un code afin de sélectionner le format de stockage des données.
<table>
<tr><th>Code Alphanumérique</th><th>Format de données</th></tr>
<tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr>
<tr><td>FP2</td><td>Campbell Scientific à 2 octets et virgule flottante</td></tr>
</table></td></tr>
<tr><td>DisableVar
Constante, Variable, ou Expression</td><td>Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Si la variable de désactivation est différente de 0, la valeur du moment ne sera pas prise en compte pour l’obtention du nouveau minimum. Le minimum qui sera sauvegardé sera le minimum des valeurs en entrée qui se seront produites lorsque la variable de désactivation portait la valeur 0.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On prend en compte la valeur du moment</td></tr>
<tr><td>≠ 0</td><td>On ne prend pas en compte la valeur du moment</td></tr>
</table></td></tr>
<tr><td>Time
Constante</td><td>Option afin d’enregistrer le moment où a eu lieu le minimum. Quand le temps est sauvegardé, les minimums de chaque répétition sont enregistrés, suivis par le moment auxquels ils se sont déroulés.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On n’enregistre pas l’heure</td></tr>
<tr><td>1</td><td>On enregistre l’heure</td></tr>
</table></td></tr>
</table>

RainFlow
(Source, DataType, DisableVar, MeanBins, AmpBins, LowLim, UpLimMinAmp, Form)

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Source
Variable</td><td>La variable qui est testée afin de déterminer quel intervalle est sélectionné.</td></tr>
<tr><td>DataType
Constante</td><td>Un code afin de sélectionner le format de stockage des données.
<table>
<tr><th>Code Alphanumérique</th><th>Format de données</th></tr>
<tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr>
<tr><td>FP2</td><td>Campbell Scientific à 2 octets et virgule flottante</td></tr>
</table></td></tr>
<tr><td>DisableVar
Constante, Variable, ou Expression</td><td>Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Si la variable de désactivation est différente de 0, la valeur du moment ne sera pas prise en compte dans l’histogramme. L’histogramme qui sera sauvegardé sera le minimum des valeurs en entrée qui se seront produites lorsque la variable de désactivation portait la valeur 0.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On prend en compte la valeur du moment</td></tr>
<tr><td>≠ 0</td><td>On ne prend pas en compte la valeur du moment</td></tr>
</table></td></tr>
<tr><td>MeanBins
Constante</td><td>Ce paramètre permet de trier par rapport à la valeur moyenne du signal durant un cycle de contrainte-tension. Le nombre entré est le nombre d’intervalles ou de sous intervalle à l’intérieur desquels on va trier par rapport à la valeur moyenne. Si on entre « 1 » on ne prend pas en compte la valeur du signal et on triera selon l’amplitude du signal uniquement. L’étendue de chaque intervalle est égale à UpLim - LowLim divisé par le nombre d’intervalles. La valeur mini. de l’intervalle le plus bas et la valeur maxi. de l’intervalle le plus haut, sont les limites hautes et basses de l’intervalle total.</td></tr>
<tr><td>AmpBins
Constante</td><td>Le nombre d’intervalles ou de sous-intervalles dont on trie l’amplitude selon un cycle de contrainte-tension. La largeur de chaque sous intervalle est égal à UpLim - LowLim divisé par le nombre d’intervalles.</td></tr>
<tr><td>LowLim
Constante</td><td>La limite basse du signal d’entrée et de la moyenne des intervalles.</td></tr>
<tr><td>UpLim
Constante</td><td>La limite haute du signal d’entrée et de la moyenne des intervalles.</td></tr>
<tr><td>MinAmp
Constante</td><td>L’amplitude minimale que doit avoir le cycle de contrainte-tension afin d’être comptabilisé.</td></tr>
<tr><td>Form
Constante</td><td>Le code est d’une forme à 3 chiffres - ABC
<table>
<tr><th>Code</th><th>Form</th></tr>
<tr><td>A = 0</td><td>Ré-initialise l’histogramme après chaque sauvegarde.</td></tr>
<tr><td>A = 1</td><td>Ne ré-initialise pas l’histogramme.</td></tr>
<tr><td>B = 0</td><td>Divise les intervalles par le nombre total de comptages.</td></tr>
<tr><td>B = 1</td><td>Sauvegarde le total dans chaque intervalle.</td></tr>
<tr><td>C = 0</td><td>Forme ouverte. Prend en compte les valeurs hors de l’étendue spécifiée.</td></tr>
<tr><td>C = 1</td><td>Forme fermée. Exclu les valeurs hors de l’étendue spécifiée.</td></tr>
</table>
101 signifie : Ne pas ré-initialiser. Diviser les intervalles par le nombre de comptages. Forme fermée.</td></tr>
</table>

On utilise le traitement des données par l’instruction Rainflow et son algorithme de comptage afin d’estimer le dommage de fatigue cumulée sur des composants subissant des cycles de contrainte-tension. Les données peuvent être fournies en effectuant des mesures en mode standard ou en mode salve. L’instruction Rainflow peut effectuer un traitement sur une fenêtre de données à la suite d’un mode salve, ou bien “en ligne” comme les autres instructions de traitement.

Ce qui est sauvegardé est un histogramme de répartition des pluies (*Rainflow Histogram*) à deux dimensions. Une dimension est l'amplitude du cycle de boucle fermée (soit la distance entre le pic et la vallée) ; l'autre dimension est la moyenne du cycle (soit [valeur au pic + valeur à la vallée] / 2). La valeur de chaque élément (intervalle) de l'histogramme peut être soit le véritable nombre de cycles de boucle fermée qui avaient l'amplitude et la valeur moyenne associée avec l'intervalle, ou la fraction du nombre total de cycles comptabilisés qui ont été associés avec cet intervalle (soit le nombre total de cycles par intervalle divisé par le nombre total de cycles comptabilisés).

L'utilisateur entre le nombre d'intervalles moyens, le nombre d'intervalle d'amplitude, et les limites hautes et basses des données en entrée.

Les valeurs pour les intervalles d'amplitude sont déterminées par la différence entre la limite haute et basse sur les données en entrée, et par le nombre d'intervalles. Si par exemple la limite basse est 100 et que la limite haute est 150, qu'il y a 5 intervalles d'amplitude, alors le maximum d'amplitude est 150 – 100 = 50. Le changement d'amplitude entre les intervalles et la limite supérieure du plus petit intervalle d'amplitude est 50/5 = 10. Les cycles avec une amplitude A, inférieure à 10, seront comptabilisés dans le premier intervalle. Le second intervalle est pour $10 \leq A < 20$, le troisième pour $20 \leq A < 30$, etc.

Afin de déterminer les étendues pour les intervalles moyens, les véritables valeurs des limites sont utilisées, en plus de la différence entre elles. La limite basse des données en entrée est aussi la limite basse du premier intervalle. Prenons encore 100 comme limite basse et 150 comme limite haute, avec 5 intervalles moyens. Dans ce cas le premier intervalle est pour les cycles qui ont des valeurs moyennes M avec tout d'abord $100 \leq M < 110$, puis $110 \leq M < 120$, etc.

Si $C_{m,a}$ est le nombre de comptages pour l'étendue de moyenne m d'amplitude a, et M et N sont les nombres d'amplitude et de moyenne d'intervalles respectivement, alors la sauvegarde lorsque le paramètre de répétition est égal à 1, sera ordonnée séquentiellement ($C_{1,1}$, $C_{1,2}$, ... $C_{1,N}$, $C_{2,1}$, $C_{2,2}$, ... $C_{M,N}$). Un nombre de répétitions supérieur à 1 sera séquentiel en mémoire. En deux dimensions cela donnera :

$C_{1,1}$	$C_{1,2}$	.	.	.	$C_{1,N}$
$C_{2,1}$	$C_{2,2}$	.	.	.	$C_{2,N}$
.	.	.			.
.	.		.		.
.	.			.	.
$C_{M,1}$	$C_{M,2}$	.	.	.	$C_{M,N}$

L'histogramme peut avoir une forme ouverte ou bien fermée. En forme ouverte, un cycle qui a une amplitude supérieure à la limite haute, sera comptabilisé dans l'intervalle maximum ; un cycle qui a une valeur moyenne inférieure à la limite basse sera comptabilisé dans l'intervalle minimum ; un cycle qui a une valeur moyenne inférieure à la limite basse ou supérieure à la limite haute, sera comptabilisé dans l'intervalle minimum ou maximum. En forme fermée un cycle qui est en dessous de l'amplitude ou de la limite moyenne, n'est alors pas comptabilisé.

La distance minimum entre le pic et la vallée, MinAmp, détermine le cycle de plus petite amplitude qui sera comptabilisé. La distance devrait être inférieure à la largeur de l'amplitude de l'intervalle ([limite haute – limite basse]/nombres d'intervalles d'amplitude) ou aux cycles dont les cycles ayant l'amplitude du premier intervalle ne seront pas comptabilisés. Cependant si la valeur est trop petite, le temps de traitement sera consommé afin de comptabiliser des « cycles » qui ne sont en réalité que du bruit.

Sauvegarde générée : Nombre d'intervalles moyens x Nombre d'intervalles d'amplitude x Répétitions

Sample (Reps, Source, DataType)

Cette instruction sauvegarde la ou les valeur(s) actuellement présente(s) dans la variable ou la ligne de donnée de variables spécifiées (prend un échantillon).

Paramètres *& type de donnée*	**Entrée**
Reps *Constante*	Le nombre d'échantillons à prendre. Quand le nombre de répétitions est supérieur à 1, la source doit être une ligne de données.
Source *Variable*	Comprend le nom de la variable qui doit être échantillonnée.
DataType *Constante*	Un code afin de sélectionner le format de stockage des données.

Code Alphanumérique	**Format de donnée**
IEEE4	IEEE à 4 octets et virgule flottante
FP2	Campbell Scientific à 2 octets et virgule flottante

SampleMaxMin (Reps, Source, DataType, DisableVar)

L'instruction SampleMaxMin est utilisée afin d'échantillonner une ou plusieurs variables lorsqu'une autre variable (ou bien une quelconque variable comprise dans une ligne de données) atteint son maximum ou minimum pour la période définie en tant qu'intervalle de sauvegarde.

L'instruction SampleMaxMin est placée à l'intérieur de la déclaration DataTable, à la suite de l'instruction Minimum ou Maximum qui sera utilisée afin d'effectuer le basculement de l'échantillon. SampleMaxMin effectue un échantillon à chaque fois qu'un minimum ou un maximum est détecté par l'instruction précédente. Lorsqu'un nouvel échantillon est pris, le(les) valeur(s) précédente n'est plus prise en compte. Les échantillons enregistrés dans le tableau de donnée seront ceux qui ont été pris après qu'un minimum ou un maximum se sera produit.

Le nombre de valeurs sauvegardées par SampleMaxMin n'est déterminé que par les paramètres de source et de destination ; le nombre de répétition dans l'instruction précédente importe peu. Lorsque le paramètre de répétitions de l'instruction Minimum ou Maximum qui précède, est supérieur à 1, alors SampleMaxMin échantillonnera à chaque fois qu'un nouveau minimum ou maximum se produit dans une quelconque des variables de ligne de donnée source. Afin de s'assurer que l'échantillon est pris uniquement lorsqu'un nouveau minimum ou maximum se produit pour une variable bien spécifique, l'instruction minimum ou maximum qui précède doit avoir le paramètre de répétitions = 1.

Paramètre *& Type de donnée*	**Entrée**
Reps *Constante*	Le nombre de valeurs à échantillonner. Lorsqu'il y a plus d'une répétition, la source doit être une ligne de données.
Source *Variable*	La source est le nom de la variable ou de la ligne de données qui est échantillonnée lorsqu'un nouveau minimum ou maximum se produit pour l'instruction Maximum ou Minimum qui la précède.
DataType *Constante*	On sélectionne le format de stockage des données. **Entrée** : **Description** IEEE4 : IEEE à 4 octets et virgule flottante FP2 : Campbell Scientific à 2 octets et virgule flottante
DisableVar *Constante, Variable ou Expression*	DisableVar est une constante, variable, ou expression qui est utilisée afin de déterminer si la mesure actuelle est à inclure ou non dans les valeurs à évaluer afin d'avoir le maximum ou le minimum. **Valeur** : **Résultat** 0 : On prend en compte la valeur de la variable du moment ≠0 : On ne prend pas en compte la valeur de la variable du moment

StdDev (Reps, Source, DataType, DisableVar)

L'instruction StdDev calcule l'écart type de la (les) source(s) durant la période définie pour l'intervalle de sauvegarde.

$$\sigma(x) = \left(\left(\sum_{i=1}^{i=N} x_i^2 - \left(\sum_{i=1}^{i=N} x_i \right)^2 / N \right) / N \right)^{\frac{1}{2}}$$

Où σ (x) est l'écart type de x, et N est le nombre d'échantillons.

Paramètres *& type de donnée*	**Entrée**
Reps *Constante*	Le nombre d'écarts type à calculer. Quand le nombre de répétitions est supérieur à 1, la source doit être une ligne de données.
Source *Variable*	Comprend le nom de la variable qui est la donnée source.
DataType *Constante*	Un code afin de sélectionner le format de stockage des données. **Code Alphanumérique** : **Format de donnée** IEEE4 : IEEE à 4 octets et virgule flottante FP2 : Campbell Scientific à 2 octets et virgule flottante
DisableVar *Constante, Variable, ou Expression*	Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Si la variable de désactivation est différente de 0, la valeur de la variable du moment ne sera pas prise en compte pour le calcul de l'écart type. L'écart type qui sera sauvegardé sera celui calculé à partir des valeurs en entrée, lorsque la variable de désactivation portait la valeur 0. **Valeur** : **Résultat** 0 : On prend en compte la valeur de la variable du moment ≠ 0 : On ne prend pas en compte la valeur de la variable du moment

Totalize (Reps, Source, DataType, DisableVar)

Cette instruction enregistre le total de la (les) variable(s) source, sur la durée de l'intervalle de sauvegarde.

<table>
<tr><th>Paramètres
& type de donnée</th><th>Entrée</th></tr>
<tr><td rowspan="2">Reps
Constante
Variable</td><td>Le nombre de totaux à calculer. Quand le nombre de répétitions est supérieur à 1, la source doit être une ligne de données.</td></tr>
<tr><td>Comprend le nom de la variable qui est la donnée source.</td></tr>
<tr><td>DataType
Constante</td><td>Un code afin de sélectionner le format de stockage des données.
<table>
<tr><th>Code Alphanumérique</th><th>Format de donnée</th></tr>
<tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr>
<tr><td>FP2</td><td>Campbell Scientific à 2 octets et virgule flottante</td></tr>
</table></td></tr>
<tr><td>DisableVar
Constante,
Variable, ou
Expression</td><td>Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Si la variable de désactivation est différente de 0, la valeur du moment pour la variable spécifiée ne sera pas prise en compte pour le calcul du total. Le total qui sera sauvegardé sera celui calculé à partir des valeurs en entrée, lorsque la variable de désactivation portait la valeur 0.
<table>
<tr><th>Valeur</th><th>Résultat</th></tr>
<tr><td>0</td><td>On prend en compte la valeur de la variable du moment</td></tr>
<tr><td>≠ 0</td><td>On ne prend pas en compte la valeur de la variable du moment</td></tr>
</table></td></tr>
</table>

WindVector (Repetitions, Speed/East, Direction/North, DataType, DisableVar, Subinterval, SensorType, OutputOpt)

L'instruction WindVector calcule la vitesse et la direction du vent à partir de capteurs de type polaires (vitesse et direction du vent) ou orthogonaux (hélices fixées à l'Est et au Nord). Elle utilise les données brutes afin de générer le vecteur unitaire principal de direction du vent, la magnitude principale du vecteur vent, et la direction principale du vecteur vent sur l'ensemble de l'intervalle de sauvegarde. Il existe deux façons différentes de calculer la direction du vecteur vent (et l'écart type de la direction du vent), dont l'une qui pondère la vitesse du vent.

<table>
<tr><td>Paramètres
& Type de Donnée</td><td>Entrée</td></tr>
<tr><td>Repetitions
Constante</td><td>Le nombre de jeux de données de vent (vitesse/direction ou Est/Nord) pour lesquels on calculera un résultat.</td></tr>
<tr><td>Speed / East
Dir / North
Variables ou
Lignes de variables</td><td>Les variables source pour la vitesse et la direction du vent, ou dans le cas des capteurs orthogonaux, pour le vent d’Est et de Nord. Si la répétition est supérieure à 1, les variables sources doivent être des lignes de données contenant des éléments pour chacune des répétitions.</td></tr>
<tr><td>DataType
Constante</td><td>Un code afin de sélectionner le format de stockage des données.<table><tr><th>Code Alphanumérique</th><th>Format de donnée</th></tr><tr><td>IEEE4</td><td>IEEE à 4 octets et virgule flottante</td></tr><tr><td>FP2</td><td>Campbell Scientific à 2 octets et virgule flottante</td></tr></table></td></tr>
<tr><td>DisableVar
Constante,
Variable, ou
Expression</td><td>Une valeur différente de 0 désactivera le traitement intermédiaire. En général on entre la valeur 0 afin que toutes les entrées soient traitées. Si la variable de désactivation est différente de 0, la valeur de la variable du moment ne sera pas prise en compte pour le calcul. Le résultat qui sera sauvegardé sera celui calculé à partir des valeurs en entrée, lorsque la variable de désactivation portait la valeur 0.<table><tr><th>Valeur</th><th>Résultat</th></tr><tr><td>0</td><td>On prend en compte la valeur de la variable du moment</td></tr><tr><td>≠ 0</td><td>On ne prend pas en compte la valeur de la variable du moment</td></tr></table></td></tr>
<tr><td>Subinterval
Constante</td><td>Le nombre d’échantillons de calcul par sous intervalle. On entre la valeur 0 si on ne souhaite pas avoir de sous intervalle.</td></tr>
<tr><td>SensorType
Constante</td><td>Type du capteur de vent :<table><tr><th>Valeur</th><th>Type de capteur</th></tr><tr><td>0</td><td>Vitesse et direction</td></tr><tr><td>1</td><td>Vitesse Est et Nord</td></tr></table></td></tr>
<tr><td>OutputOpt
Constante</td><td><table><tr><th>Valeur</th><th>Type de donnée sauvegardée (pour chaque répétition)</th></tr><tr><td>0</td><td>1. Vitesse moyenne du vent horizontal, S
2. Vecteur unitaire de la direction moyenne du vent, Θ1
3. Ecart type de la direction du vent, σ(Θ1)
L’écart type est calculé avec l’algorithme de Yamartino. Cette option est en accord avec les recommandations de l’EPA, sur les modèles de dispersion rectiligne Gaussien pour la modélisation du transport d’un panache.</td></tr><tr><td>1</td><td>1. Vitesse moyenne du vent horizontal, S
2. Vitesse unitaire de la direction moyenne du vent, Θ1</td></tr><tr><td>2</td><td>1. Vitesse moyenne du vent horizontal, S
2. Résultante de la vitesse moyenne du vent, U
3. Résultante de la direction moyenne du vent, ΘU
4. Ecart type de la direction du vent, σ(ΘU)
Cet écart type est calculé à partir de l’algorithme de Campbell Scientific, pondéré par la vitesse du vent.
L’utilisation de la résultante de la direction moyenne du vent horizontal n’est pas recommandée pour le modèle de dispersion rectiligne Gaussien, mais peut être utilisé pour la modélisation de la direction de transport dans une modélisation à trajectoire variable.</td></tr></table></td></tr>
</table>

Quand un échantillon de vitesse de vent est de 0, l'instruction utilise 0 pour traiter le scalaire ou le vecteur de vitesse de vent résultant, mais l'échantillon n'est pas comptabilisé dans le calcul de la direction du vent. L'utilisateur peut souhaiter ne pas utiliser un échantillon inférieur au seuil de démarrage du capteur utilisé, pour le calcul de l'écart type. Si tel est le cas, écrivez un programme afin de vérifier la vitesse du vent et, si elle est inférieure au seuil, donnez la valeur 0 à la vitesse du vent avant d'appeler le tableau de sauvegarde.

L'écart type peut être calculé de deux façons : 1) en utilisant les échantillons pris durant la période de sauvegarde (on donne alors 0 à la valeur du **Subinterval**), ou 2) en moyennant les écarts type calculés pour des sous-intervalles plus courts que la période de sauvegarde. Le fait de moyenner les écarts type des sous-intervalles minimisera les effets de méandre sous des conditions de vent léger, et donnera des informations plus complètes pour des périodes de transition[1].

L'écart type pour la fluctuation horizontale du vent à partir de sous-intervalles, est calculé de la façon suivante :

$$\sigma(\Theta)=[((\sigma\Theta_1)^2+(\sigma\Theta_2)^2 \ldots+(\sigma\Theta_M)^2)/M]^{1/2}$$

où $\sigma(\Theta)$ est l'écart type sur la période de sauvegarde, et $\sigma\Theta_1 \ldots \sigma\Theta_M$ sont les sous-intervalles des écarts type.

Un sous-intervalle est spécifié en tant que nombre de scrutations (*number of scans*). Le nombre de scrutations pour un sous-intervalle est donné par :

Sous-intervalle souhaité (secs) / temps de scrutation (secs)
Si par exemple le temps de scrutation est de 1 seconde et que l'intervalle de sauvegarde est de 60 minutes, l'écart type est calculé pour les 3600 scrutations, lorsque le paramètre de **Subinterval** est fixé à la valeur 0. Avec un sous-intervalle 900 scrutations (15 minutes), l'écart type est la moyenne de quatre écarts type de sous-intervalles. Le dernier sous-intervalle est pondéré dans le cas où il ne contiendrait pas le nombre de scrutations normalement attendues.

Valeurs brutes mesurées :
S_i = vitesse horizontale du vent
Θ_i = direction horizontale du vent
Ue_i = composante Est-Ouest du vent
Un_i = composante Nord-Sud du vent
N = nombre d'échantillons

Calculs :

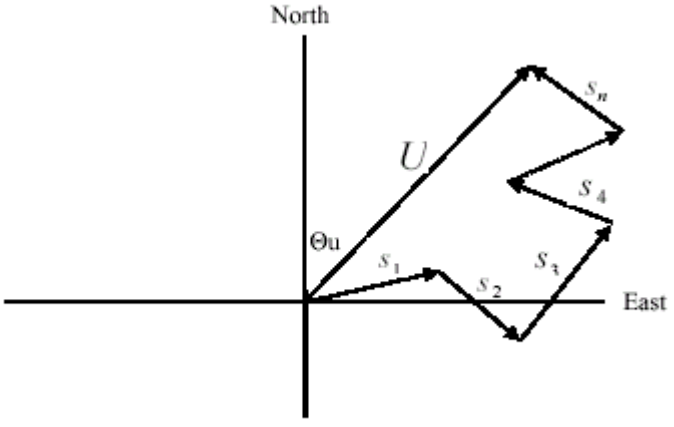


FIGURE 6.4-3 Vecteurs échantillonnés en entrée.

[1] EPA On-site Meteorological Program Guidance for Regulatory Modeling Applications.

Sur la figure 6.4-1, les petits vecteurs sont des vecteurs échantillonnés en entrée décrits par S_i et Θ_i, les échantillons de vitesse et de direction du vent, ou par Ue_i et Un_i, les composantes Est et Nord du vecteur échantillon. A la fin de l'intervalle de sauvegarde T, la somme des vecteurs échantillons est décrite par le vecteur de magnitude U et de direction Θu. Si l'intervalle de scrutation est t, le nombre d'échantillons dans l'intervalle de sauvegarde T est N = T / t. La magnitude du vecteur moyen est $\overline{U} = U / N$.

Moyenne scalaire de la vitesse du vent horizontal, S:

$$S=(\Sigma S_i)/N$$

avec dans le cas de capteurs orthogonaux :

$$S_i=(Ue_i^2+Un_i^2)^{1/2}$$

Vecteur unitaire moyen de la direction du vent, $\Theta 1$:

$$\Theta 1=\text{Arctan } (Ux/Uy)$$

où

$$Ux=(\Sigma \sin \Theta_i)/N$$
$$Uy=(\Sigma \cos \Theta_i)/N$$

et avec, dans le cas des capteurs orthogonaux :

$$Ux=(\Sigma(Ue_i/U_i))/N$$
$$Uy=(\Sigma(Un_i/U_i))/N$$
$$\text{où } U_i=(Ue_i^2+Un_i^2)^{1/2}$$

Ecart type de la direction du vent, $\sigma(\Theta 1)$, algorithme de Yamartino :

$$\sigma(\Theta 1)=\text{arc sin}(\varepsilon)[1+0{,}1547\ \varepsilon^3]$$

où

$$\varepsilon=[1-((Ux)^2+(Uy)^2)]^{1/2}$$

et Ux et Uy sont définis comme ci-dessus.

Moyenne de la résultante de la vitesse du vent horizontal, $\bar{U}$:

$$\bar{U} =(Ue^2+Un^2)^{1/2}$$

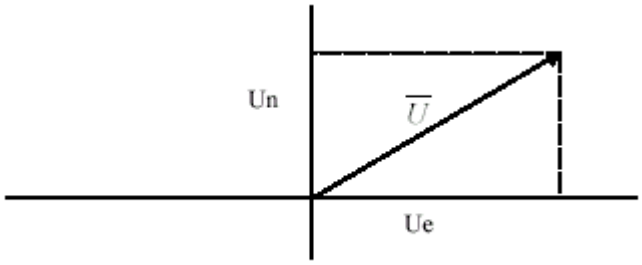


FIGURE 6.4-4 Vecteur vent moyen

Pour les capteurs polaires :

$$Ue=(\Sigma S_i \sin \Theta_i)/N$$
$$Un=(\Sigma S_i \cos \Theta_i)/N$$

ou, dans le cas de capteurs orthogonaux :

$$Ue=(\Sigma Ue_i)/N$$
$$Un=(\Sigma Un_i)/N$$

Moyenne de la résultante de la direction du vent, Θu :

$$\Theta u= \text{Arctan } (Ue/Un)$$

Ecart type de la direction du vent σ(Θu),en utilisant l'équation de Campbell Scientific :

$$\sigma(\Theta u)= 81(1- \bar{U} /S)^{1/2}$$

L'algorithme pour σ(Θu) est développé en notant que (voir figure 6.4-4)

$$\text{Cos } (\Theta_i') = U_i/s_i \text{ ; où } \Theta_i' = \Theta_i-\Theta_u$$

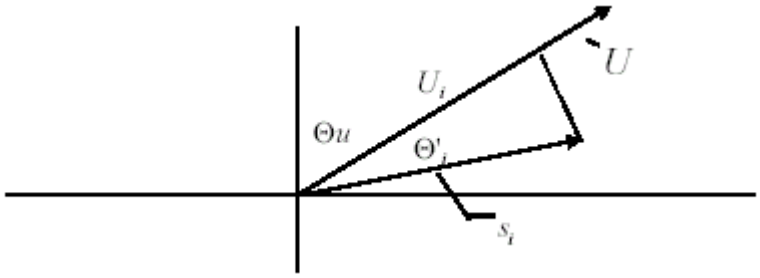


FIGURE 6.4-5 Ecart type de la direction.

La série de Taylor pour la fonction Cosinus, tronquée après 2 termes, est :

$$\text{Cos } (\Theta_i') = 1- (\Theta_i')^2/2$$

Pour des déviations de moins de 40 degrés, l'erreur dans l'approximation est de moins de 1%. A des déviations de 60 degrés, l'erreur est de 10%.

La vitesse d'échantillonnage peut être exprimée en tant que déviation autour de la vitesse moyenne,

$$s_i = s_i'+S$$

On pose l'équation pour les deux expressions pour Cos (θ') et on utilise l'équation précédente pour s_i ;

$$1-(\Theta_i')^2/2 = U_i/(s_i'+S)$$

La résolution de cette équation avec $(\Theta_i')^2$ donne :

$$(\Theta_i')^2 = 2-2U_i / S-(\Theta_i')^2 s_i'/S+2s_i'/S$$

La somme de $(\Theta_i')^2$ sur N échantillons et en divisant par N, conduit à la variance sur Θu. Notez bien que la somme du dernier terme est égale à 0.

$$\sum_{i=1}^{N}(\sigma(\Theta u'))^2=\sum (\Theta_i')^2/N = \sum_{i=1}^{N} 2(1-\bar{U}/S)- \sum(((\Theta_i')^2 s_i')/NS$$

Le terme $\sum(((\Theta_i')^2 s_i')/NS$, est égal à 0 si les déviations par rapport à la vitesse, ne sont pas corrélées avec une déviation de la direction. Cette hypothèse a été vérifiée lors de tests effectués sur les données de vent provenant de différents sites, par Campbell Scientific aux USA au « Air Resources Laboratory, NOAA, Idaho Falls, ID; and MERDI, Butte, MT. » Lors de ces test, la différence maximale dans l'équation

$$\sigma(\Theta u) = (\sum(\Theta_i')^2/N)^{1/2} \text{ et } \sigma(\Theta u) = (2(1- \bar{U}/S))^{1/2}$$

n'a jamais été supérieure à quelques degrés.

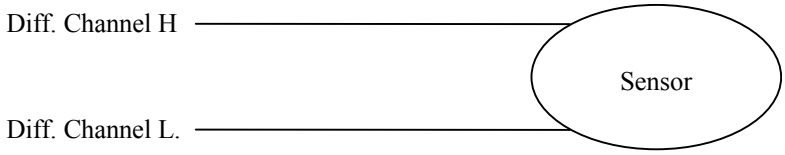
On parvient à la formule finale en convertissant les radians en degrés (57.296 degrés / radian).

$$\sigma(\Theta u) = (2(1- \bar{U}/S))^{1/2} = 81(1- \bar{U}/S)^{1/2}$$

Chapitre 7. Instructions de mesure

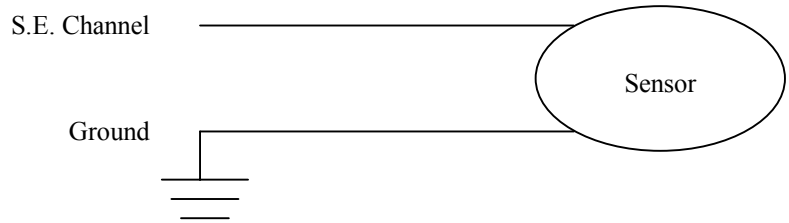
7.1 Mesures de tension

VoltDiff
(Dest, Reps, Range, DiffChan, RevDiff, SettlingTime, Integ, Mult, Offset)



Cette instruction mesure la différence de tension entre les entrées H. et L. d'une voie différentielle. Les entrée High et Low doivent chacune être comprise dans les ± 5V par rapport à la masse de la centrale de mesure (voir le chapitre « Mode Commun », paragraphe 3.2). Avec un multiplicateur égal à 1, et un offset égal à 0, le résultat est en millivolts ou Volts, selon l'étendue sélectionnée.

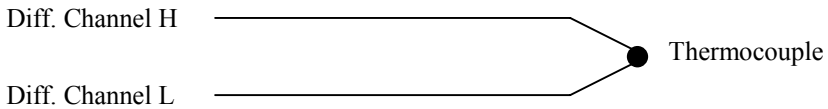
VoltSE
(Dest, Reps, Range, SEChan, MeasOfs, SettlingTime, Integ, Mult, Offset)



Cette instruction mesure la tension d'une entrée unipolaire, par rapport à la terre. Avec un multiplicateur égal à 1, et un offset égal à 0, le résultat est en millivolts ou Volts, selon l'étendue sélectionnée.

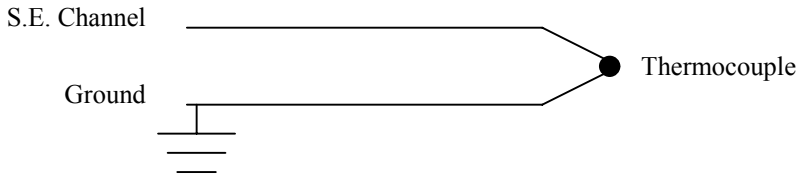
7.2 Mesures de thermocouple

TCDiff
(Dest, Reps, Range, DiffChan, TCType, TRef, RevDiff, SettlingTime, Integ, Mult, Offset)



Cette instruction mesure un thermocouple par une mesure de tension différentielle et calcule la température du thermocouple (en °C) en fonction du type de thermocouple sélectionné. L'instruction ajoute la tension mesurée à la tension équivalente calculée pour la température de référence par rapport à 0°C, et convertit la tension combinée
en température en °C. Les étendues qui sont spécifiées avec un code se terminant par C (par exemple mV2_5C) connectent de façon brève (10 µs) la voie d'entrée différentielle à la tension de référence avant d'effectuer la mesure de tension, afin de s'assurer qu'elle est dans l'étendue de mode commun et que le thermocouple n'est pas « ouvert ».

TCSE (Dest, Reps, Range, SEChan, TCType, TRef, MeasOfs, SettlingTime, Integ, Mult, Offset)



Cette instruction mesure un thermocouple par une mesure de tension unipolaire et calcule la température du thermocouple (en °C) en fonction du type de thermocouple sélectionné. L'instruction ajoute la tension mesurée à la tension équivalente calculée pour la température de référence par rapport à 0°C, et convertit la tension combinée en température en °C.

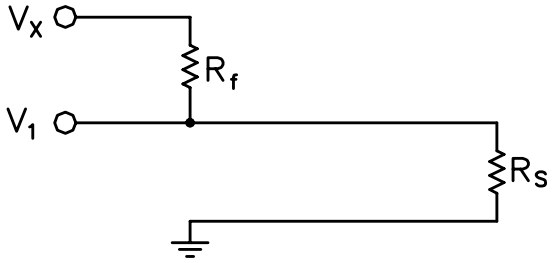
Tension et paramètres du thermocouple

<table>
<tr><td>Paramètre
& Type de Donnée</td><td colspan="3">Entrée</td></tr>
<tr><td>Dest
Variable ou ligne de données</td><td colspan="3">La variable dans laquelle on stocke le résultat de l'instruction. Lorsque les répétitions sont utilisées, les résultats sont stockés dans une ligne de données ayant le nom de la variable. La ligne de données doit être dimensionnées avec au moins autant de répétitions que celles utilisées.</td></tr>
<tr><td>Reps
Constante</td><td colspan="3">Le nombre de répétitions pour la mesure ou l'instruction.</td></tr>
<tr><td rowspan="13">Range
Constante</td><td>Code Alphanumérique</td><td>Etendue de mesure</td><td></td></tr>
<tr><td>mV5000</td><td>± 5000 mV</td><td></td></tr>
<tr><td>mV2500</td><td>± 2500 mV</td><td></td></tr>
<tr><td>mV250</td><td>± 250 mV</td><td></td></tr>
<tr><td>mV25</td><td>± 25 mV</td><td></td></tr>
<tr><td>mV7_5</td><td>± 7,5 mV</td><td></td></tr>
<tr><td>mV2_5</td><td>± 2,5 mV</td><td></td></tr>
<tr><td>Autorange</td><td>mV2_5 – mV5000</td><td>Choisit l'étendue (voir paragraphe 3.1)</td></tr>
<tr><td>mV250C</td><td>± 250 mV</td><td rowspan="5">Les étendues mV250C, mV25C, mV7_5C, et mV2_5C raménent la voie à l'intérieur de l'étendue de mode commun et vérifient qu'on n'a pas de circuit ouvert.</td></tr>
<tr><td>mV25C</td><td>± 25 mV</td></tr>
<tr><td>mV7_5C</td><td>± 7,5 mV</td></tr>
<tr><td>mV2_5C</td><td>± 2,5 mV</td></tr>
<tr><td>AutorangeC</td><td>mV2_5 – mV250</td><td>Choisit l'étendue en faisant un test « C »</td></tr>
<tr><td>DiffChan
Constante</td><td colspan="3">Le numéro de voie différentielle sur lequel on effectue la première mesure. Lorsqu'on utilise les «Reps», les mesures successives seront effectuées automatiquement sur les voies consécutives. Si la voie de mesure est négative, toutes les «Reps» se feront sur la même voie.</td></tr>
<tr><td>SEChan
Constante</td><td colspan="3">Le numéro de voie unipolaire sur lequel on effectuera la première mesure. Lorsque les «Reps» sont utilisées, les mesures successives seront automatiquement effectuées sur les voies consécutives. Si la voie de mesure est négative, toutes les «Reps» se feront sur la même voie.</td></tr>
<tr><td rowspan="8">TCType
Constante</td><td colspan="2">Code Alphanumérique</td><td>Type de Thermocouple</td></tr>
<tr><td colspan="2">TypeT</td><td>Cuivre Constantan</td></tr>
<tr><td colspan="2">TypeE</td><td>Chromel Constantan</td></tr>
<tr><td colspan="2">TypeK</td><td>Chromel Alumel</td></tr>
<tr><td colspan="2">TypeJ</td><td>Fer Constantan</td></tr>
<tr><td colspan="2">TypeB</td><td>Platine Rhodium</td></tr>
<tr><td colspan="2">TypeR</td><td>Platine Rhodium</td></tr>
<tr><td colspan="2">TypeS</td><td>Platine Rhodium</td></tr>
<tr><td>TRef
Variable</td><td colspan="3">Le nom de la variable qui est la référence de température pour la mesure de thermocouple.</td></tr>
</table>

<table>
<tr><td>Paramètre
& Type de Donnée</td><td colspan="3">Entrée</td></tr>
<tr><td>RevDiff
Constante</td><td>Code</td><td>Valeur</td><td>Résultat (L’inversion demande deux fois plus de temps)</td></tr>
<tr><td></td><td>False</td><td>0</td><td>Le signal est mesuré avec le côté haut par rapport au côté bas</td></tr>
<tr><td></td><td>True</td><td>≠ 0</td><td>Une seconde mesure est effectuée afin d’annuler les offsets</td></tr>
<tr><td>MeasOfs
Constante</td><td>Code</td><td>Valeur</td><td>Résultat : l’offset de tension de masse est soustrait de la mesure unipolaire.</td></tr>
<tr><td></td><td>False</td><td>0</td><td>La tension d’offset est corrigée à partir de l’étalonnage de fond</td></tr>
<tr><td></td><td>True</td><td>≠ 0</td><td>La tension d’offset est mesurée à chaque scrutation</td></tr>
<tr><td>SettlingTime
Constante</td><td colspan="3">Le temps en microsecondes qu’on laissera entre la configuration de la mesure (se connecter à la voie, configurer l’excitation) et la prise de la mesure (résolution d’ 1 microseconde)
<table>
<tr><th>Entrée</th><th>Etendue de mesure</th><th>Intégration</th><th>Durée de stabilisation</th></tr>
<tr><td>0</td><td>Toutes</td><td>250 µS</td><td>450 µS (défaut)</td></tr>
<tr><td>0</td><td>Toutes</td><td>_50Hz, _60 Hz</td><td>3 mS (défaut)</td></tr>
<tr><td>>=100</td><td>Toutes</td><td>Toutes</td><td>Nombre de µS entrées</td></tr>
</table></td></tr>
<tr><td>Integ
Constante</td><td colspan="3">Le temps passé pour l’intégration en microsecondes pour chacune des voies mesurées.
<table>
<tr><th>Entrée</th><th>Intégration</th></tr>
<tr><td>250</td><td>250 µS</td></tr>
<tr><td>_60Hz ou 16667</td><td>16,667 µS (réjection du bruit à 60 Hz)</td></tr>
<tr><td>_50 Hz ou 20000</td><td>20,000 µS (réjection du bruit 50 Hze)</td></tr>
</table></td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de donnée ou Expression</td><td colspan="3">Un multiplicateur et un offset par lequel on met à l’échelle les résultats de mesure brute. Voir la description de la mesure pour connaître l’unité de la mesure brute ; un multiplicateur de 1 et un offset de 0 sont nécessaires pour conserver les valeurs brutes. L’instruction TCDiff mesure un thermocouple et fournit une température en °C par exemple. Un multiplicateur de 1,8 et un offset de 32 convertira la température en °F.</td></tr>
</table>

7.3 Demi ponts

BrHalf
(Dest, Reps, Range, SEChan, ExChan, MeasPEx, ExmV, RevEx, SettlingTime, Integ, Mult, Offset)

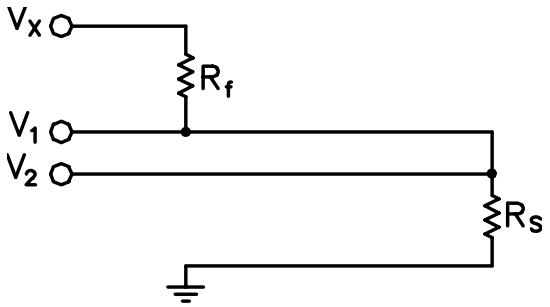


X = résultat avec :
mult = 1, offset = 0

$$X = \frac{V_1}{V_x} = \frac{R_s}{R_s + R_f}$$

Cette instruction applique une tension d’excitation, attend un certain temps, puis effectue une mesure de tension unipolaire. Le résultat de la mesure effectuée avec un multiplicateur égal à 1 et un offset égal à 0, est le rapport de la mesure de tension divisée par le tension d’excitation.

BrHalf3W
(Dest, Reps, Range, SEChan, ExChan, MeasPEx, ExmV, RevEx, SettlingTime, Integ, Mult, Offset)



X = résultat avec :
mult = 1, offset = 0

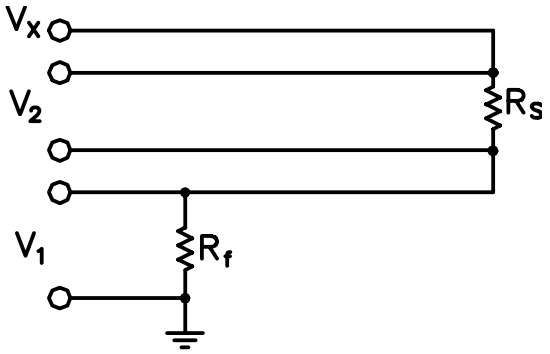
$$X = \frac{2V_2 - V_1}{V_x - V_1} = \frac{R_s}{R_f}$$

Cette instruction est utilisée afin de déterminer le rapport de la résistance du capteur par rapport à une résistance connue, en utilisant un fil propre pour la tension d'excitation en provenance du capteur afin de compenser la résistance du câble.

La séquence de mesure consiste à appliquer une tension d'excitation et à effectuer deux mesures de tension unipolaires sur des vois adjacentes : la première mesure est effectuée sur la résistance de référence, la seconde est effectuée sur le fil d'envoi de tension en provenance du capteur.

Les deux mesures sont utilisées afin de calculer la valeur résultante (multiplicateur = 1, offset = 0) qui est le rapport de la tension au travers du capteur par rapport à la tension au travers de la résistance de référence.

BrHalf4W
(Dest, Reps, Range1, Range2, DiffChan, ExChan, MeasPEx, ExmV, RevEx, RevDiff, SettlingTime, Integ, Mult, Offset)



X = résultat avec :
mult = 1, offset = 0

$$X = \frac{V_2}{V_1} = \frac{R_s}{R_f}$$

Cette instruction applique une tension d'excitation et effectue deux mesures de tension différentielles, puis inverse la polarité de l'excitation et répète les mesures. Les mesures sont effectuées sur des voies qui se suivent. Le résultat est la tension mesurée sur la seconde voie (V_2) divisé par la tension mesurée sur la première voie (V_1). Les branchements sont effectuées de telle sorte que V_1 est la chute de tension au travers de la résistance fixe (R_f), et que V_2 soit la chute de tension au travers du capteur (R_s). Le résultat avec un multiplicateur de 1 et un offset de 0 est V_2 / V_1 , ce qui est égal à R_S / R_f.

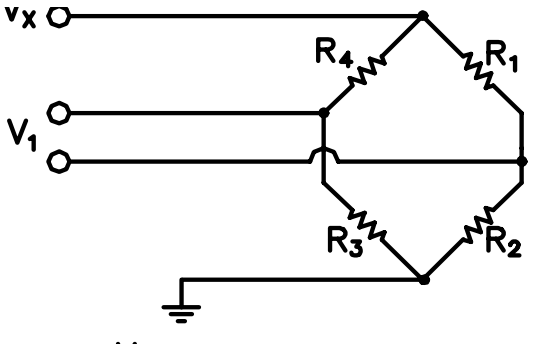
Paramètres des instructions «Half Bridge »

<table>
<tr><td>Paramètre
& Type de Donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Variable ou Ligne de données</td><td>La variable dans laquelle on stocke le résultat de l’instruction. Lorsque les répétitions sont utilisées, les résultats sont stockés dans une ligne de données ayant le nom de la variable. La ligne de données doit être dimensionnées avec au moins autant de répétitions que celles utilisées.</td></tr>
<tr><td>Reps
Constante</td><td>Le nombre de répétitions pour la mesure ou l’instruction.</td></tr>
<tr><td>Range
Constante</td><td>
<table>
<tr><th>Code Alphanumérique</th><th>Etendue de mesure</th><th></th></tr>
<tr><td>mV5000</td><td>± 5000 mV</td><td></td></tr>
<tr><td>mV2500</td><td>± 2500 mV</td><td></td></tr>
<tr><td>mV250</td><td>± 250 mV</td><td></td></tr>
<tr><td>mV25</td><td>± 25 mV</td><td></td></tr>
<tr><td>mV7_5</td><td>± 7,5 mV</td><td></td></tr>
<tr><td>mV2_5</td><td>± 2,5 mV</td><td></td></tr>
<tr><td>Autorange</td><td>mV2_5 – mV5000</td><td>Choisit l’étendue (voir paragraphe 3.1)</td></tr>
<tr><td>mV250C</td><td>± 250 mV</td><td rowspan="4">Les étendues mV250C, mV25C, mV7_5C et mV2_5C raménent la voie à l’intérieur du mode commun et vérifient qu’on n’a pas de circuit ouvert en entrée.</td></tr>
<tr><td>mV25C</td><td>± 25 mV</td></tr>
<tr><td>mV7_5C</td><td>± 7,5 mV</td></tr>
<tr><td>mV2_5C</td><td>± 2,5 mV</td></tr>
<tr><td>AutorangeC</td><td>mV2_5 – mV250</td><td>Choisit l’étendue en faisant un test « C »</td></tr>
</table>
</td></tr>
<tr><td>SEChan
Constante</td><td>Le numéro de voie unipolaire sur lequel on effectuera la première mesure. Lorsque les «Reps» sont utilisées, les mesures successives seront automatiquement effectuées sur les voies unipolaires suivantes. Si la voie de mesure est négative, toutes les «Reps» se feront sur cette même voie.</td></tr>
<tr><td>ExChan
Constante</td><td>Entrer le numéro de voie d’excitation qui excitera la première mesure.
<table>
<tr><th>Code Alphanum.</th><th>Code / Voie</th><th>Résultat</th></tr>
<tr><td>VX1
VX2
VX3</td><td>1
2
3</td><td>Les voies d’excitation commutées sont commutées à la tension d’excitation demandée durant la mesure, et sont désactivées entre les mesures.</td></tr>
</table>
</td></tr>
<tr><td>MeasPEx
Constante</td><td>Le nombre de capteurs à exciter avec la même voie d’excitation avant d’avancer automatiquement à la voie d’excitation suivante. Pour exciter tous les capteurs avec la même voie d’excitation ce nombre devra être égal au nombre de répétitions.</td></tr>
<tr><td>ExmV
Constante</td><td>La tension d’excitation en millivolts. L’étendue allouable est ± 2500mV. RevEx peut être utilisée afin d’exciter avec une polarité positive ou négative afin d’annuler les tensions d’offset.</td></tr>
<tr><td>RevEx
Constante</td><td>
<table>
<tr><th>Code</th><th>Valeur</th><th>Résultat (L’inversion demande 2 fois plus de temps pour s’exécuter)</th></tr>
<tr><td>False</td><td>0</td><td>Excite uniquement avec la tension entrée</td></tr>
<tr><td>True</td><td>≠0</td><td>Une seconde mesure est effectuée avec la tension à polarité inverse afin d’annuler les offsets.</td></tr>
</table>
</td></tr>
<tr><td>RevDiff
Constante</td><td>
<table>
<tr><th>Code</th><th>Valeur</th><th>Résultat (L’inversion demande 2 fois plus de temps pour s’exécuter)</th></tr>
<tr><td>False</td><td>0</td><td>Le signal est mesuré avec le côté positif par rapport au côté négatif.</td></tr>
<tr><td>True</td><td>≠0</td><td>Une seconde mesure est effectuée après l’inversion en entrée, afin d’annuler les offsets.</td></tr>
</table>
</td></tr>
<tr><td>SettlingTime
Constante</td><td>Le temps à attendre, en microsecondes, entre la configuration de la mesure (commuter à la voie, configurer l’excitation) et la prise de mesure. (la résolution est d’1 microseconde)
<table>
<tr><th>Entrée</th><th>Etendue de mesure</th><th>Intégration</th><th>Durée de stabilisation</th></tr>
<tr><td>0</td><td>Toutes</td><td>250 µS</td><td>450 µS (défaut)</td></tr>
<tr><td>0</td><td>Toutes</td><td>_50Hz, _60 Hz</td><td>3 mS (défaut)</td></tr>
<tr><td>>=100</td><td>Toutes</td><td>Toutes</td><td>µS entrées</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Paramètre
& Type de Donnée</td><td>Entrée</td></tr>
<tr><td>Integ
Constante</td><td>Le temps en microsecondes, passé sur l’intégration pour chacune des voies mesurées.
<table>
<tr><th>Entrée</th><th>Intégration</th></tr>
<tr><td>250</td><td>250 µS</td></tr>
<tr><td>_60Hz or 16667</td><td>16,667 µS (réjection du bruit à 60 Hz)</td></tr>
<tr><td>_50 Hz or 20000</td><td>20,000 µS (réjection du bruit à 50 Hz)</td></tr>
</table></td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de données ou Expression</td><td>Un multiplicateur et un offset par lequel mettre à l’échelle les résultats de mesure brute.</td></tr>
</table>

7.4 Ponts complets

BrFull
(Dest, Reps, Range, DiffChan, ExChan, MeasPEx, ExmV, RevEx, RevDiff, SettlingTime, Integ, Mult, Offset)

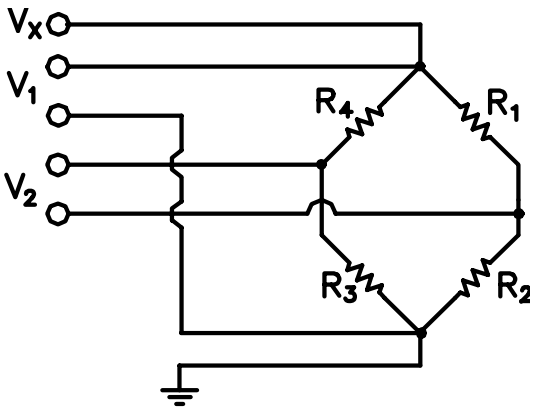


X = résultat avec
mult = 1, offset = 0

$$X = 1000\frac{V_1}{V_x} = 1000\left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$$

Cette instruction applique une tension d’excitation à un pont complet et effectue une mesure de tension différentielle à la sortie du pont complet. La valeur résultante (avec un multiplicateur = 1, offset = 0), est la tension mesurée en millivolts divisé par la tension d’excitation en volts (c’est à dire des millivolts par volt).

BrFull6W
(Dest, Reps, Range1, Range2, DiffChan, ExChan, MeasPEx, ExmV, RevEx, RevDiff, SettlingTime, Integ, Mult, Offset)



X = résultat avec
mult = 1, offset = 0

$$X = 1000\frac{V_2}{V_1} = 1000\left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$$

Cette instruction applique une tension d'excitation et effectue deux mesures de tension différentielle. Les mesures sont effectuées sur des voies de mesures qui se suivent. Le résultat est la tension mesurée sur la seconde voie (V_2) divisé par la tension mesurée sur la première voie (V_1). Le résultat est de 1000 fois V_2 / V_1 ou encore en unité de millivolts en sortie, par volt d'excitation. Les branchements sont effectués de telle sorte que V_1 est la mesure de la chute de tension à travers du pont complet, et V_2 est la mesure de la sortie du pont.

