Centrales de mesure de la série CR200
Manuel d'utilisation

Issued 01.03.2007
Traduction du 01/01/2008

Garantie

Cet équipement est garanti contre tout vice de matériau et de façon.

Cette garantie demeurera en vigueur pendant une période de douze 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'il soient renvoyés port payé, à notre Usine en Angleterre après diagnostique avec le support technique. 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,
1, rue de Terre Neuve
Miniparc du Verger
Bât. H - Les Ulis
91967 COURTABOEUF CEDEX, FRANCE
Tél. : (+33) 1 69 29 96 77
Fax : (+33) 1 69 29 96 65
Email : info@campbellsci.fr
http://www.campbellsci.fr/

Sommaire

Figure

Tableau

Aperçu de la CR200

La CR200 permet d'avoir une capacité de mesures fiables à moindre coût, avec un matériel robuste et alimenté par batterie. La CR200 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 CR206, CR211 et CR216, combinent la centrale de mesure CR200 ainsi qu'un émetteur radio pour la transmission de données. Les différents modèles ont des bandes de fréquence spécifiques :

CR206	*915 MHz*	*(anciennement CR205)*	*U.S.A. / Canada*
CR211	*922 MHz*	*(anciennement CR210)*	*Australie / Israël*
CR216	*2,4GHz*	*(anciennement CR215)*	*Europe / Le reste du monde*

La plupart des informations contenues dans ce manuel sont valables pour toutes les centrales de la série CR200. Quand on utilisera les termes CR200 ou CR2XX, les caractéristiques devront être appliqués aux différents modèles. Certaines données sont spécifiques à un type de centrale, et cela est alors spécifié dans le texte. C'est le cas des consommations de courant qui varient en fonction du modèle ; les chiffres donnés dans ce manuel sont valables pour les modèles ***CR200 / CR216****.*

La CR295 comprend un port série supplémentaire afin de communiquer avec le transmetteur pour satellite GOES (le TX312), voir annexe B.

Figure OV1-1 Centrale de mesure CR206 avec radio à 900MHz.

NOTE: Merci de vous référer au manuel associé, celui de la RF416, afin de trouver les détails sur l'utilisation de la CR216 – plus particulièrement en relation avec les licences radio

OV1 Description Physique

La figure OV1-2 montre le bornier de la CR200 et les instructions de programme qui y sont associées. A moins que ce ne soit noté, ce sont des instructions de mesure (voir chapitre 7).

OV1.1 Mesures en entrée

OV1.1.1 Mesures analogiques (Analog Inputs)

Il y a 5 entrées unipolaires pour la mesure de tensions entre 0 et +2,5V.

La résolution est de 0,6 millivolts.