Paramètres de l'instruction « Full Bridge »

<table>
<tr><td>Paramètre
& Type de Donnée</td><td colspan="3">Entrée</td></tr>
<tr><td>Dest
Variable ou Ligne de données</td><td colspan="3">La variable dans la quelle on stocke le résultat de l'instruction. Lorsque les répétitions sont utilisées, les résultats sont stockés dans une ligne de données ayant le nom de la variable. La ligne de données doit être dimensionnées avec au moins autant de répétitions que celles utilisées.</td></tr>
<tr><td>Reps
Constante</td><td colspan="3">Le nombre de répétitions pour la mesure ou l'instruction.</td></tr>
<tr><td rowspan="13">Range
Constante</td><td>Code Alphanumérique</td><td>Etendue de mesure</td><td></td></tr>
<tr><td>mV5000</td><td>± 5000 mV</td><td></td></tr>
<tr><td>mV2500</td><td>± 2500 mV</td><td></td></tr>
<tr><td>mV250</td><td>± 250 mV</td><td></td></tr>
<tr><td>mV25</td><td>± 25 mV</td><td></td></tr>
<tr><td>mV7_5</td><td>± 7,5 mV</td><td></td></tr>
<tr><td>mV2_5</td><td>± 2,5 mV</td><td></td></tr>
<tr><td>Autorange</td><td>mV2_5 – mV5000</td><td>Choisit l'étendue (voir paragraphe 3.1)</td></tr>
<tr><td>mV250C</td><td>± 250 mV</td><td rowspan="4">Les étendues mV250C, mV25C, mV7_5C et mV2_5C ramènent la voie à l'intérieur du mode commun et vérifient qu'on n'a pas de circuit ouvert en entrée.</td></tr>
<tr><td>mV25C</td><td>± 25 mV</td></tr>
<tr><td>mV7_5C</td><td>± 7,5 mV</td></tr>
<tr><td>mV2_5C</td><td>± 2,5 mV</td></tr>
<tr><td>AutorangeC</td><td>mV2_5 – mV250</td><td>Choisit l'étendue en faisant un test « C »</td></tr>
<tr><td>DiffChan
Constante</td><td colspan="3">Le numéro de voie différentielle sur lequel on effectue la première mesure. Lorsqu'on utilise les «Reps», les mesures successives seront effectuées automatiquement sur les voies consécutives. Si la voie de mesure est négative, toutes les «Reps» se feront sur la même voie.</td></tr>
</table>

<table>
<tr><td>Paramètre
& Type de donnée</td><td colspan="4">Entrée</td></tr>
<tr><td>ExChan
Constante</td><td colspan="4">Entrer le numéro de voie d'excitation qui excitera la première mesure.
<table><tr><th>Code Alphanum.</th><th>Code / Voie</th><th>Résultat</th></tr><tr><td>VX1
VX2
VX3</td><td>1
2
3</td><td>Voies d'excitation commutées qui sont commutées à la tension fixée puis désactivées entre deux mesures.</td></tr></table></td></tr>
<tr><td>MeasPEx
Constante</td><td colspan="4">Le nombre de capteurs à exciter avec la même voie d'excitation avant d'avancer automatiquement à la voie d'excitation suivante. Pour exciter tous les capteurs avec la même voie d'excitation ce nombre devra être égal au nombre de répétitions.</td></tr>
<tr><td>ExmV
Constante</td><td colspan="4">La tension d'excitation en millivolts. L'étendue allouable est ± 2500mV. RevEx peut être utilisée afin d'exciter avec une polarité positive ou négative afin d'annuler les tensions d'offset.</td></tr>
<tr><td>RevEx
Constante</td><td colspan="4"><table><tr><th>Code</th><th>Valeur</th><th>Résultat (L'inversion demande 2 fois plus de temps pour s'exécuter)</th></tr><tr><td>False</td><td>0</td><td>Excite uniquement avec la tension entrée</td></tr><tr><td>True</td><td>≠0</td><td>Une seconde mesure est effectuée avec la tension à polarité inverse afin d'annuler les offsets.</td></tr></table></td></tr>
<tr><td>RevDiff
Constante</td><td colspan="4"><table><tr><th>Code</th><th>Valeur</th><th>Résultat (L'inversion demande 2 fois plus de temps pour s'exécuter)</th></tr><tr><td>False</td><td>0</td><td>Le signal est mesuré avec le côté positif par rapport au côté négatif.</td></tr><tr><td>True</td><td>≠0</td><td>Une seconde mesure est effectuée après l'inversion en entrée, afin d'annuler les offsets.</td></tr></table></td></tr>
<tr><td>SettlingTime
Constante</td><td colspan="4">Le temps à attendre, en microsecondes, entre la configuration de la mesure (commuter à la voie, configurer l'excitation) et la prise de mesure. (la résolution est d'1 microseconde)
<table><tr><th>Entrée</th><th>Etendue de mesure</th><th>Intégration</th><th>Durée de stabilisation</th></tr><tr><td>0</td><td>Toutes</td><td>250 µS</td><td>450 µS (défaut)</td></tr><tr><td>0</td><td>Toutes</td><td>_50Hz, _60 Hz</td><td>3 mS (défaut)</td></tr><tr><td>>=100</td><td>Toutes</td><td>Toutes</td><td>µS entrées</td></tr></table></td></tr>
<tr><td>Integ
Constante</td><td colspan="4">Le temps en microsecondes, passé sur l'intégration pour chacune des voies mesurées.
<table><tr><th>Entrée</th><th>Intégration</th></tr><tr><td>250</td><td>250 µS</td></tr><tr><td>_60Hz ou 16667</td><td>16,667 µS (réjection du bruit à 60 Hz)</td></tr><tr><td>_50 Hz ou 20000</td><td>20,000 µS (réjection du bruit à 50 Hz)</td></tr></table></td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de donnée ou Expression</td><td colspan="4">Un multiplicateur et un offset par lequel on met à l'échelle les résultats de mesure brute.</td></tr>
</table>

7.5 Excitation

ExciteV (ExChan, ExmV, XDelay)

Cette instruction fixe la voie d'excitation commutée spécifiée à la tension spécifiée. Le paramètre XDelay est utilisé afin de spécifier la durée de temps pendant laquelle l'excitation sera effective; après cette durée la voie d'excitation est désactivée et la centrale de mesure passe à la prochaine instruction. Si XDelay est à la valeur "0", la voie d'excitation sera commutée à la tension spécifiée et la tension sera tenue jusqu'à la fin de la scrutation du programme ou jusqu'à ce qu'une autre instruction ne change la configuration de la voie d'excitation.

Paramètre *& Type de donnée*	**Entrée**		
ExChan *Constante*	Entrer le numéro de la voie d'excitation qui excitera la première mesure.		
	Code Alphanum.	**Code/ Voie**	**Résultat**
	VX1 VX2 VX3	1 2 3	Voies d'excitation commutées qui sont commutées à la tension fixée puis désactivées entre deux mesures.
ExmV *Constante*	La tension d'excitation en millivolts. L'étendue allouable est ± 2500mV.		
XDelay	Spécifie la durée pendant laquelle l'excitation est active; après celle-ci la voie est désactivée et la centrale de mesure passe à la prochaine instruction. Si XDelay est fixée à 0, l'excitation sera active jusqu'à la fin de la scrutation du programme ou jusqu'à ce qu'une autre instruction configure l'excitation.		

SW12 (State)

Cette instruction est utilisée afin de fixer le 12V commuté à l'état haut ou bas.

La centrale de mesure a une sortie 12V commuté. Ce 12V commuté est utilisé afin de fournir une alimentation 12V continue à des périphériques externes. A température ambiante le 12V commuté peut fournir 900mA entre la voie SW-12 et la Masse (*Ground*). Le paramètre d'état (*State*) indique si le 12V commuté est à l'état haut (différent de zéro) ou bas (0).

NOTE

Le 12V commuté est non-régulé et peut fournir 900mA à 20°C et jusqu'à 630mA à 50°C. Un fusible à polymère ré-initialisable protège le 12V commuté contre les demandes trop fortes en courant. La ré-initialisation est effectuée en enlevant la charge ou en désactivant le 12V commuté durant quelques secondes.

Paramètre *& Type de donnée*	**Entrée**	
State *Constante Variable ou Expression*	La valeur de ce paramètre détermine si le 12V est actif ou non.	
	Valeur	**Résultat**
	0	Le port SW-12 est désactivé (0V)
	≠0	Le port SW-12 est activé (relié à la tension d'alimentation)

7.6 Mesures autonomes - Self Measurements

Battery (Dest)

Cette instruction lit la tension de la batterie et en stocke la valeur dans sa variable de destination. L'unité pour la tension de la batterie est le Volt.

Calibrate (Dest, AllRanges)

L'instruction "Calibrate" met l'étalonnage automatique de la CR1000, sous le contrôle du programme d'acquisition. Le fait d'incorporer l'instruction **Calibrate** au programme désactivera l'étalonnage automatique qui se produit normalement en tâche de fond (voir chapitre 3.8).

Les paramètres de l'instruction sont optionnels et ne sont utilisés qu'afin de mettre les résultats de l'étalonnage dans une ligne de donnée de variables. Si elle est utilisée sans paramètres, l'instruction **Calibrate** ne donnera aucune valeur en retour.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Ligne de donnée</td><td>Si elle est présente, la ligne de données doit contenir au moins 60 éléments (plus que cela si des excitations sont utilisées dans le programme). Sans paramètre, aucune donnée n’est affichée en retour.</td></tr>
<tr><td>AllRanges
Constante</td><td>Option afin d’étalonner les étendues de mesures non utilisées. Dest doit être entré avant le paramètre AllRanges
<table>
<tr><th>Code Alphanum.</th><th>Valeur</th><th>Résultat</th></tr>
<tr><td>False</td><td>=0</td><td>N’étalonne que les étendues de mesures utilisées.</td></tr>
<tr><td>True</td><td>≠0</td><td>Etalonne toutes les étendues de mesure.</td></tr>
</table></td></tr>
</table>

NOTE

Dans la plupart des cas l’étalonnage en tâche de fond est adéquat et l’instruction **Calibrate** ne devrait pas être utilisée dans le programme.

Il existe trois cas où l’on peut utiliser l’instruction **Calibrate** :

1) Avec le jeu normal des instructions il n’y a pas assez de temps pour accomplir l’étalonnage en tâche de fond, mais le programme peut périodiquement s’arrêter de prendre des mesures et lancer l’étalonnage sans une scrutation séparée.
2) La CR1000 sera sujette à des changements de températures très rapides et dans ce cas l’instruction **Calibrate** permettra d’étalonner la centrale avant chaque nouveau jeu de mesures prises.
3) Le programme est lancé par un technicien de maintenance qui regarde spécialement les résultats de l’instruction **Calibrate**. (Les valeurs d’étalonnage sont aussi disponibles dans le tableau d’état, sans avoir besoin d’utiliser un programme spécifique.)

S’il n’y a pas assez de temps laissé lors d’une scrutation rapide (*fast scan*), afin que l’étalonnage en tâche de fond s’effectue, le message suivant sera affiché à la compilation : “Warning when Fast Scan x is running background calibration will be disabled.” (x est le nombre de scrutations rapides où la première scrutation entrée dans le programme est 1, la suivante est 2 etc.). Si vous voyez ce message vous avez le choix de laisser la scrutation s’effectuer sans étalonnage (si la température reste constante il y aura peu de dérive, voir chapitre 3.8), de réduire le nombre de mesures ou le temps qu’il faut pour les effectuer (par exemple réduire les durées d’intégration), ou de passer de façon périodique dans une scrutation qui exécute l’étalonnage.

Dans le cas de changements rapides de température, tel que faire passer un véhicule d’un état d’équilibre à –30°C à une journée ensoleillée d’Arizona ou en Provence, le fait d’exécuter l’instruction d’étalonnage à l’intérieur du programme peut améliorer la précision des mesures. Le changement doit être rapide afin de nécessiter cela; l’étalonnage de fond filtre les nouvelles lectures et a une constante de temps d’approximativement 36 secondes (63% de réponse à un changement d’état). Lorsque l’instruction d’étalonnage est exécutée dans le programme, l’étalonnage est effectué de façon complète à chaque fois.

A moins que l’option “AllRanges” soit sélectionnée, l’instruction d’étalonnage ne mesure que les combinaisons d’étendues de mesure et d’intégration qui se produisent dans l’exécution du programme de mesure. Pour les étalonnages d’intégration zéro et 250 μs, de multiples mesures sont moyennées afin d’obtenir les valeurs d’étalonnage. L’étalonnage à 250 μs d’intégration moyenne cinq mesures et l’intégration à zéro moyenne dix mesures.

L’instruction d’étalonnage peut être intégrée dans séquence de scrutation rapide ou lente. Dans une scrutation rapide, la totalité de l’étalonnage est effectuée d’une traite. Dans une séquence de scrutation lente, l’étalonnage est séparé en sections qui peuvent être mises en tranches et effectuées à la suite de scrutations rapides.

S'il est nécessaire de mettre à jour l'étalonnage plus rapidement que cela n'est effectué par l'étalonnage en tâche de fond, il est bon d'essayer de lancer l'instruction **Calibrate** dans la scrutation rapide, là où sont effectuées les mesure. S'il n'y a pas assez de temps pour que l'instruction s'exécute à cet endroit, l'instruction peut être placée dans une séquence de scrutation lente, mais souvenez-vous que si la scrutation lente ne s'exécute pas plus souvent que toutes les 40 secondes, l'étalonnage ne sera pas rafraîchi plus rapidement qu'il ne le serait avec l'étalonnage en tâche de fond.

Le fait de lancer l'instruction **Calibrate** dans une séquence de scrutation lente n'est pas une bonne option lorsqu'il n'y a pas assez de temps pour effectuer l'étalonnage en tâche de fond. L'instruction nécessite plus de temps à cause des mesures multiples pour les intégrations à zéro et 250 µs.

Lorsque les résultats de l'étalonnage sont placés dans une ligne de données, celle-ci doit contenir au minimum 60 éléments, et plus que cela si le programme comprend des instructions qui utilisent des voies d'excitation. Les valeurs de l'étalonnage seront ordonnées de la façon suivante, suivies des valeurs de l'étalonnage des excitations s'il y en a. Si une étendue de mesure n'est pas étalonnée, la valeur retournée sera 0 pour le gain et l'offset.

TABLEAU 7.7-1. Décodage des valeurs retournées pour le étalonnage	
Elément de la ligne de données	**Description**
1	Intégration zéro, offset de voie unipolaire 5000 mV
2	Intégration zéro, offset de voie différentielle 5000 mV
3	Intégration zéro, gain de 5000 mV
4	Intégration zéro, offset de voie unipolaire 1000 mV
5	Intégration zéro, offset de voie différentielle 1000 mV
6	Intégration zéro, gain de 1000 mV
7	Intégration zéro, offset de voie unipolaire 200 mV
8	Intégration zéro, offset de voie différentielle 200 mV
9	Intégration zéro, gain de 200 mV
10	Intégration zéro, offset de voie unipolaire 50 mV
11	Intégration zéro, offset de voie différentielle 50 mV
12	Intégration zéro, gain de 50 mV
13	Intégration zéro, offset de voie unipolaire 20 mV
14	Intégration zéro, offset de voie différentielle 20 mV
15	Intégration zéro, gain de 20 mV
16	250µSec d'intégration, offset de voie unipolaire 5000 mV
17	250µSec d'intégration, offset de voie différentielle 5000 mV
18	250µSec d'intégration, gain de 5000 mV
19	250µSec d'intégration, offset de voie unipolaire 1000 mV
20	250µSec d'intégration, offset de voie différentielle 1000 mV
21	250µSec d'intégration, gain de 1000 mV
22	250µSec d'intégration, offset de voie unipolaire 200 mV
23	250µSec d'intégration, offset de voie différentielle 200 mV
24	250µSec d'intégration, gain de 200 mV
25	250µSec d'intégration, offset de voie unipolaire 50 mV
26	250µSec d'intégration, offset de voie différentielle 50 mV
27	250µSec d'intégration, gain de 50 mV
28	250µSec d'intégration, offset de voie unipolaire 20 mV
29	250µSec d'intégration, offset de voie différentielle 20 mV
30	250µSec intégration « scriptsizeintegrate », gain de 20 mV
31	Réjection 60 Hz, offset de voie unipolaire 5000 mV
32	Réjection 60 Hz, offset de voie différentielle 5000 mV
33	Réjection 60 Hz, gain de 5000 mV
34	Réjection 60 Hz, offset de voie unipolaire 1000 mV
35	Réjection 60 Hz, offset de voie différentielle 1000 mV
36	Réjection 60 Hz, gain de 1000 mV
37	Réjection 60 Hz, offset de voie unipolaire 200 mV
38	Réjection 60 Hz, offset de voie différentielle 200 mV
39	Réjection 60 Hz, gain de 200 mV
40	Réjection 60 Hz, offset de voie unipolaire 50 mV
41	Réjection 60 Hz, offset de voie différentielle 50 mV
42	Réjection 60 Hz, gain de 50 mV
43	Réjection 60 Hz, offset de voie unipolaire 20 mV
44	Réjection 60 Hz, offset de voie différentielle 20 mV
45	Réjection 60 Hz, gain de 20 mV
46	Réjection 50 Hz, offset de voie unipolaire 5000 mV
47	Réjection 50 Hz, offset de voie différentielle 5000 mV
48	Réjection 50 Hz, gain de 5000 mV
49	Réjection 50 Hz, offset de voie unipolaire 1000 mV
50	Réjection 50 Hz, offset de voie différentielle 1000 mV
51	Réjection 50 Hz, gain de 1000 mV
52	Réjection 50 Hz, offset de voie unipolaire 200 mV
53	Réjection 50 Hz, offset de voie différentielle 200 mV
54	Réjection 50 Hz, gain de 200 mV
55	Réjection 50 Hz, offset de voie unipolaire 50 mV
56	Réjection 50 Hz, offset de voie différentielle 50 mV
57	Réjection 50 Hz, gain de 50 mV
58	Réjection 50 Hz, offset de voie unipolaire 20 mV
59	Réjection 50 Hz, offset de voie différentielle 20 mV
60	Réjection 50 Hz, gain de 20 mV

InstructionTimes (Dest)

L'instruction **InstructionTimes** donne comme résultat le temps d'exécution de chaque instruction dans le programme.

L'instruction **InstructionTimes** met en mémoire la ligne de donnée "Dest" avec le temps que prend chaque instruction du programme, à s'exécuter (en microsecondes). **InstructionTimes** doit être écrite avant la balise BeginProg.

Chaque élément de la ligne de donnée correspond à une ligne du programme. Pour que toutes les instructions puissent être prise en compte, la ligne de donnée doit comporter autant ou plus d'éléments que de lignes écrites dans le programme, lignes blanches et commentaires compris. La ligne de donnée Dest doit aussi être dimensionnée en tant qu'entier de type long (par exemple : Public Array(20) AS LONG).

A noter que le temps d'exécution pour une instruction peut varier. Cela prendra plus de temps, par exemple, d'exécuter une instruction lorsque la centrale de mesure est en train de communiquer avec un autre appareil.

PanelTemp (Dest, Integ)

Cette instruction mesure la température du bornier en °C.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Variable</td><td>La variable dans laquelle on stock le résultat de l'instruction.</td></tr>
<tr><td>Integ
Constante</td><td>Le temps d'intégration (en micro seconds) pour chaque voie mesurée.
<table>
<tr><th>Entrée</th><th>Intégration</th></tr>
<tr><td>250</td><td>250 µS</td></tr>
<tr><td>_60Hz or 16667</td><td>16,667 µS (reject 60 Hz noise)</td></tr>
<tr><td>_50 Hz or 20000</td><td>20,000 µS (reject 50 Hz noise)</td></tr>
</table></td></tr>
</table>

7.7 Digital I/O

CheckPort (Port)

CheckPort est une fonction qui donne comme résultat l'état d'un port de contrôle. CheckPort donne la valeur Vrai [True (-1)] si le port de contrôle spécifié est à l'état haut, ou Faux [False (0)] si le port est à l'état bas. CheckPort peut être utilisée en juxtaposition avec une expression (par exemple Variable = CheckPort (Port)) ou en tant qu'expression.

CheckPort n'a qu'un seul paramètre, "Port", qui est le numéro de port de contrôle (entre 1 et 8) à tester.

Attention : La variable envoyée en retour ne sera peut être pas valide si on utilise le port de contrôle en tant que port série ou port de comptage d'impulsion.

PeriodAvg (Dest, Reps, Range, SEChan, Threshold, PAOption, Cycles, Timeout, Mult, Offset)

Cette instruction mesure la période d'un signal sur n'importe laquelle des voies de mesure unipolaires. Le nombre de cycles spécifiés est temporisé avec une résolution de 92ns, donnant une résolution à la période mesurée qui sera de 92ns divisé par le nombre de cycles choisis.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Variable or Ligne de donnée</td><td>La variable dans laquelle on enregistre les résultats de l’instruction. Lorsque les répétitions sont utilisées, les résultats sont stockés dans des lignes de données avec le nom de la variable. Un ligne de donnée doit être dimensionnée afin d’avoir les éléments pour toutes les répétitions.</td></tr>
<tr><td>Reps
Constante</td><td>Le nombre de répétitions pour l’instruction, à exécuter sur les voies adjacentes.</td></tr>
<tr><td>Range
Constante</td><td>L’étendue de mesure en tension pour la mesure, qui détermine le gain appliqué au signal avant une détection de dépassement de zéro (zero-crossing detector). La fréquence maximum décroît lorsque le gain augmente.
<table>
<tr><th rowspan="2">Code d’étendue de mesure</th><th rowspan="2">Gain</th><th colspan="2">Signal (pk-pk)[1]</th><th rowspan="2">Largeur d’impulsion Minimum</th><th rowspan="2">Fréquence[2] Maximum</th></tr>
<tr><th>Min</th><th>Max</th></tr>
<tr><td>mV250</td><td>1</td><td>500 mV</td><td>10.0 V</td><td>2.50 µs</td><td>200 kHz</td></tr>
<tr><td>mV25</td><td>10</td><td>10 mV</td><td>2.0 V</td><td>10.00 µs</td><td>50 kHz</td></tr>
<tr><td>mV7_5</td><td>33</td><td>5 mV</td><td>2.0 V</td><td>62.00 µs</td><td>8 kHz</td></tr>
<tr><td>mV2_5</td><td>100</td><td>2 mV</td><td>2.0 V</td><td>100.00 µs</td><td>5 kHz</td></tr>
</table>
[1] Les signaux doivent dépasser le seuil afin de faire basculer le comparateur de tensions.
[2] Fréquence Max.=1/(deux fois la largeur d’impulsion mini.) pour 50% du cycle des signaux</td></tr>
<tr><td>SEChan
Constante</td><td>Le numéro de la voie unipolaire sur laquelle effectuer la première mesure. Si on entre un nombre négatif, toutes les répétitions utilisent la même voie.</td></tr>
<tr><td rowspan="2">Threshold
Constante
PAoption</td><td>La tension en millivolts, que le signal en entrée doit dépasser afin qu’un comptage soit effectué. Pour un signal centré sur la masse de la CR1000 (Figure 7.7-1), le seuil devrait être 0. Si le signal en entrée est un signal CMOS 0 à 5 V, alors un seuil de 2500mV ferait en sorte que le comparateur de tensions commute à 2,5V.</td></tr>
<tr><td>Spécifie si on veut sauvegarder le résultat de la période en µs ou la fréquence en Hz.
<table>
<tr><th>Code Numérique</th><th>Etendue de mesure</th></tr>
<tr><td>0</td><td>Ce qui est retourné est la période du signal</td></tr>
<tr><td>1</td><td>Ce qui est retourné est la fréquence du signal</td></tr>
</table></td></tr>
<tr><td>Cycles
Constante</td><td>Le nombre de cycles à mesurer afin de calculer la moyenne.</td></tr>
<tr><td>Timeout
Constante</td><td>La durée maximum (en msec) que la centrale de mesure attendra afin que le nombre de cycles spécifiés soit détecté pour calculer la moyenne.</td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de donnée ou Expression</td><td>Un multiplicateur et un offset par lequel vous mettez à l’échelle les résultats de mesure brute.</td></tr>
</table>

Les signaux à bas niveau sont amplifies avant d’être soumis au comparateur de tension afin d’effectuer la mesure de période moyenne. Le comparateur de tension interne est référencé au seuil rentré par l’utilisateur. Le paramètre du seuil permet à un utilisateur de mettre une référence au comparateur de tension interne, qui soit différente de 0V. Par exemple un seuil de 2500mV permet à un signal variant entre 0 et 5V d’être compris par le comparateur de tension, sans avoir besoin de mettre en place un quelconque circuit de conditionnement supplémentaire. Le seuil permet de connecter des signaux numériques standard, mais il n’est pas recommandé pour des capteurs à signaux à faible amplitude. Pour des capteurs dont l’amplitude est inférieure à 20mV pic à pic (*pk-pk*), il est recommandé d’utiliser une capacité de blocage CC afin de centrer le signal autour de la masse de la CR1000 (seuil = 0); voir la figure 7.7-1. C’est à cause de la variation de la tension d’offset en plus de la précision (± 10 mV) et de la résolution (1,2 mV) limitée, lorsqu’on utilise un seuil différent de 0.

Les besoins minimum pour la largeur de l'impulsion augmentent (la fréquence maximum diminue) lorsque le gain augmente, comme cela est indiqué dans le paramètre « range ». Les signaux plus larges que le maximum spécifié pour une étendue, satureront le gain et éviteront qu'il y ait un fonctionnement jusqu'au maximum spécifié. Des diodes positionnées dos à dos, comme à la figure 7.7-1, sont recommandées afin de limiter les signaux à large amplitude afin qu'ils restent à l'intérieur de la gamme spécifiée par l'étendue de mesure.

ATTENTION

Les signaux bruités avec une tension de transition lente au niveau du seuil, peuvent induire des comptages supplémentaires au niveau du point de comparaison. Un comparateur de tension avec une hystérésis de 20mV suit les étapes de gain en tension. L'hystérésis à laquelle on se réfère effectivement est égale à 20mV divisé par le gain de tension sélectionné. L'hystérésis à laquelle on se réfère effectivement sur l'étendue de mesure ±25mV est de 2mV; par conséquent 2mV de bruit sur le signal en entrée peut causer des comptages supplémentaires. **Pour de meilleurs résultats il est préférable de choisir l'étendue de mesure en entrée la plus large (le plus petit gain) qui sera compatible avec les besoins minima du signal en entrée.**

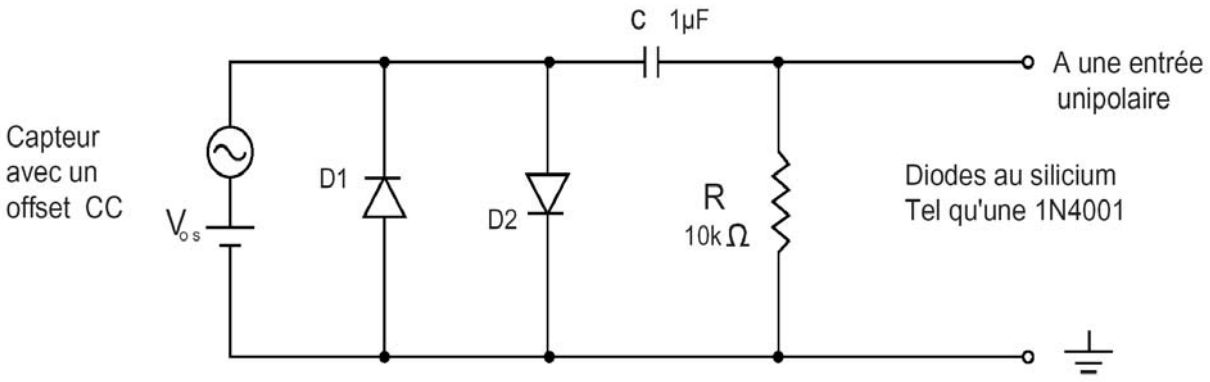


FIGURE 7.7-1. Circuit de conditionnement en entrée pour des fréquences moyennes de bas / haut niveau

La Figure 7.7-1 montre un circuit qui couple de façon capacitive un signal en entrée afin de le centrer autour de la masse et aussi afin de limiter l'amplitude du signal en entrée pour qu'il reste dans l'étendue acceptable. La capacité C bloque le courant continu afin de retirer l'offset de tension. La résistance R1 est utilisée afin de biaiser le côté de la voie d'entrée de la centrale de mesure, jusqu'à la masse. La réactance de la capacité de blocage CC ($Xc = (2 \cdot \pi \cdot f \cdot C)-1$) et de la résistance R1 forment un diviseur de tension à faibles fréquences ($R1/(R1 + Xc)$), qui atténuent le signal appliqué. Cette atténuation donne une valeur plus basse pour les fonctionnements en basse fréquence, et la taille minimum de R1. Le circuit atténue le signal d'entrée par un facteur 2 à 16 Hz.

Les diodes D1 et D2 au silicium, montées dos à dos (ou en opposition), fournissent une protection ESD (décharge électrostatique) entre la capacité C et le capteur, et limitent aussi l'amplitude des capteurs à signaux de forte amplitude. Ces diodes limitent les signaux de forte amplitude à approximativement 1,4V pic à pic, ce qui est compris dans l'étendue de mesure recommandée en entrée pour tous les codes d'étendue de mesure. Les diodes D1 et D2 ainsi que la résistance R1, sont recommandées afin de limiter les signaux de capteurs à forte amplitude, même lorsqu'une capacité de cloquage CC n'est pas utilisée. Les capteurs qui fournissent de fortes tensions peuvent engendrer le passage de forts courants au travers des diodes montées dos à dos. Une résistance de limitation de courant peut être souhaitable afin de minimiser ces courants dans certaines situations.

Le courant qui passe au travers des diodes inversées, peuvent aussi induire des offsets de mesure de tension unipolaire si les retours sont dirigés vers les voies ⏚ . Des offsets de tension unipolaire allant jusqu'à 2 µV/mA de courant qui passent au travers de la borne ⏚, peuvent être induits au travers du bornier supérieur. Les diodes montées dos à dos peuvent être reliées aux bornes G au lieu des bornes ⏚ , si cela pose un problème.

PortsConfig (Mask, Function)

L'instruction PortsConfig est utilisée afin de configurer un ou plusieurs ports de contrôle en temps qu'entrée ou sortie.

Par défaut les ports sont configurés en entrée. L'instruction PortsConfig peut être nécessaire si un port est configuré en sortie par le biais d'une instruction WriteIO ou PortSet et qu'elle nécessite ensuite de fonctionner en tant qu'entrée.

Paramètre *& Type de donnée*	**Entrée**
Mask	Le paramètre "Mask" est utilisé afin de sélectionner quel port sera affecté par cette instruction. C'est une représentation binaire du port (en lisant de gauche à droite les ports sont représentés dans l'ordre 8, 7, 6.. 1). Si l'option d'un port est à l'état "1", la centrale configure ce port. Les nombres compris en tant que nombre binaire s'ils sont précédés de "&B". Les zéros en début de ligne peuvent être ignorés. Ainsi &B110 permettra de configurer les ports 2 et 3 sur la base du paramètre de la fonction.
Function	Le paramètre de la fonction est utilisé afin de configurer le port. Une valeur binaire est entrée afin de configurer chaque emplacement des ports. 0 configure le port en tant qu'entrée; 1 le configure en tant que sortie. En utilisant l'exemple de "mask" ci-dessus, si le paramétre de la fonction est mis à &B110, les ports 3 et 2 seront configurés en sortie (le port 1 utilise le code pour être configuré en entrée mais il n'est pas affecté étant donné le « mask » utilisé).

PortGet (Dest, Port)

La fonction PortGet est utilisée afin de lire l'état de l'un des huit ports de contrôle.

Remarques

Cette instruction lira l'état des ports spécifié et placera le résultat dans la variable de destination "Dest".

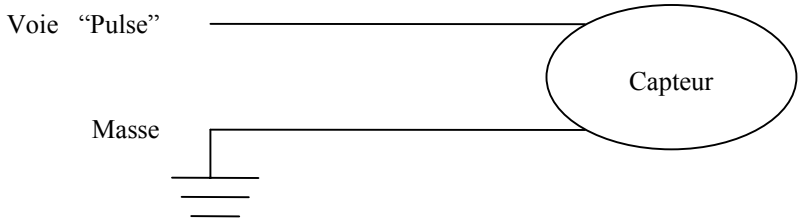
Paramètre *& Type de donnée*	**Entrée**
Dest *Variable*	La variable dans laquelle on stock le résultat de l'instruction. Il sera stocké la valeur 1 si le port est à l'état haut ; 0 s'il est à l'état bas.
Port *Constante*	Le numéro du port de contrôle (entre 1 et 8) pour lequel on souhaite connaître l'état.

PortSet (Port, State)

Cette instruction configurera le port de contrôle à l'état haut ou bas.

Paramètre *& Type de donnée*	**Entrée**		
Port *Constant, Variable, ou Expression*	Le numéro de port de contrôle (entre 1 et 8) à configurer via cette instruction.		
State *Constante, Variable, ou Expression*	L'état (haut ou bas) dans lequel on souhaite configurer le port de contrôle.		
	Code	**Valeur**	**Etat**
	True	0	Bas (*Low*)
	False	≠ 0	Haut (*High*)

PulseCount (Dest, Reps, PChan, PConfig, POption, Mult, Offset)



Paramètre *& Type de donnée*	**Entrée**
Dest *Variable ou Ligne de donnée*	La variable dans laquelle on stock les résultats de l'instruction. Lorsqu'on utilise des répétitions, les résultats sont stockés dans la ligne de donnée portant le nom de la variable. Une ligne de donnée doit être dimensionnée afin de contenir autant d'élément que de répétitions.
Reps *Constante*	Le nombre de repetitions pour la mesure ou l'instruction.
PChan *Constante*	Le numéro de la voie d'impulsion (voie « Pulse » 1 ou 2) pour la mesure. Le port de contrôle peut être utilisée afin de mesurer des hautes fréquence ou des contacts secs. On place alors un 1 devant le numéro du port de contrôle afin de l'identifier ; le port de contrôle 7 Sera par exemple défini par le code 17.
PConfig *Constante*	Un code qui spécifie le type d'impulsion à mesurer en entrée. **Code** : **Configuration en entrée** 0 : Haute fréquence (*High Frequency*) 1 : CA bas niveau (*Low Level AC*) 2 : Contact sec (*Switch Closure*)
POption	Un code qui détermine si les valeurs brutes (multiplicateur = 1 et offset = 0) sont données en tant que comptage ou fréquence. La moyenne glissante peut être utilisée afin de lisser les lectures lorsqu'une faible fréquence, comparée à la scrutation du programme, cause de fortes fluctuations dans les mesures lues. **Code** : **Résultat** 0 : Comptages 1 : Fréquence (Hz) ; comptages / intervalle de scrutation en seconde >1 : Moyenne glissante de la fréquence. Le nombre entré est la période de temps sur laquelle la fréquence est moyennée, en millisecondes.
Mult, Offset *Constante, Variable, Ligne de données ou Expression*	Un multiplicateur et un offset par lesquels on met à l'échelle les valeurs brutes mesurées. Voir la description de la mesure pour les unités des valeurs brutes ; un multiplicateur des 1 et un offset de 0 sont nécessaires afin d'obtenir les unités brutes. L'instruction **TCDiff** mesure par exemple un thermocouple et fournit des °C. Un multiplicateur de 1,8 et un offset de 32 donneront une température en °F.

L'instruction PulseCount est utilisée afin de mesurer des comptages ou des fréquences sur une des voies d'impulsion.

NOTE L'instruction PulseCount ne peut pas être utilisée dans une scrutation lente.

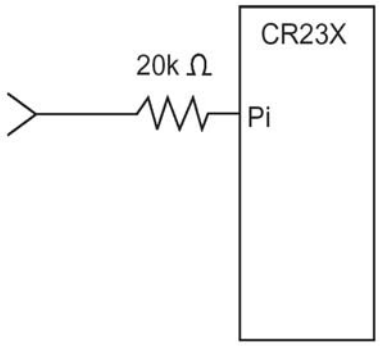


FIGURE 7.7-2. Conditionnement d'impulsions à forte tension

La tension d'entrée maximum sur une voie d'impulsion est de ±20 V. Référez-vous à la figure 7.7-2 s'il est nécessaire de réduire la tension en entrée.

- Voies d'impulsion (*Pulse Channels*)
  Tension maxi. En entrée (*Maximum Input Voltage*) : ±20 V

Entrée haute fréquence

- Voies d'impulsion (*Pulse Channels*)
  Largeur mini. de l'impulsion (*Minimum Pulse Width*) : 1,2 µs
  Fréquence maxi. (*Maximum Frequency*) : 400 kHz
  (50% Duty Cycle)
  Seuil de transition basse (*Lower Threshold*) : 1.5 V*
  Seuil de transition haute (*Upper Threshold*) : 3.5 V*

Lorsqu'une voie d'impulsion est configurée pour des impulsions à haute fréquence, il y a une résistance interne de 100 kohm qui ramène le signal à 5V sur la voie d'impulsion. Cette résistance est adaptée pour les appareils qui fournissent un signal de type collecteur ouvert (open-collector) pour des entrées haute fréquence.

* Des transitions plus importantes sont nécessaires à fréquence importante à cause du filtre RC qui a une constante de temps de 1,2 µs. Des signaux de fréquence pouvant aller jusqu'à 400 kHz seront comptabilisé s'ils sont centrés autour de +2,5 V avec une déviation ≥ ± 2,5 V et ≥ 1,2 µs.

CA bas niveau (voie d'impulsion uniquement)

Hystérésis en entrée : 15 mV
Tension maximum en entrée : 20 V pic à pic
Tension en entrée et étendue de fréquence

20 mV	de 1.0 Hz à 20 Hz
200 mV	de 0.5 Hz à 200 Hz
2000 mV	de 0.3 Hz à 10,000 Hz
5000 mV	de 0.3 Hz à 20,000 Hz

Contact sec

- Voies d'impulsion (*Pulse Channels*)
  Un contact sec est branché entre P1 ou P2 et une voie de masse analogique. Lorsque le contact est ouvert, la CR1000 ramène le signal de la voie d'impulsion à 5V au travers de la résistance de 100 kOhm. Lorsque le contact est fermé, la voie d'impulsion est ramenée vers la masse. Le comptage est incrémenté lorsque le contact s'ouvre.
  Temps minimum de contact fermé : 5 ms
  Temps minimum de contact ouvert : 6 ms
  Durée maximum de rebond : 1 ms ouvert sans être comptabilisé

PulseCountReset

PulseCountReset est utilisée afin de ré-initialiser les compteurs d'impulsion et la moyenne glissant utilisée dans l'instruction de comptage d'impulsion. Le compteur 16 bits peut aller jusqu'à la valeur 65535. S'il se produit plus de 65535 impulsions, cela conduira à une valeur hors gamme. A chaque nouvelle scrutation la CR1000 lit les comptages accumulés depuis la scrutation précédente puis ré-initialise le compteur. Si la scrutation s'arrête, comme cela est le cas dans un programme qui possède plus d'une boucle de scrutation, le compteur continue à accumuler des comptages jusqu'à ce qu'une autre scrutation soit initialisée, ou jusqu'à ce qu'il y ait une valeur hors gamme. Si la moyenne glissante est utilisée, les valeurs hors gamme seront ajoutées à la moyenne jusqu'à ce que la période pour effectuer la moyenne soit terminée (par exemple avec une moyenne glissante sur 1000 milisecondes, la valeur hors gamme sera la valeur provenant de l'instruction **PulseCount** jusqu'à ce que la seconde soit terminée). Le fait de ré-initialiser la moyenne avant de (re) démarrer la scrutation, évite cela.

PulsePort (Port, Delay)

Cette instruction bascule l'état d'un port de contrôle, effectue une temporisation de la durée spécifiée, bascule l'état du port puis effectue un délai de nouveau. La présence du second délai permet de créer un cycle à 50% de temps d'utilisation. L'instruction a les paramètres suivants :

Paramètre *& Type de donnée*	**Entrée**
Port *Constante*	Le port de contrôle qui doit être piloté par cette instruction. Le numéro du port (entre 1 et 8) est entré.
Delay *Ligne de donnée*	La durée, en microsecondes, que l'instruction devra attendre après chaque basculement d'état du port.

ReadIO (Dest, Mask)

ReadIO est utilisée afin de lire l'état des ports de contrôle de la CR1000. Il y a 8 ports I/O. L'état de ces ports est considéré comme un nombre binaire avec un port activé (+5V) lorsque le nombre vaut 1, et un port désactivé (0 V) lorsque celui-ci vaut 0. Si par exemple les ports n° 1 et 3 sont activés, et que les autres ports sont désactivés, la représentation binaire de l'état des ports sera 00000101, ou bien la valeur décimale 5. Le paramètre de masque est utilisé afin de sélectionner quel est le port que l'on souhaite lire, et ce masque est aussi une représentation binaire des ports de contrôle. L'utilisation du 1 signifiera que l'on souhaite prendre en compte l'état du port de contrôle ; l'utilisation du 0 signifiera qu'on souhaite l'ignorer (le masque donne la valeur 1 si l'état du port doit être contrôlé et que sont état est activé ; il donne la valeur 0 s'il doit être ignoré, quel que soit son état). CRBasic donne la possibilité d'entrer les nombres en format binaire si l'on précède les nombres par « &B ». Si par exemple on entre un masque de « &B100 » (les zéros qui précèdent peuvent être omis lorsqu'on utilise le format binaire, tout comme avec le format décimal) et que les ports 1 et 3 sont activés comme cela est le cas dans l'exemple qui précède, alors le résultat de l'instruction sera 4 (en décimal, et qui équivaut en binaire 100) ; si le port de contrôle numéro 3 est désactivé, le résultat sera 0.

Exemples

```
ReadIO(Port3, &B100)     ' lire l'état du port de contrôle n°3
                         ' si le port 3 est activé alors Port3 = 4
                         'si le port 3 est désactivé alors Port3 = 0
```

SDI12Recorder (Dest, SDIPort, SDIAddress, SDICommand, Multiplier, Offset)

L'instruction SDI12Recorder est utilisée afin de recevoir les données d'un capteur SDI-12.

Chaque exécution de l'instruction SDI12Recorder envoie les caractères « (adresse)M! » puis « (adresse)D0! ».

M! dit au capteur d'effectuer une mesure; D0! Lui demande d'envoyer les données.

Merci de consulter l'aide en ligne de l'éditeur CRBasic afin d'avoir les descriptions détaillées des commandes et des paramètres.

VibratingWire (Dest, Reps, Range, SEChan, ExChan, StartFreq, EndFreq, TSweep, Steps, DelMeas, NumCycles, DelReps, Multiplier, Offset)

L'instruction VibratingWire est utilisée afin de mesurer un capteur à corde vibrante sur une fenêtre de fréquence (d'une fréquence basse à une fréquence plus élevée).

La période de la réponse est mesurée, et $1/T^2$ est calculé. T est la période du signal mesuré, en millisecondes.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Variable ou ligne de donnée</td><td>Le paramètre Dest est la variable dans laquelle on stocke le résultat de la mesure.</td></tr>
<tr><td>Reps
Constante</td><td>Le paramètre Reps est le nombre de fois que l'instruction devra être exécutée. Les mesures sont effectuées sur des voies consécutives. Si le paramètre de Reps est supérieur à 1, alors le paramètre Dest doit être une ligne de donnée variable.</td></tr>
<tr><td>Range</td><td>Le paramètre Range est l'étendue de mesure du signal. On peut entrer le code numérique ou alphanumérique :
<table>
<tr><th>Alphanumérique</th><th>Numérique</th><th>Description</th></tr>
<tr><td>mV250</td><td>2</td><td>+250,00 mV</td></tr>
<tr><td>mV25</td><td>3</td><td>+25,00 mV</td></tr>
<tr><td>mV7_5</td><td>4</td><td>+7,50 mV</td></tr>
<tr><td>mV2_5</td><td>5</td><td>+2,50 mV</td></tr>
</table></td></tr>
<tr><td>SEChan
Constante</td><td>SEChan est le numéro de voie unipolaire sur laquelle on effectue la première mesure. Si le paramètre Reps est supérieur à 1 alors les mesures suivantes seront effectuées sur les voies consécutives. Si le paramètre SEChan est une valeur négative, alors toutes les Reps seront mesurées sur la même voie.</td></tr>
<tr><td>ExChan</td><td>ExChan est utilisé afin de spécifier la voie d'excitation à utiliser pour la première mesure. Si le paramètre Reps est supérieur à 1, la voie d'excitation utilisée sera incrémentée pour chaque nouvelle mesure. On peut entrer un code numérique ou alphanumérique tel que :
<table>
<tr><th>Alphanumérique</th><th>Numérique</th><th>Description</th></tr>
<tr><td>VX1</td><td>1</td><td>Voie d'excitation n° 1</td></tr>
<tr><td>VX2</td><td>2</td><td>Voie d'excitation n° 2</td></tr>
<tr><td>VX3</td><td>3</td><td>Voie d'excitation n° 3</td></tr>
</table></td></tr>
<tr><td>StartFreq</td><td>Le paramètre StartFreq est la fréquence en Hertz, à laquelle débute l'excitation. StartFreq doit être supérieure à 20 Hz.</td></tr>
<tr><td>EndFreq</td><td>Le paramètre EndFreq est la fréquence en Hertz, à laquelle se termine l'excitation. EndFreq doit être inférieure à 5000 Hz.</td></tr>
<tr><td>TSweep</td><td>TSweep est la durée, en millisecondes, du champ de fréquence.</td></tr>
<tr><td>Steps</td><td>Steps est le nombre d'étapes de changement de fréquence à parcourir afin de relier les fréquences de début (StartFreq) et de fin (EndFreq).</td></tr>
<tr><td>DelMeas</td><td>DelMeas est utilisé afin de spécifier le nombre de microsecondes à attendre avant de mesurer le signal de résonance une fois que le champ de fréquence à été balayé.</td></tr>
<tr><td>NumCycles</td><td>NumCycles est le nombre de cycles à mesurer.</td></tr>
<tr><td>DelReps</td><td>DelReps est utilisé afin de spécifier le nombre de microsecondes de délai entre chaque répétition de la mesure.</td></tr>
<tr><td>Mult, Offset
Constante, variable, ligne de donnée ou expression</td><td>Mult et Offset sont utilisés afin de mettre à l'échelle le résultat de la mesure. Avec un multiplicateur de 1 et un offset de 0, la valeur fournie en sortie est $1/T^2$, où T est la période du signal mesuré, en millisecondes.</td></tr>
</table>

WriteIO (Mask, Source)

WriteIO est employé afin de changer l'état des ports I/O (E/S) choisis sur la CR1000. (Voir également l'instruction « PortSet ».) Il y a 8 ports I/O. L'état de ces ports est considéré comme un nombre binaire avec un port activé (+5 V) signifiant 1 et un port désactivé (0 V) signifiant 0. Si par exemple les ports 1 et 3 sont activés et que le reste des ports I/O sont désactivés, la représentation binaire est 00000101, ou la décimale 5. La valeur de source est interprétée en tant que nombre binaire et les ports sont configurés en conséquence. Le paramètre de masque est employé pour choisir lesquels des ports sont à configurer ; c'est aussi une représentation binaire des ports. Le 1 indique qu'il faut configurer le port selon la représentation de la variable source ; le 0 indique qu'il ne faut pas changer l'état du port I/O. CRBasic permet d'entrer des nombres au format binaire en les précédant des caractères « &B ». Par exemple si le masque est &B110 (les zéros précédent le premier « 1 » peuvent être omis, au format binaire, tout comme ils peuvent l'être au format décimal) et que la source est la décimale 5 (ou 101 en binaire), le port I/O numéro 3 sera activé et le port I/O numéro 2 sera désactivé. Le masque indique que seuls les ports n° 3 et 2 devraient être configurés. Bien que la valeur de la source soit également à 1 pour le port 1, on l'ignore parce que le masque indique que l'état du port n° 1 ne devrait pas être modifié.

Exemple

```
WriteIO (&B100, &B100)    ' Active le port de contrôle n°3
```

Paramètre *& Type de donnée*	**Entrée**
Mask *Constante*	Le masque (**Mask**) permet à l'instruction de n'agir que sur certains ports de contrôle. Le masque fonctionne en tant que condition « AND » avec les valeurs obtenues à la lecture, ainsi qu'avec la **Source**, avant l'écriture.
Source *Constante Variable*	La variable ou le nombre qui doit être écrit sur les ports I/O.

7.8 Capteurs spécifiques

CS110 (Dest, Leakage, Status, Integ, Mult, Offset)

L'instruction CS110 est utilisée afin de mesurer le champ électrique au moyen du capteur CS110 (*electric field meter*).

Merci de consulter l'éditeur CRBasic ou le manuel du CS110 pour plus de détails au sujet de cette instruction.

CS616 (Dest, Reps, SEChan, Port, MeasPerPort, Mult, Offset)

L'instruction CS616 est utilisée afin d'activer et de mesurer le réflectomètre à teneur en eau CS616. Cette instruction donne en sortie une mesure de période en microsecondes.

Paramètre *& Type de donnée*	**Entrée**
Dest *Variable ou ligne de donnée*	Le paramètre **Dest** est une variable ou une ligne de données où stocker le résultat de la mesure. **Dest** doit avoir la dimension du nombre de **Reps**, au minimum.
Reps	**Reps** est le nombre de mesures qui devra être effectué en utilisant cette instruction. Si **Reps** >1, alors **Dest** doit avoir cette dimension ou plus.
SEChan	**SEChan** est le numéro de voie unipolaire sur lequel on effectue la première mesure. Si le **Reps** est supérieur à 1 alors les mesures suivantes seront effectuées sur les voies suivant celle indiquée au paramètre **SEChan**.
Port	Le paramètre de **Port** est le n° de port de contrôle (entre 1 et 8) qui sera utilisé afin d'activer le capteur CS616.
MeasPerPort	Le paramètre **MeasPerPort** est le nombre de port de contrôle à utiliser afin de contrôler les capteurs CS616. S'il y a 4 **Reps** et que **MeasPerPort** = 4, alors un seul port de contrôle sera utilisé pour piloter les 4 CS616. Si **MeasPerPort** = 1 alors 4 ports de contrôle consécutifs seront utilisés ; **MeasPerPort** = 2 utilisera un port de contrôle pour les 2 premières CS616, et le port suivant pour les 2 CS616 suivantes.
Mult, Offset	**Mult** et **Offset** sont une constante, variable, ou ligne de donnée utilisée afin de mettre à l'échelle les résultats de la mesure.

HydraProbe (Dest, SourceVolts, ProbeType, SoilType)

L'instruction HydraProbe est utilisée pour mesurer le capteur « Hydra Probe » de Stevens Vitel.

Paramètre *& Type de donnée*	**Entrée**
Dest *Ligne de donnée*	La variable de ligne de donnée qui contiendra les valeurs retournées par le capteur « Hydra Probe ». Cette variable doit être dimensionnée à 11. Le capteur donne les valeurs suivantes : Type de sol (1 = sable, 2 = vase, 3 = argile), constante diélectrique réelle, constante diélectrique imaginaire, température, constante diélectrique réelle avec correction en température, constante diélectrique imaginaire avec correction en température, teneur volumique en eau (fraction par volume), salinité (grammes de NaCl par litre), conductivité du sol (S/m), conductivité du sol avec correction en température (S/m), conductivité de l'eau du sol, corrigée en température (S/m).
SourceVolts *Ligne de donnée*	La variable de ligne de donnée qui contiendra les valeurs de tension retournées par le capteur (V1, V2, V3 et V4). La variable doit être dimensionnée à 4.
ProbeType *Constante*	Un code pour identifier la version du capteur « Hydra Probe » qu'on utilise. **Code** : **Description** 0 : Capteur standard (5V en sortie pour V4) 1 : Capteur Type A (2,5 V en sortie pour V4)
SoilType *Constante*	Un code pour indiquer le type de sol **Code** : **Description** 1 : Sable *(Sand)* 2 : Vase *(Silt)* 3 : Argile *(Clay)*

SlowAntenna

Voir le manuel du CS110.

Therm107 (Dest, Reps, SEChan, ExChan, SettlingTime, Integ, Mult, Offset)

Therm108 (Dest, Reps, SEChan, ExChan, SettlingTime, Integ, Mult, Offset)

Therm109 (Dest, Reps, SEChan, ExChan, SettlingTime, Integ, Mult, Offset)

Les instructions Therm107, Therm108, et Therm109 sont utilisées afin de mesurer les thermistances 107, 108, et 109 respectivement.

La syntaxe pour chacune de ces instructions, est la même.

Les instructions effectuent une mesure de demi pont et calculent le résultat en utilisant l'équation de Steinhart-Hart. Le résultat est une température en °C.

<table>
<tr><th>Paramètre
& Type de donnée</th><th>Entrée</th></tr>
<tr><td>Dest</td><td>Dest est la variable dans laquelle le résultat de la mesure sera stocké. Si le paramètre de Reps est supérieur à 1, Dest doit être une ligne de ligne de données de dimension égale ou supérieure au nombre de répétitions.</td></tr>
<tr><td>Reps</td><td>Le paramètre Reps est le nombre de fois que la mesure devra être effectuée. Les mesures sont faites sur des voies consécutives. Si le paramètre Reps est supérieur à 1, le paramètre Dest doit être une ligne de donnée de variables.</td></tr>
<tr><td>SEChan</td><td>SEChan est le numéro de la voie unipolaire sur laquelle on effectuera la mesure (entre 1 et 16). Si le paramètre Reps est supérieur à 1, les mesures suivantes seront effectuées sur des voies séquentielles.</td></tr>
<tr><td>ExChan</td><td>ExChan est la voie d'excitation (entre 1 et 3) à utiliser afin d'alimenter la thermistance. Si plusieurs thermistances sont mesurées par une seule instruction, toutes les répétitions utiliseront la même voie d'excitation. On peut entrer un code alphanumérique ou numérique.</td></tr>
<tr><td>SettlingTime</td><td>Le paramètre SettlingTime est le délai à attendre après avoir configuré la voie de mesure, et avant d'effectuer la mesure. Le tableau ci-dessous configure les délais de SettlingTime par défaut.
<table>
<tr><th>Entrée</th><th>Etendues de mesure</th><th>Intégration</th><th>Settling Time</th></tr>
<tr><td>0</td><td>Toutes</td><td>250 ms</td><td>450 ms (défaut)</td></tr>
<tr><td>0</td><td>Toutes</td><td>_50Hz</td><td>3 ms (défaut)</td></tr>
<tr><td>0</td><td>Toutes</td><td>_60Hz</td><td>3 ms (défaut)</td></tr>
<tr><td>>100</td><td>Toutes</td><td>All</td><td>ms que l'on entre</td></tr>
</table></td></tr>
<tr><td>Integ</td><td>Le paramètre Integ est la durée en microsecondes, pendant laquelle on integre le signal sur la voie que l'on mesure.
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>250</td><td>Effectue une intégration de 250 microsecondes.</td></tr>
<tr><td>_60Hz</td><td>Intégration de 16,667 millisecondes; filtre le bruit à 60 Hz.</td></tr>
<tr><td>_50Hz</td><td>Intégration de 20 millisecondes; filtre le bruit à 50 Hz.</td></tr>
</table></td></tr>
<tr><td>Mult, Offset</td><td>Mult et Offset sont des paramètres qui sont des constantes, variables, lignes de données ou expressions afin de mettre la mesure à l'échelle. Avec un multiplicateur égal à 1 et un offset de 0, la température sera donnée en °C. Un Mult de 1,8 et Offset 32, donnent une température en °F.</td></tr>
</table>

7.9 Appareils périphériques

AM25T
(Dest, Reps, Range, AM25TChan, DiffChan, TCType, Tref, ClkPort, ResPort, VxChan, RevDiff, SettlingTime, Integ, Mult, Offset)

Cette instruction contrôle le multiplexeur AM25T.