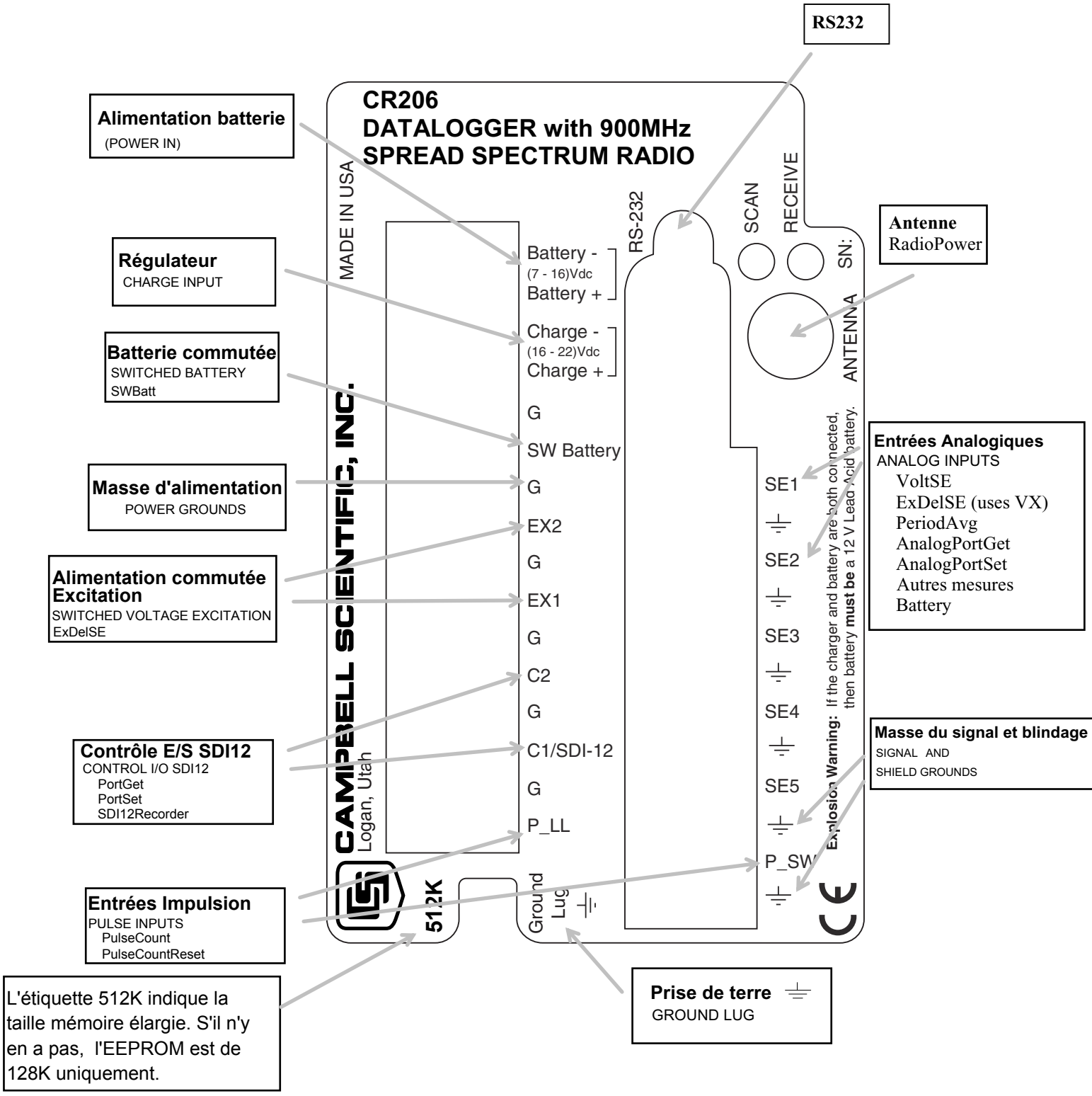


Figure OV1-2 Bornier de la CR206, et les instructions associées

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.

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

Les voies G (masse d'alimentation) sont utilisées pour relier les masses des alimentations délivrés par la voie SW-Batt ou la batterie.

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 Voies d'excitation commutée (Switched Voltage Excitation VX)

Deux sources d'excitation commutée permettent d'obtenir des tensions d'excitation précises et programmables (+2,5 et +5V) afin d'effectuer des ponts de mesure. Chaque sortie analogique fournira jusqu'à 20mA à 2,5V ou 10mA à 5V.

OV1.1.6 Entrées d'impulsion (Pulse Inputs)

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

OV1.1.7 Contrôle E/S (Control I/O)

Ces deux voies numériques d'Entrée / Sortie (0V niveau bas, 5V niveau haut) servent à effectuer des mesures de fréquence, du contrôle numérique ou de la communication SDI-12.

OV1.1.8 Entrée d'alimentation (Power In)

Les bornes Batt – et Batt + servent à relier une batterie externe à la CR200. La CR200 fonctionnera entre 7 et 16V CC. Ce sont les seules voies qui seront utilisées afin de recevoir le courant de la batterie ; la borne SW-Batt est une sortie uniquement. Le courant nécessaire à la charge d'une batterie 12V (entre 16 et 22V CC), doit être relié aux bornes Charge+ et Charge-.

OV1.1.9 Batterie commutée (Switched Battery)

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

OV1.1.10 Alimentation et adaptateur secteur (Power Supply and AC Adapter)

La CR200 n'a pas de batterie interne mais dispose de bornes de connexion pour une batterie externe et un régulateur de charge intégré afin de recharger une batterie 12V de type acide-plomb, à partir d'une source de courant externe. Le courant de charge peut provenir d'une entrée 16 à 22V CC, tel qu'un panneau solaire.

OV1.2 Communication et stockage des données

OV1.2.1 RS-232 pour ordinateur

Le port de communication est un port RS-232 qui peut être connecté directement à un ordinateur (voir figure OV1-3). Le port RS-232 de la CR200 est un port DCE. Le RS232 disponible est une version limitée, sans contrôle de flux. Le tableau OC1-1 donne une brève description des broches de la RS-232.

Le port RS-232 de la CR200 n'est pas électriquement isolé. Le branchement à un ordinateur alimenté par le secteur peut causer des boucles de courant et générer des problèmes de mesure.

La tension maximum en entrée est de +/- 25V

La tension maximum en sortie est de +/- 13V

Le signal typique en sortie est de + /- 5,4V

Tableau OV1-1 Description des broches RS-232 pour l'ordinateur

ABR = Abréviation pour le nom de la fonction
PIN = Numéro de broche (Pin n°)
O = Signal Out (en sortie) de la CR200, vers un appareil RS-232
I = Signal Into (en entrée) de la CR200, vers un appareil RS-232

PIN	ABR	I/O	Description
1			Pas de connexion
2	TX	O	Transmission asynchrone
3	RX	I	Réception asynchrone
4			Pas de connexion
5	GND		Masse (ground)
6	DSR	O	+5 V
7			Pas de connexion
8	CTS	O	Demande pour envoyer du +5V (request to send +5V)
9			Pas de connexion

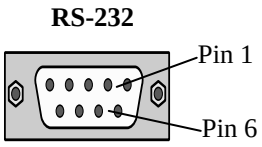


Figure OV1-3 Positionnement des broches sur la RS-232

OV1.2.2 Antenne

Plusieurs types d'antennes sont disponibles, pour répondre aux besoins des stations de base et des stations distantes. Ces antennes ont été testées sur site et sont conformes aux limites d'émission autorisées. Toutes les antennes (ou les câbles des antennes) ont un connecteur SMA femelle afin de communiquer avec la CR200. Le fait d'utiliser une antenne non autorisée pourrait augmenter la force du signal, ce qui le ferait être en désaccord avec les règles locale, et qui pourrait générer une sanction envers l'utilisateur. Les CR215 / CR216 utilisent des antennes à 2,4 GHz.