<table>
<tr><td>Paramètre
& Type de donnée</td><td colspan="3">Entrée</td></tr>
<tr><td>Dest
Variable ou Ligne de donnée</td><td colspan="3">Dest est la variable dans laquelle le résultat de la mesure sera stocké. Si le paramètre de Reps est supérieur à 1, Dest doit être une ligne de données de dimension égale ou supérieure au nombre de répétitions.</td></tr>
<tr><td>Reps</td><td colspan="3">Le nombre de voies à mesurer sur l’AM25T. Entrez 0 pour mesurer uniquement la température de référence.</td></tr>
<tr><td>Range
Constante</td><td>Code Alphanum.</td><td>Etendue de tension</td><td></td></tr>
<tr><td></td><td>mV5000</td><td>± 5000 mV</td><td></td></tr>
<tr><td></td><td>mV2500</td><td>± 2500 mV</td><td></td></tr>
<tr><td></td><td>mV250</td><td>± 250 mV</td><td></td></tr>
<tr><td></td><td>mV25</td><td>± 25 mV</td><td></td></tr>
<tr><td></td><td>mV7_5</td><td>± 7,5 mV</td><td></td></tr>
<tr><td></td><td>mV2_5</td><td>± 2,5 mV</td><td></td></tr>
<tr><td></td><td>Autorange</td><td>mV2_5 – mV5000</td><td>Choisit l’E. de mesure (Chap. 3.1)</td></tr>
<tr><td></td><td>mV250C</td><td>± 250 mV</td><td rowspan="5">Les étendues mV250C, mV25C, mV7_5C, et mV2_5C raménent la voie dans l’étendue de mode commun et vérifient s’il y a une entrée ouverte.</td></tr>
<tr><td></td><td>mV25C</td><td>± 25 mV</td></tr>
<tr><td></td><td>mV7_5C</td><td>± 7,5 mV</td></tr>
<tr><td></td><td>mV2_5C</td><td>± 2,5 mV</td></tr>
<tr><td></td><td>AutorangeC</td><td>mV2_5 – mV250</td><td>Choisit l’E. de mesure « C »</td></tr>
<tr><td>Am25tChan
Constante</td><td colspan="3">La voie du multiplexeur où débute la mesure.</td></tr>
<tr><td>DiffChan
Constante</td><td colspan="3">La voie différentielle qui sera utilisée afin de prende la mesure en provenant du multiplexeur. Si la voie est indiquée en tant que nombre négatif, toutes les répétitions se produisent sur cette voie.</td></tr>
<tr><td>TCType
Constante</td><td colspan="3">Le code du type de thermocouple. Entrer « -1 » afin de recevoir des volts.</td></tr>
<tr><td>Tref
Variable</td><td colspan="3">La variable où on stocke puis lit la valeur de température de référence de l’AM25T.</td></tr>
<tr><td>ClkPort
Constante</td><td colspan="3">Le port de contrôle qui sera utillisé pour piloter la ligne « clock » de l’AM25T. On peut utiliser une voie « clock » pour plusieurs AM25Ts.</td></tr>
<tr><td>ResPort
Constante</td><td colspan="3">Le port de contrôle qui sera utilisé pour activer l’AM25T. Chaque AM25T doit avoir sa propre ligne de « Reset ».</td></tr>
<tr><td>VxChan
Constante</td><td colspan="3">Le numéro de voie d’excitation qui alimentera la sonde PRT pour la température de référence. Si on entre «0», la température n’est pas mesurée.</td></tr>
<tr><td>RevDiff
Constante</td><td>Code</td><td>Valeur</td><td>Resultat (l’inversion nécessite deux fois plus de temps)</td></tr>
<tr><td></td><td>False</td><td>0</td><td>Le signal est mesuré avec le plus referencé au moins</td></tr>
<tr><td></td><td>True</td><td>≠0</td><td>Une seconde mesure est effectuée après avoir inversé les entrées, afin de supprimer les offsets</td></tr>
</table>

<table>
<tr><td>SettlingTime
Constante</td><td>Le temps en microsecondes, à attendre entre la configuration de la mesure (se connecter à la voie, configurer l'excitation) et la prise de mesure. (résolution de 1 microseconde)
<table>
<tr><th>Entrée</th><th>Etendue de tension</th><th>Intégration</th><th>"Settling Time"</th></tr>
<tr><td>0</td><td>Toutes</td><td>250 µS</td><td>450 µS (défaut)</td></tr>
<tr><td>0</td><td>Toutes</td><td>_50Hz, _60 Hz</td><td>3 mS (défaut)</td></tr>
<tr><td>>=100</td><td>Toutes</td><td>Toutes</td><td>µS entrées</td></tr>
</table></td></tr>
<tr><td>Integ
Constante</td><td>Le temps passé , en mmicrosecondes, pour l'intégration de chaque voie mesurée.
<table>
<tr><th>Entrée</th><th>Intégration</th></tr>
<tr><td>250</td><td>250 µS</td></tr>
<tr><td>_60Hz or 16667</td><td>16,667 µS (réjection du bruit à 60 Hz)</td></tr>
<tr><td>_50 Hz or 20000</td><td>20,000 µS (réjection du bruit à 50 Hz)</td></tr>
</table></td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de donnée ou Expression</td><td>Un multiplicateur et un offset afin de mettre à l'échelle les valeurs brutes mesurées. Voir la description de la mesure afin de connaître l'unité de mesure des valeurs brutes. Un multiplicateur de 1 et un offset de 0 sont nécessaires afin de garder la grandeur physique déterminée par la description de la mesure.</td></tr>
</table>

CS7500 (Dest, Reps, SDMAddress, CS7500Cmd)

Permet de communiquer avec le capteur CS7500 open path, qui mesure le CO_2 et H_2O. Voir le manuel du CS7500 pour plus d'information.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest</td><td>Dest est le paramètre pour le nom de la variable d'entrée où on stockera les résultats provenant de chaque donnée du CS7500 associée à cette instruction. La longueur de la ligne de donnée de la variable, dépendra du nombre de répétitions et de la commande sélectionnée.
<table>
<tr><th>Commande</th><th>Longueur de la variable d'entrée pour chaque CS7500</th></tr>
<tr><td>0 et 1</td><td>2</td></tr>
<tr><td>2</td><td>4</td></tr>
<tr><td>3</td><td>3</td></tr>
<tr><td>4</td><td>11</td></tr>
<tr><td>5</td><td>3</td></tr>
<tr><td>6</td><td>4</td></tr>
</table></td></tr>
<tr><td>Reps</td><td>Le paramètre Reps détermine le nombre d'analyseurs de gaz CS7500 avec lequel l'instruction communiquera. Les CS7500s doivent avoir des adresses SDM consécutives si le nombre Reps est supérieur à 1.</td></tr>
<tr><td>SDMAddress</td><td>Le paramètre SDMAddress définit l'adresse du CS7500 avec lequel on communiquera. Les adresses valides sont entre 0 et 14. L'adresse 15 est réservée pour l'instruction SDMTrigger. Si le paramètre Reps est supérieur à 1, la centrale de mesure incrémentera l'adresse SDM pour chaque nouveau CS7500 avec lequel elle communiquera.
L'adresse SDM est entrée en nombre de base 10, contrairement aux instructions plus anciennes où on utilisait les jumpers et les adresses base 4.</td></tr>
</table>

<table>
<tr><td>CS7500Cmd</td><td>CS7500Cmd permet de demander de rappatrier les données du capteur. La commande est envoyée en premier à l’appareil dont l’adresse est spécifiée au paramètre SDMAddress. Si Reps est supérieur à 1, les CS7500s suivant recevront la même commande lors de chaque répétition. Le résultat de la commande sera envoyé à la ligne de donnée spécifiée au paramètre Dest. Un code numérique permet de demander les données :
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Recevoir CO2 & H2O en densité molaire (mmol/m3)</td></tr>
<tr><td>1</td><td>Recevoir l’ absorptance en CO2 & H2O</td></tr>
<tr><td>2</td><td>Recevoir l’estimation interne de la pression (kPa), la mesure auxiliaire A, la mesure auxiliaire B et « cooler voltage » (V)</td></tr>
<tr><td>3</td><td>Recevoir la valeur du diagnostique de la cellule, la largeur de bande en sortie (output bandwidth (Hz)), et le délai programmé [230 + (delai * 6.579)] (msec)</td></tr>
<tr><td>4</td><td>Recevoir toutes les données (densité molaire de CO2 (mmol/m3), densité molaire de H2O (mmol/m3), absorptance de CO2, absorptance d’H2O, estimation de la pression interne (kPa), mesure auxiliaire A, mesure auxiliaire B, « cooler voltage » (V), valeur du diagnostique de la cellule, largeur de bande en sortie (Hz), et le délai programmé [230 + (delay * 6.579)] (msec))</td></tr>
<tr><td>5</td><td>Recevoir la densité molaire de CO2 & H2O (mmol/m3) et l’estimation de la pression interne (kPa)</td></tr>
<tr><td>6</td><td>Recevoir la densité molaire de CO2 & H2O (mmol/m3) l’estimation de la pression interne (kPa) et la valeur du diagnostique de la cellule</td></tr>
</table></td></tr>
</table>

CSAT3 (Dest, Reps, SDMAddress, CSAT3Cmd, CSAT3Opt)

Pour communiquer avec l’anémomètre sonique tridimentionnel CSAT3. Voir le manuel du CSAT3 pour plus de détails.

Paramètre *& Type de donnée*	**Entrée**
Dest	**Dest** est le paramètre pour le nom de la variable où on stockera les résultats provenant de la mesure. Cette variable doit avoir une dimension de 5 afin de contenir Ux, Uy, Uz, la vitesse du son et une donnée de diagnostique.
Reps	Le paramètre **Reps** détermine le nombre de fois que l’intruction sera effectuée. Les CSAT3s doivent avoir des adresses SDM consécutives si le nombre **Reps** est supérieur à 1. Si **Reps** est supérieur à 1, **Dest** doit être une ligne de donnée de variables.
SDMAddress	Le paramètre **SDMAddress** définit l’adresse du CSAT3 avec lequel on communiquera. Les adresses valides sont entre 0 et 14. L’adresse 15 est réservée pour l’instruction SDMTrigger. Si le paramètre **Reps** est supérieur à 1, la centrale de mesure incrémentera l’adresse SDM pour chaque nouveau CSAT3 avec lequel elle communiquera. L’adresse SDM est entrée en nombre de base 10, contrairement aux instructions plus anciennes où on utilisait les jumpers et les adresses base 4.
Command	Les commandes 90 à 92 envoient une demande de mesure au CSAT3 qui a l’adresse SDM spécifiée à SDMAddress. Le CSAT3 envoie aussi les données à la centrale de mesure. Les options 97 à 99 reçoivent les données après une demande de groupe, SDMTrigger(), provenant du CSAT3 spécifié à SDMAddress, et sans demander de nouveau à recevoir la mesure. L’instruction CSAT3() devra être précédée de l’instruction SDMTrigger() si l’on souhaite utiliser les options 97 à 99.

	Code	Description
	90	Déclenchement et réception des données de vitesse du vent et de vitesse du son
	91	Déclenchement et réception des données de vitesse du vent et de température sonique
	92	Déclenchement et réception des données de vitesse du vent et de vitesse du son moins 340 m/s
	97	Réception des données de vitesse du vent et de vitesse du son moins 340 m/s après une demande de groupe (*Group Trigger*)
	98	Réception des données de vitesse du vent et de température sonique après une demande de groupe (*Group Trigger*)
	99	Réception des données de vitesse du vent et de vitesse du son après une demande de groupe (*Group Trigger*)

Rate

L'argument **Rate** configure le paramètre d'exécution du CSAT3. Ce paramètre dit au CSAT3 quel paramètre de mesure utiliser et à quelle fréquence attendre les déclenchements de mesure provenant de la centrale de mesure. Voir le tableau ci-dessous pour une description brève de des paramètres, ou le manuel du CSAT3 pour plus de détails.

Code	Description
1	Configurer le paramètre d'exécution à 1 Hz
2	Configurer le paramètre d'exécution à 2 Hz
3	Configurer le paramètre d'exécution à 3 Hz
5	Configurer le paramètre d'exécution à 5 Hz
6	Configurer le paramètre d'exécution à 6 Hz
10	Configurer le paramètre d'exécution à 10 Hz
12	Configurer le paramètre d'exécution à 12 Hz
20	Configurer le paramètre d'exécution à 20 Hz
30	Configurer le paramètre d'exécution à 30 Hz
60	Configurer le paramètre d'exécution à 60 Hz
61	Configurer le paramètre d'exécution à 60 Hz et 10 Hz de mode l'échantillonnage (*Oversample Mode*)
62	Configurer le paramètre d'exécution à 60 Hz et 20 Hz de mode l'échantillonnage (*Oversample Mode*)

SDMAO4 (Source, Reps, SDMAdress)

Cette instruction est utilisée afin de configurer la tension en sortie du SDM-AO4, appareil à 4 voies analogiques en sortie.

L'instruction SDMAO4 a les paramétres suivant :

Paramètre *& Type de donnée*	**Entrée**
Source	**Source** est la variable qui contient la tension, en millivolts, qui devrait être envoyée au SDM-AO4. Si plusieurs SDM-AO4s sont déclenchés par une instruction, ce paramètre doit être une ligne de données dont la taille est égale aux **Reps**.
Reps	**Reps** détermine le nombre d'appareils SDM-AO4 qui fourniront la tension en utilisant cette instruction. Les SDM-A04s doivent avoir des adresses SDM consécutives, si le paramètre **Reps** est supérieur à 1.
SDMAddress	**SDMAddress** définit l'adresse du SDM-AO4 vers lequel la tension devra être appliquée. Les adresses SDM valides sont entre 0 et 14. L'adresse 15 est réservée à l'instruction SDMTrigger. L'adresse SDM est entrée en nombre de base 10, contrairement aux instructions plus anciennes où on utilisait les jumpers et les adresses base 4.

SDMCAN (Dest, SDMAddress, TimeQuanta, TSEG1, TSEG2, ID, DataType, StartBit, NumBits, NumVals, Multiplier, Offset)

L'instruction SDMCAN est utilisée afin de mesurer et de contrôler l'interface SDM-CAN.

Plusieurs instructions SDM SDM peuvent être utilisées dans un programme. La fonction initiale de l'instruction est de configurer l'interface SDM-CAN lorsque le programme de la centrale de mesure est compilé. Des instructions supplémentaires peuvent être employées pour déterminer quelles données sont passées entre le réseau CAN-bus et la centrale de mesure, afin de configurer et/ou de lire l'état des switch internes du SDM-CAN, et afin de lire et/ou ré-initialiser les erreurs détectées.

L'instruction SDMTrigger peut être utilisée afin de déclencher des mesures simultanées à partir de un ou plusieurs SDM-CANs et d'autres appareils SDM connectés à la centrale de mesure. Lorsque l'instruction SDMTrigger est lue dans le programme, elle envoie un message SDM spécial qui a pour effet de faire transmettre les dernières données capturées sur le bus CAN, en direction de la mémoire tampon de travail, et ce pour tous les appareils SDM-CAN. Reportez-vous au manuel du SDM-CAN pour plus de détails.

L'instruction CANBUS est constituée les paramètres suivants :
Pour des raisons techniques certaines instructions ne sont pas traduites. A l'occasion d'une mise à jour de ce manuel dans le futur, nous ne manquerons pas d'achever la traduction de ce manuel.

Paramètre *& Type de donnée*	**Entrée**
Dest	The Dest parameter is a variable array in which to store the results of the measurement. It must be an array of sufficient size to hold all of the values that will be returned by the function chosen (defined by the DataType parameter).
SDMAddress	The SDMAddress parameter defines the address of the SDM-CAN with which to communicate. Valid SDM addresses are 0 through 14. Address 15 is reserved for the SDMTrigger instruction. The SDM address is entered as a base 10 number, unlike older, jumper-settable SDM instruments that used base 4.
TimeQuanta	Three time segments are used to set the bit rate and other timing parameters for the CAN-bus network, TimeQuanta, TSEG1, and TSEG2. These parameters are entered as integer numbers. The relationship between the three time segments is defined as: $t_{bit} = t_q + t_{TSEG1} + t_{TSEG2}$ The first time segment, the synchronization segment (S-SG), is defined by the TimeQuanta parameter. To calculate a suitable value for TimeQuanta, use the following equation: $\text{TimeQuanta} = t_q * 8 * 10^6$ where tq = the TimeQuanta. There are between 8 and 25 time quanta in the bit time. The bit time is defined as 1/baud rate.
TSEG1	The second time segment, TSEG1, is actually two time segments known as the propagation segment and phase segment one. The value entered is determined by the characteristics of the network and the other devices on the network. It can be calculated as: $T_{SEG1} = t_{TSEG1} / t_q$
TSEG2	The third time segment, TSEG2 (the phase segment two), is defined by the TSEG2 parameter. The value of TSEG2 can be calculated using the equation: $T_{SEG2} = t_{TSEG2} / t_q$ The relative values of TSEG1 and TSEG2 determine when the SDM-CAN samples the data bit.

ID	Each device on a CAN-bus network prefaces its data frames with an 11 or 29 bit identifier. The ID parameter is used to set this address. The ID is entered as a single decimal equivalent. Enter a positive value to signify a 29 bit ID or a negative value to signify an 11 bit ID.	
DataType	The DataType parameter defines what function the CANBUS instruction will perform. This instruction can be used to collect data, buffer data for transmission to the CAN-bus, transmit data to the CAN-bus, read or reset error counters, read the status of the SDM-CAN, read the SDM-CAN's OS signature and version, send a remote frame, or read or set the SDM-CAN's internal switches. Enter the numeric value for the desired option:	
	Value	**Description**
	1	Retrieve data; unsigned integer, most significant byte first
	2	Retrieve data; unsigned integer, least significant byte first
	3	Retrieve data; signed integer, most significant byte first
	4	Retrieve data; signed integer, least significant byte first
	5	Retrieve data; 4-byte IEEE floating point number; most significant byte first
	6	Retrieve data; 4-byte IEEE floating point number; least significant byte first
	7	Build data frame in SDM-CAN memory; unsigned integer, most significant byte first. Overwrite existing data.
	8	Build data frame in SDM-CAN memory; unsigned integer, least significant byte first. Overwrite existing data.
	9	Build data frame in SDM-CAN memory; signed integer, most significant byte first. Overwrite existing data.
	10	Build data frame in SDM-CAN memory; signed integer, least significant byte first. Overwrite existing data.
	11	Build data frame in SDM-CAN memory; 4-byte IEEE floating point number; most significant byte first. Overwrite existing data.
	12	Build data frame in SDM-CAN memory; 4-byte IEEE floating point number; least significant byte first. Overwrite existing data.
	13	Build data frame in SDM-CAN memory; unsigned integer, most significant byte first. Logical "OR" with existing data.
	14	Build data frame in SDM-CAN memory; unsigned integer, least significant byte first. Logical "OR" with existing data.
	15	Build data frame in SDM-CAN memory; signed integer, most significant byte first. Logical "OR" with existing data.
	16	Build data frame in SDM-CAN memory; signed integer, least significant byte first. Logical "OR" with existing data.
	17	Build data frame in SDM-CAN memory; 4-byte IEEE floating point number; most significant byte first. Logical "OR" with existing data.
	18	Build data frame in SDM-CAN memory; 4-byte IEEE floating point number; least significant byte first. Logical "OR" with existing data.
	19	Transmit data value to the CAN-bus; unsigned integer, most significant byte first.
	20	Transmit data value to the CAN-bus; unsigned integer, least significant byte first.
	21	Transmit data value to the CAN-bus; signed integer, most significant byte first.
	22	Transmit data value to the CAN-bus; signed integer, least significant byte first.

	23	Transmit data value to the CAN-bus; 4-byte IEEE floating point number; most significant byte first.
	24	Transmit data value to the CAN-bus; 4-byte IEEE floating point number; least significant byte first.
	25	Transmit previously built data frame to the CAN-bus.
	26	Set up previously built data frame as a Remote Frame Response.
	27	Read Transmit, Receive, Overrun, and Watchdog errors. The errors are placed consecutively in the array specified by the Dest parameter.
	28	Read Transmit, Receive, Overrun, and Watchdog errors. The errors are placed consecutively in the array specified by the Dest parameter. Reset error counters to 0 after reading.
	29	Read SDM-CAN status; result is placed into the array specified in the Dest parameter. The result codes are as follows:

		Status	**Description**
		0000	The SDM-CAN is involved in bus activities; error counters are less than 96.
		0001	The SDM-CAN is involved in bus activities; one or more error counters is greater than or equal to 96.
		0002	The SDM-CAN is not involved in bus activities; error counters are less than 96.
		0003	The SDM-CAN is not involved in bus activities; one or more error counters is greater than or equal to 96.

	30	Read SDM-CAN operating system and version number; results are placed in two consecutive array variables beginning with the variable specified in the Dest parameter.
	31	Send Remote Frame Request.
	32	Set SDM-CAN's internal switches. The code is stored in the array specified in the Dest parameter and is entered in the form of ABCD.

		Switch	**Code**	**Description**
		A	0	Currently not used; set to 0.
		B	0	SDM-CAN returns the last value captured from the network, even if that value has been read before (default).
			1	SDM-CAN returns -99999 if a data value is requested by the datalogger and a new value has not been captured from the network since the last request.
			2-9	Currently not used.
		C	0	Disable I/O interrupts (default).
			1	Enable I/O interrupts, pulsed mode.
			2	Enable I/O interrupts, fast mode.
			3-7	Currently not used.
			8	Place the SDM-CAN into low power stand-by mode
			9	Leave switch setting unchanged.
		D	0	Listen only (error passive) mode. CAN transmissions are not confirmed.

			1	Transmit once. Data will not be retransmitted in case of error or loss of arbitration. Frames received without error are acknowledged.
			2	Self-reception. A frame transmitted from the SDM-CAN that was acknowledged by an external node will also be received by the SDM-CAN but no retransmission will occur in the event of loss of arbitration or error. Frames received correctly from an external node are acknowledged.
			3	Normal, retransmission will occur in the event of loss of arbitration or error. Frames received correctly from an external node are acknowledged. This is the typical setting to use if the SDM-CAN is to be used to transmit data.
			4	Transmit once; self-test. The SDM-CAN will perform a successful transmission even if there is no acknowledgement from an external CAN node. Frames received correctly from an external node are acknowledged.
			5	Self-reception; self -test. The SDM-CAN will perform a successful transmission even if there is no acknowledgement from an external CAN node. Frames received correctly from an external node are acknowledged. SDM-CAN will receive its own transmission.
			6	Normal; self-test. The SDM-CAN will perform a successful transmission even if there is no acknowledgement from an external CAN node. Frames received correctly from an external node are acknowledged.
			7	Not defined.
			8	Not defined.
			9	Leave switch setting unchanged.
	33	Read SDM-CAN's internal switches. Place results in the array specified in the Dest parameter.		
StartBit	The StartBit parameter is used to identify the least significant bit of the data value within the CAN data frame to which the instruction relates. The bit number can range from 1 to 64 (there are 64 bits in a CAN data frame). The SDM-CAN adheres to the ISO standard where the least significant bit is referenced to the right most bit of the data frame. If a negative value is entered, the least significant bit is referenced to the left most bit of the data frame.			
NumBits	The NumBits parameter is used to specify the number of bits that will be used in a transaction. The number can range from 1 to 64 (there are 64 bits in a CAN data frame). The SDM-CAN can be configured to notify the datalogger when new data is available by setting a control port high. This allows data to be stored in the datalogger tables faster than the program execution interval. This interrupt function is enabled by entering a negative value for this parameter. **Note: This parameter may be overridden by a fixed number of bits, depending upon the data type selected.**			

NumVals	The NumVals parameter defines the number of values (beginning with the value stored in the Dest array) that will be transferred to or from the datalogger during one operation. For each value transferred, the Number of Bits (NumBits) will be added to the Start Bit number so that multiple values can be read from or stored to one data frame.
Mult, Offset	The Mult and Offset parameters are each a constant, variable, array, or expression by which to scale the results of the measurement.

NOTE Si plus d'une instruction Canbus est utilisée dans un programme de centrale de mesure, les valeurs utilisées pour TimeQuanta, TSEG1 et TSEG2 doivent être les mêmes à chaque instruction.

Exemple CANBUS

L'exemple suivant lit la valeur de la vitesse d'un moteur à 16 bits provenant d'un réseau CAN-bus fonctionnant à 250K baud.

```
'Set Scan Rate
Const Period=1
Const P_Units=2
'\\\\\\\\\\\\ CANBUS Constants / / / / / / / / / / / /
'------------------------ Physical Network Parameters ------------
'Set SDM-CAN to 250K
Const TQUANT=4
Const TSEG1=5
Const TSEB2=2
'----------------------- Data Frame Parameters -------------------
'_____________________ Canbus Block1_____________________
'Collect and retrieve 16-bit data value
'Data Type 1, unsigned integer, most significant byte first
Const CANREP1=1                 'Repetitions
Const ADDR1=0                   'Address of SDM-CAN module
Const DTYPE1=1                  'Data values to collect
Const STBIT1=33                 'Start position in data frame
Const NBITS1=16                 'Number of bits per value
Const NVALS1=1                  'Number of values
Const CMULT1=0.4                'Multiplier
Const COSET1=0                  'Offset
Dim CANBlk1(CANREP1)            'Dimensioned Dest
'\\\\\\\\\\\\ Aliases and other Variables / / / / / / / /
Alias Canblk1(1)=Engine_Speed
'\\\\\\\\\\\\ PROGRAM / / / / / / / / / / / / / / / / /
BeginProg
      Scan(PERIOD,P_UNITS,0,0)
      '_____________________ CAN Blocks_____________________
      'Retrieve Data from CAN-bus network
      Canbus (CANBLK1(), ADDR1, TQUANT, TSEG1, TSEG2, 217056256,
DTYPE1, STBIT1, NBITS1, NVALS2, CMJLT1, COSET1)
      Next Scan
EndProg
```

SDMCD16AC (Source, Reps, SDMAddress)

L'instruction SDMCD16AC est utilisée afin de contrôler un appareil à 16 relais/ports de contrôle tels le SDM-CD16AC, SDM-CD16, ou SDM-CD16D.

Un port du SDM-CD16xx est activé/désactivé (mis à l'état « on » ou « off ») suite à l'envoi d'une valeur via l'instruction SDMCD16AC. Une valeur différente de zéro va activer le port ; zéro le désactivera. Les valeurs à envoyer au SDM-CD16AC sont contenues dans une ligne de donnée Source.

Paramètre *& Type de donnée*	**Entrée**
Source *Ligne de donnée*	La ligne de donnée qui contient les valeurs qui seront envoyées au SDM-CD16AC afin d'activer/désactiver les ports. Un SDM-CD16AC a 16 ports; la ligne de donnée de source doit donc avoir une dimension de 16 fois le nombre de répétitions (nombre de SDM-CD16AC à contrôler). Par exemple, avec une ligne de donnée CDCtrl(32), la valeur contenue à l'emplacement CDCtrl(1) sera envoyée au port 1, la valeur contenue dans CDCtrl(2) envoyée au port 2, etc. La valeur contenue dans CDCtrl(32) serait envoyée au port 16 du second SDM-CD16AC.
Reps *Constante*	**Reps** est le nombre d'appareils SDM-CD16AC qui sont contrôlés par cette instruction.
SDMAddress *Constante*	C'est l'adresse du premier SDM-CD16A qui sera contrôlé par l'instruction. Les adresses valides sont entre 0 et 15. Si l'instruction SDMTrigger est utilisée dans le programme, l'adresse 15 ne devrait pas être utilisée. Si **Reps** est supérieur à 1, la centrale de mesure incrémentera les adresses SDM pour les appareils consécutifs avec lesquels elle communiquera.

SDMINT8 (Dest, Address, Config8_5, Config4_1, Funct8_5, Funct4_1, OutputOpt, CaptureTrig, Mult, Offset)

Cette instruction permet d'utiliser le SDM-INT8, mesureur d'intervalles de temps à 8 voies, avec la CR1000. Le SDM-INT8 est un appareil « (S)ynchronous (D)evice for the (M)easurement » pour des intervalles, de comptages entre des évènements, des fréquences, des périodes, et/ou des durées depuis un événement. Voir le manuel du SDM-INT8 afin d'avoir plus de détails sur les capacités de cette interface.

Pour des raisons techniques certaines instructions ne sont pas traduites. A l'occasion d'une mise à jour de ce manuel dans le futur, nous ne manquerons pas d'achever la traduction de ce manuel.

<table>
<tr><th>Paramètre
& Type de donnée</th><th>Entrée</th></tr>
<tr><td>Dest
Variable ou Ligne de données</td><td>The array where the results of the instruction are stored. For all output options except Capture All Events, the Dest argument should be a one-dimensional array with as many elements as there are programmed INT8 channels. If the "Capture All Events" OutputOption is selected, then the Dest array must be two dimensional. The magnitude of first dimension should be set to the number of functions (up to 8), and the magnitude of the second dimension should be set to at least the number of events to be captured. The values will be loaded into the array in the sequence of all of the time ordered events captured from the lowest programmed channel to the time ordered events of the highest programmed channel.</td></tr>
<tr><td>Address
Constante</td><td>The address is entered as a base 10 number. Valid addresses are 0 to 15. The INT8 is addressable using internal jumpers. The jumpers are set at the factory for address 00. See Appendix A of the INT8 manual for details on changing the INT8 address.</td></tr>
<tr><td>Config8_5
Config4_1
Constantes</td><td>Each of the 8 input channels can be configured for either high or low level voltage inputs, and for rising or falling edges. Config8_5 is a four digit code to configure the INT8's channels 5 through 8. Config4_1 is a four digit code to configure the INT8's channels 1 through 4. The digits represent the channels in descending order left to right. For example, the code entered for Config8_5 to program channels 8 and 6 to capture the rising edge of a high level voltage, and channels 5 and 7 to capture the falling edge of a low level voltage would be "0303". See section 2 of the INT8 manual for information about the specification requirements of high and low level voltage signals.
<table>
<tr><th>Digit</th><th>Edge</th></tr>
<tr><td>0</td><td>High level, rising edge</td></tr>
<tr><td>1</td><td>High level, falling edge</td></tr>
<tr><td>2</td><td>Low level, rising edge</td></tr>
<tr><td>3</td><td>Low level falling edge</td></tr>
</table></td></tr>
<tr><td>Funct8_5
Funct4_1
Constantes</td><td>Each of the 8 input channels can be independently programmed for one of eight different timing functions. Funct8_5 is a four digit code to program the timing functions of INT8 channels 5 through 8. Funct4_1 is a four digit code to program the timing functions of INT8 channels 1 through 4. See section 5.3 of the INT8 manual for further details about these functions.
<table>
<tr><th>Digit</th><th>Results</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>Period (msec) between edges on this channel</td></tr>
<tr><td>2</td><td>Frequency (kHz) of edges on the channel</td></tr>
<tr><td>3</td><td>Time between an edge on the previous channel and the edge on this channel (msec)</td></tr>
<tr><td>4</td><td>time between an edge on channel 1 and the edge on this channel (msec)</td></tr>
<tr><td>5</td><td>Number of edges on channel 2 between the last edge on channel 1 and the edge on this channel using linear interpolation</td></tr>
<tr><td>6</td><td>Low resolution frequency (kHz) of edges on this channel</td></tr>
<tr><td>7</td><td>Total number of edges on this channel since last interrogation</td></tr>
<tr><td>8</td><td>Integer number of edges on channel 2 between the last edge on channel 1 and the edge on this channel.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Paramètre
& Type de données</th><th>Entrée</th></tr>
<tr><td></td><td>For example, 4301 in the second function parameter means to return 3 values: the period for channel 1, (nothing for channel 2) the time between an edge on channel 2 and an edge on channel 3, and the time between an edge on channel 1 and an edge on channel 4. The values are returned in the sequence of the channels, 1 to 16.
Note: the destination array must be dimensioned large enough to hold all the functions requested.</td></tr>
<tr><td>OutputOpt</td><td>Code to select one of the five different output options. The Output Option that is selected will be applied to the data collection for all of the INT8 channels. The numeric code for each option is listed below with a brief explanation of each. See the INT8 manual for detailed explanations of each option.
<table>
<tr><th>Code</th><th>Result</th></tr>
<tr><td>0:</td><td>Average of the event data since the last time that the INT8 was interrogated by the datalogger. If no edges were detected, 0 will be returned for frequency and count functions, and 99999 will be returned for the other functions. The INT8 ceases to capture events during communications with the logger, thus some edges may be lost.</td></tr>
<tr><td>32768</td><td>Continuous averaging, which is utilized when input frequencies have a slower period than the execution interval of the datalogger. If an edge was not detected for a channel since the last time that the INT8 was polled, then the datalogger will not update the input location for that channel. The INT8 will capture events even during communications with the datalogger.</td></tr>
<tr><td>nnnn</td><td>Averages the input values over "nnnn" milliseconds. The datalogger program is delayed by this instruction while the INT8 captures and processes the edges for the specified time duration and sends the results back to the logger. If no edges were detected, 0 will be returned for frequency and count functions, and 99999 will be returned for the other functions.</td></tr>
<tr><td>-nnnn</td><td>Instructs the INT8 to capture all events until "nnnn" edges have occurred on channel 1, or until the logger addresses the INT8 with the CaptureTrig argument true, or until 8000 (storage space limitation) events have been captured. When the CaptureTrig argument is true, the INT8 will return up to the last nnnn events for each of the programmed INT8 channels, reset its memory and begin capturing the next nnnn events. The Dest array must be dimensioned large enough to receive the captured events.</td></tr>
<tr><td>-9999</td><td>Causes the INT8 to perform a self memory test. The signature of the INT8's PROM is returned to the datalogger.
<table>
<tr><th>RESULT CODE</th><th>DEFINITION</th></tr>
<tr><td>0</td><td>Bad ROM</td></tr>
<tr><td>-0</td><td>Bad ROM, & bad RAM</td></tr>
<tr><td>positive integer</td><td>ROM signature, good RAM</td></tr>
<tr><td>negative integer</td><td>ROM signature, bad RAM</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Paramètre *&Type de données*	**Entrée**
CaptureTrig *Constante, Variable, ou Expression*	This argument is used when the "Capture All Events" output option is used. When CaptureTrig is true, the INT8 will return the last *nnnn* events.
Mult, Offset *Constante, Variable, Ligne de données, ou Expression*	A multiplier and offset by which to scale the raw results of the measurement. See the measurement description for the units of the raw result; a multiplier of one and an offset of 0 are necessary to output in the raw units. For example, the **TCDiff** instruction measures a thermocouple and outputs temperature in degrees C. A multiplier of 1.8 and an offset of 32 will convert the temperature to degrees F.

SDMIO16
(Dest, Status, Address, Command, Mode Ports 16-13, Mode Ports 12-9, Mode Ports 8-5, Mode Ports 4-1, Mult, Offset)

L'instruction SDMIO16 est utilisée afin de piloter le module d'extension des ports de contrôle SDM-IO16.

Les ports du SDM-IO16 peuvent être configurés en entrée ou en sortie. Lorsqu'ils sont configurés en entrée, le SDM-IO16 peut mesurer l'état logique de chaque port, peut compter des impulsions, et peut mesurer la fréquence et déterminer le cycle d'utilisation (duty cycle) des signaux appliqués. Le module peut aussi être programmé afin de générer un signal d'interruption vers la centrale de mesure, lorsqu'un ou plusieurs des ports de contrôle change d'état. Lorsqu'il est configuré en sortie, chaque port peut être mis à 0 ou 5V par la centrale de mesure. En plus de pouvoir piloter des niveaux logiques normaux en entrée, lorsqu'une sortie est activée il y a un circuit de « boost » qui est en fonctionnement et permet au module de fournir un courant allant jusqu'à 100mA, ce qui permet de contrôler directement des tensions faibles, relais etc.

Paramètre *& Type de données*	**Entrée**
Dest	La variable ou la ligne de variable dans laquelle est stockée le resultat de la mesure (codes d'opération 1 - 69, 91, 92, 99) ou la valeur de source pour les codes d'opération (70 - 85, 93 - 98). La ligne de variable pour ce paramètre doit être dimensionnée pour l'adapter au nombre de valeurs retournées (ou à envoyer) par l'instruction.
Status	La variable dans laquelle est stockée le résultat de la commande provenant de l'instruction. Si la commande est réussie un 0 est retourné ; autrement, la valeur est incrémentée de 1 aprés chaque échec.
SDMAddress	Le paramètre de SDMAddress définit l'adresse du SDM-IO16 avec lequel on communique. Les adresses valides de SDM vont 0 à 14. L'adresse 15 est réservée pour l'instruction SDMTrigger. Si le paramètre de Reps est plus grand que 1, la centrale de mesure incrémentera l'adresse SDM pour chaque SDM-IO16 suivant avec lequel elle communique. Note : Les centrales de mesure de type CRBasic emploient la base 10 pour adresser les SDM. Les centrales de mesure qui utilisent EDLOG (par exemple : CR10X, CR23X) utilisent la base 4 pour l'adressage.

Paramètre *& Type de données*	**Entrée**	
Commande	Ce qui suit sont des options valides de commande :	
	Code de Commande	**Description**
	1	Lire les comptages du port 1 accumulés dans la variable (Dest)
	2	Lire les comptages du port 2 accumulés dans la variable
	3	Lire les comptages du port 3 accumulés dans la variable
	4	Lire les comptages du port 4 accumulés dans la variable
	5	Lire les comptages du port 5 accumulés dans la variable
	6	Lire les comptages du port 6 accumulés dans la variable
	7	Lire les comptages du port 7 accumulés dans la variable
	8	Lire les comptages du port 8 accumulés dans la variable
	9	Lire les comptages du port 9 accumulés dans la variable
	10	Lire les comptages du port 10 accumulés dans la variable
	11	Lire les comptages du port 11 accumulés dans la variable
	12	Lire les comptages du port 12 accumulés dans la variable
	13	Lire les comptages du port 13 accumulés dans la variable
	14	Lire les comptages du port 14 accumulés dans la variable
	15	Lire les comptages du port 15 accumulés dans la variable
	16	Lire les comptages du port 16 accumulés dans la variable
	17	Lire les comptages des ports 1-4 accumulés dans la variable (la variable doit être dimensionnée à 4)
	18	Lire les comptages des ports 5-8 accumulés dans la variable (la variable doit être dimensionnée à 4)
	19	Lire les comptages des ports 9-12 accumulés dans la variable (la variable doit être dimensionnée à 4)
	20	Lire les comptages des ports 13-16 accumulés dans la variable (la variable doit être dimensionnée à 4)
	21	Lire les comptages des ports 1-8 accumulés dans la variable (la variable doit être dimensionnée à 8)
	22	Lire les comptages des ports 9-16 accumulés dans la variable (la variable doit être dimensionnée à 8)
	23	Lire les comptages des ports 1-16 accumulés dans la variable (la variable doit être dimensionnée à 16)
	24	Lire le port 1 en fréquence dans la variable
	25	Lire le port 2 en fréquence dans la variable
	26	Lire le port 3 en fréquence dans la variable
	27	Lire le port 4 en fréquence dans la variable
	28	Lire le port 5 en fréquence dans la variable
	29	Lire le port 6 en fréquence dans la variable
	30	Lire le port 7 en fréquence dans la variable
	31	Lire le port 8 en fréquence dans la variable
	32	Lire le port 9 en fréquence dans la variable
	33	Lire le port 10 en fréquence dans la variable
	34	Lire le port 11 en fréquence dans la variable
	35	Lire le port 12 en fréquence dans la variable
	36	Lire le port 13 en fréquence dans la variable
	37	Lire le port 14 en fréquence dans la variable

Paramètre *& Type de données*	**Entrée**	
	38	Lire le port 15 en fréquence dans la variable
	39	Lire le port 16 en fréquence dans la variable
	40	Lire les ports 1-4 en fréquence dans la variable : Dest (la variable doit être dimensionnée à 4)
	41	Lire les ports 5-8 en fréquence dans la variable (la variable doit être dimensionnée à 4)
	42	Lire les ports 9-12 en fréquence dans la variable (la variable doit être dimensionnée à 4)
	43	Lire les ports 13-16 en fréquence dans la variable (la variable doit être dimensionnée à 4)
	44	Lire les ports 1-8 en fréquence dans la variable (la variable doit être dimensionnée à 8)
	45	Lire les ports 9-16 en fréquence dans la variable (la variable doit être dimensionnée à 8)
	46	Lire les ports 1-16 en fréquence dans la variable (la variable doit être dimensionnée à 16)
	47	Lire le rapport cyclique (duty cycle) du port 1 dans la variable Read port 1's duty cycle into Dest
	48	Lire le rapport cyclique du port 2 dans la variable : Dest
	49	Lire le rapport cyclique du port 3 dans la variable
	50	Lire le rapport cyclique du port 4 dans la variable
	51	Lire le rapport cyclique du port 5 dans la variable
	52	Lire le rapport cyclique du port 6 dans la variable
	53	Lire le rapport cyclique du port 7 dans la variable
	54	Lire le rapport cyclique du port 8 dans la variable
	55	Lire le rapport cyclique du port 9 dans la variable
	56	Lire le rapport cyclique du port 10 dans la variable
	57	Lire le rapport cyclique du port 11 dans la variable
	58	Lire le rapport cyclique du port 12 dans la variable
	59	Lire le rapport cyclique du port 13 dans la variable
	60	Lire le rapport cyclique du port 14 dans la variable
	61	Lire le rapport cyclique du port 15 dans la variable
	62	Lire le rapport cyclique du port 16 dans la variable
	63	Lire le rapport cyclique (duty cycle) des ports 1-4 dans la variable : Dest (la variable doit être dimensionnée à 4)
	64	Lire le rapport cyclique (duty cycle) des ports 5-8 dans la variable (la variable doit être dimensionnée à 4)
	65	Lire le rapport cyclique des ports 9-12 dans la variable (la variable doit être dimensionnée à 4)
	66	Lire le rapport cyclique des ports 13-16 dans la variable (la variable doit être dimensionnée à 4)
	67	Lire le rapport cyclique des ports 1-8 dans la variable (la variable doit être dimensionnée à 8)
	68	Lire le rapport cyclique (duty cycle) des ports 9-16 dans la variable (la variable doit être dimensionnée à 8)
	69	Lire le rapport cyclique (duty cycle) des ports 1-16 dans la variable (la variable doit être dimensionnée à 16)
	70	Configurer le temps d'anti-rebond (debounce) du port 1 à partir de la variable

Paramètre *& Type de données*	**Entrée**	
	71	Configurer le temps d'anti-rebond (debounce) du port 2 à partir de la variable
	72	Configurer le temps d'anti-rebond du port 3 à partir de la variable
	73	Configurer le temps d'anti-rebond du port 4 à partir de la variable
	74	Configurer le temps d'anti-rebond du port 5 à partir de la variable
	75	Configurer le temps d'anti-rebond du port 6 à partir de la variable
	76	Configurer le temps d'anti-rebond du port 7 à partir de la variable
	77	Configurer le temps d'anti-rebond du port 8 à partir de la variable
	78	Configurer le temps d'anti-rebond du port 9 à partir de la variable
	79	Configurer le temps d'anti-rebond du port 10 à partir de la variable
	80	Configurer le temps d'anti-rebond du port 11 à partir de la variable
	81	Configurer le temps d'anti-rebond du port 12 à partir de la variable
	82	Configurer le temps d'anti-rebond du port 13 à partir de la variable
	83	Configurer le temps d'anti-rebond du port 14 à partir de la variable
	84	Configurer le temps d'anti-rebond du port 15 à partir de la variable
	85	Configurer le temps d'anti-rebond du port 16 à partir de la variable
	86	Configurer les ports port 16-13 à partir des paramètres Mode
	87	Configurer les ports port 12-9 à partir des paramètres Mode
	88	Configurer les ports port 16-13 à partir des paramètres Mode
	89	Configurer les ports port 4-1 à partir des paramètres Mode
	90	Configurer les ports port 16-13 à partir des paramètres Mode
	91	Lire l'état des ports 1-16 dans une variable (DEST). Le résultat est une représentation décimale de 16 bits de 0 à 65535.
	92	Lire l'état des ports 1-16 dans 16 variables séparées (la variable DEST doit être dimensionnée à 16). DEST (1) donne l'état du port 1, DEST (2) du port 2, etc. L'état est représenté par 0 ou 1.
	93	Configurer l'état des ports 1-16 d'une variable simple (DEST). La variable DEST devra être une représentation décimale de 16 bits de 0 à 65535.
	94	Configurer l'état des ports 1-16 de 16 variables séparées (la variable DEST doit être dimensionnée à 16). DEST (1) place l'état du port 1, DEST (2) du port 2, etc. L'état est représenté par 0 ou 1.
	95	Configurer la direction des ports 1-16 à partir d'une variable simple (DEST). La variable DEST devra être une représentation décimale de 16 bits de 0 à 65535.
	96	Configurer la direction des ports 1-16 à partir de 16 variables séparées (la variable DEST doit être dimensionnée à 16). DEST (1) place la direction du port 1, DEST (2) du port 2, etc. la direction est représentée par 0 ou 1.
	97	Configurer le masque (mask) d'interruption des ports 1-16 d'une variable simple (DEST). La variable DEST devra être une représentation décimale de 16 bits de 0 à 65535.
	98	Configures le masque (mask) d'interruption des ports 1-16 à partir de 16 variables séparées (la variable DEST devra être dimensionnée à 16). DEST (1) place le port 1, DEST (2) le port 2, etc. Le masque est représenté par 0 ou 1.
	99	Lire la signature du système d'exploitation (OS), la version d'OS, les compteurs pour des contrôles de chien de garde "Watchdogs" et les erreurs de communication dans 4 variables séparées (la variable DEST doit être dimensionnée à 4). L'emploi de cette commande remet à zéro également les compteurs.

<table>
<tr><td>Paramètre
& Type de données</td><td>Entrée</td></tr>
<tr><td>Mode</td><td>Le paramètre Mode configure un nombre de 4 ports lorsque l’on utilise les codes commande de 86 à 90 (si un autre code de commande est utilisé, entrez 0 pour le paramètre Mode) Le Mode est entrée comme 4 digits, où chaque digit indique la configuration pour un port. Les ports sont représentés du numéro de port le plus haut à gauche au n° le plus bas à droite (par exemple ; 16 15 14 13; 12 11 10 9; 8 7 6 5; 4 3 2 1). Il y a un mode pour les ports 16 - 13, 12 - 9, 8 - 5, et 4 - 1.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Sortie logique état bas</td></tr>
<tr><td>1</td><td>Sortie logique état haut</td></tr>
<tr><td>2</td><td>Entrée numérique, pas de filtre anti-rebond (debounce filter)</td></tr>
<tr><td>3</td><td>Entrée contact sec de 3.17 msec, filtre anti-rebond</td></tr>
<tr><td>4</td><td>Entrée numérique d’interruption activée, pas de filtre anti-rebond.</td></tr>
<tr><td>5</td><td>Entrée contact sec d’interruption activée 3.17 msec, filtre anti-rebond</td></tr>
<tr><td>6</td><td>Indéfinie</td></tr>
<tr><td>7</td><td>Indéfinie</td></tr>
<tr><td>8</td><td>Indéfinie</td></tr>
<tr><td>9</td><td>Pas de changement</td></tr>
</table></td></tr>
<tr><td>Mult, Offset</td><td>Les paramètres Mult et Offset sont des constantes, variables, ligne de données ou expression pour mettre la mesure à l’échelle.</td></tr>
</table>

SDMSIO4 (Dest, Reps, SDMAddress, Mode, Command, Param1, Param2, ValuesPerRep, Multiplier, Offset)

L’instruction SDMSIO4 est utilisée afin de contrôler et de transmettre / recevoir les données provenant d’une interface SDM-SIO4 de Campbell Scientific (appareil pour 4 voie entrée / sortie série). Voir le manuel de l’interface SDM-SIO4 pour plus de détails sur son focntionnement.

Pour les instructions utilisées afin d’effectuer des communication série d’entrée / sortie sans le SDM-SIO4, voir le chapitre 12.

Paramètre *& Type de donnée*	**Entrée**
Dest : la variable	Le paramètre de DEST est la variable dans laquelle on stocke les résultats de l'instruction en recherchant les données du SIO4. Si des données sont envoyées au SIO4, alors la variable DEST devient la ligne de donnée source pour que les données à envoyer. La ligne de donnée DEST doit être au moins aussi grande que la valeur du paramètre de Reps multipliée par la valeur du paramètre de ValuesPerRep.
Reps	Le paramètre de Reps définit le nombre de SIO4s séquentiel qui sera appellé par l'instruction. La centrale de mesure interrogera le SIO4 avec l’adresse configurée par le paramètre d'adresse en premier, recevra ou enverra le nombre de valeurs configurées dans le paramètre ValuesPerRep ensuite, interrogera alors le SIO4 avec l’adresse séquentielle suivante. Si le paramètre de Reps est 2, ValuesPerRep est 3, et le paramètre de commande est configuré pour recevoir, alors trois valeurs du premier SIO4 seront envoyées aux trois premiers éléments de la rangée de la variable DEST, et trois valeurs du deuxième SIO4 seront reçues et écrites aux éléments 4 à 6 de la rangée de la variable DEST.

<table>
<tr><th>**Paramètre**
& Type de données</th><th>**Entrée**</th></tr>
<tr><td>**SDMAddress**</td><td>Le paramètre de SDMAddress définit l'adresse du SIO4 avec lequel on communique. Les adresses valides de SDM vont de 0 à 14. L'adresse 15 est réservée pour l'instruction SDMTrigger. Si le paramètre de Reps est plus grand que 1, la centrale de mesure incrémentera l'adresse du SDM pour chaque SIO4 suivant avec lequel elle communique.
L'adresse du SDM est entrée comme un nombre en base 10, à la différence des anciens SDM équipés de détrompeur (jumper) qui utilisait la base 4.</td></tr>
<tr><td>**Mode**</td><td>Le paramètre de mode définit quel port sera affecté par l’instruction.
<table><tr><th>Code</th><th>Description</th></tr><tr><td>1</td><td>Port 1 Envoyer/Recevoir Send/Receive Port 1</td></tr><tr><td>2</td><td>Port 2 Envoyer/Recevoir</td></tr><tr><td>3</td><td>Port 3 Envoyer/Recevoir</td></tr><tr><td>4</td><td>Port 4 Envoyer/Recevoir</td></tr><tr><td>5</td><td>Envoi à tous les ports (global)</td></tr></table></td></tr>
<tr><td>**Command**</td><td>Le paramètre de commande est employé pour configurer le SIO4. Les commandes sont énumérées brièvement ci-dessous. Voir le manuel SDM-SIO4 pour les détails.
<table><tr><th>Code</th><th>Description</th></tr><tr><td>1</td><td>Interrogation des données disponibles.</td></tr><tr><td>2</td><td>Prendre les signatures de l’EPROM et de la mémoire</td></tr><tr><td>3</td><td>Vider toutes les données reçues des buffers.</td></tr><tr><td>4</td><td>Envoyer les données à la centrale de mesure</td></tr><tr><td>5</td><td>Recevoir le nombre d’erreur de chien de garde « watchdog », de commande invalide exécutée, et la tension de la batterie au lithium.</td></tr><tr><td>6</td><td>Vider le buffer de transmission.</td></tr><tr><td>7</td><td>Activer la ligne de commande.</td></tr><tr><td>8</td><td>Interroger les buffers TX pour les données.</td></tr><tr><td>9</td><td>Vider les données converties du buffer.</td></tr><tr><td>66</td><td>Envoyer un seul bit de donnée à la centrale de mesure.</td></tr><tr><td>67</td><td>Prendre le code de retour.</td></tr><tr><td>320</td><td>Envoyer le bit de donnée au SDM-SIO4.</td></tr><tr><td>321</td><td>Exécuter la commande de la ligne de commande.</td></tr><tr><td>1024</td><td>Envoyer la chaîne de caractères au SIO4.</td></tr><tr><td>1025</td><td>Transmettre un bit.</td></tr><tr><td>1026</td><td>Etat du port série.</td></tr><tr><td>1027</td><td>Mode poigné de main « handshake » manuel.</td></tr><tr><td>2049</td><td>Paramètres de communication.</td></tr><tr><td>2054</td><td>Configure le filtre de réception.</td></tr><tr><td>2304</td><td>Transmettre la chaîne de caractère et/ou la donnée à l’appareil (formater/filtrer).</td></tr><tr><td>2305</td><td>Transmettre les bits.</td></tr></table></td></tr>
<tr><td>**Param1**</td><td>Param1 est le premier paramètre qui devrait être transmis au SIO4 pour la commande choisie. Se référer au manuel SDM-SIO4 pour les détails.</td></tr>
<tr><td>**Param2**</td><td>Param2 est le deuxième paramètre qui devrait être transmis au SIO4 pour la commande choisie. Se référer au manuel SDM-SIO4 pour des détails.</td></tr>
<tr><td>**ValuesPerRep**</td><td>Le paramètre de ValuesPerRep est le nombre de valeurs à être envoyé ou reçue de chaque SIO4 chaque fois que cette instruction est effectuée.</td></tr>
<tr><td>**Mult, Offset**</td><td>Ces paramètres sont le multiplicateur et l'offset (décalage du zéro) avec lesquels les mesures, reçues par la centrale de mesure et en provenance du SIO4, sont mises à l’echelle.</td></tr>
</table>

SDMSpeed (BitPeriod)

Cette instruction change la vitesse utilisée afin de synchroniser les données SDM. Le fait de ralentir la vitesse de synchronisation peut être nécessaire lorsque de longs câbles sont utilisés afin de connecter des appareils SDM à une CR1000.

Paramètres *& Type de donnée*	**Entrée**
BitPeriod *Constante ou variable*	La durée par bit, en microsecondes. Le bit le plus court disponible est de 30 µs. 30 µs sera la valeur par défaut pour la variable BitPeriod, si l'instruction SDMSpeed n'est pas présente dans le programme.

SDMTrigger

Lorsque l'instruction SDMTrigger est exécutée, la CR1000 envoie une indication de mesure de groupe à tous les appareils SDM connectés (*"measure now" group trigger*). Les initiales SDM remplacent « Synchronous Device for Measurement ». Les appareils SDM effectuent des mesures indépendamment et envoient les résultats à la centrale en série. L'instruction SDMTrigger permet à la CR1000 de synchroniser le moment où les mesures ont été effectuées. Plusieurs instructions communiquent avec les appareils SDM afin de collecter le résultat des mesures. Tous les appareils SDM ne comprennent pas les mesures de groupe (*group trigger*); vérifiez le manuel de l'appareil afin d'avoir plus de détails.