NOTE: Il est nécessaire de connecter une des antennes autorisées à la CR215/216. Vous devez en choisir une d'entre elles.

2.4-ANT1 – Antenne pour utilisation en intérieur uniquement, 2.4 GHz omni ½ onde, Whip 0dBd

2.4-ANT2 – Antenne résistante à l'eau montée sur un coffret de Campbell Scientific, 2.4 GHz 2.5dBi

2.4-ANT3 – Antenne pour fenêtre, avec 3m de câble

Figures OV1-4 Trois antennes typiques, approuvées pour leur fonctionnement avec les CR216

A noter : Pour être en conformité avec les règles Européennes, le gain total de l'antenne (moins la perte due au câble) ne doit par excéder +3dB.

OV2 Concept de la mémoire et de la programmation

OV2.1 Mémoire

La CR200 a 2K de SRAM qui sert de mémoire tampon pour la communication, les calculs et les variables, 60K de flash EEPROM pour le système d'exploitation et le programme de l'utilisateur. La CR200 avait auparavant 128K de flash EEPROM pour le stockage des données ; elle en a actuellement 512k (lorsque cela est mentionné sur une étiquette présente sur le bornier).

Figures OV2-1 L'étiquette présente sur le CR206 indique qu'elle a une mémoire de 512k.

OV2.2 Mesures, traitement et stockage des données

La CR200 est souvent utilisée pour fournir une interface sans fil entre un capteur et une CR10X-Pakbus ou une autre centrale de mesure qui traite et stocke les résultats. Dans ce cas la CR200 effectue la mesure et envoie les résultats à une centrale de mesure maître selon un intervalle de temps défini par le programme de la centrale de mesure maître.

La CR200 peut aussi fonctionner en tant que centrale de mesure standard, en enregistrant les données sous forme de tableau, dans sa mémoire EEPROM.

OV2.3 Tableaux de données (Data Tables)

La CR200 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.

Tableau OV2-1 Exemple typique de tableau de données

TOA5	1	CR2XX		v1.0	EXPLS4.CR2	45828	AvgTemp
TMSTAMP	RECNBR	SoilT_Avg(1)	SoilT_Avg(2)	SoilT_Avg(3)	SoilT_Avg(4)		
TS	RN	DegC	DegC	DegC	DegC		
		Avg	Avg	Avg	Avg		
3/20/02 11:00	1	15.498	15.9926	18.516	19.5019		
3/20/02 12:00	2	15.4996	15.9993	18.5069	19.502		
3/20/02 13:00	3	15.4963	16.0042	18.4975	19.496		

OV2.4 Communication PakBus avec la CR200

La CR200 utilise le protocole PakBus afin de communiquer avec l'ordinateur, et d'autres appareils Pakbus. PakBus est un terme utilisé par Campbell Scientific, pour définir le protocole de communication utilisé pour effectuer le routage de packets de données. Les packets de données transmis entre les appareils PakBus comprennent des informations d'en-tête qui sont utilisées afin de faire parvenir le packet de données à sa destination finale.

Chaque appareil PakBus a besoin d'avoir une adresse Pakbus afin de recevoir, d'envoyer ou de transmettre des packets de données. Dans un réseau Pakbus, chaque appareil doit avoir une adresse Pakbus unique. La CR200 est configurée avec l'adresse 1 par défaut à sa sortie d'usine. Il est recommandé de donner une adresse comprise entre 1 et 3999, pour une CR200.

Un guide sur les communications PakBus est disponible sur le site Internet de Campbell Scientific (*PakBus Networking Guide*) ; celui-ci décrit et donne des exemples détaillés pour plusieurs configurations PakBus.

OV2.5 Communication série ASCII avec la CR200

Voir annexe A.

OV3 Configuration de la CR200 à l'aide de Device Configuration Utility

Device Configuration Utility (DevConfig) est un utilitaire fourni avec LoggerNet, PC400 et PC200W ; il est disponible gratuitement sur le site Internet de Campbell Scientific.

DevConfig est utilisé afin de configurer les centrales de mesures et les périphériques, avant que ceux-ci ne soient installés sur le terrain. Certaines fonctionnalités clé de DevConfig sont :

- DevConfig ne fonctionne qu'avec un lien série direct entre le PC et l'appareil à configurer
- DevConfig vous permet de déterminer le type et la version du système d'exploitation de votre appareil
- DevConfig fournit un résumé de la configuration actuelle de l'appareil ; celle-ci peut être affichée à l'écran ou imprimée. Cette configuration peut aussi être enregistrée sous forme de fichier, à l'aide duquel on peut rétablir la configuration d'un appareil, ou dont on peut se servir pour copier une configuration à un autre appareil.

OV3.1 Alimentation, et lien de communication avec la CR200

Pour connecter l'alimentation (entre 7 et 16 V CC) à la CR200, il faut insérer le fil positif à la borne « Battery + », puis le fil négatif à la borne « Battery - », tel que cela est montré sur la figure OV 3-1.

On connecte alors le câble série fourni (n° 10873) entre le port RS-232 de la CR200 et un port série d'ordinateur. Pour les ordinateurs qui n'ont qu'un port USB, il faut absolument les équiper d'un adaptateur USB-Série (non fourni).

Figures OV3-1 Connexion de l'alimentation et du lien série sur une CR200.

OV3.2 Utilisation de DevConfig afin de configurer l'adresse PakBus

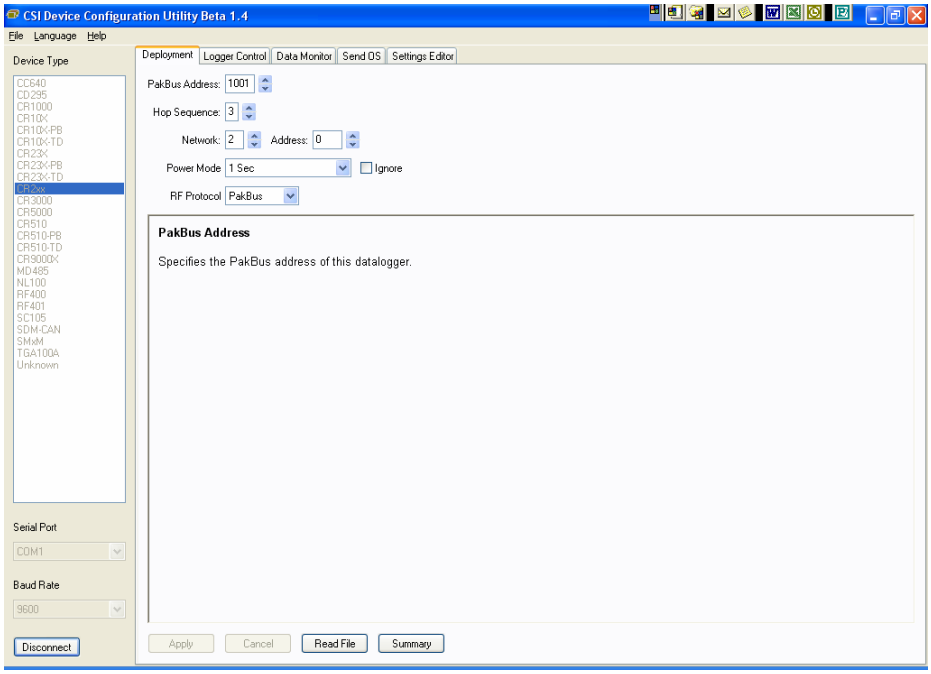
L'adresse PakBus par défaut est 1. Si la CR200 n'est pas utilisée en réseau, il n'y a sans doute pas de raison de changer ses configurations par défaut. Si la CR200 est utilisée en réseau, on utilisera DevConfig tel que décrit ci-dessous.

Pour configurer l'adresse PakBus, la CR200 doit être alimentée et connectée à un ordinateur de la façon décrite au paragraphe OV3-1.

On lance l'utilitaire DevConfig. Cet utilitaire est divisé en deux parties distinctes : la partie gauche qui regroupe les types d'appareils ; la partie droite qui détaille les onglets disponibles pour chaque appareil. On commence par sélectionner le type d'appareil (ici CR2xx), puis le port COM qu'utilisera l'ordinateur.

Figures OV3-2 Ecran principal de DevConfig.

On clique alors sur le bouton « Connect » afin d'établir la communication. L'utilitaire établit la communication, et l'écran suivant est alors affiché.



Figures OV3-3 Configuration de l'adresse PakBus.

Pour fixer l'adresse PakBus d'une CR2XX, il faut utiliser les flèches situées à côté de la boite « Pakbus Address », ou bien entrer au clavier le nombre souhaité (par exemple 10), puis appuyer sur « Apply ». Si vous souhaitez continuer avec le tutoriel de démarrage rapide qui suit (paragraphe OV4), il faut laisser l'adresse à sa valeur « 1 ». Vous pouvez alors vous déconnecter via le bouton « Disconnect ».

OV3.3 Configurer la radio

Les radios des CR206, CR211 et CR216 et RF416/RF415, ont des adresses, des fréquences et des configurations d'alimentation. Ces adresses ne sont pas des adresses PakBus, mais une adresse que la radio encode à l'intérieur de son message. Pour que deux radios communiquent entre elles, l'adresse et la fréquence de communication doivent être les mêmes entre les 2 radios.

NOTE: Dans un réseau PakBus, les adresses et les fréquences de TOUTES les radios doivent être les mêmes.

Les radios sont configurées à partir de l'onglet « Deployment », comme cela est décrit au paragraphe OV3.3. Les configurations radio sont décrites ci-dessous.

OV3.3.1 Séquence de saut de fréquence radio (Radio Hop Sequence)

Les radios à étalement de spectre ont une bande de fréquences qu'elles peuvent utiliser. Les radios « sautent » (hop) d'une fréquence à une autre à l'intérieur de cette bande, permettant à plusieurs jeux de radios de communiquer en même temps sans qu'elles n'interfèrent les unes sur les autres. La séquence de saut de fréquence détermine la séquence avec laquelle les sauts sont effectués. Les radios doivent être configurées sur la même séquence de saut de fréquence afin qu'elles s'entendent les unes aux autres (c'est à dire, afin que la radio qui écoute, soit synchronisée avec la radio qui transmet, et qu'elles sautent de fréquence au même moment.). Mettez le même saut de fréquence sur toutes les radios.