SDMSW8A (Dest, Reps, SDMAddress, FunctOp, SW8AStartChan, Mult, Offset)

L'instruction SDMSW8A est utilisée afin de contrôler le module à 8 voies de contacts secs SDM-SW8A, et afin de stocker les résultats de ces mesures dans une ligne de donnée variable.
Pour des raisons techniques certaines instructions ne sont pas traduites. A l'occasion d'une mise à jour de ce manuel dans le futur, nous ne manquerons pas d'achever la traduction de ce manuel.

Paramètre *& Type de donnée*	**Entrée**
Dest *Variable or Ligne de donnée*	The variable in which to store the results of the SW8A measurement. The variable array for this parameter must be dimensioned to the number of Reps.
Reps *Constante*	The number of channels that will be read on the SW8A. If (StartChan +Reps –1) is greater than 8, measurement will continue on the next sequential SW8A. In this instance, the addresses of the SDM devices must be consecutive.
SDMAddress *Constante*	The address of the first SW8A with which to communicate. Valid SDM addresses are 0 through 15. If the SDMTrigger instruction is used in the program, address 15 should not be used. If the Reps parameter used more channels than are available on the first SW8A, the datalogger will increment the SDM address for each subsequent device that it communicates with.

<table>
<tr><td>Paramètre
& Type de donnée</td><td colspan="3">Entrée</td></tr>
<tr><td>FunctOp
Constante</td><td colspan="3">The FunctOp is used to determine the result that will be returned by the SW8A.</td></tr>
<tr><td></td><td>Numeric Code</td><td colspan="2">Function</td></tr>
<tr><td></td><td>0</td><td colspan="2">Returns the state of the signal at the time the instruction is executed. A 0 is stored for low and a 1 is stored for high.</td></tr>
<tr><td></td><td>1</td><td colspan="2">Returns the duty cycle of the signal. The result is the percentage of time the signal is high during the scan interval.</td></tr>
<tr><td></td><td>2</td><td colspan="2">Returns a count of the number of positive transitions of the signal.</td></tr>
<tr><td></td><td>3</td><td colspan="2">Returns a value indicating the condition of the module:</td></tr>
<tr><td></td><td></td><td>positive integer:</td><td>ROM and RAM are good</td></tr>
<tr><td></td><td></td><td>negative value:</td><td>RAM is bad</td></tr>
<tr><td></td><td></td><td>Zero:</td><td>ROM is bad</td></tr>
<tr><td>StartChan
Constante</td><td colspan="3">The first channel that should be read on the SW8A. If the Reps parameter is greater than 1, measurements will be made on sequential channels.</td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de donnée ou Expression</td><td colspan="3">A multiplier and offset by which to scale the raw results of the measurement. See the measurement description for the units of the raw result; a multiplier of one and an offset of 0 are necessary to output in the raw units. For example, the TCDiff instruction measures a thermocouple and outputs temperature in degrees C. A multiplier of 1.8 and an offset of 32 will convert the temperature to degrees F.</td></tr>
</table>

SDMX50 (SDMAddress, Channel)

L'instruction SDMX50 permet à des contacts (switches) individuels d'être activés indépendamment de l'instruction pilotant le TRD100.

SDMX50 est utilisée afin de sélectionner un capteur en particulier ou afin de résoudre un problème ou de déterminer la longueur apparente du câble.

Comme il est souvent aisé d'entendre le contact du ou des multiplexeurs, l'instruction SDMX50 est un outil pratique afin de tester l'adressage et le câblage de plusieurs niveaux de multiplexeurs : on programme la centrale de mesure afin de scruter toutes les x secondes l'adresse SDM du multiplexeur et la voie 8. L'instruction débutera toujours par la voie 1 et se connectera par contact à la voie demandée dans le programme. Le contact destiné à la voie 8 produira le bruit avec la plus longue durée.

Il faut se rappeler que chaque niveau de multiplexeur doit avoir une adresse SDM différente. Les multiplexeurs de niveau 1 devront avoir une adresse supérieure de 1 par rapport à celle du TDR100, les multiplexeurs de niveau 2 devront avoir une adresse supérieure de 2 par rapport à celle du TDR100, les multiplexeurs de niveau 3 devront avoir une adresse supérieure de 3 par rapport à celle du TDR100. Si les multiplexeurs SDMX50 d'un niveau donné sont connectés et configurées correctement avec leurs bonnes adresses, ils devraient tous faire du bruit en même temps.

Paramètre *& Type de données*	**Entrée**
SDMAddress *Constante*	Le SDMAddress du SDMX50 à commuter. Les adresses valides du SDM vont de 0 à 14.
Channel *Constante*	La voie du SDMX50 à commuter de (1à 8).

TDR100 (Dest, SDMAddress, Option, Mux/ProbeSelect, WaveAvg, Vp, Points, CableLength, WindowLength, ProbeLength, ProbeOffset, Mult, Offset)

L'instruction peut être utilisée afin de mesurer un capteur TDR connecté directement au TDR100, ou de multiples capteurs TDR connectés à un ou plusieurs multiplexeurs SDMX50.

<table>
<tr><th>Paramètre
& Type de donnée</th><th>Entrée</th></tr>
<tr><td>Dest</td><td>Le paramètre de DEST est une variable ou une ligne de donnée de variables dans laquelle sont stockées les résultats de la mesure. La variable doit être dimensionnée pour s'adapter à toutes les valeurs retournées par l'instruction, qui est déterminée par le paramètre d'option.</td></tr>
<tr><td>SDMAddress</td><td>Le paramètre de SDMAddress définit l'adresse du TDR100 avec lequel vous communiquez. Les adresses valides de SDM vont 0 à 14. L'adresse 15 est réservée pour l'instruction SDMTrigger. Si le paramètre de Reps est plus grand que 1, la centrale de mesure incrémentera l'adresse du SDM pour chaque TDR100 suivant avec lequel elle communique.
Note : Les centrales de mesure de type CRBasic sont programmées en base 10 de 0 à 14, les centrales de mesure de type EDLOG sont programmées en base 4 (par exemple, CR10X, CR23X)</td></tr>
<tr><td>Option</td><td>Le paramètre d'option détermine la sortie de l'instruction.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Mesure La/L (rapport de la longueur apparente, sur la longueur physique de la tige de la sonde)</td></tr>
<tr><td>1</td><td>Collecter les valeurs de forme d'onde – sauvegarde les valeurs de forme d'onde de réflexion en tant que ligne de donnée de nombres à virgule flottante compris entre -1 et 1. Les valeurs de forme d'onde sont préfacées par un en-tête contenant les valeurs des paramètres principaux pour cette instruction (faire la moyenne, vitesse de propagation, points, longueur de câble, longueur de la fenêtre, longueur de la tige, offset de la sonde, multiplicateur, décalage du zéro (offset))</td></tr>
<tr><td>2</td><td>Collecter la forme d'onde plus le premier dérivé – Retourne les valeurs (2*n-5) +9 où n est le nombre de valeurs de réflexion de forme d'onde indiquées par le paramètre des « Points ».</td></tr>
<tr><td>3</td><td>Mesurer la conductivité électrique - sort une valeur qui une fois multipliée par le paramètre du multiplicateur détermine la conductivité électrique du sol en S/m.</td></tr>
</table></td></tr>
<tr><td>Mux/ ProbeSelect</td><td>Le paramètre Mux/Probe Select est employé pour définir la configuration de tous les multiplexeurs et sondes connectés au système. Le système d'adressage utilisé est ABCR, où A = voie du multiplexeur de niveau 1, B = voie du multiplexeur de niveau 2, C = voie du multiplexeur de niveau 3, et R = le nombre de sondes consécutives à lire, commençant par le canal indiqué par la valeur d'ABC (maximum de 8). 0 est écrit pour n'importe quel niveau non utilisé.</td></tr>
</table>

Paramètre *& Type de données*	**Enter**
WaveAvg	Le paramètre WaveAvg est employé pour définir le nombre de réflexions de forme d'onde moyennées par le TDR100 pour donner un seul résultat. Une valeur moyennée de 4 fournit un bon rapport signal/bruit pour des applications typiques. Dans des conditions élevées de bruit le nombre de moyennes peut être augmenté. Le maximum de moyennes possibles est de 128.
Vp	Le paramètre de Vp vous permet d'entrer la vitesse de propagation d'un câble lorsque vous utilisez l'instruction pour déterminer des longueurs ou des défauts de câble. L'ajustement de Vp n'est pas nécessaire pour la teneur en eau du sol ou la mesure de conductivité électrique et devra être configuré à 1.0 pour l'option de sortie 1, 2, ou 3.
Points	Le paramètre de « Points » est employé pour définir le nombre de valeurs dans la forme d'onde affichée ou collectée (20 à 2048). Une entrée de 251 est recommandée pour des mesures de l'eau dans le sol. La forme d'onde est constituée du nombre de points équidistants au travers de la fenêtre « WindowLength ».
CableLength	Le paramètre de CableLength (longueur de câble) est utilisé pour indiquer la longueur de câble, en mètres, des sondes TDR. Si un 0 est entré pour le paramètre d'option, la longueur de câble est employée par l'algorithme d'analyse pour commencer à rechercher la sonde TDR. Si un 1 ou 2 est écrit pour le paramètre d'option, la longueur mentionnée est la distance sur le câble, à partir de laquelle débutera la collecte de la forme d'onde. La valeur utilisée pour CableLength est mieux déterminée en utilisant le logiciel PCTDR100 avec le Vp = 1.0. Ajuster les valeurs de CableLength et de WindowLength dans PCTDR100 jusqu'à ce que la réflexion de la sonde puisse être affichée. Soustraire environ 0.5 mètre de la distance associée au commencement de la réflexion de la sonde. Noter que le « CableLength » indiqué s'applique à toutes les sondes lues par cette instruction ; donc, toutes les sondes doivent avoir les mêmes longueurs de câble.
WindowLength	Le paramètre de WindowLength indique la longueur, en mètres, de la forme d'onde à collecter ou à analyser. La forme d'onde commence à « CableLength » et se termine à « CableLength + WindowLength ». C'est une longueur apparente parce que la valeur donnée pour Vp peut ne pas être la vitesse réelle de propagation. Pour des mesures de teneur en eau, « WindowLength » doit être assez grand pour contenir la réflexion entière de la sonde, pour des sondes ayant des tiges de 20 à 30 centimètres. Un Vp = 1 et une longueur de fenêtre (Window length) = 5 est recommandé.
ProbeLength	Le paramètre de ProbeLength indique la longueur, en mètres, des tiges de la sonde qui sont exposées au milieu mesuré. La valeur de ce paramètre n'a d'importance que l'option 0, La/L, est employée pour la mesure.
ProbeOffset	« ProbeOffset » est une valeur apparente de longueur employée pour corriger la partie des tiges de sonde qui sont encapsulées dans de l'époxy et qui ne sont pas entourées par de la terre ou tout autre milieu mesuré. Cette valeur est fournie par Campbell Scientific pour les sondes que nous fabriquons. La valeur de ce paramètre n'a d'effet que lorsque l'option 0, La/L, est employée pour la mesure.
Mult, Offset	Les paramètres Mult et l'offset sont chacun une constante, variable, ligne de données, ou expression par lesquel les résultats de la mesure sont mis à l'échelle.

Chapitre 8. Instructions mathématiques de calculs

Opérateurs

^	Mettre à la puissance
*	Multiplier
/	Diviser
+	Additionner
-	Soustraire
=	Egal
<>	Différent de
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à

ABS (Source)

Donne comme résultat, la valeur absolue d'un nombre.

Syntaxe
x = ABS (*source*)
Remarques

La source peut être n'importe quelle expression numérique valide. La valeur absolue d'un nombre est sa grandeur (magnitude), sans signe. Par exemple **ABS(**-1**)** et **ABS(**1**)** donnent tous les deux un résultat de 1.

Exemple de fonction ABS

L'exemple ci-dessous trouve la valeur absolue d'une racine carrée. Il utilise la fonction ABS afin de déterminer la différence absolue entre deux nombres.

```
Dim Precision, Value, X, X1, X2        'Déclaration des variables.
Precision = .00000000000001
Value = Volt(3)                        'Evaluation de Volt(3).
X1 = 0: X2 = Value                     'Effectue deux premières hypothèses.
'Boucle jusqu'à ce que la différence entre les deux hypothèses soit moins
'importante que la précision.
Do Until Abs(X1 - X2) < Precision
X = (X1 + X2) / 2
If X * X * X - Value < 0 Then          'Adjuster les hypothèses.
      X1 = X
Else
      X2 = X
End If
Loop

'X est maintenant la racine cubique de la variable Volt(3).
```

ACOS (Source)

La fonction ACOS donne comme résultat l'arc cosinus d'un nombre.

Syntaxe

x = ACOS *(source)*

Remarques

La source peut être n'importe quelle expression numérique valide dont la valeur est comprise entre –1 et 1 y compris.

La fonction ACOS prend le rapport des deux côtés de l'angle droit et donnent le résultat correspondant. Le rapport est la longueur du côté adjacent à l'angle, divisé par la longueur de l'hypoténuse. Le résultat est exprimé en radians et doit être compris entre -π/2 et π/2 radians.

Pour convertir des degrés en radians, on multiplie les degrés par π/180. Pour convertir des radians en degrés, on multiplie les radians par 180/π.

ACOS est l'inverse trigonométrique de la fonction COSINUS, qui prend un angle pour argument, et donne comme résultat le rapport de longueur du côté adjacent de l'angle, par rapport à l'hypoténuse.

Exemple de fonction ACOS

L'exemple utilise ACOS afin de calculer π. Par définition, un cercle plein mesure 2π radians. ACOS (0) est π/2 radians (90 Degrés).

```
Public Pi                          'Déclaration de la variable
Pi = 2 * ACOS (0)                  'Calcule la valeur de Pi
```

AddPrecise (PrecisionVariable, X)

La fonction AddPrecise vous permet d'effectuer des cumuls de haute précision sur des variables ou des manipulations de haute précision sur ces variables.

Syntaxe

AddPrecise (PrecisionVariable, X)

Remarques

Dans cette fonction, la variable X est ajoutée à la variable « PrecisionVariable ». Chaque référence à la variable « PrecisionVariable » causera une extension de 32 bit de la mantisse, utilisée et sauvegardée en interne. Une précision normale d'une valeur flottante a 24 bits de mantisse ; ainsi cette nouvelle précision se fera avec une mantisse à 56 bits. Cette fonction peut être utilisée lorsqu'on cherche à définir une différence entre deux variables de haute précision.

La variable **PrecisionVariable** est celle qui sera affectée par l'ajout de précision.

La variable **X** est la valeur qui sera ajoutée à la variable PrecisionVariable. Elle peut être ou ne pas être une variable de haute précision, selon qu'elle ait été déclarée dans une instruction de type AddPrecise ou MovePrecise.

AND

Cette instruction est utilisée afin d'effectuer une comparaison de bits entre deux nombres.

Syntaxe

Résultat = nombre 1 **And** nombre 2

Remarques

Si, et seulement si les deux expressions sont évaluées en tant que Vrai, alors le résultat est Vrai. Si l'une ou l'autre des expressions est fausse, le résultat est Faux. Le tableau suivant illustre la façon dont le résultat est déterminé.

If nombre 1 **est**	**And** nombre 2 **est**	Le Résultat **est**
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

L'opérateur And effectue une comparaison binaire de bits positionnés de façon identique dans un nombre, et met dans la variable résultante le bit correspondant, selon le tableau suivant :

If bit in nombre 1 **est**	**And bit in** nombre 2 **est**	Le Résultat **est**
0	0	0
0	1	0
1	0	0
1	1	1

Exemple d'opérateur And

L'exemple assigne une valeur à Msg, qui dépend de la valeur des variables A, B et C, en considérant qu'aucune variable n'est nulle. Si A = 10, B = 8, et C = 6, les deux expressions sont évaluées étant vraies. Comme ces deux expressions sont vraies, l'expression And est aussi vraie.

```
Dim A, B, C, Msg                'Déclaration de variables.
A = 10: B = 8: C = 6            'Assigne les valeurs.
If A > B And B > C Then          'Evaluation de l' expressions.
      Msg = True
Else
      Msg = False
End If
```

ASIN (Source)

La fonction ASIN donne comme résultat l'arc sinus du nombre.

Syntaxe
x = ASIN (*source*)

Remarque
La source peut être n'importe quelle expression numérique valide qui a une valeur comprise entre –1 et 1.

La fonction ASIN prend le rapport des deux côtés d'un angle droit, et donne en retour l'angle correspondant. Le rapport est celui de la longueur du côté opposé de l'angle, divisé par la longueur de l'hypoténuse. Le résultat est exprimé en radian et est compris entre $-\pi/2$ et $\pi/2$ radians.

Pour convertir des degrés en radian, il faut multiplier les degrés par $\pi/180$. Pour convertir des radians en degrés, il faut multiplier les radians par $180/\pi$.

ASIN est l'inverse trigonométrique de la fonction SINUS, qui prend un angle pour argument, et donne comme résultat le rapport de la longueur du côté opposé de l'angle, par rapport à l'hypoténuse.

Exemple de fonction ASIN

L'exemple utilise ASIN afin de calculer π. Par définition, un cercle plein mesure 2π radians. ASIN (1) est $\pi/2$ radians (90 degrés).

```
Public Pi                          'Déclaration de la variable
Pi = 2 * ASin (1)                  'Calcule la valeur de Pi
```

ATN (Source)

Cette fonction donne pour résultat l'arc tangente d'un nombre.

Syntaxe

x = ATN (source)

Remarques

La source peut être n'importe quelle expression numérique valide.

La fonction **Atn** prend le rapport des deux côtés d'un angle droit, et donne comme résultat l'angle qui y correspond. Le rapport est la longueur du côté opposé à l'angle droit, sur la longueur du côté adjacent à l'angle. Le résultat est exprimé en radians et est compris entre $-\pi/2$ et $\pi/2$ radians.

Pour convertir des degrés en radian, il faut multiplier les degrés par $\pi/180$. Pour convertir des radians en degrés, il faut multiplier les radians par $180/\pi$.

Atn est l'inverse trigonométrique de la fonction Tan, qui prend comme argument la valeur d'un angle, et donne comme résultat le rapport des deux côtés opposés à cet angle droit. Ne confondez pas Atn avec la cotangente, qui est l'inverse simple de la tangente (1/tangente).

Exemple de fonction Atn

L'exemple utilise ATN afin de calculer π. Par définition, un cercle plein mesure 2π radians. ATN (1) est π/4 radians (45 Degrés).

```
Dim Pi                  'Déclaration de la variable
Pi = 4 * Atn (1)        'Calcule la valeur de Pi
```

ATN2()

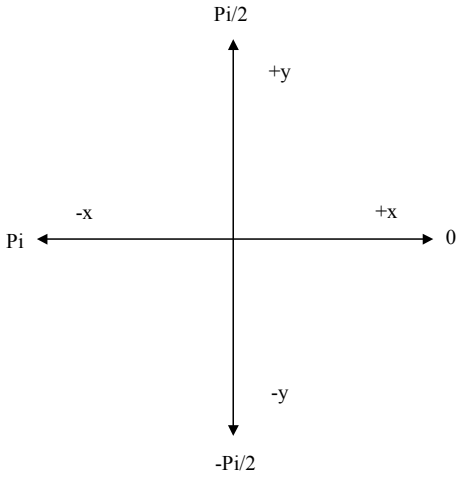
Cette fonction ATN2 donne pour résultat l'arc tangente d'un rapport y/x.

Syntaxe

x = ATN2 (Y, X)

Remarques
La fonction ATN2 calcule les arguments de Y/X et donne comme résultat une valeur comprise entre Pi et –Pi radians, avec le signe pour chacun des paramètres, afin de déterminer dans quel quadrant on donne le résultat. ATN2 est définie pour tous les points, à l'exception de l'origine (X = 0 et Y = 0). X et Y peuvent être des variables, des constantes ou des expressions.

Pour convertir des degrés en radians, il faut multiplier les degrés par π/180. Pour convertir des radians en degrés, il faut multiplier les radians par 180/π.



ATN2 est l'inverse trigonométrique de la fonction Tan, qui prend comme argument la valeur d'un angle, et donne comme résultat le rapport des deux côtés d'un angle droit. Ne confondez pas ATN2 avec la cotangente, qui est l'inverse simple de la tangente (1/tangente).

Exemple de fonction ATN2

L'exemple utilise ATN afin de calculer π. Par définition, un cercle plein mesure 2π radians. ATN2 (1, 1) est π/4 radians (45 degrés).

```
Dim Pi                  'Déclaration de la variable
Pi = 4 * ATN2 (5, 5)    'Calcule la valeur de Pi
```

AvgSpa (Dest, Swath, Source)

Cette fonction calcule la moyenne spatiale des valeurs contenues dans la ligne de données source.

Syntaxe
AvgSpa (*Dest, Swath, Source***)**

Remarques

L'instruction calcule la moyenne des valeurs de la ligne de données correspondante, et place le résultat dans la variable de *Destination*. La source doit être un élément particulier d'une ligne de données (par exemple Temp(1)) ; c'est le premier élément d'une ligne de donnée à inclure dans la moyenne. La fenêtre (*Swath*) est le nombre d'éléments à inclure dans la moyenne.

$$Dest = \frac{\sum_{i=j}^{i=j+swath} X(i)}{swath}$$

Où *X(j)* = Source

Paramètres *& Type de donnée*	**Entrée**
Dest *Variable*	La variable dans laquelle on stocke le résultat de l'instruction.
Swath *Constante*	Le nombre de valeurs à moyenner, contenues dans la ligne de données source.
Source *Ligne de données*	Le nom de la ligne de donnée (variable) qui est en entrée pour cette instruction.

Exemple de fonction Moyenne Spatiale

Cet exemple utilise la fonction AvgSpa afin de calculer la moyenne des valeurs des 5 éléments compris entre Temp(6) et Temp(10), et stocker le résultat dans la variable MoyTemp.

```
AvgSpa (MoyTemp, 5, Temp(6))
```

AvgRun (Dest, Reps, Source, Number)

Cette fonction calcule la moyenne flottante des mesures ou des valeurs calculées.

Syntaxe
AvgRun (*Dest, Reps, Source, Number*)

Remarques

La fonction AvgRun est utilisée afin de créer une moyenne flottante. La moyenne flottante est la moyenne de N valeurs où N est un nombre de valeurs.

$$Dest = \frac{\sum_{i=1}^{i=N} X_i}{N}$$

Où X_N est la valeur la plus récente de la variable source, et X_{N-1} est la valeur précédente (X_1 est la valeur la plus ancienne contenue dans la moyenne, c'est à dire que N-1 valeurs sont prises en compte antérieurement à la valeur la plus récente).

Paramètres *& Type de donnée*	**Entrée**
Dest *Variable ou Ligne de donnée*	La variable ou la ligne de données dans laquelle on stocke le résultat de la (des) moyenne(s).
Reps *Constante*	Quand la source est une ligne de données, c'est le nombre de variables dans la ligne de données, pour lesquelles on va calculer la moyenne. Lorsque la source n'est pas une ligne de données, ou que seule une variable de la ligne de données doit être moyennée, le nombre de répétition doit être 1.
Nombre *Constante*	Le nombre de valeurs à inclure dans la moyenne flottante.
Source *Ligne de données*	Le nom de la ligne de donnée (variable) qui doit être moyennée.

Exemple

```
BeginProg                          'Le programme débute ici
    Scan( RATE, RUNITS, 0, 0 )  'Scrutation de 1mSecs
    '_____________________ Volt Blocks _____________________
    VoltDiff(HiVolts, VREP1, VRNG1, 5, 1, 0, VDLY1, VINT1, VMULT1,
        VOSET1)
    AvgRun(AvgOut,1,HiVolts,100  )  'Mettre la moyenne des 100
        'variables HiVolts dans la variable AvgOut
    CallTable MAIN        'Retourner au début du programme et appeler le
     'tableau MAIN
    Next Scan             'Répéter la boucle pour la prochaine scrutation
EndProg            'Le programme se termine ici
```

Cos (Source)

Cette fonction donne comme résultat le cosinus d'un angle spécifié en radian.

Syntaxe
x = Cos (*Source***)**

Remarques
La source peut être n'importe quelle expression numérique valide exprimée en radian.

La fonction Cosinus prend la valeur d'un angle, et donne en retour le rapport des deux côtés d'un angle droit. Le rapport est celui du côté adjacent à l'angle, divisé par la longueur de l'hypoténuse. Le résultat est inclus entre –1 et 1.

Pour convertir des degrés en radian, il faut multiplier les degrés par $\pi/180$. Pour convertir des radians en degrés, il faut multiplier les radians par $180/\pi$.

Exemple de fonction Cos
Cet exemple utilise la fonction Cosinus afin de calculer le cosinus d'un angle ayant un nombre de degrés spécifié par l'utilisateur.

```
Dim Degrees, Pi, Radians, Ans          'Déclaration des variables
BeginProg
Pi = 4 * Atn(1)                        'Calcul de Pi
Degrees = Volts (1)                    'Acquiert la valeur à convertir
Radians = Degrees * (Pi / 180 )        'Convertit les degrés en radians
Ans = Cos(Radians)                     'Donne le cosinus de la valeur 'en
                                       degrés
EndProg
```

CosH (Source)

Cette fonction donne comme résultat le cosinus hyperbolique d'une expression ou d'une valeur.

Syntaxe

x = COSH (*Source***)**

Remarques

La fonction COSH prend la valeur et donne en retour le cosinus hyperbolique [COSH (x) = 0.5 (ex + e-x)] pour cette valeur.

Exemple de fonction COSH

Cet exemple utilise la fonction COSH afin de calculer le cosinus hyperbolique d'une valeur d'entrée en volts, et stocke le résultat dans la variable Ans.

```
Public Volt1, Ans          'Déclaration des variables
BeginProg
Scan (1, Sec, 3, 0)
VoltSe (Volt1,1,1,1.0,0)   'Donne comme résultat la tension de la voie 1
Ans = COSH(Volt1)
Nextscan
EndProg
```

Spatial Covariance

L'instruction CovSpa calcule la(les) covariance(s) de jeux de données qui sont chargés dans des lignes de données.

Syntaxe

CovSpa(Dest, NumOfCov, SizeOfSets, CoreArray; DatArray)

CovSpa calcule la(les) covariance(s) entre les données présentes dans la ligne de donnée « CoreArray » et un ou plusieurs jeux de données présent(s) dans la ligne de données « DatArray ». La covariance du jeu de données *X* et *Y* est calculée en tant que :

$$Cov(X,Y) = \frac{\sum_{i=1}^{n} X_i \cdot Y_i}{n} - \frac{\sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i}{n^2}$$

Où n est le nombre de valeurs de chaque jeu de données (**SizeofSets**). X_i et Y_i sont les valeurs individuelles de X et Y.

Paramètre *& Type de données*	**Entrée**
Dest *Variable ou Ligne de données*	La variable où l'on stocke le résultat de l'instruction. Lorsque des covariances multiples sont calculées, les résultats sont stockés dans une ligne de données avec le nom de la variable. Une ligne de donnée doit avoir au moins la dimension du nombre de valeurs de « NumOfCov ».
NumOfCov *Constante*	Le nombre de covariances à calculer. Si quatre jeux de données doivent être comparés par rapport à un cinquième, ce paramètre aura la valeur de quatre.
SizeOfSets *Constante*	Le nombre de valeurs dans les jeux de données, pour les calculs de covariance.
CoreArray *Ligne de données*	La ligne de donnée qui contient le jeu de données de cœur (core). La covariance de ce jeu de données par rapport aux autres jeux de données, est calculée indépendamment. Les données ont besoin d'être à la suite dans la ligne de données. Si la première valeur de la donnée n'est pas le premier point de la ligne de données, le premier point du jeu de données doit être spécifié dans ce paramètre.
DatArray *Ligne de données*	La ligne de données qui contient le(s) jeu(x) de données à comparer au jeu de données de cœur. Lorsque plusieurs covariances sont calculées, les jeux de données doivent être chargées de façon consécutive dans une seule ligne de données. La ligne de données doit avoir la dimension d'au moins la valeur de NumOfCov multiplié par SizeOfSets. Par exemple si chacun des jeux de données a 100 éléments (SizeOfSets), et que 4 covariances (NumOfCov) sont à calculer, alors DatArray devra avoir la dimension de 4 x 100 = 400. Si la première valeur du premier jeu de données n'est pas le premier point de la ligne de donnée, le premier point du jeu de donnée doit être spécifié dans ce paramètre.

DewPoint (Dest, Temp, RH)

Paramètre *& Type de donnée*	**Entrée**
Dest *Variable*	La variable où stocker la température de point de rosée (°C).
Temp *Variable*	La variable qui contient la température de l'air (°C).
RH	La variable qui contient l'Humidité Relative (%).

L'instruction de calcul du point de rosée calcule la température du point de rosée à partir des valeurs de température et HR mesurées précédemment. Les résultats finaux ne seront peut être pas aussi précis que s'ils étaient calculés à partir d'un capteur de point de rosée dédié à cela, mais ils seront acceptables pour la plupart des applications d'utilisation commune

Calcul du point de rosée

Mesurez l'Humidité Relative (HR) et de la température de l'air (T_a; en °C) avec les instructions appropriées pour le capteur que vous avez.

La température de point de rosée est calculée de la façon suivante :

1. La pression de vapeur saturante (S_{vp}; en kPa) est calculée en utilisant l'équation de Lowe (voir l'instruction SatVP).

2. La pression de vapeur (V_p; en kPa) est calculée à partir de $V_p = RH * S_{vp} / 100$).

3. Le point de rosée (Td; en °C) est calculé à partir de l'inverse d'une version de l'équation de Tetens, optimisée pour les points de rosée de l'étendue de mesure -35 à 50°C :

   $T_d = (C_3 * \ln(V_p / C_1)) / (C_2 . \ln(V_p / C_1))$
   Où :
   $C_1 = 0.61078$
   $C_2 = 17.558$
   $C_3 = 241.88$

Erreur dans l'estimation du point de rosée

L'équation de Tetens est une approximation de la véritable variation de pression de vapeur saturante en fonction de la température. Cependant, les erreurs dues à l'utilisation de la forme inversée de l'équation, résultent dans des erreurs de point de rosée bien inférieures à 0,1°C.

Le facteur d'erreur le plus important, en réalité, provient d'erreurs absolues de l'étalonnage du capteur d'HR et de température.

La figure 8-1 montre la façon dont le point de rosée varie en fonction de la température et de l'humidité. On peut voir que la réponse est non linéaire par rapport à ces deux variables. Les erreurs dans la mesure de l'HR et de la température forment alors une fonction complexe en relation avec l'erreur résultante dans la valeur estimée pour le point de rosée. En pratique, l'effet des erreurs de calibrage dans la mesure de la température de l'air, peuvent être prise en compte afin d'estimer une erreur de calcul pour le point de rosée ; si par exemple l'erreur sur la température de l'air est une surestimation de 0,2°C, alors la valeur du point de rosée est surestimée d'environ 0,2°C. La figure 8-2 montre les erreurs sur le point de rosée en fonction du "pire des cas" d'une erreur de calibrage de 5% sur le capteur d'HR.

Pour des capteurs installés sur le terrain, il y a des erreurs additionnelles associées à l'exposition du capteur, comme par exemple le fait que les capteurs présents dans des abris non ventilés donnent une température de l'air légèrement plus importante que celle de l'air réel, dans des conditions de faible vent et de rayonnement important. Cependant, si les capteurs d'HR et de température sont installés dans le même abri, et sont par conséquent exposés de façon identique, alors l'estimation du point de rosée n'est pas assujettie à la même erreur que celle qui devrait être celle de la température de l'air. Ceci est du au fait que le capteur de température mesurera réellement la température du capteur d'HR, ce qui est demandé afin de dériver la pression de vapeur de l'air, et par conséquent le point de rosée.

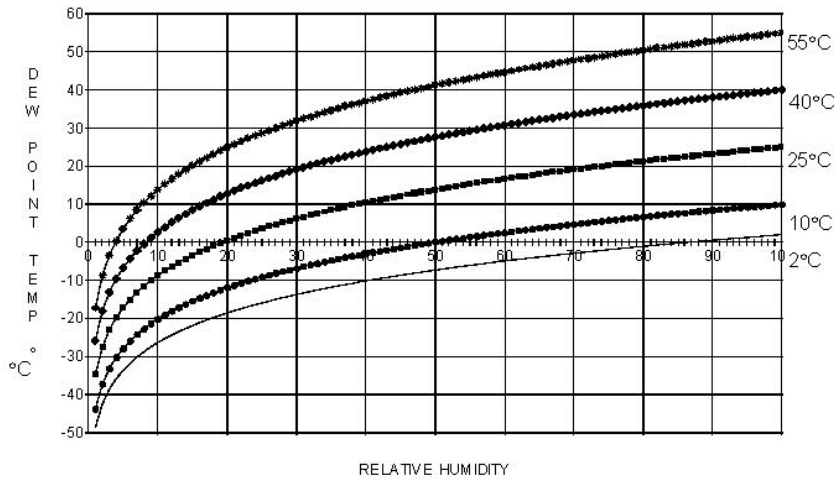


FIGURE 8-1. Température de point de rosée selon l'HR et les températures sélectionnées.

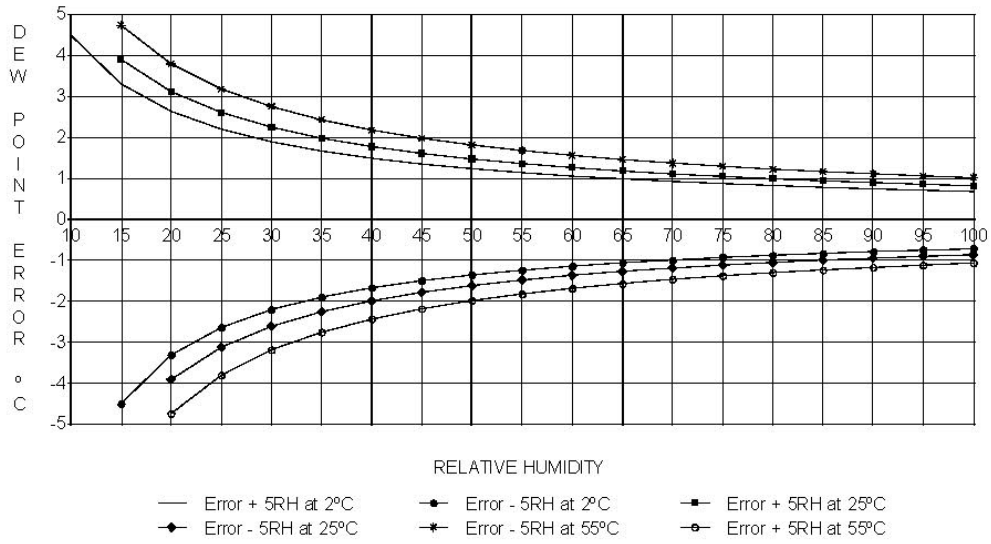


FIGURE 8-2. Effet des erreurs d'HR sur le point de rosée calculé (erreur de ±5 unités d'HR à trois températures de l'air)

Exp

Cette instruction donne comme résultat l'exponentielle « e » (la base du logarithme népérien) à la puissance donnée.

Syntaxe
x = Exp (source**)**

Remarques
Si la valeur de la source excède 709,782 712 893, une erreur de type « Overflow » se produit. La valeur constante de « e » est approximativement 2,718282.

NOTE

La fonction exponentielle fait le complément d'action de la fonction Log et est parfois appelée l'antilogarithme

Exemple de fonction Exp

L'exemple utilise la fonction Exp afin de calculer la valeur de « e ». Exp(1) est la valeur de e montée à la puissance 1.

Exp(x) est e^x donc Exp(1) est e^1 ou e.

```
Dim ValeurDeE          'Déclaration de la variable
Beginprog
ValeurDeE = Exp(1)          'Calcule la valeur de "e"
EndProg
```

FFTSpa (Dest, N, Source, Tau, Units, Option)

Pour de raisons techniques, nous n'avons pas traduit ce paragraphe.

The FFTSpa performs a Fast Fourier Transform on a time series of measurements stored in an array and places the results in an array. It can also perform an inverse FFT, generating a time series from the results of an FFT. Depending on the output option chosen, the output can be: 0) The real and imaginary parts of the FFT; 1) Amplitude spectrum. 2) Amplitude and Phase Spectrum; 3) Power Spectrum; 4) Power Spectral Density (PSD); or 5) Inverse FFT.

The difference between the FFT instruction (Section 6) and FFTSpa is that FFT is an output instruction that stores the results in a data table and FFTSpa stores its results in an array.

<table>
<tr><th>Paramètre
& Type de donnée</th><th>Entrée</th></tr>
<tr><td>Dest
Ligne de donnée</td><td>The array in which to store the results of FFT.</td></tr>
<tr><td>Source
Variable</td><td>The name of the Variable array that contains the input data for the FFT.</td></tr>
<tr><td>N
Constante</td><td>Number of points in the original time series. The number of points must be a power of 2 (i.e., 512, 1024, 2048, etc.).</td></tr>
<tr><td>Tau
Constant</td><td>The sampling interval of the time series.</td></tr>
<tr><td>Units
Constante</td><td>The units for Tau.
<table>
<tr><th>AlphaCode</th><th>Numeric Code</th><th>Units</th></tr>
<tr><td>USEC</td><td>0</td><td>microseconds</td></tr>
<tr><td>MSEC</td><td>1</td><td>milliseconds</td></tr>
<tr><td>SEC</td><td>2</td><td>seconds</td></tr>
<tr><td>MIN</td><td>3</td><td>minutes</td></tr>
</table></td></tr>
<tr><td>Options
Constante</td><td>A code to indicate what values to calculate and output.
<table>
<tr><th>Code</th><th>Result</th></tr>
<tr><td>0</td><td>FFT. The output is N/2 complex data points, i.e., the real and imaginary parts of the FFT. The first pair is the DC component and the Niquist component. This first pair is an exception because the DC and Niquist components have no imaginary part.</td></tr>
<tr><td>1</td><td>Amplitude spectrum. The output is N/2 magnitudes. With Acos(wt); A is magnitude.</td></tr>
<tr><td>2</td><td>Amplitude and Phase Spectrum. The output is N/2 pairs of magnitude and phase; with Acos(wt - ϕ); A is amplitude, ϕ is phase (- p,p).</td></tr>
<tr><td>3</td><td>Power Spectrum. The output is N/2 values normalized to give a power spectrum. With Acos(wt - ϕ), the power is $A_2 / 2$. The summation of the N/2 values yields the total power in the time series signal.</td></tr>
<tr><td>4</td><td>Power Spectral Density (PSD). The output is N/2 values normalized to give a power spectral density (power per herz). The Power Spectrum multiplied by T = N*tau yields the PSD. The integral of the PSD over a given bandwidth yields the total power in that band. Note that the bandwidth of each value is 1/T herz.</td></tr>
<tr><td>5</td><td>Inverse FFT. The input is N/2 complex numbers, organized as in the output of option 0, which is assumed to be the transform of some real time series. The output is the time series whose FFT would result in the input array.</td></tr>
</table></td></tr>
</table>

T = N*tau: the length, in seconds, of the time series.

Processing field: "FFT,N,tau,option". Tick marks on the x axis are 1/(N*tau) Herz. N/2 values, or pairs of values, are output, depending upon the option code.

Normalization details:

Complex FFT result i, i = 1 .. N/2: ai*cos(wi*t) + bi*sin(wi*t).
wi = $2\pi(i-1)/T$.
ϕi = atan2(bi,ai) (4 quadrant arctan)
Power(1) = $(a1^2 + b1^2)/N^2$ (DC)
Power(i) = $2*(ai^2 + bi^2)/N^2$ (i = 2..N/2, AC)
PSD(i) = Power(i) * T = Power(i) * N * tau
A1 = $sqrt(a1^2 + b1^2)/N$ (DC)
Ai = $2*sqrt(ai^2 + bi^2)/N$ (AC)

Notes:

- Power is independent of the sampling rate (1/tau) and of the number of samples (N).
- The PSD is proportional to the length of the sampling period (T=N*tau), since the "width" of each bin is 1/T.
- The sum of the AC bins (excluding DC) of the Power Spectrum is the Variance (AC Power) of the time series.
- The factor of 2 in the Power(i) calculation is due to the power series being mirrored about the Niquist frequency N/(2*T); only half the power is represented in the FFT bins below N/2, with the exception of DC. Hence, DC does not have the factor of 2.
- The Inverse FFT option assumes that the data array input is the transform of a real time series. Filtering is performed by taking an FFT on a data set, zeroing certain frequency bins, and then taking the Inverse FFT. Interpolation is performed by taking an FFT, zero padding the result, and then taking the Inverse FFT of the larger array. The resolution in the time domain is increased by the ratio of the size of the padded FFT to the size of the unpadded FFT. This can be used to increase the resolution of a maximum or minimum, as long as aliasing is avoided.

Frac (Source)

Cette instruction donne comme résultat la partie fractionnelle du nombre.

Syntaxe

x = Frac (source)

Remarques

Cette instruction donne comme résultat la partie fractionnelle du *nombre* qui est entre les parenthèses.

GetRecord (Dest, TableName, RecsBack)

Cette instruction reçoit un enregistrement provenant d'une table de données.

Syntaxe

GetRecord (Dest, TableName, RecsBack)

Remarques

L'instruction « GetRecord » récupère un enregistrement contenu dans une table de données. La ligne de données dans laquelle est placé le résultat de l'instruction, doit être suffisamment grand pour contenir tous les champs de l'enregistrement.

Paramètres *& type de donnée*	**Entrée**
Dest *Ligne de données*	La variable de ligne de données dans laquelle on stocke la valeur prise dans la table de données. La ligne de données doit être dimensionnée de façon à accepter touts les champs de l'enregistrement.
TableName *Nom*	Le nom de la table de données depuis laquelle on récupère les données.
RecsBack *Constante ou variable*	Le nombre d'enregistrement avant le plus récent, pour lequel on va récupérer la donnée (« 1 » indique la valeur la plus récente).

IfTime

Cette instruction est utilisée afin de donner comme résultat une valeur Vraie (-1) ou fausse (0), sur la base de l'horloge en temps réel de la centrale de mesure.

Syntaxe

IfTime (TintoInt, Interval, Units)

La fonction IfTime donne des expressions de type Vrai: *True* (-1) ou Fausse *False* (0) basées sur l'échantillonnage de l'horloge. L'horloge est sauvegardée en interne par la centrale de mesure comme le temps écoulé depuis le 1 Janvier 1990, à 00:00:00 heure. L'intervalle est synchronisé avec ce temps écoulé (par exemple, La commande est exécutée chaque fois que le temps réel correspond au temps spécifié dans l'intervalle). Le temps dans cet intervalle permet un offset dans cet intervalle. L'instruction IfTime peut être utilisée pour mettre la valeur d'une variable ou peut être utilisée comme une expression d'une condition.

The scan clock that the IfTime function checks has the time resolution of the scan interval (i.e., it remains fixed for an entire scan and increments for the next scan). IfTime must be within a scan to function.

L'instruction IfTime ne donnera la valeur « vraie » qu'une fois par intervalle. Par exemple, un programme ayant une scrutation par seconde, et une instruction IfTime (0,10, min), exécutera 60 fois l'instruction IfTime durant l'intervalle où l'instruction pourrait être vraie. Cette instruction ne retournera le résultat « vrai », que la première fois qu'elle est exécutée. Elle ne donnera le résultat vrai à nouveau, que lors de l'intervalle de 10 minutes suivant.

Paramètres *& Type de donnée*	**Entrée**
TintoInt *Constante*	Le temps à l'intérieur de l'intervalle, permet de décaler par rapport à l'horloge de la centrale de mesure, le moment où l'instruction IfTime sera vraie. Par exemple si l'intervalle est fixé à 60 minutes et que la valeur de TintoInt est fixée à 5, alors l'instruction donnera le résultat vrai à chaque cinquième minute de l'heure de la centrale de mesure. Si TintoInt est fixée à 0, la condition vraie sera à chaque heure pile.
Interval *Constante*	C'est la périodicité à laquelle l'instruction IfTime sera vraie.
Units *Constante*	C'est l'unité qui est commune à **TintoInt** et **Interval** **Code Alphanumérique** — **Unités** Usec — microsecondes Msec — millisecondes Sec — secondes Min — minutes Hr — Heures Day — Jours

IIF

La fonction IIF évalue une variable ou une expression, et donne comme retour une ou deux valeurs, en fonction de l'évaluation.

Syntaxe

Résultat = **IIF**(Expression, TrueValue, FalseValue)

Paramètre *& Type de donnée*	**Entrée**
Expression *Expression ou Variable*	La variable ou l'expression à tester. **Valeur** : **Résultat** ≠0 : Vrai : retour **TrueValue** 0 : Faux : retour **FalseValue**
TrueValue *Constante, Variable ou Expression*	La valeur (ou l'expression, selon la valeur) à retourner si la condition du test est vraie.
FalseValue *Constante, Variable ou Expression*	La valeur (ou l'expression, selon la valeur) à retourner si la condition du test est fausse.

IMP

La fonction IMP est utilisée afin d'effectuer une implication logique sur deux expressions.

Syntaxe

Résultat = expression1 IMP expression2

Remarques

Le tableau suivant illustre la façon dont les résultats sont déterminés :

Si expression1 est	Et expression2 est	Le résultat est
Vraie	Vraie	Vrai
Vraie	Fausse	Faux
Fausse	Vraie	Vrai
Fausse	Fausse	Vrai
Fausse	Nulle	Vrai

L'opérateur IMP effectue une comparaison de bits sur des bits positionnés au même endroit dans deux expressions numériques, et donne au bit en question la valeur associée, par rapport au tableau suivant :

Si le bit dans expression1 est	Et le bit dans expression2 est	Le résultat est
0	0	1
0	1	1
1	0	0
1	1	1

Int, Fix

Donne comme résultat la portion entière d'un nombre.

Syntaxe

x = Int (*source*)

x = Fix (*source*)

Remarques

La *source* peut être n'importe quelle expression numérique valide. Les fonctions **Int** et **Fix** retirent toutes les deux la partie fractionnelle de la *source* et donnent comme résultat une valeur entière.

Si le résultat de l'expression numérique est une valeur hors gamme (Not-a-Number), **Int** et **Fix** donneront comme résultat une valeur « Not-a-Number ».

La différence entre **Int** et **Fix** est visible dans le cas où le nombre est négatif; **Int** donnera comme résultat un entier négatif de valeur inférieure ou égale au nombre, lorsque **Fix** donnera comme résultat le premier entier négatif de valeur supérieure ou égale au nombre. Par exemple **Int** convertira -8.4 en -9, et **Fix** convertira -8.4 en -8.

Exemple de fonctions Int et Fix

Cet exemple illustre l'utilisation de Int et de Fix.

```
Dim A, B, C, D          'Déclaration des variables.
BeginProg
A = Int(-99.8)          'Retourne -100
B = Fix(-99.8)          'Retourne -99
C = Int(99.8)           'Retourne 99
D = Fix(99.8)           'Retourne 99
EndProg
```

Ln (Source)

Log (Source)

Cette instruction donne comme résultat le logarithme d'un nombre.

Syntaxe
x = Log (*source***)**

Remarques
La source peut être une quelconque valeur ou expression numérique dont la valeur est supérieure à 0. Le logarithme naturel est le logarithme base « e ». La constante « e » a la valeur approximative de 2.718282.

On peut calculer le logarithme en base « n » pour n'importe quel nombre « x », en divisant le logarithme naturel de « x » par le logarithme naturel de « n », tel que décrit ci-dessous :

Logn(x) = **Log(**x**) / Log(**n**)**

L'exemple suivant illustre la procédure à suivre pour calculer le logarithme base 10 :

Log10 = **Log**(X) / **Log**(10)

Exemple de fonction Log

L'exemple ci-dessous calcule la valeur de « e », puis utilise la fonction Log afin de calculer le logarithme naturel de « e » à la puissance un, deux ou trois.

```
Dim I, M          'Déclare les variables
BeginProg
M = Exp(1)
For I = 1 to 3              'A faire 3 fois
   M = Log(Exp(1) ^I)
Next I
EndProg
```

LOG10 (number)

La fonction LOG10 donne comme résultat le logarithme base 10 d'un nombre.

Syntaxe
LOG10(nombre)

Remarques
La fonction LOG10 donne comme résultat le logarithme base 10 d'un nombre.

Le nombre de l'argument peut être une quelconque expression numérique valide qui a une valeur supérieure à 0. On peut calculer le logarithme base « n » de n'importe quel nombre « x », en divisant le logarithme base 10 de « x » par le logarithme base 10 de « n », comme décrit ci-dessous :

LOGN(x) = LOG10(x) / LOG10(n)

Exemple de fonction LOG10
Cet exemple utilise l'instruction LOG10 afin de calculer le log base 2 de 1000.

```
Dim LOG2_1000          'Déclaration des variables
LOG2_1000 = LOG10(1000) / LOG10(2)
```

MaxSpa (Dest, Swath, Source)

Cette instruction permet d'obtenir la valeur maximum d'une ligne de données.

Syntaxe
MaxSpa(Dest, Swath, Source**)**

Remarques

L'instruction trouve le maximum spatial dans une fenêtre de données consécutives présentes dans une ligne de données. La source doit être un élément particulier d'une ligne de données (par exemple, Temp(1)) ; c'est le premier élément de la ligne de données, à partir duquel il faut chercher la valeur maximum.

Paramètres *& type de donnée*	**Entrée**
Dest *Ligne de données*	La ligne de données dans laquelle on stocke la valeur maximum et l'emplacement où elle était à l'intérieur de la ligne.
Swath *Constante*	Le nombre de valeurs à l'intérieur de la ligne de données source (la fenêtre), sur lequel on va chercher la valeur maximum.
Source *Ligne de données*	Le nom de la variable de données qui contient les valeurs d'entrée utilisées par l'instruction.

Exemple de fonction MaxSpa

Cet exemple utilise l'instruction MaxSpa afin de chercher la valeur maximum des 5 éléments entre Temp(6) et Temp(10), et afin de stocker le résultat dans la variable MaxTemp.

```
MaxSpa(MaxTemp, 5, Temp(6))
```

MinSpa (Dest, Swath, Source)

Cette instruction permet d'obtenir la valeur minimum d'une ligne de données.

Syntaxe
MinSpa(Dest, Swath, Source**)**

Remarques

L'instruction trouve le minimum spatial dans une fenêtre de données consécutives présentes dans une ligne de données. La source doit être un élément particulier d'une ligne de données (par exemple, Temp(1)) ; c'est le premier élément de la ligne de données, à partir duquel il faut chercher la valeur minimum. La fenêtre (*Swath*) est le nombre d'éléments sur lesquels on va chercher le minimum.

Paramètres *& type de donnée*	**Entrée**
Dest *Ligne de données*	La ligne de données dans laquelle on stocke la valeur minimum et l'emplacement où elle était à l'intérieur de la ligne.
Swath *Constante*	Le nombre de valeurs à l'intérieur de la ligne de données source (la fenêtre), sur lequel on va chercher la valeur minimum.
Source *Ligne de données*	Le nom de la variable de données qui contient les valeurs d'entrée utilisées par l'instruction.

Exemple de fonction MinSpa

Cet exemple utilise l'instruction MinSpa afin de chercher la valeur minimum des 5 éléments entre Temp(6) et Temp(10), et afin de stocker le résultat dans la variable MinTemp.

```
MinSpa(MinTemp, 5, Temp(6))
```

Mod

Cette fonction divise un nombre par un autre, et ne donne comme résultat, que le reste de la division.

Syntaxe
Résultat = operand1 Mod operand2

Remarques

Pour effectuer un modulo, ou un « reste », l'opérateur divise operand1 par operand2 (en arrondissant les nombres à virgules, en entiers) et donne comme résultat le reste seul en tant que *résultat*. Par exemple, dans l'expression A = 19 **Mod** 6.7, A (qui est le résultat) est égal à 5. Les operand peuvent être n'importe quelle valeur ou expression numérique.

Exemple d'opérateur Mod

L'exemple utilise l'opérateur **Mod** afin de déterminer si une année à 4 chiffres est une année bissextile ou non.

```
Dim TestYr, LeapStatus                                  'Déclare les variables
TestYr = 1995
If TestYr Mod 4 = 0 And TestYr Mod 100 = 0 Then         'Divisible par 4 ?
   If TestYr Mod 400 = 0 Then                           'Divisible par 400 ?
            LeapStatus = True
   Else
            LeapStatus = False
   EndIf
Else TestYr Mod 4 = 0 Then
   LeapStatus = True
Else
   LeapStatus = False
End If
```

Move (Dest, Reps, Source, Reps)

Déplace un block ou bien remplis une ligne de donnée.

Syntaxe

Move(Dest, Reps, Source, Reps**)**

Paramètre *& Type de donnée*	**Entrée**
Dest *Variable ou ligne de donnée*	La variable dans laquelle on stockera les valeurs à partir de la source.
Reps *Constante*	Le nombre d'éléments à remplir dans la ligne de donnée de destination.
Source *Ligne de donnée ou Expression*	Le nom de la variable de ligne de donnée ou bien de l'expression qui est la source des valeurs à déplacer.
Reps *Constante*	Le nombre de répétitions pour l'instruction ou la mesure. Si le nombre de répétitions de la source est moins important que le nombre de répétitions de la destination, le reste de la destination sera complétée par la dernière valeur de la source.

Exemple de fonction Move

L'exemple utilise la fonction Move afin de :

```
Move(x, 20, y, 20)      'déplacer la ligne de données y vers x
Move(x, 20, 0.0, 1)     'remplir x avec des 0.0.
```

MovePrecise (PrecisionVariable, X)

La fonction MovePrecise vous permet de déplacer une variable de haute précision vers une autre mémoire d'entrée (variable).

Syntaxe

MovePrecise (PrecisionVariable, X)

Remarques

Dans cette fonction, la variable « X » est déplacée vers la « PrecisionVariable » en tant que valeur à haute précision. Chaque référence à la « PrecisionVariable fera passer la mantisse utilisée pour la sauvegarde et les calculs internes, à une mantisse étendue de 32-bit. Une précision normale a une mantisse de 24 bits ; la nouvelle précision sera donc à 56 bits.

PrecisionVariable : La “PrecisionVariable” est la variable qui sera affectée par le déplacement de précision.