OV3.3.2 Adresses radio, adresses radio réseau (Radio Address / Radio Net Address)

Les adresses radio « Radio Address » et les adresses radio réseau « Radio Net Address » sont combinées et sont envoyées en tant que partie intégrante de l'en-tête de paquet avec chaque message. Même si une radio est sur la même séquence de saut de fréquence, et qu'elle peut entendre une autre radio, elle ignore le message, à moins que ce message ait son adresse dans l'en-tête du message. Mettez les mêmes « Radio Address » et « Radio Net Address » sur toutes les radios.

NOTE: Les radios RF415/416 qui communiquent avec une CR206, CR211 ou CR216, doivent aussi être configurées avec les mêmes « Radio & Radio Net Address ». Voir le manuel de la RF415/416 pour les conseils afin de configurer ces paramètres sur la RF415/416

OV3.3.3 Modes d'alimentation de la radio (Radio Power Mode)

Le mode de configuration de l'alimentation de la radio détermine la fraction de temps pendant laquelle la radio est activée et « écoute » les transmissions entrantes, et par conséquent combien de courant est utilisée par la radio / la centrale de mesure, à cet effet.

RF_ON : La radio au repos consomme un courant inférieur à 24mA (CR206, CR211) / inférieur à 36mA (CR216). Le récepteur radio est toujours actif. Cela offre la réponse la plus rapide lorsque la centrale de mesure sera interrogée, mais à cause de la consommation en courant, cette configuration n'est recommandée que pour les sites où il y a une batterie de secours rechargée par un adaptateur secteur, ou lorsque la vitesse de transmission est une nécessité.

RfpinEn : La radio au repos reçoit un courant de 0mA. Pour utiliser la centrale en tant que capteur sans fil, avec une centrale de mesure de type TD, on sélectionne RfpinEn pour le mode d'alimentation. Ce mode a la consommation en courant la plus faible parmi tous les modes disponibles. La radio est contrôlée par le programme de la CR200. Dans le cas du capteur sans fil, toutes les transmissions sont initiées par la CR200. La radio n'est alimentée que pendant la transmission et pendant un court moment après celle-ci, pendant qu'elle attend la réponse de la centrale de mesure qui est maître.

NOTE: Dans une application avec capteurs sans fil, la RF416 connectée à une centrale de mesure qui est maître, doit être configurée et mise en mode « toujours allumée ».

RF1_Sec : La radio au repos consomme un courant inférieur à 2,2mA (CR206, CR211) / inférieur à 4mA (CR216). La radio s'active une fois par seconde afin d'écouter les transmissions. Le délai maximum de réponse est de 2 secondes.

RF8_Sec : La radio au repos consomme un courant inférieur à 0,45mA (CR206, CR211) / inférieur à 0,8mA (CR216). La radio s'active une fois toutes les 8 secondes afin d'écouter les transmissions. Le délai maximum de réponse est de 16 secondes.

RF1S_LH : La radio au repos reçoit un courant inférieur à 2,2mA. La radio s'active une fois par seconde afin d'écouter les transmissions. Lorsqu'elle initie la communication, elle envoie un long en-tête (de plus d'une seconde) dans son message. Cela ne sera utilisé que si la centrale est en train d'initier la communication avec SendGetData et qu'une autre radio dans le système, utilise le mode de veille avec une seconde de délai. L'en-tête longue permet de s'assurer que les autres radios dans le réseau entendent le message.

RF8S_LH : La radio au repos reçoit un courant inférieur à 0,4mA (CR200, CR210) ; inférieur à 4mA (CR215). La radio s'active une fois toutes les 8 secondes afin d'écouter les transmissions. Lorsqu'elle initie la communication, elle envoie un long en-tête (de plus de huit secondes) dans son message. Cela ne sera utilisé que si la centrale est en train d'initier la communication avec SendGetData et que les autres radios dans le système, utilisent le mode de veille avec huit secondes de délai. L'en-tête longue permet de s'assurer que les autres radios dans le réseau entendent le message.

Une fois que vous avez choisi le mode de saut de fréquence souhaité, l'adresse et le mode d'alimentation, vous devez enregistrer la configuration en cliquant sur le bouton « Save Settings », afin qu'elle soit sauvegardée dans la radio.

OV3.3.4 Protocole radio (RF Protocol)

Ceci identifie le protocole radio qui sera utilisé pour les CR2xx. *Afin d'être compatible avec d'autres appareils de type CR2xx ou RF400, la valeur par défaut est le mode transparent*. Les valeurs suivantes sont supportées :

1. Transparent – Ce mode est compatible avec les systèmes d'exploitation des CR205, CR210, CR215 et RF400 / RF405 / RF415.
2. PakBus – Ce mode peut être utilisé dans des réseaux mettant en œuvre des RF401/411/416 ou des CR206/211/215, et utilisant la capacité de rappel (*retry*) inhérente aux évolutions de la radio MaxStream. *Ce mode n'est pas compatible avec les radios les plus anciennes*

OV4 Comment programmer la CR200 rapidement

OV4.1 Logiciels pour la CR200

Le logiciel de démarrage PC200W permet d'effectuer de la communication en direct entre l'ordinateur et le centrale de mesure (par RS232 ou radio à étalement de spectre) ; ce logiciel inclut l'éditeur de programmes Short Cut. PC200W fournit des outils de base afin de fixer l'heure de la centrale, d'envoyer des programmes vers la centrale, de collecter les données manuellement, de visualiser les données issues des capteurs, et de visionner les données. Le support pour la CR200 a été ajouté lors de la réalisation de la version 3.0 du logiciel PC200W. PC200W est disponible gratuitement sur le site Internet de Campbell Scientific USA.