X : La variable “X” est la valeur qui sera déplacée vers “PrecisionVariable”. Elle peut être ou non, une variable à haute précision; cela dépend de l’endroit où elle a été déclarée, comme par exemple via une instruction précédente de type AddPrecise ou MovePrecise.

NOT

La fonction NOT est utilisée afin d’effectuer une négation logique sur une expression.

Syntaxe
Résultat = NOT expression

Remarques

Le tableau suivant illustre la façon dont les résultats sont déterminés.

Si l’expression est	**Le résultat est**
Vraie	Faux
Fausse	Vrai
Nulle	Nul

L’opérateur NOT est aussi un opérateur de conversion de bits ; il inverse les valeurs de bits de n’importe quelle variable et fixe le bit correspondant, dans l’expression, selon le tableau suivant :

Si le bit dans *expr1* **est**	**Le** *résultat* **est**
0	1
1	0

Exemple d’opérateur NOT

L’exemple donne la valeur à la variable Msg en fonction de l’état du Flag(1).

```
Dim A, B, C, Flag(8)          'Déclare les variables.
Public Msg
If NOT Flag(1) Then           'Evalue les expressions.
      Msg = 10
Else
      Msg = 100.
End If
```

Or

Utilisée afin d’effectuer une disjonction logique entre deux expressions.
Syntaxe

resultat **=** *expr1* **Or** *expr2*

Remarques

Si l’une ou les deux expressions sont évaluées à “Vrai”, le résultat est vrai. Le tableau suivant illustre les résultats déterminés :

Si *expr1* **est**	**Et** *expr2* **est**	**Le** *résultat* **est**
Vraie	Vraie	Vrai (True)
Vraie	Fausse	Vrai
Fausse	Vraie	Vrai
Fausse	Fausse	Faux (False)

L'opérateur **Or** effectue aussi une comparaison de bits sur des bits positionnés au même endroit dans deux expressions numériques, et donne la valeur qui correspond aux résultats attendus d'après le tableau qui suit, aux bits de l'expression :

Si le bit dans *expr1* **est**	**Et que le bit dans** *expr2* **est**	**Le** *résultat* **est**
0	0	0
0	1	1
1	0	1
1	1	1

Exemple d'opérateur Or

L'exemple fixe la valeur de la variable Msg en fonction de la valeur des variables A, B, et C, en supposant qu'aucune variable n'est nulle. Si A = 10, B = 8, et C = 11, l'expression de gauche est Vraie et l'expression de droite est Fausse. Du fait qu'au moins une des deux expressions de comparaison est Vraie, l'opérateur Or évalue le résultat en tant que « Vrai ».

```
Dim A, B, C                    'Déclare les variables.
A = 10: B = 8: C = 11          'Assigne des valeurs.
If A > B Or B > C Then          'Evalue les expressions.
    Msg = True
Else
    Msg = False.
End If
```

PeakValley (DestPV, DestChange, Reps, Source, Hysteresis)

PeakValley est utilisée afin de détecter des pics et des vallées (maximum et minimum locaux) sur un signal. Lorsqu'un nouveau pic ou une nouvelle vallée sont détectés, cette valeur ainsi que le changement par rapport au pic ou à la vallée précédente, est stocké dans des variables.

Paramètre *& Type de donnée*	**Entrée**
DestPV *Variable ou ligne de donnée*	Variable or array in which to store the new peak or valley. When a new peak or valley is detected, the value of the peak or valley is loaded in the destination. PeakValley will continue to load the previous peak or valley until the next peak or valley is detected.
DestChange *Variable ou ligne de donnée*	Variable or array in which to store the change from the previous peak or valley. When a new peak or valley is detected, the change from the previous peak or valley is loaded in the destination. When a new peak or valley has not yet been reached, 0 is stored in the destination. When Reps are greater than 1, the array must be dimensioned to Reps+1. The additional element is used to flag when a new peak or valley is detected in any of the source inputs. The flag element is stored after the changes [e.g., *changevar*(Reps+1)] and is set to -1 (true) when a new peak or valley is detected and set to 0 (false) when none are detected.
Reps *Constante*	The number inputs to track the peaks and valleys for. Each input is tracked independently. When reps are greater than 1 the source and DestPV arrays must be dimensioned to at least the number of repetitions; DestChange must be dimensioned to Reps+1.
Source *Variable ou ligne de donnée*	The variable or array containing the inputs to check for peaks and valleys.
Hysteresis *Constante, Variable ou Expression*	The minimum amount the input has to change to be considered a new peak or valley. This would usually be entered as a constant.

L'exemple qui suit utilise un signal sinus et cosinus en entrée afin d'illustrer l'utilisation de PeakValley avec des répétitions. Le tableau de données PV1 stocke les pics et les vallées du signal de la courbe cosinus. PV2 stocke les pics et les vallées de la courbe sinusoïdale. PV3 stocke les pics et les vallées des deux courbes.

```
Public Dim XY(2)

Const Pi=4*ATN(1)                    'Define Pi for converting degrees to
                                     radians

DataTable(PV1,Change(1),500)         'Peaks and valleys for first signal,
                                     triggered when 'Change(1) is not 0.
    Sample(1,PeakV(1),IEEE4)         'DataTable PV1 holds the peaks and
                                     valleys for XY(1)
EndTable

DataTable(PV2,Change(2),500)         'Peaks and valleys for second signal,
                                     triggered when 'Change(2) is not 0.
    Sample(1,PeakV(2),IEEE4)         'DataTable PV2 holds the peaks and
                                     valleys for XY(2)
EndTable

'The Following table is an alternative to using separate tables for each signal.
'It stores both signals whenever there is a new peak or valley in either signal.
'The value stored for the signal that does not have a new peak will be a repeat
'of its last peak or valley.  Normally a program would not have a table storing
'peaks and valleys for several signals, it would use individual tables for the
'signals.

DataTable(PV3,Change(3),500)
    Sample(2,PeakV(1),IEEE4)
EndTable

BeginProg
    Scan(500,mSec,0,0)
        Deg=Deg+5
        XY(1)=Cos(Deg*Pi/180)        'Compute the cosine as input XY(1)
        XY(2)=Sin(Deg*Pi/180)        'Compute the sine as input XY(2)
        PeakValley(PeakV(1),Change(1),2,XY(1),0.1)   'Find the peaks and
                                                     'valleys for both
                                                     'inputs.  Hysteresis
                                                     '= 0.1
        CallTable PV1
        CallTable PV2
        CallTable PV3
    Next Scan
EndProg
```

PRT (Dest, Reps, Source, Mult, Offset)

Utilisée afin de calculer la température à partir d'une résistance RTD.

Syntaxe

PRT ***(****Dest, Reps, Source, Mult, Offset****)***

Remarques

Cette instruction utilise les résultats d'une mesure précédente de résistance de pont RTD afin de calculer la température. L'entrée (Source) doit être le rapport de Rs/R0, où Rs est la résistance RTD et R0 est la résistance de la RTD à 0° C.

La température est calculée à partir des caractéristiques de la norme DIN 43760 ajustée (1980) pour le standard « *International Electrotechnical Commission* ». L'étendue de linéarisation est entre -200° C et 850° C. L'erreur de linéarisation est inférieure à 0,001° C entre -200 et +300° C, et elle est inférieure à 0,003° C entre -180 et +830° C. L'erreur (T calculée - T standard) est de +0,006° à -200° C et –0,006° à +850° C.

Paramètre *& Type de donnée*	**Entrée**
Dest *Variable u ligne de donnée*	La variable dans laquelle on stocke la température en °C.
Reps	Le nombre de répétitions pour la mesure ou l'instruction.
Source *Variable ou ligne de donnée*	Le nom de la variable ou de la ligne de données qui contien(nen)t la(les) valeur(s) de Rs/RO.
Mult, Offset *Constante, Variable, Ligne de données ou Expression*	Un multiplicateur et un offset grâce auxquels on effectuera une mise à l'échelle les résultats des mesures brutes. Voir la description de la mesure pour connaître l'unité de la mesure brute ; il est nécessaire de mettre un multiplicateur de 1 et un offset de 0 afin de garder la même unité que la mesure brute. Par exemple, l'instruction TCDiff mesure le signal d'un thermocouple et donne des °C. Un multiplicateur de 1,8 et un offset de 32 convertiront la température en °F.

Randomize (Source)

Cette instruction permet d'activer le générateur de valeur aléatoire (random).

Syntaxe
Randomize (source)

Remarques
L'argument *nombre* peut être n'importe quelle expression numérique valide. Le *nombre* est utilisé afin d'initialiser le générateur de valeur numérique aléatoire, en lui donnant une nouvelle valeur de départ. Si vous oubliez le *nombre*, la valeur donnée par la fonction « Timer », est alors utilisée comme nouvelle valeur de départ.

Si l'instruction « Randomize » n'est pas utilisée, la fonction Rnd donne comme résultat la même séquence de nombres aléatoires à chaque fois que le programme est exécuté. Si vous souhaitez avoir la séquence de nombres aléatoires qui change à chaque exécution du programme, mettez l'instruction Randomize, sans paramètres, au début du programme.

RealTime

Cette instruction sert à prendre la mesure de l'année, du mois, du jour, de l'heure, de la minute, de la seconde, de la semaine et/ou du jour à l'intérieur de l'année, à partir de l'horloge de la CR1000.

Syntaxe

RealTime(Dest)

Remarques

La ligne de donnée de destination doit avoir une dimension de 9.

Exemple d'instruction RealTime

Cet exemple utilise **RealTime** afin de placer toutes les définitions du temps, dans la ligne de donnée de destination. Si on enlève les 9 (') qui précèdent les premières instruction Sample, et qu'on met un (') devant la dernière instruction Sample, le résultat sera exactement le même.

```
Public rTime(9)                 'déclare comme public et dimensionne rTime à 9
Alias rTime(1) = Year           'assigne la variable  alias Year à rTime(1)
Alias rTime(2) = Month          'assigne la variable  alias Month à rTime(2)
Alias rTime(3) = Day            'assigne la variable  alias Day à rTime(3)
Alias rTime(4) = Hour           'assigne la variable  alias Hour à rTime(4)
Alias rTime(5) = Minute         'assigne la variable  alias Minuteà rTime(5)
Alias rTime(6) = Second         'assigne la variable  alias Second à rTime(6)
Alias rTime(7) = uSecond        'assigne la variable  alias uSecond à rTime(7)
Alias rTime(8) = WeekDay        'assigne la variable  alias WeekDay à rTime(8)
Alias rTime(9) = Day_of_Year    'assigne la variable  alias Day_of_Year à rTime(9)

DataTable (VALUES, 1, 100)      'configuration de la table de données (data table)
DataInterval(0, 1, mSec, 0)     'configuration de la table de données ( data table)
'  Sample(1, Year, IEEE4)       'place Year dans la table VALUES
'  Sample(1, Month, IEEE4)      'place Month dans la table VALUES
'  Sample(1, Day, IEEE4)        'place Day dans la table VALUES
'  Sample(1, Hour, IEEE4)       'place Hour dans la table VALUES
'  Sample(1, Minute, IEEE4)     'place Minute dans la table VALUES
'  Sample(1, Second, IEEE4)     'place Second dans la table VALUES
'  Sample(1, uSecond, IEEE4)    'place uSecond dans la table VALUES
'  Sample(1, WeekDay, IEEE4)    'place WeekDay dans la table VALUES
'  Sample(1, Day_of_Year, IEEE4) 'place Day_of_Year dans la table VALUES
   Sample(9, rTime(), IEEE4)    'place tous les  9 segments dans la table VALUES
EndTable

BeginProg
  Scan (1, Sec, 0, 0)
    RealTime(rTime())
    CallTable VALUES
  Next Scan
EndProg
```

RectPolar (Dest, Source)

Cette instruction convertit des coordonnées rectangulaires en coordonnées polaires.

Paramètres *& type de donnée*	**Entrée**
Dest *Ligne de données*	La ligne de données variable dans laquelle on stocke 2 valeurs résultantes. La longueur du vecteur est enregistrée dans l'élément de destination spécifiée, et l'angle en radians (+ ou - π), dans l'élément suivant de la ligne de données.
Source *Ligne de données*	La ligne de données variable qui contient les coordonnées X et Y à convertir en coordonnées polaires. La valeur de X doit être dans la ligne de donnée destination spécifiée, et la valeur de Y doit être l'élément suivant dans la ligne de données.

Exemple : Dans l'exemple suivant, un compteur (Deg) est incrémenté entre 0 et 360 degrés. Le cosinus et le sinus de l'angle sont pris afin de transformer X et Y en coordonnées rectangulaires. RecPolar est alors utilisé afin de convertir les coordonnées polaires.

```
Dim XY(2), Polar(2), Deg, AngleDeg
Const Pi=4*ATN(1)

Alias XY(1)=X
Alias XY(2)=Y
Alias Polar(1) = Longueur
Alias Polar(2) = AngleRad

Data Table(RtoP,1,500)
Sample(1,Deg)
Sample(2,XY)
Sample(2,Polar)
Sample(1,AngleDeg)
EndTable
```

```
BeginProg
    For Deg=0 to 360
        XY(1)=Cos(Deg*Pi/180)      'Cosinus et Sinus fonctionnent en radian
        XY(2)=Sin(Deg*Pi/180)
        RectPolar(Polar, XY)
        AngleDeg=Polar(2)*180/Pi   'Convertit l'angle en degré pour
                                   'comparaison
        CallTable RtoP             'w/Deg
    Next Deg
EndProg
```

RMSSpa (Dest, Swath, Source)

Cette instruction est utilisée afin de calculer la racine carrée moyenne (Root Mean Square) d'une ligne de données.

Syntaxe
RMSSpa(Dest, Swath, Source)

Remarques

La racine carrée moyenne spatiale, calcule la racine carrée des valeurs contenues dans une ligne de données.

$$Dest = \sqrt{\frac{\sum_{i=j}^{i=j+swath} (X(i))^2}{swath}}$$

Où *X(j)* = Source

Paramètres *& type de donnée*	**Entrée**
Dest *Variable*	La variable dans laquelle on stocke la valeur de la racine carrée.
Swath *Constante*	Le nombre de valeurs dans la ligne de données, à inclure dans le calcul de la moyenne.
Source *Ligne de données*	Le nom de la ligne de données variable, qui contient les données d'entrée pour l'instruction.

RND (Source)

La fonction RND est utilisée afin de générer un nombre aléatoire.

Syntaxe
Variable = RND

Remarques

La fonction RND donne comme résultat une valeur inférieure à 1 mais supérieure ou égale à 0.

La même séquence de nombres aléatoires est générée à chaque fois que l'instruction est exécutée, puisque chaque appel de la fonction RND utilise le nombre aléatoire précédemment calculé, comme point de départ du calcul de nombre aléatoire suivant.

La valeur du Nombre de l'argument détermine la façon dont le nombre aléatoire sera généré :

Valeur	Description
< 0	Le même nombre à chaque fois, tel que décrit par le Nombre
> 0	Le nombre aléatoire suivant de la séquence
= 0	Le nombre généré le plus récemment
Pas de nombre	Le nombre aléatoire suivant de la séquence

Si on souhaite que le programme génère une séquence de nombres différents à chaque occurrence, il faut alors utiliser l'instruction **Randomize** avec l'argument assigné à une variable ou une expression dont la valeur change, qui servira de valeur d'initialisation pour le générateur de nombres aléatoires ayant une valeur différente avant que RND ne soit appelée.

Pour produire des entiers dans une étendue de mesure définie, il faut utiliser la formule suivante :

INT((borne_supérieure – borne_inférieure + 1) * RND + borne_inférieure)

Ici la borne_supérieure est la valeur maximale dans l'étendue de mesure, et la borne_inférieure est la valeur minimale dans l'étendue de mesure.

SatVP (Dest, Temp)

SatVP calcule la pression de vapeur saturante (au dessus de l'eau, Svpw) en kilo pascals, à partir de la température de l'air (°C), et place le résultat dans la variable de destination. L'algorithme servant à obtenir la pression de vapeur saturante Svpw à partir de la température de l'air (en °C), est prise dans la publication de : Lowe, Paul R.: 1977, "*An approximating polynomial for computation of saturation vapor pressure*," *J. Appl. Meteor*, **16**, 100-103.

La pression de vapeur saturante au dessus de la glace, (Svpi) en kilo pascals et pour une étendue de température allant de 0°C à -50°C, peut être obtenue en utilisant l'instruction SatVP et la relation suivante :

$Svpi = -.00486 + .85471\ Svp + .2441\ Svp^2$

Où Svpw est dérivée par l'instruction SatVP. Cette relation est dérivée par Campbell Scientific à partir des équations données par l'article de Lowe pour Svpw et Svpi.

Paramètre *& Type de donnée*	**Entrée**
Dest	Variable où on stocke la pression de vapeur saturante (kPa).
	Variable contenant la température de l'air (°C).

StrainCalc (Dest, Reps, Source, BrZero, BrConfig, GF, *v*)

Cette instruction convertit le signal de sortie d'un pont de mesure, en micro contrainte (*microstrain*).

Syntaxe

StrainCalc (Dest, Reps, Source, BrZero, BrConfig, GF, *v*)

Remarques

Calcule la micro contrainte (microstrain), en με, à partir de la formule appropriée pour la configuration du pont. Tous les ponts sont des ponts complets. Les jauges de contrainte en ¼ de pont, ½ pont et pont complet se réfèrent au nombre d'éléments actifs (les jauges de contrainte), soit 1, 2, ou 4 respectivement.

<table>
<tr><td>Paramètre
& Type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest</td><td>Variable to store strain in.</td></tr>
<tr><td>Reps</td><td>Number of strains to calculate, Destination, source, and zero variables must be dimensioned accordingly.</td></tr>
<tr><td>BrConfig</td><td>Bridge configuration code for strain gages The bridge configuration code can be entered as a positive or negative number:
+ code: $V_r = 0.001(Source - Zero)$; bridge configured so its output decreases with increasing strain.
- code: $V_r = -0.001(Source - Zero)$; bridge configured so output increases with strain. This is the configuration for a quarter bridge using CSI's 4WFB350 Terminal Input Module (i.e., enter the bridge configuration code as -1 for 1/4 bridge with TIM.)
<table>
<tr><th>Code</th><th>Configuration</th></tr>
<tr><td>1</td><td>Quarter bridge strain gauge $\mu\varepsilon = \frac{-4\cdot 10^6 V_r}{GF(1+2V_r)}$</td></tr>
<tr><td>2</td><td>Half bridge strain gauge, one gage parallel to strain, the other at 90° to strain:
$\mu\varepsilon = \frac{-4\cdot 10^6 V_r}{GF[(1+\nu)-2V_r(\nu-1)]}$</td></tr>
<tr><td>3</td><td>Half bridge strain gauge, one gage parallel to $+\varepsilon$, the other parallel to $-\varepsilon$:
$\mu\varepsilon = \frac{-2\cdot 10^6 V_r}{GF}$</td></tr>
<tr><td>4</td><td>Full bridge strain gage, 2 gages parallel to $+\varepsilon$, the other 2 parallel to $-\varepsilon$:
$\mu\varepsilon = \frac{-10^6 V_r}{GF}$</td></tr>
<tr><td>5</td><td>Full bridge strain gage, half the bridge has 2 gages parallel to $+\varepsilon$ and $-\varepsilon$: the other half $+\nu\varepsilon$ and $-\nu\varepsilon$:
$\mu\varepsilon = \frac{-2\cdot 10^6 V_r}{GF(\nu+1)}$</td></tr>
<tr><td>6</td><td>Full bridge strain gage, one half $+\varepsilon$ and $-\nu\varepsilon$, the other half $-\nu\varepsilon$ and $+\varepsilon$.:
$\mu\varepsilon = \frac{-2\cdot 10^6 V_r}{GF[(\nu+1)-V_r(\nu-1)]}$</td></tr>
</table></td></tr>
<tr><td>Source</td><td>The source variable array for the measurement(s), the input is expected as millivolts out per volt in (the result of the full bridge instruction with a multiplier of 1 and an offset of 0.</td></tr>
<tr><td>BrZero</td><td>The variable array that holds the unstrained reading(s) in millivolts out per volt in.</td></tr>
<tr><td>GF</td><td>Gage Factor. The gage factor can be entered as a constant used for all repetitions or a variable array can be loaded with individual gage factors which are automatically used with each rep. To use an array enter the parameter as arrayname(), with no element number in the parentheses.</td></tr>
<tr><td>v</td><td>Poisson ratio, enter 0 if it does not apply to configuration.</td></tr>
</table>

Exemple d'utilisation de StrainCalc

Cet exemple utilise StrainCalc afin de trouver la valeur de la micro contrainte en sortie de pont de mesure.

```
'      Program name: STRAIN.DLD

Public Count, ZStrain, StMeas, Strain, Flag(8)     ' Declare all variables as
                                                    public

'Data Table STRAINS samples every measurement when user Sets Flag(1) High

DataTable(STRAINS,Flag(1),-1)
      DataInterval(0,0,0,100)                      'Interval = Scan, 100 lapses
      Sample (1,Strain,Ieee4)
EndTable

'DataTable ZERO_1 stores the "zero" measurements

DataTable(ZERO_1,Count>99,100)                     'Trigger on Count 100
      Average(1,ZStrain,IEEE4,0)
EndTable

'Subroutine to measure Zero, Called on first pass or when user sets Flag(2)low

Sub Zero
      Count = 0                                    'Reset Count
      Scan(10,mSec,0,100)                          'Scan 100 times
            BrFull(ZStrain,1,mV50,5,1,6,7,1,5000,1,0,0,100,1,0)
            Count = Count + 1                      'Increment Counter used By
                                                   DataTable
            CallTable ZERO_1                       'Zero_1 outputs on last scan
                                                   (Count=100)
      Next Scan
      ZStrain = ZERO_1.ZStrain_Avg(1,1)            'Set ZStrain = averaged
                                                   value
      Flag(2) = True
End Sub

BeginProg
      Scan(10,mSec,0,0)                                'Scan 10(mSecs)
            If Not Flag(2) Then Zero
            BrFull(StMeas,1,mV50,5,1,6,7,1,5000,1,0,0,100,1,0)
            StrainCalc(Strain,1,StMeas,ZStrain,-1,2,0)
            CallTable STRAINS                      'Strains outputs only when
Flag(1)=True
      Next Scan
EndProg
```

Sgn (Source)

Utilisé afin de trouver la valeur du signe d'un nombre.

Syntaxe

x = Sgn (*source*)

Remarques

Donne comme résultat un entier qui indique le signe du nombre.

L'argument peut être n'importe quelle expression numérique valide. Son signe détermine la valeur la valeur retournée par la fonction Sgn :

Si X > 0, alors Sgn(X) = 1.
Si X = 0, alors Sgn(X) = 0.
Si X < 0, alors Sgn(X) = -1.

Exemple de fonction Sgn

L'exemple utilise la fonction Sgn afin de déterminer le signe d'un nombre.

```
Dim Msg, Number                 'Déclare les variables.
Number = Volt(1)                'Effectue la mesure en entrée
Select Case Sgn(Number)         'Evalue le nombre.
    Case 0                      'Zero.
        Msg = 0
    Case 1                      'Positif.
        Msg = 1
    Case -1                     'Négatif.
        Msg = -1
End Select
```

Sin (Source)

Cette instruction donne comme résultat le sinus d'un angle.

Syntaxe
x = Sin (source)

Remarques
La source ne peut être qu'une expression numérique valide mesurée en radian.

La fonction **Sin** prends un angle, et donne comme résultat le rapport des deux côtés d'un triangle à angle droit. Le rapport est la longueur du côté opposé à l'angle, divisé par la longueur de l'hypoténuse.

Le résultat est compris entre -1 et 1.

Afin de convertir des degrés en radian, on multiplie par π/180. Pour convertir des radians en degrés, on multiplie par 180/π.

Cette instruction donne comme résultat le sinus de la valeur entre parenthèses. Cette valeur d'entrée doit être en radian.

Exemple de fonction Sin

Cet exemple utilise la fonction Sin afin de calculer le sinus d'un angle, à partir d'une donnée d'entrée en Volt.

```
Dim Msg, Nombre                 'Déclaration des variables
Pi = 4 * Atn(1)                 'Calcul de Pi
Degrés = Volt(1)                'On prend la valeur en entrée
Radians = Degrés * (Pi / 180)   'On convertit la valeur en radian
Ans = Sin(Radians)              'On calcul le sinus de l'angle
```

SinH (Source)

Cette instruction donne comme résultat le sinus hyperbolique d'une expression ou d'une valeur.

Syntaxe
x = SINH (*expression*)

Remarques
La fonction SINH donne comme résultat le sinus hyperbolique [SINH(x) = 0.5 (e^{x} - e^{-x})] trouvé pour la valeur contenue dans l'argument « *expression* ».

L'exemple utilise SINH afin de calculer le sinus hyperbolique de la tension d'entrée.

```
Public Volt1, Ans              'Déclare les variables.
BeginProg
Scan ( 1, min, 3, 0)
      VoltDiff(Volt1,1,mV5000,1,True,100,500,1,0)
      'Donne la tension de la voie diff. 1
      Ans = SINH( Volt1 )  'Donne le sinus hyperbolique de Volt1.
      NextScan
EndProg
```

Sqr (Source)

Cette instruction donne comme résultat la racine carrée d'un nombre.

Syntaxe
x = sqr (*nombre*)

Remarques
Le *nombre* peut être une quelconque expression numérique valide dont la valeur est supérieure ou égale à 0.

Le résultat de la fonction, est la racine carrée de la valeur contenue entre les parenthèses.

Exemple de fonction Sqr

L'exemple utilise la fonction Sqr afin de calculer la racine carrée de la valeur de la tension, Volt(1).

```
Dim Msg, Nombre                 'Déclaration des variables
Nombre = Volt(1)                'On prend la valeur en entrée
If Nombre < 0 Then
Msg = 0                         'On ne peut pas calculer la racine carrée d'un
                                nombre négatif
Else
Msg = Sqr(Nombre)
End If
```

StdDevSpa (Source)

Cette instruction est utilisé afin de calculer l'écart type d'une ligne de donnée.
Syntaxe
StdDevSpa (Dest, Swath, Source)

Remarques
Ecart type spatial.

$$Dest = \left(\left(\sum_{i=j}^{i=j+swath} X(i)^2 - \left(\sum_{i=j}^{i=j+swath} X(i) \right)^2 / swath \right) / swath \right)^{\frac{1}{2}}$$

Où $X_{(j)}$ = Source

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou ligne de données*	La variable dans laquelle on stocke les résultats de l'instruction.
Swath *Constante*	Le nombre de valeurs dans la ligne de données, à inclure dans le calcul.
Source *Ligne de données*	Le nom de la ligne de données qui contient les données d'entrée pour l'instruction.

Tan (Source)

Cette instruction donne comme résultat la tangente d'un angle.

Syntaxe
x = Tan (*source*)

Remarques
La *source* peut être une quelconque expression numérique valide dont la valeur est exprimée en radian.

La fonction Tan prend un angle, et donne comme résultat le rapport des deux côtés d'un angle droit. Le rapport et la longueur du côté opposé à l'angle, divisé par la longueur du côté adjacent à l'angle.

Afin de convertir des degrés en radian, on multiplie par π/180. Pour convertir des radians en degrés, on multiplie par 180/π.

Exemple de fonction Tan

L'exemple utilise la fonction Tan afin de calculer la tangente de la valeur de la tension, Volt(1).

```
Dim Degrés, Pi, Radians, Ans          'Déclaration des variables
Pi = 4 * Atn(1)                       'Calcul de Pi
Degrés = Volt(1)                      'On prend la valeur en entrée
Radians = Degrés * (Pi / 180)         'On convertit la valeur en radians
Ans = Tan(Radians)                    'On calcul la tangente de l'angle
```

TANH (Source)

Cette instruction donne comme résultat la tangente hyperbolique d'une expression ou d'une valeur.

Syntaxe
x = TANH (*source*)

Remarques

La fonction TANH donne comme résultat la tangente hyperbolique [TANH(x) = sinh(x) / cosh(x)] trouvée pour la valeur contenue dans la *source*.

Exemple de fonction TANH

L'exemple utilise TANH afin de calculer la tangente hyperbolique de la tension en entrée.

```
Public Volt1, Ans                     'Déclare les variables.
VoltDiff(Volt1,1,mV5000,1,True,100,500,1,0)
                  'Donne la valeur de la tension de la Diff1 dans la variable Volt(1)
Ans = TANH( Volt1 )                   'Calcule la Tangente hyperbolique de Volt1.
```

TimeIntoInterval (TintoInt, Interval, Units)

L'instruction TimeIntoInterval (ou IfTime) est utilisée pour donner un niveau logique vrai ou faux base sur l'horloge temps réel de la centrale de mesure.

Syntax

Variable = TimeIntoInterval(TintoInt, Interval, Units)

ou

If TimeIntoInterval (TintoInt, Interval, Units)

Remarks

When encountered by the datalogger program, the TimeIntoInterval statement is evaluated True (-1) or False (0) based on the datalogger's real-time clock. Time is kept internally by the datalogger as the elapsed time since January 1, 1990, at 00:00:00 hours. When the Interval divides evenly into this elapsed time, the TimeIntoInterval is set True. The TimeIntoInterval instruction can be used to set the value of a variable to -1 or 0 (first syntax example), or it can be used as an expression for a Condition (second syntax example).

The TimeIntoInterval instruction has the following parts:

TintoInt The TintoInt, or time into interval, argument allows the programmer to define an offset from the Interval at which the TimeIntoInterval statement will be evaluated true. For example, if the Interval is set at 60 minutes, and TintoInt is set to 5, TimeIntoInterval will be True at 5 minutes into the hour, every hour, based on the datalogger's real-time clock. If the TintoInt is set to 0, the TimeIntoInterval statement is True at the top of the hour.

Interval The Interval is how frequently the TimeIntoInterval statement will be evaluated True, based on the datalogger's real-time clock.

Units The Units argument is used to specify the units on which the TintoInt and Interval arguments will be based. The options are microseconds, milliseconds, seconds, minutes, hours, or days.

Notes:

TimeIntoInterval must be placed within a scan to function.

This instruction is also known as IfTime. Either keyword can be used within the program.

VaporPressure (Dest, Temp, RH)

L'instruction VaporPressure calcule la pression de vapeur ambiante (Vp) à partir des valeurs mesurées auparavant de la température et de HR (humidité). L'instruction calcule d'abord la pression de vapeur saturée de la température de Lowe (voir l'instruction SatVp). Ensuite la pression de vapeur est calculée en multipliant la fraction HR :

Vp = SatVp x RH/100

WetDryBulb (Dest, Temp, WetTemp, Pressure)

WetDryBulb calcule la pression de vapeur en kilo pascals à partir des températures en °C des thermomètres mouillé et de l'air ambiant (sec). Cet algorithme est utilisé par le *National Weather Service*:

Vp = Svpwet - A (1 + B∗Tw)(Ta - Tw) P
Vp = pression de vapeur ambiante en kilo pascals
Svpwet = pression de vapeur saturante de la température du thermomètre mouillé en kilo pascals
Tw = température du thermomètre mouillé, °C
Ta = température de l'air ambiante, °C
P = pression de l'air en kilo pascals

A = 0.000660
B = 0.00115

Bien que l'algorithme nécessite la pression de l'air, les fluctuations journalières sont suffisamment faibles pour la plupart des applications, pour qu'une valeur constante de la pression standard de la hauteur du site suffise. Si un capteur de pression est employé, la pression courante peut être utilisée.

Paramètre *& Type de données*	*Entré*
Dest	La variable dans lequel est stockée Vp (kPA).
Temp	La variable contenant la température de l'air (dry-bulb °C).
RH	La variable contenant RH (%).
WetTemp	La variable contenant la température du thermomètre mouillé (°C).
Pressure	La variable contenant la pression atmosphérique (kPa).

XOR

La fonction XOR est utilisée afin d'effectuer une exclusion logique sur deux expression.

Syntaxe

Résultat = expression1 XOR expression2

Remarques

Si seule une des deux expressions est évaluée à Vrai, le résultat est Vrai. Si l'une des expressions est Nulle, le résultat est aussi Nul. Lorsque aucune des expressions n'est Nulle, le résultat est déterminé selon le tableau suivant :

Si *expr1* **est**	**Et** *expr2* **est**	**Le** *résultat* **est**
Vraie	Vraie	Faux
Vraie	Fausse	Vrai
Fausse	Vraie	Vrai
Fausse	Fausse	Faux

L'opérateur XOR effectue une comparaison de bits sur des bits positionnés à l'identique dans 2 expressions numériques, et fixe la valeur du bit correspondant selon la table de vérité suivante :

Si le bit dans l'expression 1 est :	**Et si le bit dans l'expression 2 est :**	**Alors le résultat est :**
0	0	0
0	1	1
1	0	1
1	1	0

Exemple d'opérateur XOR

L'exemple donne une valeur à la variable Msg en fonction de la valeur des variables A, B, et C, en supposant qu'aucune des variables n'est Nulle. Si A = 10, B = 8 et C = 11, l'expression de gauche est Vraie et l'expression de droite est Fausse. Du fait qu'une seule expression est vraie, le résultat de l'évaluation par l'opérateur XOR est Vrai.

```
Dim A, B, C                 'Déclare les variables.
A = 10: B = 8: C = 11       'Assigne des valeurs.
If A > B XOR B > C Then      'Evalue les expressions.
    Msg = True
Else
    Msg = False.
End If
```

Fonctions mathématiques dérivées

La liste suivante est une liste de fonctions mathématiques non intrinsèques, qui peuvent être dérivées à partir des fonctions mathématiques intrinsèques fournies avec le CRBasic :

Fonction	Equivalent en CRBasic
Secant	Sec = 1 / Cos(X)
Cosecant	Cosec = 1 / Sin(X)
Cotangent	Cotan = 1 / Tan(X)
Inverse Secant	Arcsec = Atn(X / Sqr(X * X - 1)) + Sgn(Sgn(X) -1) * 1.5708
Inverse Cosecant	Arccosec = Atn(X/Sqr(X * X - 1)) + (Sgn(X) - 1) * 1.5708
Inverse Cotangent	Arccotan = Atn(X) + 1.5708
Hyperbolic Secant	HSec = 2 / (Exp(X) + Exp(-X))
Hyperbolic Cosecant	HCosec = 2 / (Exp(X) - Exp(-X))
Hyperbolic Cotangent	HCotan = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))
Inverse Hyperbolic Sine	HArcsin = Log(X + Sqr(X * X + 1))
Inverse Hyperbolic Cosine	HArccos = Log(X + Sqr(X * X - 1))
Inverse Hyperbolic Tangent	HArctan = Log((1 + X) / (1 - X)) / 2
Inverse Hyperbolic Secant	HArcsec = Log((Sqr(-X * X + 1) + 1) / X)
Inverse Hyperbolic Cosecant	HArccosec = Log((Sgn(X) * Sqr(X * X + 1) +1) / X)
Inverse Hyperbolic Cotangent	HArccotan = Log((X + 1) / (X - 1)) / 2
Logarithm	LogN = Log(X) / Log(N)

Chapitre 9. Instructions de contrôle de programme

BeginProg … EndProg

L'instruction BeginProg est utilisée afin de marquer le début du programme. EndProg marque la fin du programme.

Syntaxe

BeginPog

…

…

EndProg

Remarques

Toutes les instructions du programme principal, se trouvent entre les instructions BeginProg et EndProg. Les variables du programme, les tables de données (DataTables) et les sous-programmes (Subroutines), doivent être définis avant le programme principal.

Exemple de balises BeginProg

Le code qui suit montre le déroulement d'un programme typique de centrale d'acquisition, et l'utilisation des balises BeginProg/EndProg. Les variables du programme et les tableaux de données « DataTable » sont définis, suivis par le code du programme principal.

```
'Définit les Variables pour WindSpeed et Rain
'Dimensionnement de la ligne de donnée de RealTime
PUBLIC WINDSP
PUBLIC RAIN
DIM TIME(9)
ALIAS TIME(1)=YEAR
ALIAS TIME(2)=MONTH
ALIAS TIME(3)=DAY
ALIAS TIME(4)=HOUR
ALIAS TIME(5)=MINUTES
ALIAS TIME(6)=SECONDS
ALIAS TIME(7)=mSECONDS
ALIAS TIME(8)=DAY_OF_WEEK
ALIAS TIME(9)=DAY_OF_YEAR

'Definit le tableau de donnée METDATA
DataTable (METDATA,1,1000)
    DataInterval (0,1,Min,10)
    Sample (1,WINDSP,FP2)
    Totalize (1,RAIN,FP2,False )
EndTable

'Programme principal – Lire l'heure de la centrale de mesure
'Mesure 2 voies de comptage d'impulsion et appeler le tableau de
'données
BeginProg
    Scan (1,Sec,3,0)
    RealTime (TIME)
    PulseCount (WINDSP,1,1 ,1,1,1.0,0)
    PulseCount (RAIN,1,2,2,0,1.0,0)
    CallTable METDATA
    NextScan
EndProg
```

Call

L'instruction Call sert à transférer le contrôle du programme, du programme principal vers un sous-programme.

Syntaxe

Call Name (liste de variables)

Remarques

L'utilisation du mot « Call » est une option, lorsqu'on souhaite appeler un sous-programme.

La fonction « Call » a trois parties :

Call	C'est un mot optionnel, afin de transférer le contrôle du programme à un sous programme.
Name	C'est le nom du sous-programme à appeler.
Liste de variables ou Constantes	La liste peut contenir des variables, des constantes, ou des expressions qui évaluent une constante (c'est à dire qu'elle ne contiennent pas de variable) et qui devraient être mises dans les variables déclarées dans le sous-programme. Les valeurs ou les variables passées, peuvent être altérées par le sous-programme. Si le sous-programme change la valeur de la variable déclarée dans le sous-programme, il change alors la valeur de la variable qui était précédemment là. Si une constante est mise dans un des sous-programmes appelés « variable », la « variable » devient une constante et sa valeur ne peut plus être changée par le sous-programme.

Exemple d'utilisation de « Call »

Voir l'exemple donné pour la description de l'instruction « Sub » au chapitre 5.

CallTable

Cette instruction est utilisée afin d'appeler une table de données.

Syntaxe

CallTable Name

Remarques

L'instruction « CallTable » est utilisée dans le programme principal, afin d'appeler un tableau de données (DataTable). Les tableaux de données sont listés dans la partie « Déclaration » du programme, avant la balise « BeginProg ». Lorsque le tableau de données est appelé, il effectuera un traitement des données de la façon indiquée dans le programme, et il effectuera un test sur les conditions de sauvegarde.

Exemple d'utilisation de CallTable

Cet exemple utilise l'instruction CallTable afin d'appeler la table ACCEL

CallTable ACCEL

Data … Read … Restore

Utilisé afin de marquer le début d'une liste de données.

Syntaxe

Data *liste de constantes*

Read [VarExpr]

Restore
Remarques

La fonction **Data**: Une *liste* de constantes à virgule flottante, peuvent être lues (en utilisant **Read**) vers une ligne de donnée variable.

Paramètre: Une *liste* de constantes à virgule flottante.

Read Data permet de lire les données et de les mettre en place dans une ligne de données. Plusieurs instructions **Read** consécutives reprennent la liste de donnée là où la précédente instruction s'est arrêtée.

Paramètre : Variable de destination.

Restore Data permet de rediriger un pointeur vers le début de boucle. C'est utilisé en conjonction avec **Data** et **Read**.

Exemple de balise Data

Cet exemple utilise « Data » afin de contenir les valeurs des données et « Read » afin de les transférer vers des variables.

```
Data 1, 2, 3, 4, 5                'données pour x
Data 6, 7, 8, 9, 10               'données pour y
For I = 1 To 5
   Read x( I )
Next I
For I = 1 To 5
   Read y( I )
Next I
```

Cet exemple utilise « Restore » afin de lire des données 1, 2, 3 et 4 dans les variables X( ) aussi bien que Y( ).

```
Data 1, 2, 3, 4
For I = 1 To 4
   Read X( I )
Next I
Restore
For I = 1 To 4
   Read Y( I )
Next I
```

ClockSet (Source)

Cette instruction permet de fixer l'heure de la CR1000 à partir des valeurs contenues dans une ligne de données. L'utilisation la plus probable de cette instruction sera lorsque la CR1000 utilise une source de synchronisation temporelle qui est plus précise que la sienne, à savoir un récepteur GPS par exemple. La valeur du temps en entrée serait périodiquement ou continuellement convertie dans l'unité demandée pour la ligne de donnée variable, et l'instruction ClockSet serait utilisée afin de fixer l'heure de la CR1000.

Source *Ligne de données*	La source doit être une ligne de données de 7 éléments. Les éléments de 1 à 7 doivent comporter respectivement l'année, le mois, le jour, l'heure, la minute, la seconde et la microseconde.

Delay (Option, Delay, Units)

Cette instruction est utilisée afin de donner un délai au programme.

Syntaxe

Delay (Option, Delay, Units)

Remarques

L'instruction de délai est utilisée afin d'effectuer une pose sur la tâche de séquence de mesure ou les instructions de calcul, pendant une durée spécifiée par les valeurs « Delay » et « Units », avant d'effectuer la mesure suivante ou le calcul suivant.

L'intervalle de scrutation (*Scan Interval*) doit être suffisamment long pour que le programme puisse exécuter la totalité des instructions + le délai spécifié, entre deux intervalles de scrutation.

L'intervalle de scrutation devrait être suffisamment long afin de pouvoir effectuer toutes les mesures et les périodes de délai. Si le délai est appliqué à la séquence de tâche de mesure et que l'intervalle de scrutation n'est pas assez long pour effectuer toutes les mesures plus le délai, le programme ne se compilera pas lorsqu'il sera téléchargé sur la centrale de mesure. Si le délai est appliqué à la séquence de tâche de calcul, le programme se compilera mais des scrutations seront perdues.

Paramètre *& Type de donnée*	**Entrée**		
DelayOption *Constante*	**Code**	**Résultat**	
	0	Le délai affectera la séquence de tâche de mesure. Les calculs continueront à s'effectuer comme cela est nécessaire, en tâche de fond. Lorsque cette option est choisie, l'instruction de délai ne doit pas être placée dans une balise conditionnelle.	
	1	Le délai affectera les calculs. Les mesures continueront à s'effectuer à mesure que la séquence de tâche les appelle.	
Delay *Constante*	La valeur numérique pour le délai de temps.		
Units *Constante*	L'unité pour le délai.		
	Code Alphanumérique	**Code Numérique**	**Unité**
	USEC	0	microsecondes
	MSEC	1	millisecondes
	SEC	2	seconds
	MIN	3	minutes

Do … Loop

Cette instruction répète un bloc de commandes tant qu'une condition est vraie (*while*), ou jusqu'à ce qu'une conditions devienne vraie (*until*).

*** Syntaxe1**
Do [{**While** | **Until**} *condition*]
[*bloc d'instructions*]
[**Exit Do**]
[*bloc d'instructions*]
Loop
*** Syntaxe2**
Do
[*bloc d'instructions*]
[**Exit Do**]
[*bloc d'instructions*]
Loop [{**While** | **Until**} *condition*]

L'instruction **Do…Loop** contient ces parties :

Partie	Description
Do	Doit être la première instruction écrite lors d'une structure « Do… Loop »
While	Ce paramètre indique que la boucle est répétée tant que la *condition* est vraie
Until	Ce paramètre indique que la boucle est exécutée jusqu'à ce que la *condition* soit vraie.
condition	C'est une expression numérique dont le résultat est vrai (différent de 0) ou faux (0 ou Nul).
bloc d'instructions	Ce sont les ligne de programme entre les balises « Do » et « Loop ». Elles sont répétées tant que, ou jusqu'à ce que la *condition* soit vraie.
Exit Do	Cela n'est utilisé qu'à l'intérieur des structures de contrôle de type « **Do …Loop** », afin de fournir une façon alternative de sortie du « **Do …Loop** ».On peut placer autant de **Exit Do** qu'on veut et à différents endroits dans une structure « **Do …Loop** ». Souvent utilisée avec des évaluations de conditions (par exemple If …Then), **Exit Do** transfert le contrôle au code qui est écrit juste à la suite dans la **Loop**. Lorsque les « **Do …Loop** » sont imbriqués, le transfert est alors contrôlé par le « **Do …Loop** » qui est au niveau d'imbrication au dessus de celui où est écrit la balise **Exit Do**.
Loop	Cette balise termine le « **Do …Loop** »

Exemple de fonctionnement de « Do …Loop »

L'exemple crée une boucle infinie, dont on ne peut sortir que si la valeur de Volt(1) est comprise dans une certaine étendue de mesure.

```
Dim Reply                                   'Déclare la variable
Do
 Reply = Volt(1)
 If Reply > 1 And Reply < 9 Then            'Vérifie l'étendue de mesure
        Exit Do                             'Sort de la boucle « Do '…Loop »
 End If
Loop
```

La même chose peut être faite d'une autre manière en incorporant l'étendue de mesure de test à l'intérieur d'une boucle « Do …Loop » de la façon suivante :

```
Dim Reply                                   'Déclare la variable
Do
 Reply = Volt(1)
Loop Until Reply > 1 And Reply < 9
```

L'exemple suivant montre l'utilisation de **Wend**.

```
While X>Y                                   'Ancienne façon de faire des boucles
…
Wend

Do While X > Y                              'Façon plus intéressante de programmer
…
Loop
```

FileManage

L'instruction **FileManage** est utilisée afin de gérer des fichiers à partir d'un programme de centrale de mesure en fonctionnement.

Syntaxe

FileManage("Device: FileName", Attribute)

Remarques

FileManage est une fonction qui permet au programme actif de la centrale de mesure, de manipuler les fichiers programmes qui sont stockés sur la centrale de mesure.

L'instruction **FileManage** comprend les paramètres suivants :

<table>
<tr><th>Paramètre
& Type de donnée</th><th>Entrée</th></tr>
<tr><td>Device;
Filename
Texte</td><td>L'argument "Device:Filename" est le fichier qui devrait être manipulé. L'appareil sur lequel le fichier est stocké, doit être spécifié, et la chaîne de caractère en entier doit être contenue dans les guillemets. Appareil = CPU, le fichier est stocké sur la mémoire de la centrale de mesure. Appareil = CRD, le fichier est stocké sur une carte PCMCIA.</td></tr>
<tr><td>Attribute</td><td>Attribute est un code numérique afin de déterminer l'action à effectuer avec le fichier affecté par l'instruction FileManage. Le code d' « Attribute » est en fait un champ comprenant des bits. Les codes sont :
<table>
<tr><th>Bit</th><th>Décimal</th><th>Description</th></tr>
<tr><td>bit 0
bit 1
bit 2
bits 1 & 2
bit 3
bit 4</td><td>1
2
4
6
8
16</td><td>Programme inactif
Activer à la mise sous tension
Activer maintenant
Activer maintenant et à la mise sous tension
Effacer
Effacer tout</td></tr>
</table>
</td></tr>
</table>

Exemple d'utilisation de FileManage

La balise suivante utilise FileManage afin de faire fonctionner TEMPS.CR5, stocké sur la CPU (UC – Unité Centrale) de la centrale de mesure, lorsque le Flag(2) est activé.

```
If Flag(2) then FileManage( "CPU:TEMPS.CR5" 4 )   '4 pour "activer maintenant"
```

FileMark (TableName)

Paramètre *&Type de donnée*	**Entrée**
TableName *nom*	Le nom du tableau de données dans lequel on veut insérer une marque de fichier « filemark ».

FileMark est utilisé afin d'insérer une marque de fichier dans un fichier de données. La marque de fichier peut être utilisée par le logiciel de décodage afin d'indiquer que le nouveau fichier devrait débuter à l'emplacement de la marque. Cette capacité à créer de multiples fichiers n'existe qu'au niveau du convertisseur binaire vers l'ASCII. Pour utiliser cette capacité, les fichiers doivent être stockés sur une carte CF puis rapatriés à partir de l'écran se référant à la centrale de mesure, ou bien en retirant la carte et en transférant les données directement sur l'ordinateur.

FileMark est placé à l'intérieur d'une balise de conditions afin d'écrire le marqueur de fichier au moment souhaité.

For …Next

Cette instruction répète un groupe d'instruction un nombre de fois spécifié.

Syntaxe
For *counter* = *start* **To** end [**Step** *increment*]
[*bloc d'instructions*]
[**Exit For**]
[*bloc d'instructions*]
Next [*counter* [, *counter*] [, …]]

L'instruction **For…Next** contient ces parties :

Partie	**Description**
For	C'est le début d'une boucle de contrôle de type **For…Loop**. Il doit apparaître avant n'importe quelle autre partie de cette structure.
counter	Variable numérique utilisée en tant que compteur de boucle. La variable ne peut pas être une partie d'une ligne de donnée ou d'une table enregistrée.
start	Valeur initiale du compteur (*counter*)
To	Sépare les valeurs *start* et *end*
end	Valeur finale du compteur (*counter*)
Step	Indique que l'incrément (*increment*) est explicitement mentionné.
increment	Valeur par laquelle le compteur est incrémenté à chaque passage de boucle. Si vous n'avez pas besoin de définir de **Step**, l'incrément par défaut à la valeur de 1.
bloc d'instructions	Lignes de programme qui sont exécutées un nombre spécifique de fois, entre For et Next.

Exit For	Cette expression n'est utilisée qu'à l'intérieur d'une boucle de structure « **For …Next** ». On peut utiliser autant de **Exit For** que l'on souhaite à l'intérieur de la boucle « **For…Next** ». Souvent utilisée avec des évaluations de conditions (par exemple If …Then), Exit For transfert le contrôle de programme au code qui est écrit juste derrière le **Next**.
Next	Cette expression termine la boucle « **For …Next** ». C'est suite à elle, que s'ajoute la valeur de l'incrément, à celle du compteur.

La valeur de « **Step** », contrôle l'exécution de la boucle de la façon suivante :

Lorsque *Step* est	Alors la boucle (*Loop*) s'exécute si
Positif ou égal à 0	Le compteur (*counter*) est <= *end*
Négatif	Le compteur (*counter*) est >= *end*

Une fois que l'on est entré dans la boucle, et que les instructions de la boucle sont exécutées, la valeur de **Step** (un « pas ») est ajoutée au compteur (*counter*). A ce moment, soit les instructions à l'intérieur s'exécutent encore (sur la base du même test que celui qui a permis de rentrer dans la boucle), soit on sort de la boucle et l'exécution continue avec les instructions qui suivent l'instruction **Next**.

Conseil : Si vous faites en sorte de changer la valeur du compteur alors que vous êtes à l'intérieur de la boucle, vous rendrez votre programme plus compliqué à lire et à corriger (débuguer).

On peut imbriquer des instructions « **For …Next** » en intégrant une autre boucle « **For …Next** » à l'intérieur d'une boucle existante. Il faut alors donner un nom unique à chaque variable compteur. Le type de construction suivant, est correct :

```
For I = 1 To 10
    For J = 1 To 10
        For K = 1 To 10
        ...
        Next K
    Next J
Next I
```

NOTE

Si vous oubliez la variable **Next** dans votre séquence de programme, la valeur d'incrémentation de **Step** est ajoutée à la variable associée à la boucle **For** la plus récente. Si une instruction **Next** est lue avant son instruction **For** correspondante, une erreur se produit.

Exemple d'instruction « For...Next »

L'exemple utilise une boucle "For …Next" à l'intérieur d'une autre boucle.

```
Dim I, J                          'Déclaration des variables.
For J = 5 To 1 Step -1            'On recule de un à chaque fois, sur 5 fois
      For I = 1 To 12             'On effectue 12 fois la boucle
    . . . .                       'On exécute quelques instructions
      Next I
    . . . .                       'On exécute quelques instructions
Next J
. . . .                           'On exécute quelques instructions
```

Cet exemple rempli les éléments impairs de X jusqu'à 40 * Y avec des nombres impairs.

```
For I = 1 To 40 * Y Step 2
X( I ) = I
Next I
```

If … Then … Else

Cela permet une exécution conditionnelle, basée sur l'évaluation d'une expression.

```
* Syntaxe 1
If condition Then thenpart [Else elsepart]
* Syntaxe 2
If condition1 Then
        [bloc d'instructions-1]
[ElseIf condition2 Then
        [bloc d'instructions-2] ]
[Else
        [bloc d'instructions-n] ]
End If
```

*** Description de la Syntaxe 1**
La forme à une seule ligne est souvent utilisée pour les tests conditionnels courts et simples. La Syntaxe 1 comporte trois parties :

Partie	Description
If	Ceci débute la structure de contrôle simple **If … Then**
condition	C'est une expression évaluée vrai (différent de 0) ou faux (0 ou nul).
Then	Identifie l'action à effectuer si la *condition* est satisfaite
thenpart	Instructions ou commandes à effectuer quand la *condition* est vraie
Else	Identifie l'action à effectuer si la condition n'est pas satisfaite. Si la partie « **Else** » n'est pas présente, le contrôle passe aux instructions suivantes du programme.
elsepart	Instructions ou commandes exécutées lorsque la *condition* est fausse.

Les champs *thenpart* et *elsepart* ont la même syntaxe :

{*instruction* | [Go To] numéro_de_le_ligne | Go To intitulé_de_le_ligne}

La syntaxe des champs *thenpart* et *elsepart* ont cette partie :

Partie	Description
Instruction	Une ou plusieurs instructions CRBasic, séparées par des guillemets « : ».
Note	Vous pouvez avoir plusieurs instructions avec une condition, mais elles doivent être sur la même ligne et doivent être séparées par des « : », comme dans l'exemple suivant : **If** A > 10 **Then** A = A + 1 : B = B + A : C = C + B

*** Description de la syntaxe 2**

Le block de forme **If … Then … Else** permet d'avoir plus de structure et de flexibilité qu'une forme à une seule ligne. Elle est souvent plus facile à lire, à comprendre, à débuguer et donc à « entretenir » (*maintain*). La syntaxe 2 contient ces parties :

Partie	Description
If	Ceci débute la structure de contrôle **If … Then**
condition1	C'est une expression du même type que la *condition* utilisée ci-avant
Then	Identifie l'action à effectuer si la *condition* est satisfaite
bloc d'instructions-1	Une ou plusieurs instructions ou commandes en CRBasic, à effectuer quand la *condition1* est vraie
ElseIf	Identifie l'action à effectuer si la condition1 n'est pas satisfaite.
condition2	C'est une expression du même type que la *condition* utilisée ci-avant
bloc d'instructions-2	Une ou plusieurs instructions ou commandes en CRBasic, à effectuer quand la *condition2* est vraie
Else	Identifie l'action à effectuer si aucune des conditions précédentes ne sont satisfaites.
bloc d'instructions-n	Une ou plusieurs instructions ou commandes en CRBasic, à effectuer si la *condition1* et la *condition2* sont fausses
End If	C'est la fin de structure du **If …Then**

Lorsqu'on exécute un bloc de If, le CRBasic teste la *condition1*, la première expression numérique. Si l'expression est vraie, les commandes qui suivent le Then, sont alors exécutées.