Le logiciel PC400 (logiciel de milieu de gamme) permet d'utiliser un nombre varié d'options de télécommunication, permet aussi de collecter les données manuellement, et de les visualiser. PC400 inclut Short Cut ainsi que l'éditeur CRBasic. PC400 ne permet pas d'effectuer des communications en mode combiné (par exemple d'un téléphone à une radio), ni du routage PakBus® ou de l'appel automatique pour la collecte des données.

Le logiciel LoggerNet (logiciel complet) permet de mixer les modes de télécommunication, permet de faire de l'affichage de données graphique, et de l'appel automatique pour la collecte de données. Le logiciel inclut Short Cut ainsi que l'éditeur CRBasic afin de générer des programmes pour la CR200 ; il inclut aussi des outils de dépannage et de gestion de réseau de centrales de mesures

OV4.1.1 Options pour la création de programmes CR200

1. Short Cut est un générateur de programme qui crée un programme pour centrale de mesure suite à 4 étapes, et qui l'associe avec un diagramme de câblage des capteurs sélectionnés. Short Cut connaît la plupart des capteurs commercialisés par Campbell Scientific ; il est conseillé de l'utiliser afin d'avoir des programmes basiques pour mesurer et enregistrer les données d'un capteur.
2. L'éditeur CRBasic est est un programme utilisé afin de créer des programmes plus complexes. Les programmes générés par Short Cut peuvent être importés dans CRBasic, puis modifiés afin d'y incorporer des instructions ou des fonctionnalités non disponibles avec Short Cut.

OV4.2 Se connecter à la CR200

Pour se connecter à la CR200 il faut que celle-ci soit alimentée en 12V CC, et que l'ordinateur y soit relié, comme décrit au paragraphe OV3.1.

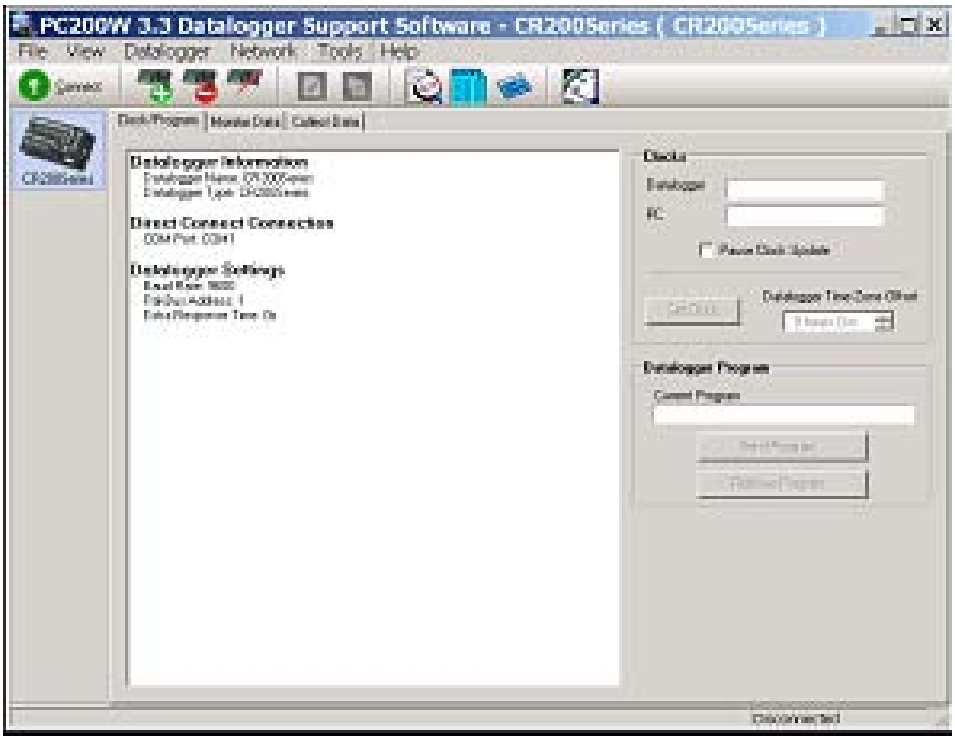
OV4.3 Le logiciel PC200W

Ce didacticiel rapide permet de faire avancer étape par étape au travers du processus de programmation de la CR200, de visualisation des mesures effectuées en direct ou enregistrées, et la visualisation graphique de ces données. Avant d'utiliser PC200W, vérifiez que son adresse Pakbus est « 1 », comme cela est mentionné au paragraphe OV3.

Lorsqu'on lance PC200W pour la première fois, l'utilitaire EZSetup s'exécute. Cliquez sur « Next » et suivez les étapes afin de choisir une CR200, le port COM de l'ordinateur que vous utiliserez, la vitesse de 9600 baud, une adresse Pakbus de « 1 », et lorsqu'on vous propose de tester la communication, cliquez sur le bouton « Finish ».