Si la première expression est fausse, le CRBasic commence à évaluer chacune à son tour, les expressions **ElseIf**. Quand le CRBasic trouve une condition qui est vraie, les commandes qui suivent le **Then**, sont alors exécutées. Si aucune des conditions **ElseIf** n'est vraie, les commandes qui suivent le **Else**, sont alors exécutées. Après avoir exécuté les commandes qui suivent le **Then**, ou le **Else**, le programme se poursuit en exécutant les instructions qui sont situées après le **End If**.

Les conditions **Else** et le **ElseIf** sont toutes les deux optionnelles. On peut avoir autant de conditions **ElseIf** que l'on souhaite à l'intérieur d'un bloc **If**, mais aucune ne doit être écrite après la condition **Else**. Chacun des blocs de conditions peut contenir des blocs de conditions **If** imbriquées.

Le CRBasic regarde ce qui est écrit après le mot clé **Then**, afin de déterminer si la commande **If** fait partie d'un bloc de commandes de type **If**. Si une quelconque écriture différente d'un commentaire, est écrite après le **Then**, la commande est traitée comme si elle était une commande de type **If** pour la ligne seulement (pas pour un block **If**).

Un bloc de commandes **If**, doit être le premier sur la ligne de commande. Les parties **Else**, **ElseIf** et **End If**, ne peuvent rien avoir d'autre que des espaces devant eux en début de ligne. Le bloc de commandes **If** doit se terminer par une commande **End If.**

Par exemple:

```
If a > 1 And a <= 100 Then
…
ElseIf a = 200 Then
...
End If
```

A noter : Il peut être plus simple d'utiliser le comparateur « Case » lorsque l'on évalue une seule expression, qui peut engendrer plusieurs actions.

Exemple de condition If … Then … Else :

Cet exemple donne plusieurs illustrations de la syntaxe If … Then … Else.

```
Dim X, Y, Temp(5)              'Déclaration des variables
If X < 10 Then
        Y = 1                  '1 digit
ElseIf X < 100 Then
        Y = 2                  '2 digits
Else
        Y = 3                  '3 digits
End If
…                              'Exécution de code
```

RunProgram

L'instruction RunProgram est utilisée afin de lancer l'activation d'un programme de centrale de mesure à partir du programme actif

Syntaxe

RunProgram ("Device:FileName", Attrib)

Remarques

RunProgram est compose des paramètres suivants :

"Device:FileName" L'argument "Device:Filename" est le nom du fichier qui doit être exécuté. L'appareil sur lequel le fichier est stocké doit être spécifié, et la chaîne de caractère doit être écrite entre des guillemets. Device = CPU (UC-Unité Centrale), le fichier est stocké dans la mémoire de la centrale de mesure. Device = CRD, le fichier est stocké sur la carte PCMCIA.

Attribute L' « Attribute » est un code numérique afin de déterminer ce qui doit se produire lorsque l'instruction RunProgram est exécutée. Les codes d' « Attribute » sont décrits ci-dessous :

Bit	Décimal	Description
bit 1	2	Activer à la mise sous tension
bit 2	4	Activer maintenant

Exemple d'instruction RunProgram
La commande ci-dessous utilise RunProgram afin de lancer le programme TC-TEMP.CR1, qui est présent dans la CPU de la centrale de mesure, lorsque le Flag(2) passe à l'état haut.

```
If Flag(2) then RunProgram ( "CPU: TC-TEMP.CR1" 4 )
```

ResetTable

Utilisé afin de ré-initialiser un tableau de donnée, sous contrôle du programme.

Syntaxe

ResetTable(TableName)

Remarques
ResetTable est une fonction qui permet à un programme qui s'exécute, d'effacer et de ré-initialiser un tableau de données. « TableName » est le nom du tableau de données à ré-initialiser.
Exemple d'utilisation de ResetTable

La ligne d'exemple de programme utilise ResetTable afin de ré-initialiser le tableau de données appelé « MAIN » lorsque le Flag(2) est à l'état haut.

```
If Flag(2) then ResetTable( MAIN )
```

Scan … NextScan

Cette instruction est utilisée afin de donner le temps de scrutation du programme.

Syntaxe
Scan(Interval, Units, Option, Count**)**
...
...**[Exit Scan]**
...

Next Scan

Les mesures, les calculs et les appels afin d'enregistrer les tableaux de sauvegarde contenues entre les instructions Scan … NextScan, déterminent la séquence et la temporisation du programme de la centrale de mesure.

L'instruction Scan détermine la fréquence (l'échantillonnage) à laquelle les mesures présentes à l'intérieur de la structure Scan…NextScan, seront effectuées ; elle contrôle aussi les capacités de mémoire tampon, et fixe le nombre de fois qu'il faudra effectuer le boucle au travers de la scrutation.

<table>
<tr><th>**Paramètre**
& Type de données</th><th>**Entrée**</th></tr>
<tr><td>**Interval**
Constante</td><td>Entrer l'intervalle de temps auquel la scrutation doit être exécutée. L'intervalle peut être en ms, s, ou minutes, selon ce qui est choisi au paramètre "**Units**". L'intervalle minimum est de 10 millisecondes. L'intervalle maximum est de 30 minutes.</td></tr>
<tr><td>**Units**
Constante</td><td>L'unité pour le paramètre de temps.
<table>
<tr><th>Code Alphanumérique</th><th>Unités</th></tr>
<tr><td>MSEC</td><td>millisecondes</td></tr>
<tr><td>SEC</td><td>secondes</td></tr>
<tr><td>MIN</td><td>minutes</td></tr>
</table></td></tr>
<tr><td>**Option**
Constante</td><td>Le paramètre d'option détermine la façon dont les données seront mise en mémoire tampon durant la séquence Scan…NextScan. Les options sont :
<table>
<tr><th>Option</th><th>Résultat</th></tr>
<tr><td>0, 1, ou 2</td><td>La centrale de mesure utilise deux mémoires tampon lorsqu'elle effectue des calculs sur les mesures effectuées. Lorsqu'une mesure débute sur une scrutation, les valeurs de la scrutation précédente sont mises dans une mémoire tampon. Cela permet aux calculs de terminer à s'effectuer alors que de nouvelles mesures sont prises sur la scrutation en cours.</td></tr>
<tr><td>>3</td><td>La centrale de mesure utilise 3 mémoires tampon ou plus lorsqu'elle effectue des calculs, en fonction du nombre de scrutations définies par cette constante.</td></tr>
</table>
Des mémoires tampon plus importantes peuvent être utilisées pour une scrutation qui aurait, de façon occasionnelle, besoin de beaucoup de capacité de calcul, comme c'est le cas avec les FFT, Histogrammes, ou lorsque les calculs pourraient être interrompus par les communications. Si une valeur de 1000 est insérée dans l'argument « BufferSize » d'une scrutation ayant 10 mesures de thermocouples, alors 40 000 octets de SRAM seront alloués pour la mémoire tampon. [(4 octetc) / (mesure) x (10 mesures)/(scrutation mise en mémoire tampon) x 1000 scrutations mises en mémoire)]. La taille de la mémoire tampon plus la taille de n'importe quelle tableau de données, ne doit pas être supérieure à 2 Mo.

Si les calculs sont en retard et n'ont pas assez de mémoire tampon qui leur est allouée, la centrale de mesure ne tiendra pas compte des valeurs mises en mémoire et se synchronisera de nouveau par rapport aux mesures actuelles.

L'instruction SlowSequence ne permet pas d'utiliser ce schéma de mise en mémoire tampon, même si on utilise une balise Scan afin de spécifier le début d'une scrutation en séquence lente. En mode SlowSequence, les mesures sont stockées dans une seule mémoire tampon. Les calculs de cette mémoire tampon sont effectués avant que les mesures de NextScan soient effectuées.</td></tr>
<tr><td>**Count**
Entier</td><td>Le nombre de fois qu'il faudra exécuter la boucle Scan/NextScan. On entre la valeur 0 si la répétition doit être infinie.</td></tr>
</table>

SelectCase … EndSelect

Cette instruction exécute les commandes présentes à l'intérieur du bloc de commandes, en fonction de la valeur d'une expression.

Syntaxe

SelectCase expression_de_test
[**Case** liste_d_expression_1
    [bloc_d_instruction_1]]
[**Case** liste_d_expression_2
    [bloc_d_instruction_2]]
[**CaseElse** liste_d_expression_n
    [bloc_d_instruction_n]]
EndSelect

La syntaxe de l'expression Select Case contient ces parties :

Partie	Description
Select Case	Ceci débute la structure de contrôle **SelectCase**. Elle doit être écrite avant n'importe laquelle des autres parties de la structure de **Select Case.**
expression_de_test	C'est une expression numérique ou une chaîne de caractères. Si l'*expression_de_test* est en accord avec le *bloc_d_instruction* qui lui est associé à l'intérieur de la clause **Case**, alors le *bloc_d_instruction* qui suit la clause **Case**, est exécuté jusqu'à ce qu'une autre clause **Case** soit lue (ou jusqu'à ce que **End Select** soit lu, pour la dernière clause). Si l'*expression_de_test* est en accord avec plus d'une clause **Case**, seules les instructions suivant la première clause, sont exécutées.
Case	Identifie un groupe de commande CRBasic à effectuer si l'expression dans *liste_d_expression* est en accord avec l'*expression_de_test.*
liste_d_expression	Une *liste_d_expression* est constituée de une ou plusieurs formes d'expression parmi les suivantes, délimitées par des virgules : expression expression **To** expression expression comparée via un opérateur comparateur bloc d'instructions Les *bloc_d_instruction_1* à *bloc_d_instruction_n* sont des instructions de CRBasic qui seront écrites sur une ou plusieurs lignes.
Case Else	Identifie le *bloc_d_instruction* à effectuer si aucune concordance n'est satisfaite, avec les *expression_de_test* ou les *liste_d_expression* précédentes. Lorsqu'il n'y a pas d'instruction **Case Else** et qu'aucune des expressions listée dans les clause **Case**, ne permet de concorder avec l'*expression_de_test*, l'exécution du programme se poursuit jusqu'aux instructions qui suivent le **End Select**.
End Select	C'est la fin de structure du **Select Case**. Cette instruction doit apparaître à la suite de toutes les autres instructions nécessaires au bloc de contrôle **Select Case**.

La liste des arguments de *liste_d_expression* contient ces parties :

Partie	Description
expression	N'importe quelle expression numérique
To	Mot clé utilisé afin de spécifier une étendue de mesure de valeurs. Si on utilise le mot clé « To » pour indiquer une étendue de mesure de valeurs, la valeur la plus petite doit être placée avant le « To ».

Bien que cela ne soit pas nécessaire, il est préférable d'avoir une condition **Case Else** à l'intérieur du bloc de commandes **Select Case**, afin de tenir compte des valeurs inattendues de l'*expression_de_test.*

On peut utiliser des expressions multiples ou des étendues de mesure dans chaque clause Case. Par exemple, la ligne suivante est valide :

Case 1 **To** 4, 7 **To** 9, 11, 13

Les conditions **Select Case** peuvent être imbriquées. Chaque bloc de commandes **Select Case** doit avoir un **End Select** pour le conclure.

Exemple de condition Select Case

L'exemple utilise la clause Select Case afin de décider quelle action effectuer, en fonction des entrées données par l'utilisateur.

```
Dim X, Y                        'Déclaration des variables
If Not X = Y Then               ' X est-il égal à Y ?
    If X > Y Then
        Select Case X           'Quelle est la valeur de X
        Case 0 To 9             'X est inférieur à 10
        …                       'On exécute les instructions…
        Case 10 To 99           'X est inférieur à 100
        …                       'On exécute les instructions…
        Case Else               'X est différent des conditions énoncées
        …                       'On exécute les instructions…
        End Select
    End If
Else
    Select Case Y               'Quelle est la valeur de Y
    Case 1, 3, 5, 7, 9          'Y est impair
    …                           'On exécute les instructions…
    Case 0, 2, 4, 6, 8          'Y est pair
    …                           'On exécute les instructions…
    End Select
End If
…                               'On exécute les instructions…
```

SetSecurity (security[1], security[2], security[3])

Les valeurs de Security[I] sont des constantes. SetSecurity ne s'exécute qu'au moment de la compilation.

Si security[I] a la valeur 0, alors security[>I] ont aussi la valeur 0.

security[I] est compris dans l'étendue entre 0 et 65535.

Le niveau le plus fort de sécurité bloque toutes les communications. Le suivant bloque les communications qui permettent de changer la valeur de variables, de mettre à l'heure l'horloge, et de télécharger ou rapatrier un programme. Le dernier niveau ne bloque que le téléchargement ou le rapatriement de fichiers.

On coordonnera cela avec une configuration qui est entrée au clavier.

SetStatus ("FieldName", Value)

L'instruction SetStatus est utilisée afin de changer la valeur d'une des valeurs présentes à l'intérieur du tableau d'état (*Status table*) de la centrale de mesure.

Syntaxe
SetStatus ("FieldName", Value)

Remarques
Le paramètre FielName est le nom de la variable à modifier ; le nom doit être écrit entre des guillemets. Le paramètre de valeur est la valeur que l'on doit donner à cette variable. Si la valeur à donner est une chaîne de caractères (comme *Messages* ou *StationName*), elle doit être entre guillemets. Les variables suivantes peuvent être modifiées :

"FieldName"	Description
Low12VCount	Un compteur d'erreur qui indique le nombre de fois où l'alimentation 12 V a chuté en dessous du seuil permis.
Low5VCount	Un compteur d'erreur qui indique le nombre de fois où l'alimentation 5 V a chuté en dessous du seuil permis.
MaxProcTime	La durée maximale de temps qu'il a fallu pour exécuter le programme.
Messages	Un champ qui peut être utilisé afin de mettre une chaîne de caractères dans la table d'état de la centrale de mesure. Elle doit être écrite entre des guillemets.
SkippedScan	Un compteur d'erreur qui indique le nombre de fois où une scrutation (*Scan*) n'a pas pu s'effectuer du fait que la centrale de mesure soit occupée à effectuer une autre tâche (telle la scrutation précédente).
SkippedSlowScan	Un compteur d'erreur qui indique le nombre de fois où un *SlowScan* n'a pas pu s'exécuter.
SkippedRecord	Un compteur d'erreur qui indique le nombre de fois où un enregistrement devait être effectué, mais ne l'a pas été.
StationName	Le nom de station de la centrale de mesure.
VarOutOfBound	Un indicateur disant qu'une variable n'a pas la dimension suffisante lui permettant de contenir les valeurs qui lui sont envoyées.
WatchdogErrors	Un compteur d'erreur qui indique le nombre de fois où la centrale de mesure a eu besoin de ré-initialiser son processeur. On entre « 0 » pour ré-initialiser le compteur.

Pour tous ces champs, *Messages* et *StationName* mis à part, le fait de donner la valeur 0 à la valeur, va ré-initialiser l'indicateur. Cela peut être utile pour aider à résoudre des problèmes.

Slow Sequence

L'instruction SlowSequence est utilisée afin de marquer le début d'un paragraphe de code qui sera exécuté de façon concurrente à l'exécution du programme principal, bien que typiquement il s'exécutera moins souvent.

Syntaxe

SlowSequence

Remarques

La balise SlowSequence marque la fin du programme principal et le début d'un programme séparé, de priorité moindre. Les instructions pour le programme de séquence lente, sont exécutées lorsque le programme principal ne s'exécute pas et que le temps le permet.

Il est possible d'avoir jusqu'à quatre SlowSequences différentes pour des mesures qu'il n'est pas nécessaire d'effectuer à la même vitesse que la scrutation principale. La centrale de mesure bascule les instructions de mesure de la scrutation SlowSequences vers la scrutation normale.

La CR1000 met à jour son tableau d'étalonnage de façon automatique en mode SlowSequence.

Les mesure d'une seule "scrutation" en mode SlowSequence, peuvent être séparées sur une plus longe période de temps parce que les mesures peuvent être effectuées à l'intérieur de plusieurs scrutations principales successives.

Lorsque plus d'une section SlowSequence est utilisée dans le programme, certaines combinaisons du taux de scrutation du programme principal et du taux d'exécution de la scrutation SlowSequence, peuvent induire le fait que la partie SlowSequence ne soit jamais exécutée. Si l'intervalle d'une priorité SlowSequence plus importante, arrive avant qu'une SlowSequence de moindre priorité n'arrive à se glisser, la SlowSequence de priorité moins important ne sera jamais exécutée. Chaque combinaison de programme principal et segments de programme SlowSequence doivent être évalués afin de déterminer si il y a assez de temps pour permettre aux scrutations de priorité moindre, pour s'exécuter.

Subroutines et DataTables qui sont appelées par une instruction SlowSequence sont déclarées après l'instruction SlowSequence. Les données écrites pour les instructions SlowSequence DataTables seront estampillées avec le temps du début de la scrutation de la dernière SlowSequence.

Exemple SlowSequence

```
'Centrale de mesure CR1000
'Exemple Slow Sequence

Public Temp107, PanelT, BattVolts

DataTable (T107,True,-1)
 DataInterval (0,1,Min,10)
 Average (1,Temp107,FP2,False)
EndTable

BeginProg
 Scan (1,Sec,10,0)
      Therm107 (Temp107,1,1,Vx1,0,250,1.0,0)
      CallTable T107
 NextScan

''Premier scan Slow Sequence  toute les minutes et stocke les moyennes horaires
 SlowSequence
 DataTable (TPanel,True,-1)
      DataInterval (0,1,Hr,10)
      Average (1,PanelT,FP2,False)
 EndTable

 Scan (1,Min,3,0)
      PanelTemp (PanelT,250)
      CallTable TPanel
 NextScan

'Second scan Slow Sequence toute les  30 minutes et stocke la moyenne journalière et le min.
 SlowSequence
 DataTable (BattV,True,-1)
      DataInterval (0,1,Day,10)
      Average (1,BattVolts,FP2,False)
      Minimum (1,BattVolts,FP2,False,False)
 EndTable

 Scan (30,Min,3,0)
      Battery (BattVolts)
      CallTable BattV
 NextScan
EndProg
```

SubScan (SubInterval, Units, Count) … NextSubScan

L'instruction SubScan est utilisée pour contrôler un multiplexeur AM16/32A ou pour mesurer des voies analogiques à une vitesse supérieure de la scrutation du programme.

Syntaxe

SubScan (SubInterval, Units, Count)

Mesures et traitement

NextSubScan

Remarques
Les instructions SubScan/NextSubScan sont placées à l'intérieur des instructions du programme Scan/NextScan.

NOTE SubScans ne peut pas se nicher ou se placer dans une instruction SlowSequence. Pulse Count ou les mesures SDM ne peuvent pas être utilisées sans une instruction SubScan.

Paramètre *& Type de donnée*	**Entrée**
SubInterval *Constante*	L'intervalle de temps entre subscans. Entrer 0 pour n'avoir aucun délai entre subscans.
Units	L'unité de temps pour le SubInterval. **Code Alpha** / **Unités** usec – microsecondes msec – millisecondes sec – Secondes
Count *Constante*	Le nombre de temps pour boucler à travers le subscan chaque fois que le scan s'exécute. Le nombre maximum est 65 535.

Timer (TimNo, Units, TimOpt)

Utilisée pour donner la valeur d'une horloge (timer).

Remarques

Timer est une fonction qui donne la valeur de l'horloge. **TimOpt** est utilisé pour démarrer, arrêter, réinitialiser et démarrer, arrêter et réinitialiser, ou lire sans l'altération de l'état (Exécution ou arrêté). De multiple timers, chacun identifié par un numéro différent (TimNo), peuvent être actifs en même temps.

Syntaxe

variable = Timer(TimNo, Units, TimOpt)

Paramètre *& Type de données*	**Entré**
TimNo *Constante, Variable, ou Expression*	Un nombre entier pour le timer (ex., 0, 1, 2, . . .) Utiliser les petits nombres pour conserver la mémoire: en utilisant TimNo 100 allouera un espace pour 100 timers même si c'est le seul timer dans le programme.
Units *Constante*	Les unités dans laquelle la valeur du timer est donnée. **Code Alpha** / **Code Numerique** / **Unités** USEC – 0 – microsecondes MSEC – 1 – millisecondes SEC – 2 – secondes MIN – 3 – minutes
TimOpt *Constante*	L'action sur l'horloge (timer). La fonction timer retourne la valeur du timer après que l'action se soit accomplie. **Code** / **Résultat** 0 – démarrer 1 – arrêter 2 – réinitialiser et démarrer 3 – arrêter et réinitialiser 4 – Lire seulement

While … Wend

Les instructions **While … Wend** sont utilisées afin d'exécuter une série d'instructions à l'intérieur d'une boucle, jusqu'à tant que la condition soit vraie.

Syntaxe
While Condition
[*bloc_d_instruction*]
Wend

Remarques
Les boucles de type **While … Wend**, peuvent être imbriquées.

Les instructions **While … Wend** ont les paramètres suivants :

While	Cette balise débute la structure de la boucle de contrôle **While … Wend.**
Condition	La condition est une expression quelconque qui peut être évaluée en tant que vraie (différent de zéro) ou faux (0 et Nul). Si la Condition est vraie, toutes les commandes du *bloc_d_instruction* sont exécutées jusqu'à ce que le **Wend** soit lu. Le contrôle retourne alors jusqu'à la balise **While**, et la condition est de nouveau testée. Si la Condition est encore vraie, le processus est répété. Si la condition n'est pas vraie, l'exécution de la boucle s'arrête, et les instructions suivant la balise **Wend** sont alors exécutées.
bloc_d_instruction	C'est la portion du programme qui devra être répétée jusqu'à ce que la boucle se termine. Ces instructions sont écrites entre les balises **While** et **Wend**.
Wend	Cette balise termine la structure de la boucle de contrôle **While … Wend.**

NOTE Le bloc de contrôle **Do … Loop** permet d'effectuer des boucles d'une façon plus flexible encore, que la boucle **While … Wend**.

Exemple de programme avec While … Wend
Ce programme crée une boucle While … Wend, dans laquelle on ne sort que si « Reply » (Reponse) est contenue dans une certaine étendue de mesure.

```
Dim Reponse                    'Déclaration de la Variable
While Reponse < 90
        Reply = Reply + 1
Wend
```

Chapitre 10. Menus d'affichage clavier personnalisés

Le langage CRBasic dispose de la capacité de créer un menu d'affichage clavier personnalisé pour un programme de centrale de mesure. Le menu personnalisé peut alors apparaître en tant que sous-menu du menu standard de la CR1000, ou bien en tant que menu principal, avec le menu standard contenu en sous-menu. Un élément du menu personnalisé peut faire l'une des quatre choses suivantes : 1) Afficher la valeur d'une variable ou d'un champ présent dans un tableau de données. 2) Afficher la valeur d'une variable / d'un drapeau, et permettre à l'utilisateur de changer cette valeur. 3) Effectuer un lien vers un autre menu personnalisé. 4) Effectuer un lien vers le menu standard.

Syntaxe

```
DisplayMenu (MenuName, 0)
    DisplayValue ("MenuItemName", tablename.fieldname )
    MenuItem ("MenuItemName", Variable )
        MenuPick (Item1, Item2, Item3...Item7 )
    SubMenu (MenuName )
        MenuItem ("MenuItemName", Variable )
    EndSubMenu
EndMenu
```

Les instructions « DisplayMenu » et « EndMenu » marquent le début et la fin de la définition personnalisée. Les variables et les données enregistrées peuvent être affichées en tant qu'éléments dans un menu grâce à l'instruction « DisplayValue ». L'instruction « MenuItem » crée un élément qui affiche la valeur d'une variable et permet à la variable d'être éditée. « MenuItem » peut être configurée afin d'être éditée via l'entrée d'une valeur numérique au clavier, ou en sélectionnant une option dans un menu à choix multiples. « MenuPick » est utilisée afin de créer un menu à choix multiples à utiliser avec « MenuItem ». Un lien vers un autre menu utilisateur peut être créé avec les fonctions « SubMenu » et « EndSubMenu ».

Exemple :

```
' Exemple de menu personnalisé pour CR1000

'Déclare les variables pour la température du bornier, deux thermocouples, un 'compteur
vers le [bas] et un drapeau pour déterminer si le compteur est actif ou non :

Public Tpnl, Ttc(2)
Public Counter, CountFlag

'Déclare les constantes pour le menu d'affichage :
Const Yes = True
Const No = False

'Définition du tableau de données Temp :
DataTable (Temp,1,1000)
    DataInterval (0,60,Sec,10)
    Average (1,Tpnl,IEEE4,0)
    Average (2,Ttc(),IEEE4,0)
EndTable

'Définition du menu personnalisé :
DisplayMenu ("Example Custom Menu",1)
    SubMenu("Current Temperatures")
        DisplayValue("Panel Temp",Tpnl)
        DisplayValue("TC 1",Ttc(1))
        DisplayValue("TC 2",Ttc(2))
    EndSubMenu
    SubMenu("Last 1 Min.  Averages")
        DisplayValue("Panel Temp",Temp.Tpnl_Avg(1,1))
        DisplayValue("TC 1",Temp.Ttc_Avg(1,1))
        DisplayValue("TC 2",Temp.Ttc_Avg(2,1))
    EndSubMenu
    SubMenu ("Play with Down Count")
        MenuItem ("Enable",CountFlag)
            MenuPick (Yes,No)     'Crée un choix multiple avec des constantes
        MenuItem("Down Count",Counter)
            MenuPick(15,30,45,60)                'Crée un choix multiple pour le compteur.
        ' Alors que compteur peut être ré-initialisé à partir des éléments du menu
    ' précédent, le fait d'utiliser un sous-menu permet d'avoir un peu plus de texte :
        SubMenu("Reload Down Counter")
            MenuItem("Pick Count",Counter)
                MenuPick(15,30,45,60)            'Crée un choix multiples pour le compteur
            MenuItem("Enter No.",Counter)        'pas de choix multiple = entrer #
        EndSubMenu
    EndSubMenu
EndMenu

'Programme principal
BeginProg
    Scan (1,Sec,3,0)
        PanelTemp (Tpnl,250)
        TCDiff (Ttc(),2,mV20C ,1,TypeT,Tpnl,True ,0,250,1.0,0)
        If CountFlag Then
            Counter=Counter-1
            If Counter <=0 Then Counter=0
        EndIf
        CallTable Temp
    NextScan
EndProg
```

DisplayMenu/EndMenu

Syntaxe :

DisplayMenu ("MenuName", AddtoSystem)

Définition du menu (DisplayValue, MenuItem, and SubMenu)

EndMenu

Les instructions **DisplayMenu/EndMenu** sont utilisées afin de marquer le début et la fin d'un menu personnalisé. Les instructions **DisplayValue, MenuItem**, et **SubMenu/EndSubMenu** sont utilisées afin de définir ce qui sera affiché sur le menu personnalisé.

Paramètre *& Type de donnée*	**Entrée**	
MenuName *Texte*	Le texte qui sera affiché en en-tête pour le menu personnalisé. La chaîne est limitée à 20 caractères, et doit être mise entre guillemets.	
AddtoSystem *Constante*	Cette constante détermine si le menu personnalisé est un sous-menu ou remplace le menu standard.	
	Valeur	**Résultat**
	0	Le menu standard est le sous-menu du menu personnalisé
	≠0	Le menu personnalisé est le sous-menu du menu standard

DisplayValue ("MenuItemName", Source)

L'instruction « DisplayValue » est utilisée afin de définir le texte du menu et les variables associées ou bien les champs de tableau de données qui doivent être affichés dans le menu personnalisé.

Le paramètre « MenuItemName » est le texte qui apparaîtra sur la gauche de la ligne, dans le menu personnalisé. Il peut y avoir jusqu'à 10 caractères d'affichées en plus de la valeur de la source. Le nom doit être contenu dans des guillemets. La source doit être une variable ou un champs provenant d'un tableau de données. Les valeurs affichées en utilisant « DisplayValue » ne peuvent pas être éditées.

Note : « DisplayValue » ne permet pas à l'utilisateur du clavier de changer la valeur. Il faut utiliser « MenuItem » afin d'afficher une variable et de permettre à l'opérateur de changer la valeur.

Paramètre *& Type de donnée*	**Entrée**
MenuItemName *Texte*	Le texte qui sera affiché en tant qu'en-tête pour le menu personnalisé. La chaîne est limitée à 20 caractères, et doit être comprise entre des guillemets.
Source *Variable ou TableName.Field*	La source de la valeur à afficher à la droite du texte « MenuItemName ». La source doit être une variable ou un champ d'un tableau de données. Les variables affichées via « DisplayValue » ne peuvent pas être éditées.

MenuItem ("MenuItemName", Source)

L'instruction « **MenuItem »** est utilisée afin d'afficher la valeur d'une variable, et permettre à l'utilisateur de la changer. Du texte peut être affiché à la place d'une valeur numérique, si l'instruction « MenuPick » est utilisée afin de créer une liste de choix de constantes. Les constantes doivent être définies dans le programme.

Le paramètre « MenuItemName » est le texte qui apparaît sur la gauche de la ligne dans le menu personnalisé. Le nom est limité à 20 caractères, mais seuls 10 caractères seront affichés lorsque la valeur de la variable est affichée (la totalité des 20 caractères seront affichés lorsque la variable est éditée). « MenuItemName » doit être continue dans des guillemets.

Le paramètre « Variable » est le nom de la valeur à être affichée. Les valeurs affichées via « MenuItem » peuvent être éditées, soit en tapant la valeur à l'intérieur de la valeur directement, soit en créant un menu de valeurs à choix multiples en utilisant « MenuPick ».

Note : Il faut utiliser « DisplayValue » si on souhaite afficher des variables sans permettre qu'elles puissent être modifiées.

Paramètres *& Type de donnée*	**Entrée**
MenuItemName *Texte*	Le texte qui sera affiché en tant qu'en-tête pour le menu personnalisé. La chaîne est limitée à 20 caractères, et doit être comprise entre guillemets.
Source *Variable*	La source de la valeur à afficher à la droite du texte « MenuItemName ». La source doit être une variable.

MenuPick (Item1, Item2, Item3, ..., Item7)

L'instruction **MenuPick** est utilisée afin de créer un menu à choix multiple avec des valeurs qui peuvent être attribuées à la variable mentionnée dans l'instruction « MenuItem » qui précède. Lorsque « MenuPick » est utilisée, le menu à choix multiple est la seule façon que l'on aura pour donner une valeur à la variable à partir du menu personnalisé.

Le menu à choix multiple peut contenir des constantes (voir l'exemple). Les constantes doivent être définies dans le programme. Lorsque la liste contient des constantes, la valeur de la variable qui est affichée dans « MenuItem » sera affichée avec le nom de la constante (en texte) si la valeur numérique de la variable est égale à la constante.

L'instruction « **MenuPick »** doit être placée à la suite immédiate de l'instruction « **MenuItem »** pour laquelle une liste d'options est générée. Chaque élément de la liste est séparé du prochain par une virgule.

SubMenu/EndSubMenu

Syntaxe :
SubMenu ("MenuName")
Définition du menu (DisplayValue, MenuItem, and SubMenu)
EndSubMenu

Les instructions **SubMenu/EndSubMenu** sont utilisées afin de définir le début et la fin d'un écran de menu personnalisé, un niveau au dessous du menu actuel. Le paramètre «MenuName » est le texte qui sera affiché sur l'écran de la centrale de mesure pour le menu actuel, en tant qu'en-tête pour le sous menu. La chaîne est limitée à 20 caractères, et elle doit être comprise entre guillemets. « **EndSubMenu** » marque la fin de la définition du menu personnalisé. Les instructions « DisplayValue », « MenuItem », et « SubMenu » sont utilisées afin de définir les sous menus.

Paramètre *& Type de donnée*	**Entrée**
MenuName *Texte*	Le texte qui sera affiché en tant qu'en-tête pour le sous menu. La chaîne est limitée à 20 caractères, et doit être comprise entre guillemets.

Chapitre 11. Fonctions « Chaîne de caractères »

11.1 Expressions avec des chaînes de caractères

11.1.1 Chaînes de caractères constantes

Des chaînes de caractères fixes (constantes) peuvent être utilisées dans des expressions utilisant des guillemets. Si on écrit par exemple 'FirstName = "Mike" ', cela a pour résultat de donner la valeur "Mike" à la chaîne de caractères qui s'appelle 'FirstName'.

11.1.2 Ajouter des chaînes de caractères

Les chaînes peuvent être concaténées en utilisant l'opérateur '+'. Par exemple FullName = FirstName + " " + MiddleName + " " + LastName (Les " " permettent d'avoir un espace entre les noms.)

11.1.3 Soustraire des chaînes de caractères

String1- String2 donne pour résultat un entier compris entre –255 et +255. En débutant par le premier caractère de chaque chaîne de caractère, les caractères de « string2 » sont soustraits aux caractères de « string1 » jusqu'à ce que la différence soit différente de zéro, ou que jusqu'à ce que la fin de chaque chaîne (string) soit atteinte. Cette opération est utilisée la plupart du temps afin de savoir si des chaînes de caractère sont identiques ou non.

11.1.4 Conversion numérique d'une chaîne de caractères

La conversion d'une chaîne de caractère en nombre, ou d'un nombre en chaîne de caractère, est effectuée automatiquement lorsqu'une tâche est demandée à partir d'une chaîne, et vers un nombre, ou inversement, si cela est possible.

Par exemple:

```
Public Value                    ' format par défaut, IEEE4 à virgule flottante
Public SensorString AS String * 8               'lecture d'un capteur  ASCII
Value = SensorString  * 1.8 + 32  'la chaîne du capteur est convertie en valeur de la
variable Value au format IEEE4, et mise à l'échelle depuis °C vers °F.
```

Exemple : Ajouter un identificateur (ID) à la fin d'un nom :

```
Dim ID AS long
Public Names(10) AS STRING * 8

For ID = 1  to 10
    Names(ID) = "ITEM"+ID
Next ID
```

La ligne de données de Names(10) devient "ITEM1", "ITEM2",…,"ITEM10"

11.1.5 Chaînes de caractères et opérateurs de comparaison

Les opérateurs de comparaison =, >,<,<>, >= et <= fonctionnent sur les chaînes de caractères. Les opérateurs de qualité effectuent la soustraction de la chaîne de caractères mentionnée ci-dessus, puis appliquent les règles appropriées afin de donner comme résultat VRAI ou FAUX (TRUE / FALSE).

Exemple : Chercher le nom "Mike" dans ligne de donnée des "Names"

```
For ID = 1 to 10
    If Names(ID) = "Mike"
….
```

11.1.6 Sample () Type Conversions and other Output Processing Instructions

L'instruction d'échantillonnage « Sample() » effectuera les conversions nécessaires si le type de données de la source est différente du type de données de l'échantillon. La conversion de variables de type « float » ou « longs » en données de type « string », allouera 12 octets par champ afin de contenir la chaîne de caractères.

Pour toutes les autres instructions de sauvegarde, à l'exception des instructions de sauvegarde ayant une répétition égale à 1, le type de données de la source doit être le même que le type de données spécifié dans l'instruction. (Seul le premier élément de la variable source, est convertit du format « Long » vers le format « Float » si cela est nécessaire. D'où la répétition égale à 1 mentionnée ci-dessus).

Les chaînes de caractères ne sont pas autorisées avec les instructions de sauvegarde, sauf pour l'instruction « Sample() ».

11.2 Fonctions de manipulation de chaîne de caractère

CHR(c)

Utilisé de façon principale pour exprimer des caractères ASCII non imprimables.

La valeur de 'c' va de 0 à 255. A noter que 0 sera la terminaison d'une chaîne de caractères et c'est pourquoi ce caractère n'est utilisé qu'à condition qu'il doive être sauvegardé.

Exemple : Ajouter un "retour chariot / nouvelle ligne" (carriage return, line feed) à la fin d'une chaîne de caractères.

```
X = "Line"+Chr(13)+Chr(10)
```

FormatFloat (Float, FormatString)

L'instruction « FormatFloat » est utilisée afin de convertir une valeur à virgule flottante en chaîne de caractères.

Syntaxe

String = FormatFloat (Float, FormatString)

Remarques

La conversion de la chaîne de caractères de la valeur à virgule flottante est basée sur le format de « FormatString ». Voir l'aide de l'éditeur CRBasic pour plus de détails.

InStr (Start, SearchString, SoughtString, SearchOption)

L'instruction InStr est utilisée afin de trouver la position d'une chaîne de caractères à l'intérieur d'une chaîne de caractères.

Syntaxe

Variable = InStr (Start, SearchString, SoughtString, SearchOption)

Remarques

L'instruction donne comme résultat la partie entière du paramètre 'SoughtString'. Si 'SoughtString' n'est pas trouvé, l'instruction donne comme résultat la valeur 0.

Voir l'aide de l'éditeur CRBasic pour plus de détails.

LowerCase (SourceString)

Donne comme résultat la chaîne de caractère en minuscule pour ‘SourceString’.

Mid (SearchString, Start, Length)

L’instruction Mid est utilisée afin de donner comme résultat une partie d’une chaîne de caractères.

Syntaxe

String = Mid (SearchString, Start, Length)

Remarques

Les paramètres ‘Start’ et ‘Length’ sont utilisés afin de déterminer quelle partie de la chaîne ‘SearchString’ doit être donnée comme résultat. En dépit de la valeur du paramètre ‘Length’, la chaîne de caractère donnée en résultat ne sera pas plus longue que la chaîne de caractère originelle.

Voir l’aide de l’éditeur CRBasic pour plus de détails.

SplitStr (ResultString, SearchString, FilterString, NumSplit, SplitOption)

L’instruction ‘SplitStr’ est utilisée afin de donner comme résultat une ligne de donnée de chaînes de caractères ou de nombres provenant d’une chaîne de caractères.

Syntaxe

SplitStr (ResultString, SearchString, FilterString, NumSplit, SplitOption)

Remarques

Les champs ‘FilterString’ et ‘SplitOption’ aident à déterminer la ligne de donnée retournée par l’instruction ‘SplitStr’.

Voir l’aide de l’éditeur CRBasic pour plus de détails.

StrComp (String1, String2)

La fonction ‘StrComp’ est utilisée afin de comparer deux chaînes de caractères en effectuant une soustraction de caractères dans une chaîne de caractères, à partir de caractères de l’autre chaîne de caractères.

Syntaxe

Variable = StrComp (String1, String2)

Remarques

L’instruction ‘StrComp’ est généralement utilisée afin de savoir si deux chaînes de caractères sont identiques. En débutant par le premier caractère de chaque chaîne de caractères, les caractères de ‘String2’ sont soustraits aux caractères de ‘String1’ jusqu’à ce que la différence soit différente de zéro où jusqu’à ce que la fin de la variable ‘String2’ soit atteinte. Le résultat de cette instruction est un entier compris entre -255 et +255. Si le résultat est 0, c’est que les 2 chaînes de caractères sont identiques.

UpperCase (SourceString)

Donne comme résultat la chaîne de caractère en majuscule pour ‘SourceString’.

Chapitre 12. Fonctions d'Entrée / Sortie série

Ces jeux d'instructions et de fonctions sont destinés à être utilisés avec des capteurs série ou des contrôleurs (non-Pakbus), afin de composer et d'envoyer des caractères à des appareils génériques fonctionnant avec du texte. Ces instructions couvrent les fonctionnalités des P15 et P97 présentes sur les centrales de type Edlog ; de la flexibilité supplémentaire est ainsi ajoutée.

DialModem (ComPort, BaudRate, DialString, ResponseString)

L'instruction DialModem est utilisée afin de fournir en sortie d'un des ports série de la centrale de mesure, une chaîne de caractères pour qu'un modem puis composer un numéro.

Syntaxe

DialModem (ComPort, BaudRate, DialString, ResponseString)

ou

variable = DialModem (ComPort, BaudRate, DialString, ResponseString)

Remarques

DialModem effectue une instruction 'SerialOpen', plusieurs 'SerialOuts', et enfin une instruction 'SerialClose'. Si l'instruction est associée à une variable, cette variable sera à l'état –1 si 'ResponseString' est reçue avec succès, ou 0 si ce n'est pas le cas.

L'instruction DialModem peut être utilisée à l'intérieur d'une commande DialSequence/EndDialSequence afin de spécifier le chemin de communication à utiliser pour une centrale Pakbus, ou elle peut être utilisée à l'intérieur de la séquence BeginProg/EndProg afin d'envoyer la séquence de composition à chaque fois que l'instruction est exécutée. Lorsqu'elle est utilisée entre les commandes DialSequence/EndDialSequence, il faut que DialModem soit égal à la variable qui sera utilisée pour le paramètre 'DialSuccess' de EndDialSequence. On verra l'état de la variable grâce à l'instruction 'EndDialSequence'. Si l'appel n'aboutit pas, le lien sera fermé.

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

DialSequence (PakBusAddr)

Les instructions DialSequence/EndDialSequence sont utilisées afin de définir le code nécessaire pour transmettre des paquets de données à une centrale Pakbus.

Syntaxe

DialSequence (PakBusAddr)

Instruction de composition, par exemple :

DialSuccess = DialModem (ComPort, DialString, ResponseString)

EndDialSequence (DialSuccess)

Remarques

L'instruction DialSequence indique le début du code ; EndDialSequence indique la fin du code. Le code est écrit dans la partie du programme qui définit les déclarations, avant le programme principal (défini par BeginProg/EndProg).

A chaque fois qu'une instruction du programme principal nécessite de communiquer avec une centrale de mesure distante identifiée par le paramètre PakBusAddr, le code de DialSequence pour cette centrale, sera exécuté. Le code sera aussi exécuté si la centrale de mesure reçoit un message d'un autre appareil PakBus qui nécessite d'être dirigé (« routé ») vers une centrale de mesure distante.

Chaque instruction a un paramètre :

PakBusAddr Le paramètre PakBusAddr identifie l'adresse PakBus de la centrale de mesure avec laquelle la centrale de mesure hôte essaye de communiquer. Les entrées valides sont entre 0 et 4094. Chaque élément PakBus du réseau doit avoir une adresse unique.

DialSuccess Le paramètre DialSuccess est une variable dont on va suivre l'état et qui permettra de savoir si une tentative d'appel a réussi ou non. Si la tentative d'appel a échoué, le lien de communication sera fermé. Une variable contenant le résultat de l'instruction 'DialModem' peut être utilisée pour ce paramètre.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

DialVoice (DialString)

L'instruction DialVoice est utilisée afin de définir une séquence de composition de n° pour un modem à synthèse vocale COM310.

Syntaxe

DialVoice (DialString)

Remarques

Si l'instruction DialVoice est reliée à une variable et que cette variable a la valeur -1 (True / Vrai), cela signifiera que la tentative d'appel a réussi; si elle a la valeur 0 (False / Faux), cela signifiera qu'elle a échoué. VoiceHangup est utilisée après la tentative d'appel, afin de garder la ligne du modem vocal.

DialString DialString est le numéro de téléphone et les autres codes nécessaires à la composition du numéro. Une virgule insérée dans ce numéro ajoutera une pause de 2 secondes.

Lorsqu'on crée le code pour le modem vocal, l'instruction 'VoiceKey' devra être utilisée afin d'ajouter un délai avant que l'instruction 'EndVoice' soit exécutée. Sans cela, la centrale de mesure terminera la commande 'VoiceSpeak' avant que le message parlé soit terminé.

ModBusMaster (ResultCode, ComPort, BaudRate, ModBusAddr, Function, Variable, Start, Length, Tries, TimeOut)

L'instruction 'ModBusMaster' configure une centrale de mesure en tant qu'appareil ModBus maître (*master*) afin d'envoyer ou de recevoir des données à/d'un appareil ModBus esclave.

Syntaxe

ModBusMaster (ResultCode, ComPort, BaudRate, ModBusAddr, Function, Variable, Start, Length, Tries, TimeOut)

Remarques

La centrale de mesure comprend les fonctions ModBus entre 01 et 05, 15, et 16 (voir le paramètre de fonction ci-dessous). L'instruction ModBusMaster peut être placée en dehors du programme principal (défini par BeginProg/EndProg).

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

ModBusSlave (ComPort, BaudRate, ModBusAddr, DataVariable, BooleanVariable)

L'instruction ModBusSlave configure la centrale de mesure afin qu'elle soit un appareil ModBus esclave.

Syntaxe

ModBusSlave (ComPort, BaudRate, ModBusAddr, DataVariable, BooleanVariable)

Remarques

Cette instruction configure un appareil ModBus esclave afin de répondre à la demande de données d'un appareil ModBus maître. Les fonctions ModBus 01, 02, 03, 04, 05, 15, et 16 sont compatibles. Voir l'aide de l'éditeur CRBasic pour le détail des paramètres.

Notes :

La centrale de mesure communique avec les autres appareils ModBus en mode RTU (non pas en mode ASCII). Le port de communications, la vitesse en baud, le nombre de bit de données, le bit de stop et la parité sont configurés sur le pilote (driver) ModBus du PC lorsqu'on utilise des logiciels PC ou un PLC.

La centrale de mesure passe généralement en mode de veille après 40 Secondes d'inactivité sur le port de communications. Après être passé en mode veille, et avec certaines méthodes d'interfaçage, il peut être nécessaire d'envoyer un premier paquet de données afin de réveiller la communication, et il faut donc renvoyer le paquet de données afin qu'il soit réellement transmis. Par exemple le premier octet du paquet est utilisé à réveiller la SC32A/B ou la SC929, donc une tentative de renvoi du paquet dans les 40 secondes est nécessaire afin que le paquet ModBus soit reçu dans son intégralité par la centrale et qu'elle puisse le traiter. Si les paquets continuent d'arriver avant les 40 Secondes de délai, la centrale de mesure devrait répondre très rapidement aux nouveaux paquets. S'il est nécessaire, vous pouvez mettre la broche 3 du port CS I/O à 5V de façon permanente afin de garder la centrale de mesure en état de réveil. Le désavantage de cette méthode est que la consommation moyenne de courant sera supérieure à celle que la centrale de mesure aurait si elle avait la possibilité de passer en mode veille, lorsque les requêtes ModBus sont peu fréquentes.

Certains appareils ModBus (par exemple certaines RTUs de Bailey Controls qui utilisent des unités centrales (CPUs) moins communes) nécessitent des ordres de mots inversés (MSW/LSW) au format à virgule flottante. La centrale de mesure n'est pas compatible avec ce mode de communication moins commun, à l'heure actuelle. (Il y a certaines versions expérimentales pour certaines centrales de mesure, qui ont utilisé l'ordre de mots inversé.) Certaines suites de logiciel ont une configuration d'origine afin de fonctionner avec ce format ModBus d'origine. Par exemple l'option avancée (*advanced option*) de "Modicon 32-bit floating point order (0123 vs. 3210)" doit être activée pour la surveillance d'objets ModBus de National Instrument (*enabled for the ModBus object in National Instruments' Lookout).*

ModemHangup (ComPort) … EndModemHangup

Les instructions ModemHangup et EndModemHangup sont utilisées afin de contenir un code à exécuter lorsque un port COM décroche une communication.

Syntaxe

ModemHangup (ComPort)

Instructions à exécuter lors du « décrochage »

EndModemHangup

Remarques

ModemHangup indique le début du code; EndModemHangup indique la fin du code. Le code est écrit dans la partie destinée aux déclarations du programme, avant le programme principal (défini par BeginProg/EndProg). Lorsque la centrale de mesure détecte qu'un port COM est en train de décrocher, le code contenu dans ModemHangup sera exécuté.

Ce jeu d'instructions est le plus souvent utilisé pour des modems qui ont besoin de recevoir une séquence de commande afin de se déconnecter et de passer en mode de veille.

A noter que chaque port COM fonctionne indépendamment ; c'est pourquoi des commandes destinées à décrocher la ligne pour des modems, peuvent être effectuées en même temps.

Le paramètre 'ComPort' spécifie le port de communication et le mode de communication de cette instruction.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialClose (ComPort)

L'instruction SerialClose est utilisée afin de fermer un port de communications qui aura été préalablement ouvert par l'instruction 'SerialOpen'.

Syntaxe

SerialClose (ComPort)

Remarques

Si ce jeu d'instructions est égal à une variable, le résultat sera vrai [True (-1)] si le port Com était ouvert, ou Faux [False (0)] s'il était déjà fermé.

Le paramètre 'ComPort' spécifie le port de communication qui devra être fermé. Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialFlush (ComPort)

L'instruction SerialFlush est utilisée afin de supprimer les caractères présents dans la mémoire tampon du port série.

Syntaxe

SerialFlush (ComPort)

Remarques

Cette instruction vide la mémoire tampon, et laisse le port ouvert. Si la mémoire tampon en entrée doit être effacée avant chaque exécution de l'instruction 'SerialIn', il faudra placer 'SerialFlush' dans le code juste avant l'instruction 'SerialIn'.

Le paramètre ComPort spécifie la mémoire tampon du port de communication qui devra être effacée.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialIn (Dest, ComPort, TimeOut, TerminationChar, MaxNumChars)

L'instruction 'SerialIn' est utilisée afin de configurer un port de communication qui recevra des données série en entrée.

Syntaxe

SerialIn (Dest, ComPort, TimeOut, TerminationChar, MaxNumChars)

Remarques

Les données en entrée sont stockées dans la ligne de donnée de destination, jusqu'à ce que le caractère 'TerminationChar' soit reçu, que la valeur 'MaxNumChars' soit atteinte, ou que la durée du paramètre 'TimeOut' soit dépassée. Les données en entrée sont mises en mémoire tampon sur une mémoire circulaire dont la taille est définie par le paramètre 'SerialOpen'. La mémoire tampon (*buffer*) peut être effacée grâce à l'instruction 'SerialFlush'.

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialInBlock (ComPort, Dest, MaxNumberBytes)

L'instruction 'SerialInBlock' stocke des données série entrantes. Si elle est fixée à une valeur qui est une variable, ou si elle est utilisée en tant qu'expression, le résultat qu'elle donnera sera le nombre d'octets reçu.

Syntaxe

SerialInBlock (ComPort, Dest, MaxNumberBytes)

Remarques

Les données série entrantes, jusqu'à la valeur définie par le paramètre 'MaxNumberBytes', seront stockées dans le paramètre destination 'Dest'. 'SerialInBlock' n'attendra pas un caractère particulier en retour. Si aucun caractère nouveau n'a été reçu depuis la dernière fois où l'instruction a été exécutée, l'instruction donnera en retour la valeur 0. Cette instruction peut être utilisée, comme l'expression du paramètre de nombre d'octets 'NumberBytes' de l'instruction ''SerialOutBlock'.

Le paramètre 'ComPort' spécifie le port de communication et le mode de communication qui sera utilisé lorsque l'instruction recevra les données binaires.

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialOpen (ComPort, BaudRate, Format, TXDelay, BufferSize)

L'instruction 'SerialOpen' est utilisée afin de configurer un port série de la centrale de mesure pour qu'elle communique avec un appareil « non-PakBus ».

Syntaxe

SerialOpen (ComPort, BaudRate, Format, TXDelay, BufferSize)

Remarques

Lorsque l'instruction SerialOpen est exécutée, le port série est "ouvert" et les messages textuels subséquents arriveront ou sortiront du port entre des paquets PakBus. La donnée sera redirigée à partir du mode terminal d'entrée basée sur les instructions qui suivent SerialIn et SerialOut.

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

Note: SerialFlush est utilisée pour vider le buffer.

Pour la communication PakBus, BufferSize peut normalement être gardé à 0. Cependant, pendant la communication avec quelques appareils, il peut être nécessaire de limiter la dimension des paquets (BufferSize) et d'ajouter un délais (TXDelay) pour que la communication se passe dans de bonne condition. Par exemple, les paquets PakBus sont de 1000 bits. Le paquet le plus large que peut utiliser une radio de type RF95 est de 248 bits. La configuration du buffer à 240 limitera la dimension du paquet et permettra que le buffer du RF95 ne soit pas dépassé. Un délai (ex., 500,000 ms) permettra que chaque paquet ait assez de temps pour arriver à sa destination avant que le prochain paquet soit transmis.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialOut (ComPort, OutString, WaitString, NumberTries, TimeOut)

L'instruction 'SerialOut' est utilisée afin de transmettre une chaîne de caractères via un des ports de communication de la centrale.

Syntaxe

SerialOut (ComPort, OutString, WaitString, NumberTries, TimeOut)

Remarques

Si on fixe cette instruction égale à une variable, alors le nombre de caractères transmis sera le résultat de cette variable. Si un délai est nécessaire avant que la chaîne de caractères soit transmise, on doit le spécifier au paramètre 'TXDelay' de l'instruction 'SerialOpen'. Si les variables 'OutString' et 'WaitString' ne sont pas définies comme des chaînes de caractères (*string*), elles seront converties en chaînes de caractère par la centrale de mesure.

Une de ces conditions détermine le moment où la centrale de mesure devra passer à l'exécution de l'instruction suivante : lorsque le caractère 'WaitString' est reçu, que les tentatives 'NumberTries' sont dépassées, ou que la temporisation 'TimeOut' est atteinte.

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

SerialOutBlock (ComPort, Expression, NumberBytes)

L'instruction 'SerialOutBlock' est utilisée afin d'envoyer des données binaries en sortie sur un port série.

Syntaxe

SerialOutBlock (ComPort, Expression, NumberBytes)

Remarques

L'instruction est nécessaire lorsque les données qui doivent être envoyées contiennent une valeur nulle (*null value*). (L'instruction 'SerialOut' est terminée par une valeur nulle, c'est pourquoi lorsque cette valeur nulle est dans la chaîne de caractères à transmettre, il faut utiliser le mode binaire) Elle peut aussi être utilisée lorsque le nombre d'octets à transmettre est variable, ou lorsque l'appareil qui reçoit les données nécessite que celle-ci soient codées en format binaire.

L'aide de l'éditeur CRBasic donne des détails sur les paramètres.

Cette instruction s'exécute de façon séquentielle à partir de la séquence de tâche de traitement, peu importe que la centrale de mesure soit en mode séquentiel ou « pipeline ».

Chapitre 13. Instructions de communication PakBus

Ce jeu d'instructions est utilisé afin de communiquer avec d'autres appareils PakBus. En général elles spécifient un port COM et une adresse PakBus. Si la route conduisant à l'appareil n'est pas connue à l'avance, les instructions essayeront de faire une communication directe via le port COM. Si le chemin indique de passer par un voisin (*neighbor*) qui doit tout d'abord être joint, alors le jeu d'instruction lancera tout d'abord la séquence spécifiée 'DialSequence' pour joindre ce voisin.

Le paramètre d'adresse PakBus est une variable, ce qui implique qu'il peut être utilisé dans une boucle de type « For loop » par exemple.

Les constantes prédéfinies ci-dessous s'adressent au port de communication, paramètre 'ComPort' :

1 ComRS232
2 ComME
3 Com310
4 ComSDC7
5 ComSDC8
6 Com1 (C1,C2)
7 Com2 (C3,C4)
8 Com3 (C5,C6)
9 Com4 (C7,C8)

Ce paramètre configure un port Com par défaut au cas où le chemin pour accéder au nœud (*node*) distant ne soit pas encore connu.

La vitesse de communication en baud sur les ports asynchrones, sera par défaut 9600 baud, à moins que ce ne soit configuré autrement via une fonction ' SerialOpen()' définie au chapitre 5.1 ou si le port est ouvert par un packet de données entrant à une autre vitesse de communication.

Le paramètre 'Baud Rate' sur les ports asynchrones (ComRS232, ComME, Com1, Com2, Com3,and Com4) est restreint à quelques vitesses 300,1200,4800,9600,19200,38400,57600,115200, avec comme valeur par défaut 9600. (Cette vitesse de baud sur les ports synchrones, doit être configurée.)

L'instruction comprend en général une variable 'ResultCode' qui indique si l'instruction a réussi ou non. 'ResultCode' aura la valeur 0 s'il y a eu succès. Si la communication échoue, cette variable s'incrémentera. Si la communication se passe avec succès, mais qu'il y a des erreurs d'indiquées, alors le code de réponse (response code), puis le 'ResultCode' prendront une valeur négative. Les réponses possibles différentes de 0 (et négatives) sont :

1 Read Only or Permission denied – Lecture seule ou accès refusé
2 Out of Space in the remote – Plus de place sur l'appareil distant
3 Syntax Error – Erreur de syntaxe
16 Invalid Table Name or Field Name – Nom invalide pour le nom de champs ou de tableau
17 Data type conversion not supported – Type de conversion de données non supporté
18 Memory bounds violation – Violation du dimensionnement mémoire
19 Out of memory in the host – Pas assez de mémoire dans l'hôte
20 Cannot route to remote (communication no attempted)
–Impossible d'acheminer les données jusqu'à l'appareil distant (communication non aboutie)

Le paramètre ‘Timeout’ de ces instructions est en unité de 0,01 secondes. Si on utilise la valeur 0, alors la temporisation (*timeout*) par défaut définie pour le temps utilisé par le meilleur chemin, sera utilisé. Des “Hop Metrics” PakBus sont alors utilises afin de calculer ce temps.

Actuellement il n’y a pas de paramètre de nombre d’essai qui soit défini. Il est simple de programmer des tentatives de rappel avec CRBasic : For I = 1 to 3: SetSettings(ResultCode,….): if ResultCode = 0 Exit For.

(Si cela est nécessaire on peut ajouter un paramètre de rappel ou bien définir par défaut plusieurs tentatives de rappel.)

Ces instructions de communication vont attendre, par défaut, de recevoir une réponse ou d’atteindre un délai de temps spécifié, avant que le programme ne passe à l’instructions suivante. Ces instructions peuvent bien entendu être utilisées dans une partie de scrutation lente (SlowSequence Scan). Il est possible de configurer, en option, le paramètre ‘ComPort’ de façon négative; ceci fera en sorte que l’instruction n’attendra de réponse ni de délai de temps. Cela permettra que l’instruction s’exécute plus rapidement, mais toutes les données qui seront reçues, ainsi que le code ‘ResultCode’, auront des valeurs attribuées de façon asynchrone par rapport à la scrutation, c’est à dire à un quelconque moment où la communication sera effectuée.

ClockReport

‘ClockReport’ est une instruction qui envoie les valeurs de l’heure interne de la centrale de mesure, à une centrale de mesure distante du même réseau PakBus.

Syntaxe

ClockReport (ComPort, RouterAddr, PakBusAddr)

Remarques

Cette instruction initie une transmission à sens unique de la valeur de l’heure interne de la centrale de mesure, à une centrale distante. Aucune réponse n’est envoyée par la centrale distante. Si la centrale distante est programmée avec une instruction ‘PakBusClock’ indiquant l’adresse PakBus de la centrale qui envoie l’heure, la centrale distante aura son heure mise à jour par rapport à celle de la valeur de temps transmise.

Voir l’aide de l’éditeur CRBasic pour plus de détails.

A noter : Par défaut LoggerNet a l’adresse PakBus 4094 et PC400 a l’adresse 4093.

PakBusClock(PakBusAddr)

Lorsque le programme de la centrale de mesure comprend une instruction ‘PakBusClock’, la centrale de mesure mettra à jour son horloge à partir des valeurs de l’heure transmises par la centrale qui a l’adresse PakBus spécifiée dans ‘PakBusAddr’.

Syntaxe

PakBusClock (PakBusAddr)

Remarques

Le paramètre ‘PakBusAddr est l’adresse de la centrale distante pour laquelle la centrale de mesure va accepter un ‘ClockReport’. L’instruction ‘ClockReport’ est utilisée dans la centrale distante afin d’envoyer les valeurs de son horloge à cette centrale de mesure.

Routes(Dest)

L'instruction 'Routes' donne comme résultat une liste de routes ou chemins dynamiques connus pour atteindre une centrale de mesure PakBus.

Syntaxe

Routes (Dest)

Remarques

Cette instruction stocke quatre valeurs pour chaque route connue, dans le paramètre 'Dest'. Les quatre valeurs pour chaque route sont : 'ComPort' utilisé pour la communication, l'adresse PakBus du voisin 'neighbor PakBus address', l'adresse PakBus de destination, et le délai de temps attendu pour la réponse (en millisecondes). La liste de routes est terminée par '-1'. 'Dest' doit être une ligne de donnée dimensionnée afin de recevoir le nombre de routes (4*routes), plus une donnée de caractère de terminaison.

SendData (ComPort, RouterAddr, PakBusAddr, DataTable)

L'instruction 'SendData' est utilisée afin d'envoyer des valeurs les plus récentes d'un enregistrement dans un tableau de données, vers un appareil PakBus distant.

Syntaxe

SendData (ComPort, RouterAddr, PakBusAddr, DataTable)

Remarques

Cette instruction peut être utilisée afin d'envoyer les données vers un PC qui a le serveur LoggerNet d'activé. Lorsque les données sont reçues, LoggerNet stockera les données dans un fichier ayant le nom qui est en accord avec la convention de nom définie par la configuration du logiciel, dans la fenêtre « Setup » de celui-ci.

Voir l'aide de l'éditeur CRBasic pour plus de détails.

A noter : Par défaut LoggerNet a l'adresse PakBus 4094 et PC400 a l'adresse 4093.

DataTable	Le paramètre 'DataTable' est le nom du tableau de données contenant le dernier enregistrement qui devra être envoyé.

SendGetVariables ComPort, RouterAddr, PakBusAddr, Security, TimeOut, ResultCode, SendVariable, SendSwath, GetVariable, GetSwath)

L'instruction 'SendGetVariables' est utilisée dans une centrale de mesure distante afin d'envoyer une ligne de données de variables vers une centrale de mesure hôte, et/ou afin de recevoir une ligne de données en provenance de la centrale de mesure hôte.

Syntaxe

SendGetVariables (ResultCode, ComPort, RouterAddr, PakBusAddr, Security, TimeOut, SendVariable, SendSwath, GetVariable, GetSwath)

Remarques

Lorsque 'SendGetVariables' est utilisée sur une centrale de mesure, les temps de transmission des données sont contrôlés par la centrale de mesure hôte. La plupart du temps cette instruction est précédée de l'instruction 'TimeUntilTransmit' afin de conditionner l'exécution de l'instruction 'SendGetVariables'. Le programme présent dans la centrale de mesure hôte doit contenir l'instruction 'NetWork', instruction qui fixe le temps auquel la centrale de mesure distante devrait répondre.

Si la sécurité est active sur la centrale de mesure hôte, elle doit être débloquée jusqu'au niveau 2 afin que cette instruction puisse aboutir avec succès.

Voir l'aide de l'éditeur CRBasic pour plus de détails.

SendTableDef (ComPort, RouterAddr, PakBusAddr, DataTable)

L'instruction 'SendTableDef' est utilisée afin d'envoyer la définition d'un tableau de données (*table definitions*) vers un appareil PakBus distant.

Syntaxe

SendTableDef (ComPort, RouterAddr, PakBusAddr, DataTable)

Remarques

Cette instruction peut être utilisée afin d'envoyer la définition d'un tableau de données d'une centrale, vers un PC qui fait fonctionner le serveur LoggerNet.

Voir l'aide de l'éditeur CRBasic pour plus de détails.

SendVariables (ResultCode, ComPort, RouterAddr, PakBusAddr, Security, TimeOut, "TableName", "FieldName", Variable, Swath)

L'instruction 'SendVariables' est utilisée afin d'envoyer les valeurs provenant d'une variable ou d'une ligne de donnée de variables, vers un tableau de données présent sur une autre centrale de mesure.

Syntaxe

SendVariables (ResultCode, ComPort, RouterAddr, PakBusAddr, Security, TimeOut, "TableName", "FieldName", Variable, Swath)

Remarques

Les valeurs ne peuvent être envoyées qu'aux tableaux de données Public ou au tableau d'état (*Status Stable*). Les paramètres 'Dest' et 'Swath' sont utilisés afin de déterminer quelles valeurs vont être envoyées vers la centrale de mesure distante. La première valeur qui sera envoyée est définie avec 'Dest', et le nombre de valeurs envoyées est spécifié par 'Swath'. Les valeurs les plus récentes stockées dans le tableau, sont celles qui sont envoyées.

Si la sécurité est active sur la centrale de mesure hôte, elle doit être débloquée jusqu'au niveau 2 afin que cette instruction puisse aboutir avec succès.

Voir l'aide de l'éditeur CRBasic pour plus de détails.

SetSettings (ResultCode, ComPort, RouterAddr, PakBusAddr, Security, TimeOut, Settings)

L'instruction 'SetSettings' est utilisée afin de donner une ou plusieurs valeurs à un ou plusieurs paramètres dans une centrale de mesure distante.

Syntaxe

SetSettings (ResultCode, ComPort, RouterAddr, PakBusAddr, Security, TimeOut, Settings)

Remarques

Cette instruction peut être utilisée afin de fixer un ou plusieurs paramètres PakBus (ou paramètres créés par l'utilisateur) existant sur la centrale de mesure. Si la sécurité est active sur le centrale de mesure distante, elle doit être débloquée jusqu'au niveau 1 afin que cette instruction puisse aboutir avec succès.

Voir l'aide de l'éditeur CRBasic pour plus de détails.

A noter : Par défaut LoggerNet a l'adresse PakBus 4094 et PC400 a l'adresse 4093.

TimeUntilTransmit

L'instruction 'TimeUntilTransmit' donne comme résultat le temps qu'il reste, en secondes, avant que la communication soit effectuée avec la centrale de mesure hôte.

Syntaxe

TimeUntilTransmit

Remarques

La valeur de 'TimeUntilTransmit' est derivée de l'information de 'time slot' qui est envoyée par la centrale de mesure hôte. Si la centrale de mesure n'a pas encore envoyé d'information de 'time slot', cette instruction utilisera un intervalle de temps aléatoire compris entre 0 et 60 secondes, jusqu'à ce que la communication avec l'hôte soit effectuée.

Une utilisation typique de cette instruction est effectuée afin de conditionner l'exécution de l'instruction 'SendGetVariables' lorsque le 'time slot' pour la communication de la centrale de mesure se produit
(par exemple, If TimeUntilTransmit = 0 Then SendGetVariables).

Chapitre 14. Le Réseau PakBus

Ce chapitre est à considérer comme une introduction à la mise en réseau PakBus effectuée avec une CR1000. Vous verrez dans ce chapitre : une configuration « étape par étape » d'une liaison CR1000/FR416 ; une explication générale sur la configuration d'un réseau PakBus ; les concepts du protocole PakBus avec quelques détails ; une liste d'éditeurs de configurations et les capacités afin de configurer les paramètres PakBus de la CR1000 et des périphériques de communication ; un guide de résolution des problèmes qui liste certains problèmes que l'on peut rencontrer et les façons de les résoudre ; et un glossaire pour les termes du vocabulaire PakBus qui apparaissent dans ce chapitre.

14.1 Démarrage rapide – Exemple de réseau CR1000/RF416

Les pages suivantes décrivent la configuration pour le réseau présenté en exemple. LoggerNet communique avec la CR1000_10 et la CR1000_20 (voir ci-dessous) à l'aide d'une RF416 configurée en mode « PakBus Aware ». LoggerNet découvre la CR1000_10 à partir du plan de réseau (*device map*). La CR1000_10 utilise un filtre de réseau afin de découvrir la CR1000_20. La CR1000_10 simule un chemin forcé entre LoggerNet et la CR1000_20 afin de surmonter l'effet de distance, de terrain, de végétation ou de bruit, afin d'améliorer la fiabilité de la transmission (voir le chapitre 2.3 sur la « Découvert du réseau »).

S'il n'y a pas besoin d'effectuer des mesures au niveau du point de routage, une RF416 configurée en tant que routeur seul (stand-alone router) peut remplacer l'association de la RF416 + CR1000_10 configurée en tant que routeur. Une telle configuration est présentée à la fin « Démarrage rapide ».

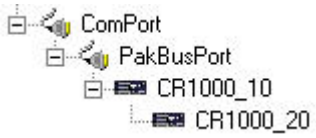
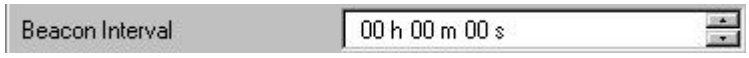
PC(LoggerNet)-RF416~~~~~RF416-CR1000_10~~~~~RF416-CR1000_20

PakBus Aware **PakBus Aware** **PakBus Aware**

Vous pouvez construire cet exemple de réseau avec les éléments ci-dessous ou leur équivalent :

- Deux CR1000s
- Deux alimentations PS100
- Trois câbles série SC12
- Trois radios RF416s
- Trois antennes pour RF416
- Un adaptateur secteur pour la RF416 de base (code # 15966)
- Un PC avec un port COM de disponible
- LoggerNet

Prenons pour élément de départ le logiciel LoggerNet; les configurations (*Setup*) des éléments du réseau décrit auparavant seront alors :

Exemple de réseau – Configuration de LoggerNet	
Setup	**Description**
Device Map	Le plan décrit la CR1000_10 en tant que routeur pour joindre la CR1000_20 :  Note : La CR1000_10 découvre la CR1000_20 grâce au filtre de voisins afin que LoggerNet puisse communiquer avec la CR1000_20 au travers de la CR1000_10.
PakBusAddresses	L'adresse par défaut de PakBusPort pour LoggerNet est 4094. On donne alors une adresse PakBus unique à chaque élément du plan de réseau : CR1000_10  CR1000_20 
Beaconing	On désactive l'envoi de balises (*beacon*) de LoggerNet (pour le PakBusPort), de ce fait il ne peut pas découvrir la CR1000_20 directement. 
Baud Rate	Mettre la vitesse en baud du PakBusPort, à la même vitesse que celle du périphérique de communication utilisé (ici le port RS232 de la RF416 de base) : 

Exemple de réseau – Configuration de la CR1000_10 (Routeur)	
Setup	**Description**
General / Général	Si les configurations de la CR1000 ont changé par rapport à ceux de l'usine, il faut alors utiliser le « Device Configuration Utility » afin de revenir aux configurations par défaut. On peut alors procéder aux configurations montrées ci-dessous.
PakBus Address	Dans l'onglet « Deployment », donner l'adresse PakBus à la centrale, de façon à ce qu'elle soit identique à celle de LoggerNet (*LoggerNet map*) ; faites attention à bien appliquer les modifications (*Apply*) : 

Neighbor Filter	Dans « Using Device Configuration Utility/Settings Editor », entrer en face des **Neighbors Allowed SDC7**, la valeur "(20, 20)" [Entrée], et en face de **Verify Interval SDC7**, la valeur "60" [Entrée]. Ceci configurera un filtre de voisins (*neighbor filter*) qui cherchera/permettra à l'adresse PakBus 20, d'être un voisin, en vérifiant sa présence chaque 60 secondes. Appliquer les modifications. Verify Interval ME: 0 Verify Interval SDC7: 60 Verify Interval SDC8: 0 Verify Interval COM1: 0 Verify Interval COM2: 0 Verify Interval COM3: 0 Verify Interval COM4: 0 Neighbors Allowed RS232: Neighbors Allowed ME: Neighbors Allowed SDC7: (20, 20) Neighbors Allowed SDC8:
Router	Dans « Device Configuration Utility/ Settings Editor », configurer la CR1000_10 en tant que routeur ; donner la valeur 1 à la case «Is Router». Is Router: 1
Central Routers	Non nécessaire. Laissez les vide ou à la valeur 0.
Beaconing	Laisser la valeur à 0 sur tous les ports.
Baud Rate SDC7	Non configurable par l'utilisateur (voir paragraphe 2.5.2).

Exemple de réseau – Configuration de la CR1000_20 (Leaf-Node)	
Setup	**Description**
General / Général	Si les configurations de la CR1000 ont changé par rapport à ceux de l'usine, il faut alors utiliser le « Device Configuration Utility » afin de revenir aux configurations par défaut. On peut alors procéder aux configurations montrées ci-dessous.
PakBus Address	Dans l'onglet « Deployment », donner l'adresse PakBus à la centrale, de façon à ce qu'elle soit identique à celle de LoggerNet (*LoggerNet map)* : PakBus Address: 20
Neighbor Filter	Pas de filtre de voisin nécessaire. (la CR1000_10 découvrira la CR1000_20 via son propre filtre de voisins.)
Router	Laisser « IsRouter » à sa valeur par défaut, « False » ou zéro.
Beaconing	La CR1000_20 est découverte par le filtre de voisin du routeur. Il n'y a aucune nécessité pour que la feuille (*leaf node*) envoie des balises (*beacon*). Pour minimiser le trafic radio, on désactive le « beaconing » (mis à 0) : Beacon Interval RS232: 0 Beacon Interval ME: 0 Beacon Interval SDC7: 0 Beacon Interval SDC8: 0 Beacon Interval COM1: 0
Baud Rate SDC7	Non configurable par l'utilisateur (voir paragraphe 2.5.2).

Exemple de réseau – Configuration des RF416s	
Setup	**Description**
General / Général	Si la configuration de base de la RF416 à été modifiée par rapport à la configuration usine, utiliser « Device Configuration Utility » pour redonner les valeurs par défaut, et procéder à la configuration décrite ci-dessous.
Active Interface (Port)	Utiliser « Device Configuration Utility » pour configurer : La RF416 de base en : AutoSense. Les RF416s distantes en : CSDC 7. Note: En réseaux PakBus, une configuration en port CSDC 7 (ou CSDC 8) est plus efficace que celle, par défaut, en port « Modem Enable » (ME).
RS-232 Baud Rate	RF416 de base (PC) : 38400 baud, en accord avec la config. de LoggerNet. RF416s distantes : valeur par défaut (9600 baud)
Hopping Sequence	Mettre toutes les RF416s à la valeur 1.
Network Address	Mettre toutes les RF416s à la valeur 2.
Radio Address	Non utilisé en mode « PakBus Aware » (toutes les RF416s ont la valeur 0).
Retry Level	Mettre toutes les RF416s à « Medium ».
Protocol	Mettre toutes les RF416s en mode « PakBus Aware ».
Standby Mode	Mettre toutes les RF416s à « <24 mA, Always on, no long header ».

Après avoir effectué les configurations ci-dessus, mettez un câble série entre le port COM de votre PC (celui utilisé par le plan de réseau « device map » de LoggerNet) et le port RS-323 de la RF416 configurée en AutoSense. Connectez les autres RF416s aux CR1000_10 et CR1000_20. Lancez LoggerNet et utilisez le menu « Connect » afin de vous connecter à la CR1000_10. La connexion prendra peut être quelques secondes avant de s'établir. Lancer « PakBus Graph » et vous devriez voir d'ici peu la configuration suivante :

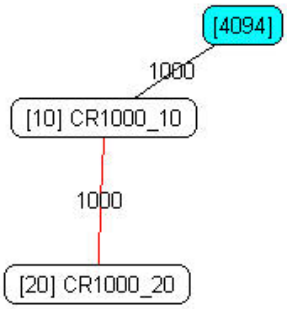


FIGURE 14-1. Vue du réseau via PakBus Graph

LoggerNet découvrira la CR1000_10 grâce au plan 'device map' (lien statique) peu après que vous ayez cliqué sur « *Connect* ». La CR1000_10 ayant un filtre de voisins, découvrira la CR1000_20. A ce point de la configuration, il est possible de se connecter à la CR1000_10 ou CR1000_20, et les fonctions de haut niveau de la centrale sont alors disponibles (mise à jour de l'horloge, envoi d'un programme, collecte des données etc.). Il est alors temps d'envoyer un programme sur chacune des centrales de mesure.

Sur le « PakBus Graph », si vous cliquez avec le bouton droit sur la CR1000_20, et que vous sélectionnez « Show Settings », vous verrez par exemples les « Routes » de la centrale (*routing table*). Le tableau de Route ci-dessous indique que le Port 4 (CSDC 7) de la CR1000_20 communique au travers d'un voisin ayant l'adresse PakBus 10, avec un appareil PakBus d'adresse 10. La CR1000_20, vu qu'elle est "leaf node", n'indique de Route que vers ses voisins. La valeur "1000" est le paramètre de '*hop metric*' (délai de réponse maximum en millisecondes) pour la communication avec cet appareil.

Routes	(4, 10, 10, 1000)

Si la CR1000_20 était configurée en tant que routeur, vous verriez le lien avec le voisin mais aussi la route vers le serveur LoggerNet. Utilisez le menu « PakBus Graph / Show Settings » afin de changer temporairement la configuration de la CR1000_20 en tant que Routeur (IsRouter à 1). Appliquez le changement et cliquez sur « Show Settings » encore une fois, vous verrez une Route d'ajoutée (voir ci-dessous). La nouvelle Route de la CR1000_20 est via le voisin CR1000_10 d'adresse PakBus 10, vers le serveur LoggerNet ayant l'adresse 4094. Le temps de réponse du second lien est 2000. Voir les concepts PakBus, les routeurs, et le glossaire pour plus de détails.

Routes	(4, 10, 10, 1000) (4, 10, 4094, 2000)

Afin d'éviter les communications non nécessaires, des nœuds 'nodes' qui ne nécessitent pas d'être des routeurs, doivent rester des 'leaf nodes'.

Si vous n'avez pas besoin d'avoir de mesure au niveau du routeur, vous pouvez alors supprimer la CR1000_10 et configurer la RF416 en tant que routeur seule (*stand-alone router*) ayant la même adresse PakBus. La configuration pour cela serait la suivante :

Exemple de réseau – Configuration de la RF416 'Stand-alone Router' (peut remplacer la CR1000_10 Routeur + RF416)	
Setup	**Description**
General	Si la configuration de base de la RF416 à été modifiée par rapport à la configuration usine, utiliser « Device Configuration Utility » pour redonner les valeurs par défaut, et procéder à la configuration décrite ci-dessous.
Active Interface (Port)	Utiliser « Device Configuration Utility » pour la configuration. Configurer en tant que « PakBus Router ».
Hopping Sequence	Lui donner la valeur 1 (comme tout le reste du réseau).
Network Address	Lui donner la valeur 2 (comme tout le reste du réseau).
Radio Address	Ne pas utiliser (laisser à la valeur 0).
Retry Level	Mettre à la valeur « Medium » (comme tout le reste du réseau).
Protocol	Configurer en tant que « PakBus Node ».
PakBus Address (APB)	Lui donner la valeur 10. (Il faut alors retirer la CR1000_10 du réseau, ou bien lui changer son adresse PakBus afin d'éviter les conflits sur le réseau.)
PakBus Beacon Interval	Lui donner la valeur 0 (pas de balises 'beacons').
PakBus Allowed Neighbors	Avec Device Configuration Utility, se connecter à la RF416, cliquer sur l'onglet PakBus, et taper "20" pour les voisins autorisés 'Allowed Neighbor' pour les en-tête 'Begin' et 'End'. Cliquer sur 'Add Range' puis 'Apply' pour appliquer.
PakBus Verify Interval	Lui donner la valeur 60 (secondes).
Central router	Ne pas utiliser (laisser la valeur à 0).

Supprimez la CR1000_10 du réseau et mettez la RF416 ‘stand-alone router’ à sa place (avec un adaptateur secteur CA). Vous devriez voir le même plan de réseau sur le graphe PakBus (*PakBus Graph network map*) qu’auparavant.

14.2. Les bases de la configuration réseau

L’approche générale lorsqu’on crée un réseau PakBus, est la suivante (l’ordre dans lequel on suit les étapes peut varier) :

CONFIGURATION DU RESEAU PAKBUS
1. Créer le schéma des ‘Router’ / ‘Leaf-node’
2. Assigner les adresses PakBus
3. Configuration de la découverte des voisins
4. Configuration du Routeur
5. Configuration de périphérique de communication
6. Création du plan de réseau de LoggerNet

14.2.1 Configuration en tant que ‘Router’ et ‘Leaf-node’

L’organisation entre les routeurs ‘routers’ et les feuilles ‘leaf-nodes’ se produit en amont de la planification du réseau. Le nombre de ‘routers’ sur le réseau, peut aller de zero à un nombre important, en partant du serveur LoggerNet et en allant vers la station la plus éloignée. Les ‘Routers’ peuvent inclure une centrale de mesure qui ne prend pas de mesure, ou bien une RF416 en mode ‘stand-alone router’ afin de transmettre uniquement les packets de données à destination d’autres centrales de mesure. Les appareils ‘leaf-node’ peuvent être ajoutés à n’importe lequel des ‘routers’.

Un ‘leaf-node’ ne peut pas transmettre de packet de données mais il peut les générer ou bien en être destinataire. La CR1000 est ‘leaf-node’ par défaut. Elle devient ‘router’ si on fait passer la configuration ‘IsRouter’ à la valeur « 1 » = « True » (voir paragraphe 4). Les CR216 sont toujours des ‘leaf-nodes’. Le réseau ci-dessous contient 4 ‘routers’ et 8 ‘leaf-nodes’. LoggerNet est toujours un ‘router’ et, en tant que tel, peut transférer des communications d’une centrale de mesure à une autre.

14.2.2 Configuration des adresses PakBus

En configuration d’usine, les CR1000s ont une adresse PakBus (APB) de 1. Pour les réseaux ayant plus d’une dizaine de stations, vous devriez avoir un système d’assignation d’APB afin de les organiser et de garantir des APBs uniques. Une façon de faire est d’assigner des APBs qui sont des multiples de dix pour les ‘*routers’* et d’assigner des APBs à leurs ‘leaf-nodes’ en remplissant ces décennies. L’exemple ci-dessous l’illustre :

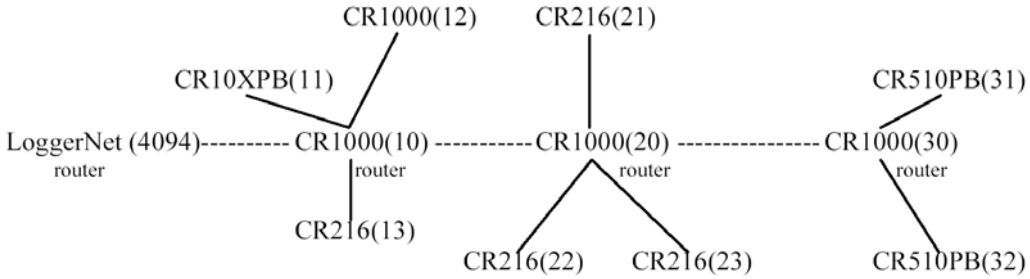


FIGURE 14.2-1. système d’assignation d’APB

Il y a plusieurs éditeurs de configuration (*Settings Editors*), tells que « Device Configuration Utility » et « PakBus Graph », qui peuvent modifier une APB de CR1000 (voir paragraphe 4). Attention aux changements de configuration d’APB sur les centrales de mesure distantes. Si vous finissez par ne plus savoir son adresse, vous aurez alors besoin de vous connecter à la centrale de mesure en *direct* à partir d’un PC utilisant « Device Configuration Utility » afin de découvrir son APB.

14.2.3 Configuration de la découverte des voisins

Les stations du réseau PakBus ont besoin de découvrir les liens qui seront nécessaires pour communiquer entre LoggerNet et les centrales de mesure, ou entre les centrales de mesure. Par exemple, dans la partie « Démarrage rapide », le filtre de voisins de la CR1000_10 établissait le lien nécessaire afin de communiquer avec la CR1000_20. LoggerNet peut découvrir un nœud déclaré lorsqu'on clique sur « Connect » dans l'écran 'Connect', à partir des informations de lien statique configurées correctement dans le plan de configuration de LoggerNet.

Si vous configurez LoggerNet et les routeurs de LoggerNet afin qu'ils envoient des balises (*beacon*), la découverte des voisins se fera automatiquement car toutes les voies imaginables utilisant la radio seront découvertes. Cependant dans certains cas, les chemins trouvés pourront être peu fiables, auquel cas ils peuvent être éliminés suite à l'ajout de filtres de voisins et la désactivation de l'envoi de certaines balises. D'un autre côté, vous pouvez trouver, via l'envoi de balises et de « *ping* » (voir ci-dessous), que des liens que vous pensiez être marginaux, sont acceptables. Si vous laissez la possibilité d'utiliser ce lien, cela gardera l'autre lien (via le routeur) comme un lien redondant, un lien de sauvegarde (*backup link*).

En utilisant des filtres de voisin, vous pouvez déterminer quels nœuds devront être considéré comme des voisins potentiels. Ceci vous permettra de forcer des packets de données à utiliser une route en particulier, la meilleure route (selon votre propre connaissance de l'installation). D'autres liens possibles peuvent être vérifiés en changeant temporairement les filtres de voisin afin de mettre en place l'envoi de balises, puis en revenant à la configuration avec filtre de voisins.

Avec l'envoi de balises ou bien avec découverte de filtres de voisins, il faut garder à l'esprit que : les liens sont établis et vérifiés à l'aide de messages assez courts (hello packets), mais lorsque les liens sont utilisés des messages plus long sont alors envoyés. La conséquence de cela est qu'un lien peut être suffisamment fiable pour une « découverte » de lien, mais pas assez fiable pour des packets de données plus importants. Ce cas de figure peut se produire pour une installation sujette à interférences radio. Le niveau de signal provenant de la radio peut paraître ample (voir manuel des RF416), mais le niveau de réussite du lien peut être par contre marginal. En raison de tout cela, l'intégrité des liens devrait être vérifiée en utilisant « PakBusGraph Ping Node ».

Les « pings » peuvent débuter avec 50 octets, puis 100, 200, et 500 octets. Le fait d'envoyer 10 « pings » de chacune de ces tailles, caractérisera le lien. Les autres sources de trafic sur le réseau (les appels automatiques pour la collecte de données, les vérifications d'horloge etc.) devront être suspendus temporairement pendant que ce test est effectué. L'envoi des « pings » devront être initiés par le PC de base et en utilisant « PakBusGraph » pour aller vers les voisins proches (1 hop away), puis à destination des nœuds qui sont à plus de *1 hop* de distance. Le gage de bon fonctionnement : 10 sur 10 est bon; 9 sur 10 est encore bon ; moins de 7-8 sur 10 (pour les packets de 500 octets) caractériseront une zone de moindre fiabilité.

Lorsque vous utilisez des balises sur un réseau, gardez les intervalles d'envoi de balises aussi long que possible lorsqu'il y a du trafic important (un grand nombre de nœuds et/ou des collectes de données fréquentes). De longs intervalles d'envoi de balises permettront de minimiser les risques de collision avec d'autres packets et les tentatives de renvoi de packets qui en découlent. L'intervalle minimum recommandé est de 60 secondes. Si vous avez un trafic plus important, vous devriez envisager de passer ce délai à quelques minutes. Si les besoins en transfert de données sont importants, vous pouvez maximiser la bande passante en créant des routeurs de « branche » (voire paragraphe 3.7), Et/ou en éliminant tous les envois de balises et en utilisant uniquement les filtres de voisins.

La configuration de filtres de voisins dans un routeur consiste en : (1) entrer les APBs des voisins autorisés (*Neighbors Allowed*) et (2) entrer l'intervalle de vérification (*Verify Interval*) 'xxx' pour le port approprié. Prenons par exemple la première CR1000 routeur dans le réseau d'APB précédent. Si vous avez besoin de configurer des filtres de voisins, vous devriez entrer les valeurs "4094, 11, 12, 13, 20" dans le champ 'Neighbors Allowed SDC7'. Pour le champ 'Verify Interval SDC7' de la CR1000, vous pouvez entrer '120' pour un intervalle de vérification égal à 2 minutes. La mention "4094" n'est pas nécessaire car toutes les APB ≥ 4000 passent tout de même au travers de tous les filtres de voisin.

Si un intervalle de vérification de filtre de voisins expire sans qu'il y ait eu de communication normale (telle que la collecte de données), le routeur essaye de ré-établir l'état du voisin, en initialisant un échange de messages « hello ». Un intervalle de vérification pour un appareil PakBus, est de 2,5 x l'intervalle d'envoi de balises. Donc une CR1000 qui envoie des balises n'a pas besoin de configuration de son intervalle de vérification ('Verify Interval' peut être laissé à la valeur 0). Si les intervalles de vérification sont différents entre des voisins, l'intervalle le plus court est utilisé pour le lien afin de garder actif le lien vers le voisin (voir les Concepts PakBus, intervalles de vérification).

Les intervalles de vérification configurés par l'utilisateur doivent être basés sur la temporisation de communications normales telles que les collectes de données en appel automatique ou les communications effectuées entre les centrales de mesure. L'idée est d'éviter que les intervalles de vérification puissent expirer avant qu'il y ait une communication normale. Si l'intervalle de vérification expire, l'appareil débutera une séquence de messages « hello » afin d'essayer de récupérer l'état du voisin, ce qui consomme un peu de bande passante.

14.2.4 Configuration du routeur

La CR1000 est configurée en 'leaf-node' par défaut (les fonctions de routage sont désactivées). Pour faire d'une CR1000 un routeur il faut donner la valeur « 1 » (ou « True »), à la variable 'IsRouter' (voir paragraphe 4). Si vous avez plus de 50 nœuds dans votre réseau, il faudra augmenter la valeur par défaut de 50, qui est attribuée aux 'PakBus Nodes Allocation'.

14.2.4.1 Routeurs seuls

Le fait de configurer une RF416 en mode 'PakBus Node' vous permet d'assigner une adresse PakBus à la radio en elle même, et de la configurer en routeur seul (voir manuel des RF416s). Tout comme un routeur, vous pouvez la configurer afin qu'elle découvre ses voisins via 'Neighbors Allowed' et 'Verify Interval' xxx, ou via 'Beacon Interval'. S'il n'y a aucun routeur de branche ou routeur central, vous pouvez lister les routeurs centraux dans la RF416 à l'aide de 'Device Configuration Utility' ou 'PakBus Graph/Settings'.

14.2.4.2 Routeurs de branche

Les RF416 servant de routeurs seuls, ont moins de mémoire de disponible pour le routage, que n'en ont les centrales de mesure qui servent de routeur. En guise de règle, on fera en sorte que si un réseau comprend environ 10 stations et un ou plusieurs routeur(s) seul(s), il sera nécessaire de configurer un « routeur de branche ».

La création de routeurs de branche est effectuée en désignant certains routeurs positionnés de façon centrale en tant que routeurs centraux, et en listant les routeurs centraux restant dans tous les routeurs restant (qui sont maintenant des branches du chemin). Voir le paragraphe 3 pour plus d'information au sujet des 'Branch Routers' et 'Central Routers'.

IsRouter	True
PakBusNodes	50
CentralRouters(1)	4094
CentralRouters(2)	11
CentralRouters(3)	21
CentralRouters(4)	0

14.2.5 Configuration du périphérique de communication

14.2.5.1 Ports Com

Sur un réseau PakBus, il est préférable d'utiliser le port de communication CSDC 7 ou CSDC 8 du périphérique, comme interface active. Celles-ci sont plus efficaces que le port par défaut qui est 'Auto-Sense', port ME.

Pour des communications initiées par le programme, telles que les transferts de centrale de mesure à centrale de mesure, l'instruction du programme peut (1) avoir son paramètre de port Com configuré à '0' afin de découvrir par lui même l'interface active du périphérique qui est attaché à son port de communication, ou (2) être configuré afin d'être en accord avec le port périphérique qui lui est connecté. Si vous avez par exemple une RF416 avec l'interface CSDC 7 active, vous pouvez configurer l'instruction du programme de deux façons qui sont :

- SendVariables (Res, **0**, 0, 1, 0, 0, "Pub", "FN", Var, Sw)
  ou
- SendVariables (Res, ComSDC7, 0, 1, 0, 0, "Pub", "FN", Var, Sw)

Il est pratique de laisser le paramètre du port Com égal à 0, de telle sorte que l'instruction fonctionnera quel que soit l'appareil connecté au port Com.

14.2.5.2 Vitesses de communication en Baud

Les vitesses en baud par défaut pour les ports 'BaudrateME' et 'BaudrateRS232' de la CR1000 sont 115200 Auto (–115200). Le signe moins qui indique 'Auto', précise au port qu'il devra se mettre à la même vitesse que la communication entrante. Après la communication d'une CR1000 configurée avec un mode auto RS-232 ou M.E. pour la vitesse en baud, le nombre avec le signe moins (Auto) changera et indiquera la vitesse en baud qui aura été utilisée le plus récemment.

Les vitesses en baud positives (non-Auto), fixent la vitesse de communication jusqu'à ce qu'un utilisateur édite ce paramètre, ou qu'une instruction du programme ne le change. L'instruction 'SetStatus' peut par exemple modifier la vitesse de communication en baud du port M.E. de la CR1000 :

SetStatus (BaudrateME, –115200)

Les ports CSDC et COM310 n'ont pas de vitesse de communication associée. Pour ces ports, les données asynchrones sont cadencées à un taux qui varie automatiquement en fonction du trafic réel. Ces ports ne sont pas configurables par l'utilisateur et le tableau d'état de LoggerNet (*LoggerNet's Status Table*) les indique avec la valeur 115200.

14.2.6 Configuration du plan de réseau de LoggerNet

Après avoir ajouté le port COM (ou autre port de communication) comme racine, au travers duquel vous souhaitez communiquer, ajoutez n'importe quel appareil dont vous avez besoin, un port PakBus ('PakBusPort'), et les centrales de mesure. Les figures 2-1A et 2-1B peuvent être des réseaux PakBus de RF4xx. A noter que les appareils RF4xx n'ont pas besoin d'être représentés sur le plan de réseau.

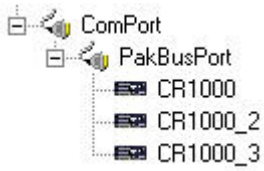


FIGURE 14.2-2A. Plan rectiligne

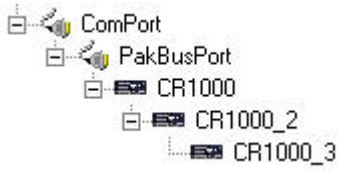


FIGURE 14.2-2B. Plan déployé

Dans les réseaux de RF4xx, quand toutes les RF sont des stations distantes, vous pouvez utiliser la configuration « rectiligne » (Figure 2-2A). Dans les réseaux RF4xx où il y a des routeurs, on utilisera la configuration « déployée » (Figure 2-2B). Ici, la CR1000 est probablement configurée en tant que routeur avec pour filtre de voisin la CR1000_2, et la CR1000_2 est probablement configurée en tant que routeur entre la CR1000 et la CR1000_3.

Sur la Figure 2-2B, la forme du plan de réseau ne permet pas une connexion directe de LoggerNet à CR1000_2 et CR1000_3, cela implique des itinéraires forcés, qui seront probablement établis par l'utilisateur à travers la configuration de filtres pour voisin (neighbor filter) dans le routeur. Ceci suppose que les balises (beacons) de LoggerNet soient arrêtées (non configurées) avec par défaut l'APB de 4094, sinon les balises de LoggerNet pénétreraient le filtre de voisin de toute la gamme ou marginalement la gamme des nœuds.

14.3. Concepts PakBus

14.3.1 Paquets

Les centrales de mesure d'ancienne génération (*Mixed-array dataloggers*) transfèrent des messages à travers des connexions fixe, d'un point à un autre, en monopolisant les ressources du chemin de communication jusqu'à ce que la tâche soit accomplie. Les centrales PakBus, d'un autre côté, transfèrent les messages en paquets « packets », ce qui permet de partager en temps réel les ressources de communication réseau en donnant la possibilité de transférer des packets de données en provenance de plusieurs stations, et grâce au même lien matériel.

Il existe deux niveaux de paquets PakBus. Les paquets « BMP5 » prennent en compte les fonctions de « haut niveau » pour les centrales de mesure, telles que la mise à jour de l'horloge, l'envoi d'un programme, la réception de données, ou l'envoi de données entre centrales de mesure. Les paquets de bas niveau « PakCtrl », prennent en compte des fonctions telles que la découverte de voisins ou les communications de routeur à routeur.

La taille des paquets est typiquement limitée à 1000 octets. S'ils font plus de volume, les données sont envoyées en plusieurs packets. Chaque paquet comporte un en-tête qui comporte l'adresse de la source, de la destination, des informations sur les voisins, des informations de contrôle, et bien entendu des données. Une fois qu'un paquet est envoyé sur le réseau, le système de routage établi le guidera jusqu'à sa destination. Les paquets peuvent être stockés dans un routeur durant quelques millisecondes afin de permettre à d'autres paquets d'être transmis via le même lien matériel.

14.3.2 Appareils PakBus

Un appareil PakBus est un appareil capable de créer et de traiter des paquets au format PakBus. Des exemples sont : La CR1000, LoggerNet, la RF416 (en utilisant le protocole 'PakBus Node'), la CR10X avec le système d'exploitation PakBus, le NL100, et la CR216.

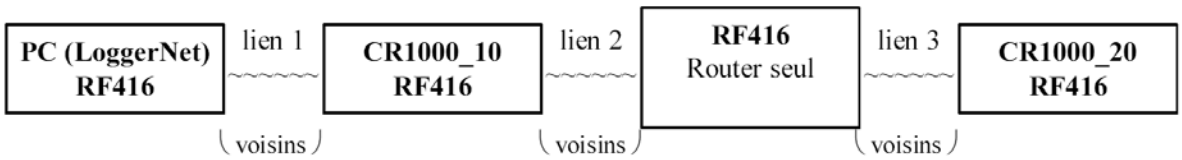
Un appareil PakBus dispose d'une adresse PakBus entière (APB) comprise entre 1 et 4094. Pour former un réseau qui fonctionne correctement, chaque appareil PakBus doit avoir une APB *unique*. L'adresse par défaut pour le serveur LoggerNet est 4094, ce qui est utilisé par toutes les plans représentant les 'PakBusPorts'. L'adresse par défaut de la CR1000 est 1.

La CR1000, par défaut, peut gérer un réseau allant jusqu'à 50 nœuds. Si le nombre d'appareils PakBus présent sur votre réseau, est supérieur à ce nombre, vous devrez changer le paramètre de 'PakBus Nodes Allocation' afin de le rendre compatible avec ce nombre d'appareils.

14.3.3 Découverte de voisins

Les termes de *découverte* et de *voisins*, ont des sens particulier lorsqu'on les attache à 'PakBus'. Pour que deux appareils PakBus transfèrent des données ou des programmes (par exemple LoggerNet et une CR1000 reliés via une paire de radios RF416), ils doivent tout d'abord devenir des *voisins*. Pour devenir voisins, un appareil PakBus doit *découvrir* l'autre appareil en lui envoyant des paquets de données de bas niveau de type *hello* (PakCtrl). Lors de la réception de ces paquets, l'autre appareil doit répondre avec un paquet de réponse-au-*hello*. Si l'échange de hello est réussi, les deux appareils seront reconnus en tant que voisins.

Si deux appareils PakBus doivent communiquer via un routeur ou des routeurs (par exemple LoggerNet et la CR1000_20 ci-dessous), alors pour chaque lien sur l'itinéraire entre les appareils (il peut y en avoir deux, trois ou plus), les deux appareils associés à chaque lien doivent se reconnaître l'un l'autre en tant que voisins suite à des échanges réussi de messages hello, et alors seulement la communication de type BMP5 (c'est à dire mise à jour de l'horloge, envoi de programmes, collecte de données etc.) peut avoir lieu.



Il y a deux façons principales de découvrir les voisins :

Dans certains réseaux de RF4xx qui comprennent des routeurs, des filtres de voisins sont employés afin d'éliminer les liens RF sur lesquels on ne peut pas compter. L'idée est d'éviter qu'un lien direct ne soit utilisé lorsqu'on souhaite qu'un routeur soit nécessaire à l'envoi du message. 'PakBusGraph Ping Node' est utilisé afin de caractériser un lien et de déterminer s'il est nécessaire de forcer l'utilisation d'un chemin à l'aide de filtres de voisins (voir paragraphe 2.3). Parfois un savant mélange d'envoi de balises et de mise en place de filtres de voisins, peuvent être employés.

Que vous utilisiez les envois de balises, les filtres de voisins ou les deux, le mécanisme de base de la découverte de voisins est l'échange de messages de *hello*. Un quelconque des événement qui suivent, peut générer l'envoi d'un packet de type *hello* vers un appareil PakBus spécifique :

LoggerNet essaye de se connecter à un voisin potentiel défini dans sa plan de réseau

- Un appareil PakBus entend une balise
- Un filtre de voisin est configuré

La réussite des échanges de messages de *hello*, dépend de quelques critères. L'appareil auquel on envoie un message de hello doit avoir assez de place dans la liste de voisins, pour ajouter le voisin qui lui envoi le hello. Les CR2xxs n'acceptent qu'un seul voisin (voir le glossaire). Pour que l'échange de messages *hello* réussisse, l'appareil auquel est envoyé le hello, ne doit pas avoir un filtre de voisin qui exclurait l'appareil qui envoi le message hello. Un appareil qui envoie un message hello et qui aurait une adresse ≥ 4000, ne peut pas être exclu par un filtre de voisins.

Si vous créez le plan de réseau de LoggerNet telle que décrit ci-dessous (avec les bonnes vitesses en baud) et que vous fournissez des liens de communication tells que des radios RF4xxs ou des câbles RS-232, alors lorsque vous ouvrirez la fenêtre de l'écran « *Connect »*, et que vous cliquerez sur l'icône *« Connect »*, LoggerNet enverra des paquets de bas niveau à destination de l'adresse PakBus 1. Si la CR1000 d'adresse 1 reçoit le hello et répond avec succès, les appareils deviennent des voisins. A partir de là la communication de haut niveau peut avoir lieu (envoi de programme, collecte de données etc.) et les envois d'itinéraire pour le partage des informations de routage de bas-niveau débute.

Les informations sur les types de packets PakBus sont décrites dans le guide 'PakBus Networking Guide' qui est disponible au format PDF sur http://www.campbellsci.com rubrique Support/Manuals.

14.3.4 Suppression des voisins

L'état des voisins entre des appareils PakBus, doit être rafraîchi de façon périodique. S'il n'y a pas de communications avec succès sous forme (1) normale ou sous forme de (2) vérification ('hello-exchange'), la communication expirera. La vérification de la communication se produira automatiquement en cas d'absence de communication normale, lorsque l'intervalle de vérification expirera.

14.3.5 Intervalles de Vérification

L'intervalle de vérification est l'intervalle de temps auquel un appareil PakBus qui a un voisin défini, mais qui n'aurait pas eu de ses nouvelles pendant le délai normal de communication entre appareils PakBus, essayera d'établir une communication avec ce voisin en lui envoyant des messages de *hello*. Cet intervalle de vérification est défini (entré par l'utilisateur dans la case 'Verify Interval xxx') ou bien calculé en multipliant le temps d'envoi de balises (*beacon interval*) par 2,5. Pour la vérification de la communication, c'est l'intervalle de vérification le plus court (entre deux voisins) qui sera utilisé. Par exemple, si l'intervalle de 'beaconing' de LoggerNet est de 120 secondes, et celui de la CR1000 est de 60 secondes, alors l'intervalle de vérification entre ces deux appareils sera de 60 × 2,5 = 150 secondes. Ceci est valable de la même façon lorsque deux CR1000 communiquent. Lorsqu'une centrale 'leaf node' est impliquée, le délai de vérification de la communication devient infini.

Lorsqu'un intervalle de vérification du lien est dépassé, les deux appareils débutent un échange de messages *hello* afin de ré-établir la table d'état des voisins.

Une CR1000 peut ne pas avoir de 'Verify Interval xxx' de défini (il peut être laissé à la valeur 0) ; cependant si vous souhaitez avoir un délai de découverte initiale plus court, et un délai plus long pour l'intervalle de vérification, vous pouvez entrer dans le cadre définie pour 'Verify Interval xxx', un nombre plus important que ('beacon interval' × 2.5) et la CR1000 vérifiera le lien à cet intervalle de temps précisément. Par exemple, pour un délai de découvert initial divisé par deux, on entrera la valeur 300 pour le 'Verify Interval SDC7' mais un 'Beacon Interval SDC7' à la valeur de 60 secondes.

La règle à utiliser afin de fixer la valeur de l'intervalle de vérification, est de configurer (verify interval) > (scheduled collection interval), de façon à ce que la collecte automatisée de données (*scheduled collection*) maintienne la liste d'état des voisins en ordre. Avec une CRxxxPB (qui ne comprend pas de case afin d'entrer un intervalle de vérification), on configurera l'intervalle d'envoi de balises tel que : (beacon interval) > (scheduled collection interval / 2.5).

Lorsqu'une CR1000 dispose d'un filtre de voisins, tous les envois de balises devraient être désactivés. Son intervalle de vérification est celui entré par l'utilisateur pour 'Verify Interval xxx'.

14.3.6 Routeurs

Un *routeur* est un appareil PakBus qui a la capacité de recevoir, de traiter, de ré-adresser et de transmettre un paquet jusqu'à sa destination. Un routeur a aussi la capacité de participer au système de routage du réseau. La plupart des appareils PakBus (à l'exception des CR2xx) peuvent être des routeurs si on les configure dans ce but. Un appareil de communication transparente peut recevoir et transmettre des paquets mais, n'ayant pas d'adresse PakBus propre, et n'étant pas capable de traiter et de ré-adresser un paquet, ni de prendre part au système de routage, il ne sera ni un appareil PakBus ni un routeur.

Chaque appareil PakBus maintien une liste de voisins qui contient l'adresse PakBus des appareils qu'il considère comme des voisins. Les appareils PakBus qui sont des routeurs, ont aussi une liste de routeurs qui indique tous les routeurs du réseau, et un tableau de lien qui indique tous les liens du réseau. Les routeurs mettent cette information dans un tableau de '*best-route algorithm'* afin de créer une liste de tableau de routage avec tous les nœuds présents sur le réseau et les voisins respectifs au travers desquels il faut passer afin d'atteindre ces nœuds. Un routeur est responsable, lorsque sa propre liste de voisins change, de la transmission de l'information aux appareils de sa branche de réseau.

Le fait d'être un routeur, utilise plus de ressource que celui d'être un 'leaf node'. Ainsi on fera en sorte de configurer le plus possible de 'leaf nodes', et on ne configurera des envois de balises qu'à partir des routeurs.

Si une CR1000 est configurée en tant que routeur (avec 'IsRouter' à la valeur « 1 » ou « True »), vous pourrez voir une liste de routage 'Routing Table' en utilisant 'PakBus Graph'. Cliquez avec le bouton droit sur le graphique de la CR1000 et sélectionnez 'Show Settings'. Vous devriez voir quelque chose de ce type:

Routes	(1, 4094, 4094, 1000) (4, 22, 22, 1000) (4, 3, 3, 1000)

Il y a trois routes dans le tableau de routages ci-dessus. La seconde route (4, 22, 22, 1000) doit être interprétée de la façon suivante :

4	Le port Com actif sur le CR1000 est le Com 4 (CSDC 7).
22	Adresse PakBus du voisin qui transmit à l'APB de destination.
22	APB de la destination.
1000	Hop Metric (temps de réponse du message en millisecondes).

Les ports Com de la CR1000 sont les suivant :

1. RS-232 Port
2. CS I/O Modem Enabled (ME)
3. CS I/O COM310 Modem
4. CS I/O CSDC 7
5. CS I/O CSDC 8
6. C1/C2
7. C3/C4
8. C5/C6
9. C7/C8

Vous pouvez copier le tableau de routage de la CR1000 en utilisant l'instruction 'Routes'; ceci met les informations dans une variable que vous pouvez définir comme 'Public', et visionner avec 'Numeric Display'. Le tableau suivant montre une partie d'un tel tableau de routage afin de décrire le chemin pour aller à l'adresse PakBus 4094. Lorsque les champs (2) et (3) contiennent la même adresse, cela indique que la CR1000 est un voisin de l'appareil auquel est destiné le packet et qui est mentionné au champ (3).