Pour changer un élément dans la configuration de la centrale de mesure il vous faut sélectionner la centrale dans la fenêtre principale, et cliquer sur « Edit ». Si une centrale n'est pas présente et que vous souhaitez l'ajouter, cliquez sur le bouton « Add » et suivez les étapes du « Wizard » (l'Assistant).

Lorsque vous quittez EZSetup, l'onglet Clock / Program apparaît, tel que montré ci-dessous. Le profil de configuration actuel de la centrale, son heure et d'autres éléments sont intégrés sur cet onglet. Les onglets suivants sont utilisés afin de visionner les valeurs (Monitor Data) ou afin de collecter les données (Collect Data). Les boutons les plus à droite sont utilisés afin d'exécuter les applications View, Split, Card Convertor ou Short Cut.

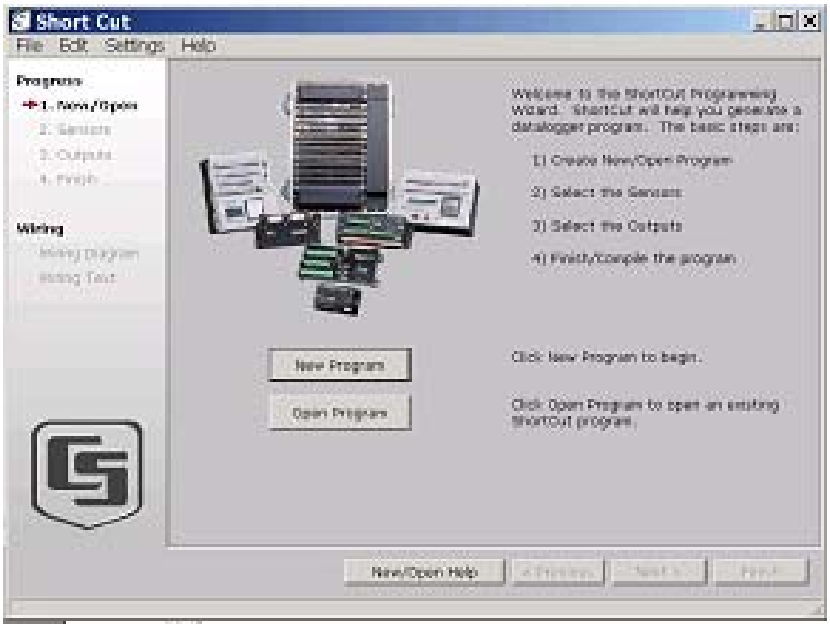


OV4.3.1 Générer un programme de CR200 à l'aide de Short Cut

Objectif : Mesurer à un intervalle de 10 secondes, la température d'une thermistance 109 et la pluviométrie (en mm) d'un capteur ARG100. Chaque minute on sauvegardera la température moyenne, le total de précipitation, et le minimum de tension batterie.

Vous pouvez suivre l'exemple de programme même si vous ne disposez pas des capteurs 109 et ARG100. Sans capteur 109, les valeurs affichées seront « NAN », sans ARG100 les mesures « 0 » seront affichées.

Cliquez sur le bouton « Short Cut » afin d'afficher l'écran ci-dessous.



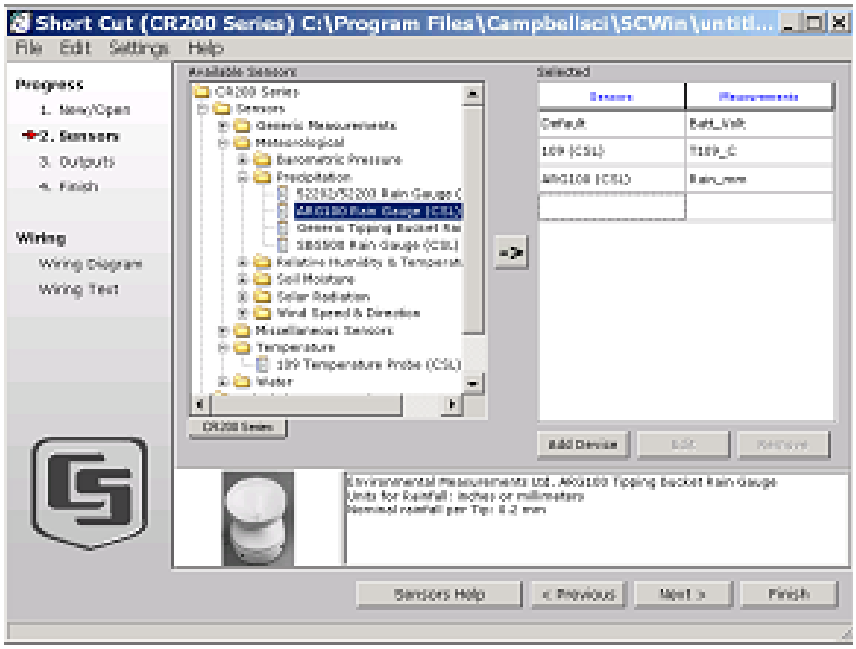
En cliquant sur l'onglet Help (Aide), vous pouvez avoir accès à l'aide de ce logiciel. On peut utiliser l'aide en plus des descriptions données ci-dessous :

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 CR200. On entre la valeur de 10 (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

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.