RoutingTable(1)	1.00
RoutingTable(2)	4094.00
RoutingTable(3)	4094.00
RoutingTable(4)	5000.00

RoutingTable(1) – Port Com de la CR1000, utilisé pour la communication

(voir détail de la liste des ports Com ci-avant)

RoutingTable(2) – Adresse PakBus du voisin de la CR1000, qui conduit à l'adresse de destination contenu dans RoutingTable(3)

RoutingTable(3) – APB de l'appareil de destination (ici : LoggerNet)

RoutingTable(4) – 'Hop metric' (temps de réponse attendu en millisecondes) pour le nœud de destination. Le mode de veille choisi pour une radio RF416 utilisant un protocole PakBus, affecte la valeur du 'hop metric'.

Un système de distribution de routage dans un réseau, est basé sur le fait que les routeurs apprennent qui sont leurs voisins et qu'ils partagent les informations avec les autres routeurs, afin que chacun puisse choisir le meilleur chemin pour atteindre un nœud et appareil PakBus. A ce niveau c'est le but d'un routeur que de connaître la liste de voisins de chaque autre routeur du réseau. Il existe une exception lorsqu'on crée des branches de réseau et des routeurs de branche (*branch router*).

14.3.7 Routeurs de branche et routeurs centraux

Un routeur de branche tâche seulement de connaître les listes de voisins des (1) routeurs centraux (*Central Routers*), (2) des routeurs présents entre lui même et les routeurs centraux, et (3) les routeurs du réseau qui sont autres que lui-même. Les routeurs centraux connaissent le réseau dans son ensemble. Le fait que les routeurs de branche aient les routeurs centraux dans leur tableau de routage, cela leur permettra d'accéder à tous les nœuds du réseau.

Le fait de configurer un routeur en routeur de branche, réduit la taille de son tableau de routage. Par exemple, les RF416s peuvent être des routeurs de branche lorsqu'elles sont configurées en tant que routeurs seuls dans un réseau qui dispose de plusieurs routeurs et/ou il y a beaucoup de trafic de communication. De façon générale, les routeurs de branche doivent être vu comme une façon de réduire le trafic radio ; cependant si le nombre de routeurs dans le réseau, est supérieur à 10, alors le fait de configurer les RF416 routeurs seuls en tant que routeurs de branche devient nécessaire, car les radios ont une ressource mémoire qui est réduite. Les routeurs centraux choisis de façon appropriée pour un réseau donné (voir exemple ci-dessous) peuvent réduire d'environ de moitié, le nombre de nœuds dont les routeurs de branche doivent garder la trace à l'intérieur de leurs tableaux de routage.

Les indications qui montrent que l'information de routage du réseau dépasse la capacité mémoire d'un routeur seul sont : dans un réseau avec envoi de balise (beconing network), un routeur seul n'a pas de route définie pour tous les nœuds du réseau, comme cela se devrait ; LoggerNet n'est pas capable de se connecter à certains nœuds qui passent par le biais de certains routeurs seuls. Vous pouvez voir le tableau de routage d'un routeur seul dans 'PakBus Graph' en cliquant sur le bouton droit sur le nœud et en sélectionnant 'Show Settings'.

Pour diviser un réseau en plusieurs branches et configurer des routeurs de branche, il est nécessaire de designer un ou plusieurs routeurs dans le réseau en tant que routeurs centraux. Les routeurs qui restent dans le réseau doivent être configurés en tant que routeurs de branche, ce qui est effectué en indiquant dans le champs 'Central Router', les APB des routeurs centraux.

Les règles du routage par branche :

1. Les routeurs centraux sont continus (aucun routeur de branche ne doit être intercalé entre les routeurs centraux)
2. Les routeurs centraux sont assez éloignés de LoggerNet pour isoler les branches
3. Tous les routeurs du réseau doivent participer soit en tant que routeur central, soit en tant que routeur de branche
4. Tous les routeurs de branche listent les même routeurs centraux dans leurs champs 'Central Router'
5. Les routeurs centraux ont « 0 » dans leurs champs 'central router'

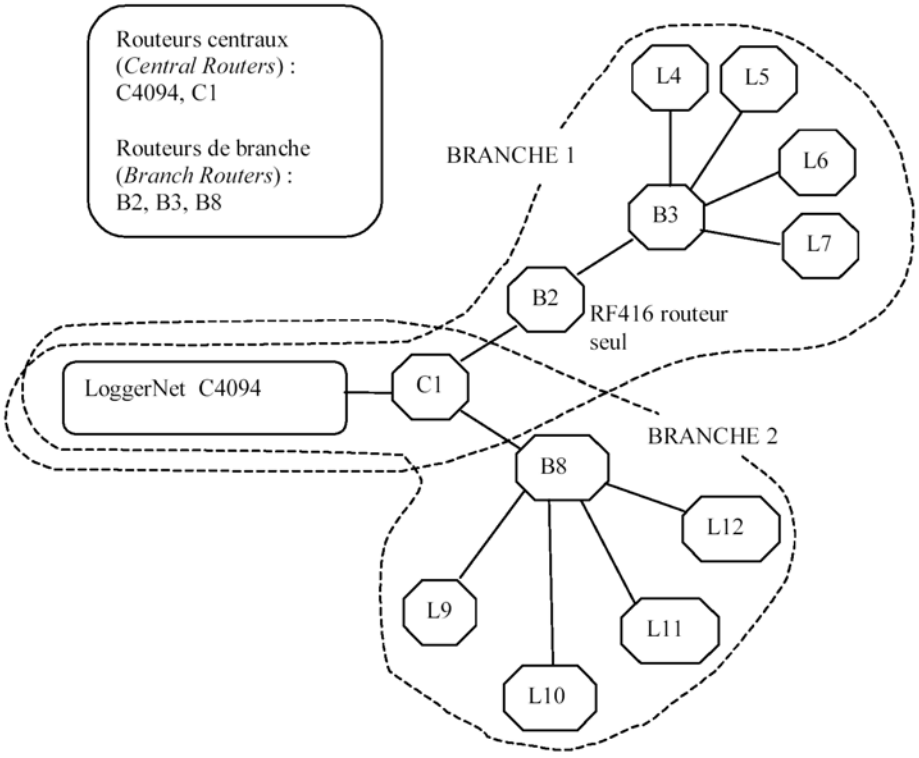


FIGURE 14.3-1. Exemple de réseau avec des routeurs de branche

C1 (routeur central) devrait être une CR1000 avec une RF416 et un câble RS-232 relié à un PC faisant fonctionner LoggerNet. Les nœuds débutant par « B » pourraient être des CR1000 routeurs de branche avec des RF416 (mentionnant 4094 et 1 en tant que routeurs centraux). Les nœuds débutant par « L » pourraient être des CR1000 'leaf-nodes' avec des RF416s (ou des CR216s). Les 'leaf nodes' ne sont pas des routeurs et n'ont pas de liste de routeurs centraux.

Dans un réseau ayant des routeurs de branche, *chaque* routeur est essentiellement un routeur central étant donné que chaque routeur tâche de connaître les listes de voisin de tous les routeurs présents sur le réseau.

L'outil de tableau d'état de l'écran 'Connect' peut être utilisé afin d'entrer la liste de routeurs centraux sur chaque routeur de branche. A noter que dans l'exemple ci-dessous LoggerNet est considéré comme un routeur central. LoggerNet n'a pas la possibilité d'être un routeur de branche (il ne peut pas lister les routeurs centraux) ; il doit donc toujours être listé en même temps que les routeurs centraux et, selon les règles, il doit être continu avec les autres routeurs centraux.

IsRouter	True
PakBusNodes	50
CentralRouters(1)	4094
CentralRouters(2)	1
CentralRouters(3)	0
CentralRouters(4)	0

14.3.8 Leaf-Nodes

Un '*leaf-node*' est un appareil PakBus qui ne transmet pas de paquet, même s'il pourrait en être capable si il était configuré dans ce but. Le tableau de routage d'un 'leaf-node' est limité à sa liste de voisins. Avec 'PakBus Graph/ Show Settings' vous n'afficherez que le lien vers le voisin du 'leaf node'.

Si un message est initié par la CR1000 (en utilisant SendVariables(), GetVariables(), SendGetVariables(), SendData(), SendTableDef(), ou ClockReport()), et si le paramètre de la variable 'neighbor address' est de « –1 », ce qui veut dire « découvre le chemin » (*discover the route*), un 'leaf-node' enverra le packet de données au premier routeur qu'il trouvera parmi la liste de voisins qu'il aura découvert.

Le fait d'être un routeur nécessite plus de ressource que le fait d'être 'leaf node'. On fait alors en sorte d'avoir le plus possible de 'leaf nodes', et que seuls les routeurs envoient des balises (*beacon*). Ceci évite que des liens inutiles soient crées entre des 'leaf-nodes' qui n'auraient aucune raison de communiquer entre eux.

Dans un réseau qui découvre les voisins suite à la mise en place de filtres et envoi de messages "hello", les 'leaf nodes' n'ont pas besoin d'avoir de filtre de voisins car ils seront découverts par leurs routeurs respectifs.

Lorsque LoggerNet n'envoi pas de balises et qu'il communique avec un 'leaf node' en direct (par exemple via RS-232 ou RF416s), l'intervalle de vérification de la communication sera infini. Une fois la communication établie, LoggerNet considèrera que le 'leaf node' est un voisin, pour toujours (ou bien jusqu'à ce que sa liste de voisin soit mise à jour, ce qui se produit via « Reset Node » dans 'PakBus Graph' ou lorsque LoggerNet est fermé).

14.3.9 Balises et filtres de voisin (Beacons and Neighbor Filters)

L'envoi de balises est une façon pratique, pour un appareil PakBus, de découvrir les voisins; ce mode de fonctionnement peut cependant conduire à l'utilisation de liens de communication direct mais peu fiable, alors qu'un lien plus fiable existerait via un routeur. Pour des réseaux de RF4xx avec des routeurs, il sera peut être nécessaire de désactiver certains envois de balises, et d'employer à la place des filtres de voisins. Vous pouvez utiliser 'Ping Node' de 'PakBus Graph' afin de vérifier les liens (voir paragraphe 2.3).

Pour créer un filtre de voisins, on procède en deux étapes. On spécifie les APB du ou des voisins autorisés ‘Allowed Neighbors’ et on spécifie une durée pour l’intervalle de vérification ‘Verify Interval xxx’ (en secondes). Un appareil PakBus qui a un filtre de voisin, enverra un packet « hello » dans le but d’essayer d’établir un lien de « voisinage » avec les appareils PakBus qui sont définis dans sa liste de voisins. Plus tard, si un voisin établi comme tel, ne donne pas de nouvelles Durant l’intervalle de vérification qui est spécifié, le filtre de voisin enverra des messages « hello » aux voisins autorisés afin d’essayer d’avoir une réponse qui restaurera l’état du voisin qui n’a pas répondu. Une communication normale qui se produira à l’intérieur de l’intervalle de vérification, évitera l’envoi des « hello », qui lorsqu’ils sont envoyés, utilisent un peu de bande passante.

14.3.10 LoggerNet et les communications avec les RF4xx

Prenons cet exemple de réseau. LoggerNet communique via une RF4xx reliée au port COM d’un PC. La CR1000B est à x mètres du PC.

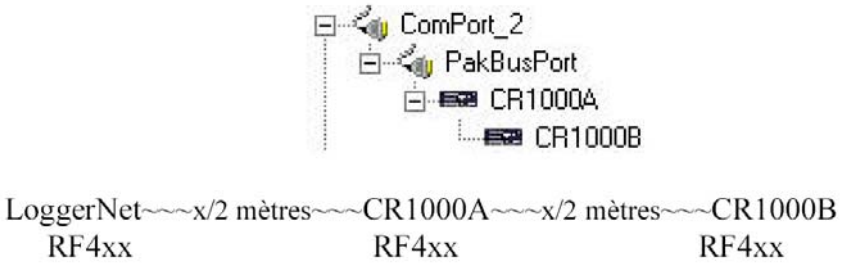


FIGURE 14.3-2. Communication en mode transparent

On suppose que l’on sait, suite à un test effectué avec ‘PakBusGraph / Ping Node’ (paragraphe 2.3), que le lien direct entre LoggerNet et la CR1000B, est peu fiable, et que la CR1000A intermédiaire est placée afin d’agir en tant que routeur. Si LoggerNet envoi des balises, la CR1000B pourra, à l’occasion, entendre les balises de LoggerNet, et y répondre avec succès, devenant alors un voisin de LoggerNet à la place du routeur. Afin d’éviter cela, on désactivera l’envoi de balises de LoggerNet et on installera une filtre de voisins dans le routeur, afin qu’il découvre la centrale distante (on configure ‘Neighbors Allowed SDC7’ et ‘Verify Interval SDC7’).

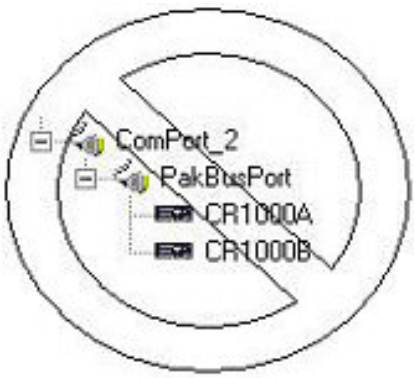


FIGURE 14.3-3 Plan de réseau rectiligne

Il peut être risqué de configurer un réseau de station en « plan de réseau rectiligne», lorsqu’on utilise un réseau de RF4xx . On utilisera au contraire un plan de réseau déployé, ce qui évite à LoggerNet de créer à partir du plan de réseau, un lien direct mais peu fiable entre LoggerNet et un nœud distant (voir ‘Static Link’ et ‘Dynamic Link’ dans le glossaire).

Avec la CR1000A mise en place de la façon mentionnée ci-dessus, LoggerNet sera capable de se connecter à la CR1000B à travers la CR1000A, pour autant que la CR1000A et la CR1000B se seront découverte l'une et l'autre. Ceci se produira en définissant un filtre de voisin dans la CR1000A, qui mentionnera l'APB de la CR1000B reconnue comme voisin.

NOTE Si vous ajoutez un routeur supplémentaire dans un réseau de RF4xx qui utilise des filtres de voisins pour la découverte du réseau, les routeurs présents dans chacun des deux nouveaux liens doivent lister les autres routeurs en tant que voisins autorisés, ou alors ils ne pourront pas devenir des voisins.

Avec des RF4xx, une autre façon de découvrir le réseau dans le cas mentionné ci-avant, est de configurer le routeur de la CR1000A afin qu'il envoie des balises (*beacon*) et qu'il découvre la CR1000B. Les balises de la CR1000A découvriraient ainsi LoggerNet (ce qui n'est pas nécessaire car LoggerNet a une route statique (*static route*) vers la CR1000A en vertu de son plan de réseau).

14.3.11 Les protocoles PakBus des RF416

Les modes 'PakBus Aware' et 'PakBus Node' ont l'avantage notable d'avoir un identifiant unique pour chaque RF416, donc il y a des accusés de réception de bas niveau pour la réception des packets de données, ainsi que des tentatives de renvoi automatiques si le packet de données n'arrive pas correctement. C'est un gros avantage lorsqu'il y a du bruit, ou qu'il y a des collisions d'ondes radio.

14.4. Editeurs de configurations

Ce tableau décrit les outils utilisés afin de configurer les paramètres de la CR1000 et du périphérique de communication dans les réseaux PakBus.

Certaines configurations s'appliquent à des ports spécifiques de la CR1000, comme par exemple le SDC7.

Editeurs de configurations pour CR1000 et périphériques de communication				
Setting (Configuration)	Possibilité de configurer avec le **Clavier/écran**	**Clients de LoggerNet**		**Device Configuration Utility**
		Status Table ConnectScreen Tool	**PakBusGraph** Settings	
PakBus Address (APB)	Oui	Oui	Oui	Oui
IsRouter	Oui	Oui	Oui	Oui
BeaconSDC7	Oui	Oui	Oui	Oui
BaudrateSDC7	Oui	Oui	Oui	Oui
Neighbors Allowed SDC7	Oui	Non	Oui	Oui
Verify Interval SDC7	Oui	Non	Oui	Oui
CentralRouters()	Non	Oui	Oui	Oui
Communication Peripheral Settings	Non	Non	Non	Oui

14.5. Résolution des problèmes sur le réseau

Symptômes	Causes Possibles	Solutions
LoggerNet ne se connecte pas à une CR1000 qui est pourtant définie dans le plan de réseau.	L'adresse PakBus dans le plan de réseau est différente de celle de la CR1000.	Changer l'une des deux adresses PakBus.
	Après que la CR1000 ait changé de configuration sur l'un de ses ports, LoggerNet continue à montrer un lien avec le port précédent de la CR1000 dans son tableau de routage.	Laisser du temps au lien précédent pour qu'il se désactive (selon la 'verification interval', sauf s'il est infini, voir paragraphe 3.8).
		Utiliser 'PakBus Graph' et effectuer un 'Reset Node' puis 'Broadcast Reset' afin de ré-initialiser le tableau de routage de LoggerNet et de la CR1000 (ré-initialiser les tableaux de routage de LoggerNet et de tous les nœuds qui sont voisins de LoggerNet).
LoggerNet ne se connecte pas à une CR1000 qui est définie, via RF4xx.	La valeur maximum de baud pour 'PakBusPort' dans LoggerNet, est différente de celle des RF4xx qui est de 9600 baud.	Changer la valeur de 'PakBusPort' pour correspondre avec la valeur maxi des RF4xx.
LoggerNet ne se connecte pas à une CR1000 qui est définie derrière un routeur, via un réseau de RF4xx.	Certaines radio ont une mauvaise configuration de *Hop*, *Net*, *Radio*, *Standby*, *Retry*, *Protocol* (RF416), ou *Active Interface*.	Configurer toutes les radios avec les même configurations de *Hop*, *Net*, *Radio*, *Standby*. Vérifier toutes les configurations.
	Le routeur n'a pas de filtre de voisin ni de paramètre de *beacon* lui permettant de devenir un voisin de la CR1000 de destination.	Activer le 'beaconing' ou installer un filtre de voisin dans le routeur afin de lister la CR1000 de destination en tant que voisin autorisé.
	La CR1000 de destination a un filtre de voisin qui interdit (ne liste pas) la communication avec le routeur.	Supprimer le filtre de voisin sur la CR1000 'leaf-node' ou bien lister le routeur dans la partie des 'neighbors allowed'.

Symptôme	Causes Possibles	Solutions
Dans un réseau de RF4xx, LoggerNet passe outre le routeur, et se connecte directement à la station en aval.	Le plan de réseau de LoggerNet est configurée en 'plan rectiligne', permettant à LN de se connecter directement aux centrales distantes via un chemin statique .	Configurer le plan de réseau de LoggerNet en forme déployée, où la station distante est liée au routeur.
Dans un réseau de RF4xx avec 2 ou 3 routeurs, LoggerNet ne se connecte pas à la CR1000.	Au moins un des routeurs n'est pas considéré en tant que voisin auprès du routeur adjacent positionné dans son chemin de liaison.	Configurer tous les filtres de voisin des routeurs afin qu'il comprennent le prochain routeur (dans les deux sens) présent sur le chemin d'un voisin potentiel.
La CR1000 ne reçoit pas de données d'une autre CR1000 avec l'instruction SendGetVariables.	SendGetVariables n'est pas programmée pour la bonne adresse PakBus de l'autre CR1000.	Corriger l'adresse PakBus.
	SendGetVariables n'est pas programmée pour utiliser le même port Com (Active Interface) que la RF4xx ou l'autre périphérique de communication attaché à la CR1000.	Changer le port Com afin qu'il concorde entre l'instruction et la configuration du périphérique attaché à la CR1000.
LoggerNet ne communique pas avec des nœuds situés derrière un routeur ou un routeur seul.	Les informations de routage du réseau dépassent la capacité mémoire de la RF416 configurée en routeur ou routeur seul.	Créer des routeurs de branche, avec les routeurs et les routeurs seuls (voir paragraphe 3.7).
Dans un réseau qui envoie des balises, un routeur seul n'a pas de routes vers les autres nœuds du réseau, comme il se devrait.		

14.6. Glossaire de la terminologie PakBus

Active Interface (Interface active)	Le port Com sélectionné sur une RF4xx ou MD485. Le choix peut être SDC, CSDC 7, CSDC 8, Modem Enabled, ou RS-232. CSDC 7 est équivalent à SDC 7.
Beacon (Balise)	Un packet transmit à tous les nœuds. Une balise est destinée à découvrir les appareils PakBus du voisinage. Les appareils qui reçoivent une balise peuvent répondre en initiant un échange de hello avec l'appareil qui envoie la balise, si ils ne sont pas déjà un voisin. Un appareil avec un filtre de réseau ignorera les balises provenant d'appareils qui n'est pas dans sa liste de voisins potentiels, sauf si l'adresse PakBus de l'appareil qui envoie la balise est ≥ 4000 (normalement LoggerNet).
Beacon Interval	Intervalle spécifié par l'utilisateur entre l'envoi des balises.
Beacon Interval SDC7	Identique à BeaconSDC7. La configuration de la CR1000 pour l'intervalle entre l'envoi des balises.
Branch Router (Routeur de branche)	Typiquement une RF416 configurée en tant que routeur seul. Un routeur de branche a une vue réduite sur le réseau et n'essaye de connaître que la liste de voisin des routeurs centraux, et des routeurs autres que lui-même présents sur le réseau.
BMP5	Block Mode Protocol 5. Le protocole de communication PakBus utilisé pour l'envoi de packets. Les hauts niveaux de packets BMP5 transmettent par exemple les programmes, les données ou les vérifications d'horloge.
Broadcast	Envoi à tous les appareils PakBus du réseau.
Buddy	Le « quasi-voisin » unique de la CR2xx. Une fois qu'elle a un « buddy », la CR2xx ignore tous les autres packets 'broadcast' tant que la vérification de la communication est effectuée. Le trafic radio est ainsi minimisé. Une CR2xx ayant un « buddy » continuera à répondre à tous les packets qui lui sont destinés.
Central Router (Routeur central)	Un routeur (par ex. une CR1000) qui est dans un réseau avec des RF416 configurées en routeur de branche ayant une vue réduite du réseau. Un routeur de branche essaye simplement de connaître la liste de voisin des routeurs centraux spécifiés, et des routeurs autres que lui-même présents entre lui et les routeurs centraux.
Client	En architecture client/serveur, un client est typiquement une application qui est lancée sur un PC qui compte sur un serveur afin d'utiliser ses ressources dans certains cas. Un exemple PakBus : Setup, ConnectScreen, et PakBusGraph sont des clients du serveur LoggerNet.
Comms	Abréviation pour communications.
Communications Verification Interval	L'intervalle de temps que l'appareil PakBus utilise afin de déterminer s'il est temps d'envoyer un message hello pour confirmer que le voisin peut encore répondre (et maintenir le voisinage). Les intervalles de vérifications se déroulent avec des échanges de hello.

Concurrent Communications	La capacité d'une centrale PakBus, à communiquer avec plusieurs périphériques en même temps (transferts intercalés).
CSDC	Concurrent Synchronous Device Communications. Se rapporte à une communication cadencée (en opposition à asynchrone) entre centrale de mesure et périphérique adressé, ce qui peut être intercalé avec des communications d'autres périphériques adressés (ou même des périphériques M.E.).
CS I/O	Interface I/O de Campbell Scientific. Une interface avec connecteur 9 broches qui ressemble à une RS-232 à 9 broches, mais qui a des fonctions et des configurations de broches différentes.
Directed Packet	Un packet qui est adressé à un appareil PakBus particulier ; on parle aussi de « communications adressée ».
Dynamic Link (Lien dynamique)	Un lien confirmé entre deux appareils PakBus une fois qu'ils sont établi comme des voisins et qu'ils sont en cours de vérifications d'intervalle de communication. Les routes statiques, au contraire, sont supposées fonctionner, mais de façon non certaine. Les liens dynamiques sont en noir sur le 'PakBusGraph'.
Flat Map (Plan rectiligne)	C'est une structure de plan de configuration de LoggerNet où tous les appareils PakBus sont des "enfants" du PakBusPort (contrairement à la forme déployée ou chaque appareil est un enfant d'un appareil auquel il est rattaché). Une structure rectiligne est intéressante lorsque tous les appareils PakBus sont à proximité et qu'ils peuvent être des voisins fiables pour LoggerNet ou d'autres routeurs.
Header (En-tête)	La partie débutant un packet PakBus, et qui contient des information telles que les voisins, la source, la destination, et l'état du lien. Pour plus d'information voir http://www.campbellsci.com et Support/Manuals.
Hello-exchange (Echange de hello)	L'envoi d'un packet de commande hello par un appareil PakBus, vers un autre appareil, et la réception de la réponse du hello depuis cet appareil. Seul un « hello-exchange » peut établir deux appareils PakBus en tant que voisins.
Hello Message (Message hello)	Un packet envoyé à un voisin potentiel ou un appareil qui envoie une balise afin d'établir un lien de voisinage.
Hop	Un lien vers un voisin.
Hop Metric	Une mesure (résultant d'un 'hello-exchange') qui indique le temps de réponse maximum attendu pour un certain nœud. Une RF416 configurée pour utiliser un protocole PakBus, indique le 'Standby Mode' comme valeur de 'hop metric' du lien.
Hopping Sequence	La configuration de la RF4xx's pour le saut de fréquence (0 – 6) de la radio à étalement de spectre.

Leaf Node	Un appareil PakBus qui n'est pas un routeur, bien qu'il puisse en avoir la capacité s'il était configuré pour cela.
ID	Identifiant, autre façon de parler d'adresse PakBus.
Link	Un chemin de communication directe (1-hop) entre deux appareils PakBus.
LogView	Logiciel indépendant permettant de visualiser et de décoder les packets PakBus dans un fichier de log bas niveau provenant de LoggerNet.
M.E.	Modem Enabled.
Modem Enabled	Périphérique qui est actif lorsque la ligne 'modem enable' (ligne 5) du port CS I/O passe à l'état haut.
Neighbor (Voisin)	Pour un appareil PakBus en particulier, un voisin est un appareil avec qui il a récemment communiqué de façon directe (pas via un routeur) par le biais de communications normales ou suite à 'hello-exchange'.
Neighbors Allowed (Voisins autorisés)	APB listées dans le filtre de voisins d'un appareil, et avec lequel l'appareil acceptera de répondre aux 'hellos' et d'établir un lien de voisin. Un appareil PakBus répond aux messages de sa liste de voisins autorisés. Identique à 'potential neighbors.'
Neighbor List (Liste de voisin)	La liste d'appareils PakBus voisins que l'appareil met à jour. Les 'leaf nodes' n'ont qu'un seul voisin. Les nœuds de routeur peuvent avoir beaucoup de voisins. Les appareils routeur s'échangent leur liste de voisins (et tous les changements nécessaires) avec les autres routeurs du réseau afin de construire et de mettre à jour le système de routage.
Neighbor Filter (Filtre de voisin)	C'est un moyen de découverte du réseau. C'est un groupe de configuration qui contrôle avec quel(s) appareil(s) PakBus une centrale de mesure essayera d'établir un lien de voisinage. Un filtre de voisin initie des 'hello-exchanges' vers les appareils de sa liste de '*neighbors allowed*'. Ainsi on filtre les paquets du réseau dont les adresses ne sont pas sur la liste autorisée, à moins que l'APB soit ≥ 4000 (l'adresse par défaut de LoggerNet est 4094).
Network Address (Adresse réseau)	Adresse réseau des RF4xx. Sur un réseau PakBus toutes les RF4xxs doivent avoir la même adresse réseau.
Node (Nœud)	Autre terme pour 'PakBus device'. Une station avec une centrale de mesure PakBus peut être appelée un appareil PakBus ou un nœud. Une RF416 configurée en routeur seul (stand-alone router) est aussi un nœud.
Packet	Une architecture, en général de 1000 octets ou moins, contenant un en-tête et des données. Les paquets transfèrent l'information entre les centrales de mesure PakBus et LoggerNet ou entre les centrales de mesure, souvent via des routeurs.
PakBus Address (Adresse PakBus)	Entier unique à 4 chiffres, assigné à un appareil PakBus dans un réseau PakBus (entre 1 et 4094).

PakBus Device (Appareil PakBus)	C'est un appareil qui peut recevoir et créer des paquets PakBus (BMP5). C'est un synonyme de nœud.
PakBusGraph	C'est un client de LoggerNet qui représente de façon graphique les appareils PakBus du réseau et qui permet de visualiser et d'éditer certains paramètres tells que le tableau de routage, la configuration des ports ou les voisins autorisés.
PakCtrl	Protocole de communication réseau au niveau de PakBus. Les paquets 'PakCtrl' de bas niveau effectuent les fonctions de découverte et de routage telles que l'envoi de « hello-exchanges » et la transmission des listes de voisins.
APB	Abréviation pour Adresse PakBus.
Port	Port Com. Une des interfaces de la centrale de mesure ou du périphérique : SDC, CSDC, Modem Enabled, RS-232, C1/C2, C3/C4, C5/C6 ou C7/C8.
Potential Neighbors (Voisins potentiels)	Les voisins potentiels sont aussi appelés « voisins autorisés ». Les adresses PakBus listées dans le filtre de voisins d'un appareil, auxquelles l'appareil répondra aux tentatives d'établissement de communication et aux messages 'hello'. Les réponses ne sont envoyées qu'aux messages provenant des voisins autorisés ou dont l'APB est ≥ 4000.
Potential Neighbor List (Liste de voisins potentiels)	Une liste d'appareils (adresses PakBus) comprise dans un filtre de voisin, avec lesquels la centrale de mesure tentera de devenir un voisin. Les appareils non listés sont exclus, sauf si leur adresse PakBus est ≥ 4000 (généralement le cas de LoggerNet). Les voisins potentiels sont aussi appelés des voisins autorisés.
Radio Address (Adresse radio)	Les adresses radio des RF4xx. Sur un réseau PakBus transparent, toutes les RF4xxs seront configurées avec la même adresse radio. En mode 'PakBus Aware' ou 'PakBus Mode', l'adresse radio n'est pas utilisée.
Represented (Représenté)	Ce terme s'applique aux appareils qui sont montrés sur un plan de réseau de LoggerNet. Certains appareils de communication telles les RF4xx, n'ont pas besoin d'être représentées sur le plan de réseau.
RF	Fréquence radio (Radio Frequency). Terme qui se rapporte aux communications sans fil (émission / réception).
Route	C'est un chemin de communication qui conduit à un nœud. Il peut y avoir plusieurs 'hops' jusqu'au nœud.
Router (Routeur)	Un appareil PakBus configure de telle sorte qu'il peut faire transiter des packets qui ne lui sont pas destinés, vers l'appareil concerné. Le routeur n'accepte de transmettre des packets, qu'à destination d'appareils dont il connaît le chemin d'accès. La CR1000, la CR23X avec OS PakBus, le NL100, une RF416 routeur seul ou LoggerNet peuvent être des routeurs. Les CR2xx ne peuvent pas être des routeurs.

Routing Table (Tableau de routage)	Une liste des itinéraires d'un routeur (les voisins conduisant à la destination) pour tous les appareils PakBus dont il a connaissance sur le réseau. Un tableau de routage est créé en utilisant un algorithme de « meilleur chemin » provenant des listes de voisins des autres routeurs.
SDC	Synchronous Device Communications – Communications série cadencées, entre centrale de mesure et périphérique adressé. Un périphérique de centrale de mesure peut utiliser ce mode adressé au lieu d'utiliser la ligne 'modem enable'.
Server (Serveur)	En architecture client/serveur, un serveur est typiquement un ordinateur ou un appareil qui s'occupe (donne accès) de ressources réseau. En réseau PakBus, le serveur LoggerNet dispose de routes ou de routeurs afin de communiquer avec d'autres appareils PakBus via des interfaces PakBus. Il communique via des transactions sur l'interface LoggerNet, avec des clients tels que 'PakBusGraph' et 'ConnectScreen'.
Static Link (Lien statique)	C'est un lien ou une route qui existe en vertu de la configuration du plan de réseau de LoggerNet ou d'une instruction de programme. Les routes *Dynamiques* sont celles qui ont été vérifiées (et qui sont à jour en terme de durée d'intervalle de vérification). Sur 'PakBusGraph', un lien statique est coloré en rouge alors qu'un lien vérifié sera (deviendra) en noir
Transaction	Un échange de packets entre des nœuds. Un 'hello-exchange' en est un exemple. La plupart des packets viennent par paire de commande / réponse.
Tree Map (Plan déployée)	Plan de configuration de LoggerNet où les appareils PakBus sont ajoutés en tant que « fils » de l'appareil précédent. Si les filtres de voisins sont corrects, un plan de voisins peut forcer un certain chemin à être suivi à destination d'un appareil PakBus particulier.
Verification Interval (Intervalle de vérification)	C'est l'intervalle de vérification de la communication ("communication verification interval"). La configuration qu'un appareil PakBus utilise afin de déterminer lorsqu'il est temps (depuis la dernière communication) d'envoyer un message 'hello' afin de confirmer que le voisin peut toujours répondre (et qu'il reste un voisin). Les intervalles de vérification se déroulent conne des 'hello-exchanges'.
Verify Interval xxx	Identique à 'Verify xxx'. C'est la configuration du port d'une CR1000 pour l'intervalle de vérification de la communication pour la centrale de mesure.

Pour un traitement plus en profondeur, des paquets de données et de communication PakBus, "PakBus Networking Guide" est disponible au format pdf sur http://www.campbellsci.com rubrique Support/Manuals.

Annexe A. Tableau d'état de la CR1000 (Table d'état)

Le tableau d'état de la CR1000 contient les informations actuelles de l'état de fonctionnement de la CR1000. Ces informations peuvent être visualisées par un PC, ou peuvent être lues par des instructions du programme de la CR1000. Il est aussi possible de voir ces informations à partir du clavier/écran. Le tableau 1 montre les variables du tableau d'état et une courte description de leurs fonctions.

Status Fieldname (Nom des variables)	Description	Type de Variable	Défaut	Valeurs Normales	Modifiable par l'utilisateur ?	Type d'info.
FileMark	Une valeur créée par le logiciel PC		_			
RecNum	Le numéro d'enregistrement de ce jeu de données		–	–	–	
TimeStamp	L'heure à laquelle l'enregistrement a été généré	Time	_	_	_	
OSVersion	La version du système d'exploitation (Operating System – OS)	Chaîne de caractères	–	–	–	Etat
OSDate	Date à laquelle l'OS a été réalisé	Chaîne de caractères	–	–	–	Etat
OSSignature	Signature de l'OS	Entier	_	_	_	Etat
SerialNumber	Numéro de série spécifique à la machine. Il est stocké en mémoire FLASH.	Entier	_	_	_	Etat
RevBoard	Numéro de révision du matériel (Hardware). Il est stocké en mémoire FLASH.	Entier	_	_	_	Etat
StationName[1]	Nom de la machine. Il est stocké en mémoire FLASH.	Chaîne de caractères	–		Oui	Config.
PakBusAddress[2]	Adresse PakBus de la centrale.	Entier	1	De 1 à 3999	Oui	Config. PB
ProgName	Nom du programme en cours d'exécution.	Chaîne de caractères	–	–	–	Etat
StartTime	Heure à laquelle le programme a été lancé initialement.	Time	_	_	_	Etat
RunSignature	Signature de la structure binaire compilée du programme exécuté. Cette valeur ne prend pas en compte les commentaires ajoutés en fin de ligne.	Entier	_	_	_	Etat
ProgSignature	Signature du programme actuelle *.CR1 ; tous les caractères sont pris en compte.	Entier	_	_	_	Etat
Battery	Valeur actuelle de la tension de la batterie. Elle est prise en tâche de fond en même temps que l'étalonnage.	A virgule flottante	–	De 9,6 à 16 Volts	–	Mesure
PanelTemp	Valeur actuelle de la température du bornier. Elle est prise en tâche de fond en même temps que l'étalonnage.	A virgule flottante	–	–	–	Mesure

Status Fieldname (Nom des variables)	**Description**	**Type de Variable**	**Défaut**	**Valeurs Normales**	**Modifiable par l'utilisateur ?**	**Type d'info.**
WatchdogErreurs[3]	Le nombre d'erreurs 'Watchdog' qui se sont produites depuis que ce programme est activé.	Entier	0	0	Pour ré-initialiser, entrer '0'	Erreur
LithiumBattery[4]	Valeur actuelle de la tension de la pile au Lithium. Elle est prise en tâche de fond en même temps que l'étalonnage.	A virgule flottante	–	De 2,7 à 3,6 Volts	–	Mesure
Low12VCount[5]	Garde trace du nombre de fois où le signal de 12V faible a été atteint. Lorsque ces conditions sont détectées, la centrale de mesure cesse d'effectuer des mesures et passe en mode veille jusqu'à ce que la tension retrouve un niveau satisfaisant.	Entier	0	0	Pour ré-initialiser, entrer '0'	Erreur
Low5VCount	Garde trace du nombre de fois où le signal de 5V faible a été atteint.	Entier	0	0	Pour ré-initialiser, entrer '0'	Erreur
CompileResults	Contient les messages d'erreur qui ont été générés par la compilation ou durant l'exécution du programme.	Chaîne de caractères	–	0	–	Erreur
StartUpCode[6]	Un code variable qui permet à l'utilisateur de savoir comment le système s'est mis en fonctionnement à la mise sous tension.	Chaîne de caractères	0	0	–	Etat / Erreur
ProgErreurs	Le nombre d'erreurs de compilation (ou d'exécution) du programme actuel.	Entier	_	0	_	Erreur
VarOutOfBound[7]	Nombre de fois qu'une variable de type 'ligne de données' (*array*) a été utilisée en dehors de sa taille pré-définie.	Entier	0	0	Pour ré-initialiser, entrer '0'	Erreur
SkippedScan	Nombre de fois que la scrutation (*scan*) a été manquée, depuis que le programme est en fonctionnement.	Entier	0	–	Pour ré-initialiser, entrer '0'	Erreur
SkippedSlowScan[8]	Nombre de fois que la scrutation (*scan*) a été manquée en « *slow sequence* ». Si l'utilisateur gère plusieurs scrutations lentes, cette variable devient une ligne de donnée ayant une place pour chaque scrutation lente.	Entier, ligne de données	0	–	Pour ré-initialiser, entrer '0'	Erreur
SkippedRecord[9]	Variable de ligne de donnée qui indique combien d'enregistrements ont été manqué sur chaque tableau de sauvegarde. Chaque tableau a une place dans cette variable.	Entier, ligne de données	0	0	Pour ré-initialiser, entrer '0'	Erreur
ErreurCalib[8]	Un compteur qui est incrémenté à chaque fois qu'une mauvaise valeur de l'étalonnage est mesurée. Cette valeur n'est alors pas prise en compte (non incluse dans la mise à jour du filtre) et la valeur ErreurCalib est incrémentée.	Entier	0	0	_	Erreur

Status Fieldname (Nom des variables)	**Description**	**Type de Variable**	**Défaut**	**Valeurs Normales**	**Modifiable par l'utilisateur ?**	**Type d'info.**
MemorySize	Quantité totale de SRAM (octets) sur cet appareil.	–	–	2097152 (2M) 4194304 (4M)	–	Etat
MemoryFree	Nombre (d'octets) non alloués sur la CPU (SRAM). L'utilisateur ne pourra peut être pas allouer toute la mémoire disponible, pour le stockage des données, car les tableaux de sauvegarde doivent être contigus. Lorsque la mémoire est allouée puis libérée, il peut y avoir des trous qui ne sont pas utilisables pour le stockage de données, mais qui seront affichés comme des octets disponibles (*free bytes*).	Entier	–	–	–	Etat
ProgMemFree	Quantité d'espace libre du CPU, utilisée pour le stockage des fichiers de programmes (ramdisk).	Entier		De 0 à 95232	_	Etat
CommsMemFree	Ligne de données avec 2 valeurs. La première est le nombre d'octets libres, la seconde est le nombre de petits blocs disponibles.	Entier, ligne de données (2 val.)	–	(1) de 2000 à 15000	–	Etat
FullMemReset	Une valeur de 98765 écrite à cet endroit engendrera une ré-initialisation complète de la mémoire. Cela ré-initialisera le RAM Disk, le stockage des données, la mémoire PakBus, et re-mettre en place les valeurs par défaut.	Entier	0	_	Entrer 98765 pour ré-initialiser	Config.
DataRecordSize	Nombre d'enregistrements dans le tableau. Chaque tableau à sa propre valeur dans cette ligne de donnée.	–	–	–	–	
SecsPerRecord	Intervalle de sauvegarde donné pour le tableau. Chaque tableau à sa propre valeur dans cette ligne de donnée.	–	–	–	–	
DataFillDays	Durée en jours afin de remplir le tableau. Chaque tableau à sa propre valeur dans cette ligne de donnée.	–	–	–	–	
CardEtat	Contient une chaîne de caractères avec les informations les plus récentes sur l'état de la carte.	Chaîne de caractères	–	–	–	Etat
CardBytesFree[10]	Donne les octets disponibles sur la carte.	Entier	–	–	–	Etat
MesureOps	C'est le nombre « d'opcodes » nécessaires au séquenceur de tâche afin d'effectuer toutes les mesures sur le système. Il inclut les opcodes de l'étalonnage (temps de compilation) et du système « slow sequence ».	Entier	–	–	–	Etat

Status Fieldname (Nom des variables)	**Description**	**Type de Variable**	**Défaut**	**Valeurs Normales**	**Modifiable par l'utilisateur ?**	**Type d'info.**
MesureTime	Le temps en microsecondes nécessaire au matériel pour effectuer les mesures dans la scrutation. La somme du temps de mesure + stabilisation. Le temps de traitement se fait en parallèle, donc la somme des temps de mesure et de traitement ne sera pas égale au temps nécessaire à chaque scrutation.	Entier	_	_	_	Etat
ProcessTime	Temps en microsecondes, pris pour effectuer le traitement sur le scan précédent. Ce temps est mesuré entre la fin de l'instruction 'EndScan' (une fois que l'évènement de mesure est activé) et le début de 'EndScan' (avant l'attente du début de l'événement de mesure) pour la scrutation (scan) suivante.	Entier	_	_	_	Etat
MaxProcTime	Le temps maximum en microsecondes, nécessaire pour effectuer le traitement dans le scan. Cette valeur est ré-initialisée lorsqu'on sort du scan.	Entier	_	_	Pour ré-initialiser, entrer '0'	Etat
LastSlowScan	La dernière fois que la scrutation lente s'est exécutée. S'il y a plusieurs scrutations lentes, cette variable devient une ligne de données.	Entier, ligne de données	–	–	_	Etat
SlowProcTime[11]	Temps en microsecondes, nécessaire pour traiter le scan en scrutation lente. S'il y a plusieurs scrutations lentes cette variable devient une ligne de données.	Entier, ligne de données	–	–	_	Etat
MaxSlowProcTime[12]	Temps maximum en microsecondes, nécessaire pour traiter le scan en scrutation lente. S'il y a plusieurs scrutations lentes cette variable devient une ligne de données.	Entier, ligne de données	–	–	_	Etat
PortEtat	Ligne de donnée de booléens qui donnent l'état des ports de contrôle. Les valeurs mise à jour toutes les 500mS.	Ligne de donnée de 8 Booléen	False (faux)	True / False (vrai ou faux)	Oui	Etat
PortConfig	Ligne de donnée de chaîne de caractères décrivant l'utilisation du port de contrôle associé. Les entrées valides sont : Entrée, Sortie, SDM, SDI-12, Tx, et Rx.	Ligne de données à 8 valeurs en chaîne de caract.	Entrée	Input / Output (entrée ou sortie)	–	Etat
Security[13]	Une ligne de donnée avec les 3 configurations de sécurité (qui sont masqués si la sécurité est activée).	Ligne de données de 3 entiers	0, 0, 0	De 0 à 65535 (0 = pas de sécurité)	Oui	Etat

Status Fieldname (Nom des variables)	**Description**	**Type de Variable**	**Défaut**	**Valeurs Normales**	**Modifiable par l'utilisateur ?**	**Type d'info.**
CommsActive[14]	Ligne de données de booléen indiquant si la communication est actuellement activée sur le port correspondant. Les alias sont pour CommActiveRS232, CommActiveME, CommActiveCOM310, CommActiveSDC7, CommActiveSDC8, CommActiveCOM1, CommActiveCOM2, CommActiveCOM3, et CommActiveCOM4	Ligne de données de 9 valeurs de booléens	Faux, excepté pour le port COM actif	True / False (vrai ou faux)	–	Etat
CommsConfig	Ligne de donnée de valeurs donnant le configuration des ports com. Les alias sont pour CommConfigRS232, CommConfigME, CommConfigCOM310, CommConfigSDC7, CommConfigSDC8, CommConfigCOM1, CommConfigCOM2, CommConfigCOM3, et CommConfigCOM4	Ligne de données de 9 valeurs d'entiers	De RS232 à SDC8 = 4 / De COM1 à COM4 = 0	0 ou 4	–	Config.
Baudrate[15]	Ligne de données de vitesses en baud pour les liens de communication. Les alias sont pour BaudrateRS232, BaudrateME, BaudrateCOM310, BaudrateSDC7, BaudrateSDC8, BaudrateCOM1, BaudrateCOM2, BaudrateCOM3, BaudrateCOM4	Ligne de donnée de 9 entiers	RS232=-115200 De ME à SDC8 = 115200 De COM1 à 4 = 0	1200, 2400, 4800, 9600, 19.2k, 38.4k, 57.6k, 115.2k	Oui, on peut aussi utiliser l'instruction 'SerialOut' pour effectuer la configuration	Config.
IsRouter	Pour configurer la CR1000 en tant que routeur ou non.	Booléen	Faux	0 ou 1	Oui	Config. PB
PakBusNodes	Nombre de nœuds (approximatif) qui existeront dans le réseau PakBus. Cette valeur est utilisée afin de déterminer combien de mémoire allouer pour le réseau.	Entier	50	>=50	Oui	Config. PB
CentralRouters(1) - (8)[16]	Ligne de donnée de (8) adresses PakBus pour des routeurs centraux.	Ligne de donnée de 8 entiers	0	–	Oui	Config. PB
Beacon (Beacon Interval)	Ligne de donnée d'intervalles des balises (en secondes) pour les ports Com. Les Alias sont : BeaconRS232, BeaconME, BeaconCOM310, BeaconSDC7, BeaconSDC8, BeaconCOM1, BeaconCOM2, BeaconCOM3, BeaconCOM4	Ligne de donnée de 9 entiers	0	De 0 à approx. 65 500	Oui	Config. PB

Status Fieldname (Nom des variables)	Description	Type de Variable	Défaut	Valeurs Normales	Modifiable par l'utilisateur ?	Type d'info.
Verify	Ligne de vérification d'intervalles (en secondes) pour les ports COM. Tels que VerifyRS232, VerifyME, VerifyCOM310, VerifySDC7, VerifySDC8, VerifyCOM1, VerifyCOM2, VerifyCOM3, VerifyCOM4	Ligne de donnée de 9 entiers	0	De 0 à approx. 65 500	_	Etat
MaxPacketSize	Nombre maximum d'octets par packet de données collecté	_	1000	_	_	
Messages	Contient une chaîne de caractères pour un message qui peut être entré par l'utilisateur.	Chaîne de caractères	–		Oui	
CalGain[17]	Tableau de calibration pour les valeurs de Gain. Chaque combinaison d'étendue de mesure / code d'intégration, a un gain qui lui est associé. Si le programme le nécessite, ces valeurs sont mises à jour en tâche de fond par une scrutation lente.	Ligne de donnée de 18 valeurs à virgule flottante	_	.	_	Calib.
CalSeOffSet[17]	Tableau de l'étalonnage pour les valeurs d'offset unipolaires. Chaque combinaison d'étendue de mesure / code d'intégration, a un gain qui lui est associé. Si le programme le nécessite, ces valeurs sont mises à jour en tâche de fond par une scrutation lente.	Ligne de donnée de 18 entiers	–	Proche de 0	–	Calib.
CalDiffOffset[17]	Tableau de l'étalonnage pour les valeurs d'offset différentielles. Chaque combinaison d'étendue de mesure / code d'intégration, a un gain qui lui est associé. Si le programme le nécessite, ces valeurs sont mises à jour en tâche de fond par une scrutation lente.	Ligne de donnée de 18 entiers	–	Proche de 0	–	Calib.

1 L'instruction StationName peut aussi être utilisée dans un programme afin d'écrire sur ce champ.

2 Les adresses PakBus entre 1 et 4094 sont valides. Les adresses >= 4000 sont généralement utilisées par le PC utilisant PC200W, PC400, ou LoggerNet.

3 Les erreurs de chien de garde (*Watchdog Errors*) sont automatiquement ré-initialisées lorsqu'on compile un nouveau programme.

4 Il faut remplacer la pile au lithium si elle est à moins de 2.7V. Voir le paragraphe 1.10.2 pour les instructions à suivre.

5 Les comparateurs de 12V faible ont certaines variantes, mais généralement ils changent d'état à environ 9,0V. La tension minimum spécifiée en entrée de 9,6 V ne causera pas un comptage de 12 V faible, mais une condition de 12V faible stoppera l'exécution du programme avant que la CR1000 ne donne de mauvaises mesures dues aux conditions de faible tension d'alimentation

6 Pas encore utilisé (12/1/2004)

7 L'erreur de variable "out of Bounds" se produit lorsqu'un programme essaye d'écrire dans une variable ligne de données, en dehors de sa taille déclarée. C'est une erreur de programmation qui cause cela, et cette erreur ne devrait pas être ignorée. Lorsque le centrale de mesure détecte qu'on essaye d'écrire en dehors des limites d'une variable, elle n'exécute pas l'ordre d'écrire mais elle incrémente alors la valeur de VOOB dans le tableau d'état. Le compilateur et le pré-compilateur ne peuvent remarquer que des choses telles qu'un nombre de répétitions trop important pour la taille d'une variable. Si une ligne de donnée est utilisée dans une boucle ou une expression, le pré-compilateur ne vérifie pas (dans la plupart des cas il ne peut pas vérifier) si une variable tentera d'être utilisée en dehors de ses limites définies (par exemple accéder à une ligne de donnée avec une variable indexée par une expression telle que arr(index) = arr(index-1), où 'index' est une variable).

8 L'étalonnage automatique est activé automatiquement dans une scrutation lente (voir paragraphe 3.8.)

9 L'ordre des tableaux est l'ordre dans lequel ils ont été déclarés.

10 Card bytes free = -1 lorsqu'aucune carte n'est présente.

11 Affiche de grands nombres jusqu'à ce qu'une scrutation lente 'SlowScan' ne s'exécute.

12 MaxSlowProcTime affiche 0 jusqu'à ce que 'SlowScan' s'exécute.

13 La sécurité peut être modifiée via DeviceConfig, CR1000KD, PBGraph, EtatTable, et l'instruction SetSecurity. Il est affiché '-1' si le code de sécurité n'a pas été donné / désactivé.

14 Lorsque l'instruction 'SerialOpen' est utilisée, CommsConfig est rempli avec le paramètre de format de cette instruction. Au (11/2004), la seule option de format disponible est 0 = Pas d'erreur de vérification. La communication PakBus peut se dérouler de façon concomitante sur le même port si le port a été ouvert auparavant (dans le cas du CP UARTS) pour PakBus, ou si le port est déjà ouvert (CS-9pin, et RS232) pour PakBus le code sera 4.

15 La valeur affichée est la vitesse initiale en baud, que la CR1000 utilisera. Une valeur négative permettra à la CR1000 d'utiliser une vitesse en baud automatique mais spécifiera à quelle vitesse débuter.

16 Une liste allant jusqu'à 8 adresses PakBus pour des routeurs qui peuvent agir en tant que routeurs centraux. Voir l'utilitaire DeviceConfig pour plus d'information à ce sujet.

17 (1) Etendue de mesure 5000 mV , étendue d'intégration 250 uS
(2) Etendue de mesure 2500 mV, étendue d'intégration 250 uS
(3) Etendue de mesure 250 mV , étendue d'intégration 250 uS
(4) Etendue de mesure 25 mV , étendue d'intégration 250 uS
(5) Etendue de mesure 7,5 mV , étendue d'intégration 250 uS
(6) Etendue de mesure 2,5 mV , étendue d'intégration 250 uS
(7) Etendue de mesure 5000 mV, étendue d'intégration 1/60 Hz
(8) Etendue de mesure 2500 mV, étendue d'intégration 1/60 Hz
(9) Etendue de mesure 250 mV , étendue d'intégration 1/60 Hz
(10) Etendue de mesure 25 mV , étendue d'intégration 1/60 Hz
(11) Etendue de mesure 7,5 mV, étendue d'intégration 1/60 Hz
(12) Etendue de mesure 2,5 mV, étendue d'intégration 1/60 Hz
(13) Etendue de mesure 5000 mV , étendue d'intégration 1/50 Hz
(14) Etendue de mesure 2500 mV , étendue d'intégration 1/50 Hz
(15) Etendue de mesure 250 mV , étendue d'intégration 1/50 Hz
(16) Etendue de mesure 25 mV , étendue d'intégration 1/50 Hz
(17) Etendue de mesure 7,5 mV , étendue d'intégration 1/50 Hz
(18) Etendue de mesure 2,5 mV , étendue d'intégration 1/50 Hz

CAMPBELL SCIENTIFIC COMPANIES

Campbell Scientific, Inc. (CSI)
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com
info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.csafrica.co.za
sales@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 444
Thuringowa Central
QLD 4812 AUSTRALIA
www.campbellsci.com.au
info@campbellsci.com.au

Campbell Scientific do Brazil Ltda. (CSB)
Rua Luisa Crapsi Orsi, 15 Butantã
CEP: 005543-000 São Paulo SP BRAZIL
www.campbellsci.com.br
suporte@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
11564 - 149th Street NW
Edmonton, Alberta T5M 1W7
CANADA
www.campbellsci.ca
dataloggers@campbellsci.ca

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk
sales@campbellsci.co.uk

Campbell Scientific Ltd. (France)
Miniparc du Verger - Bat. H
1, rue de Terre Neuve - Les Ulis
91967 COURTABOEUF CEDEX
FRANCE
www.campbellsci.fr
info@campbellsci.fr

Campbell Scientific Spain, S. L.
Psg. Font 14, local 8
08013 Barcelona
SPAIN
www.campbellsci.es
info@campbellsci.es

Campbell Scientific Ltd. (Germany)
Fahrenheitstrasse13, D-28359 Bremen
GERMANY
www.campbellsci.de
info@campbellsci.de

Please visit www.campbellsci.com to obtain contact information for your local US or International representative.