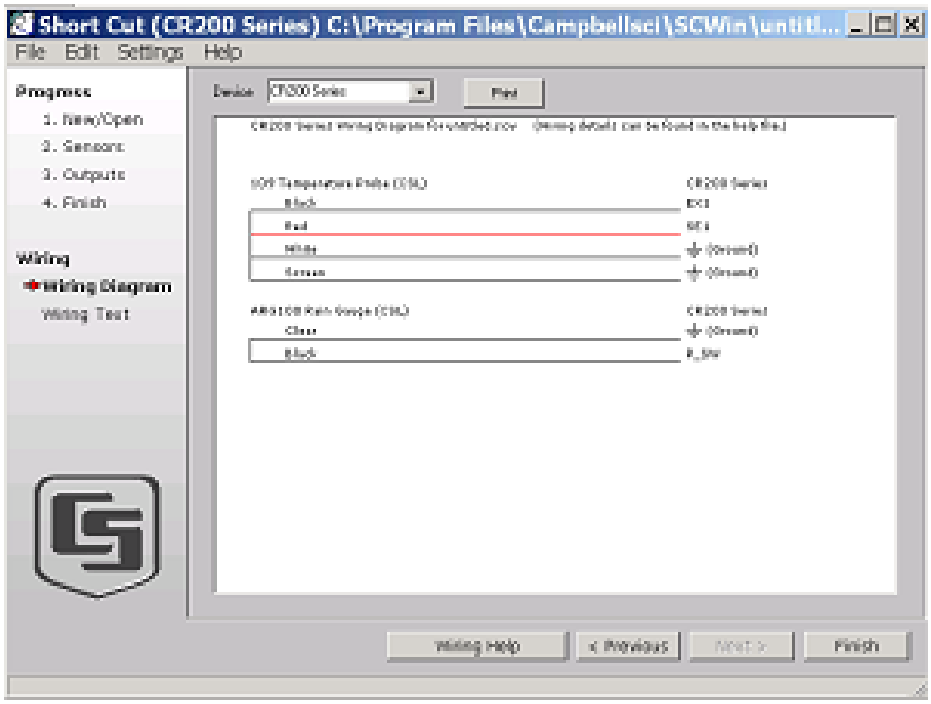


Les capteurs que vous allez programmer seront choisis à partir de la liste définie dans l'arborescence des capteurs.

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 «**109 Temperature Probe**» 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 (T109_C).

Double cliquez sur « **Meteorological »**, puis « **Precipitation** » / **ARG100**, et cliquez sur « **OK** ». Sur l'écran suivant, assurez-vous que la variable « Rain_mm » s'affiche ; vous mesurerez alors des mm de pluie.

Cliquez sur votre gauche sur « **Wiring Diagram** » afin de voir le schéma de câblage, tel que celui ci-dessous. Il faut alors câbler la thermistance 109 et le pluviomètre ARG100 comme cela est spécifié. Si vous n'avez pas d'ARG100, vous pouvez simuler sa présence avec un contact de basculement (toggle switch). Sans capteur 109, le résultat de la mesure sera « NAN » (Not A Number).



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)

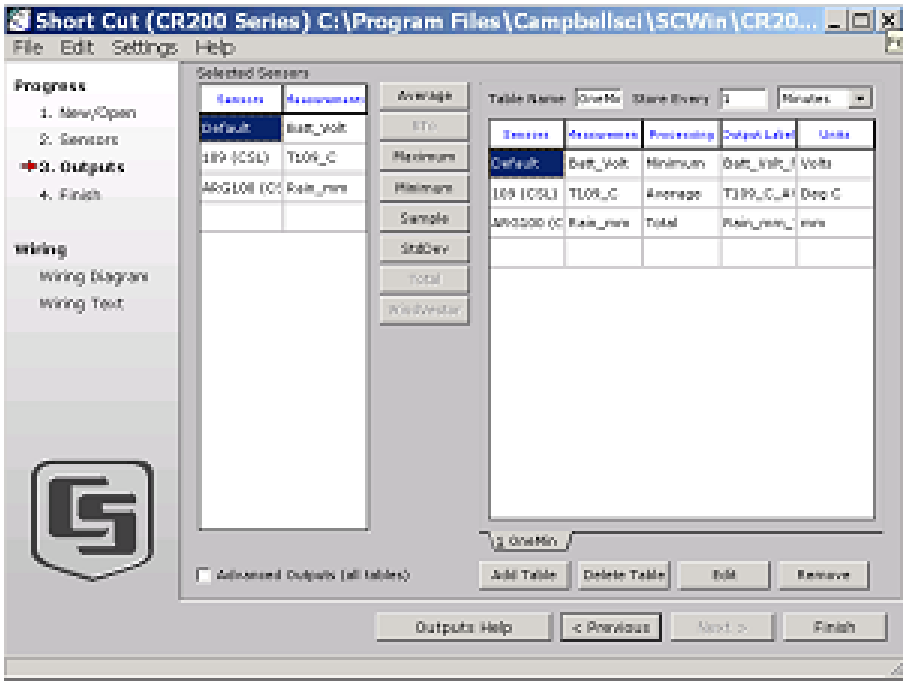
L'étape 3 permet de définir les instructions de sauvegarde pour les capteurs mesurés. Sur votre gauche, il faut cliquer sur le bouton « **Output** » (Sortie).

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.

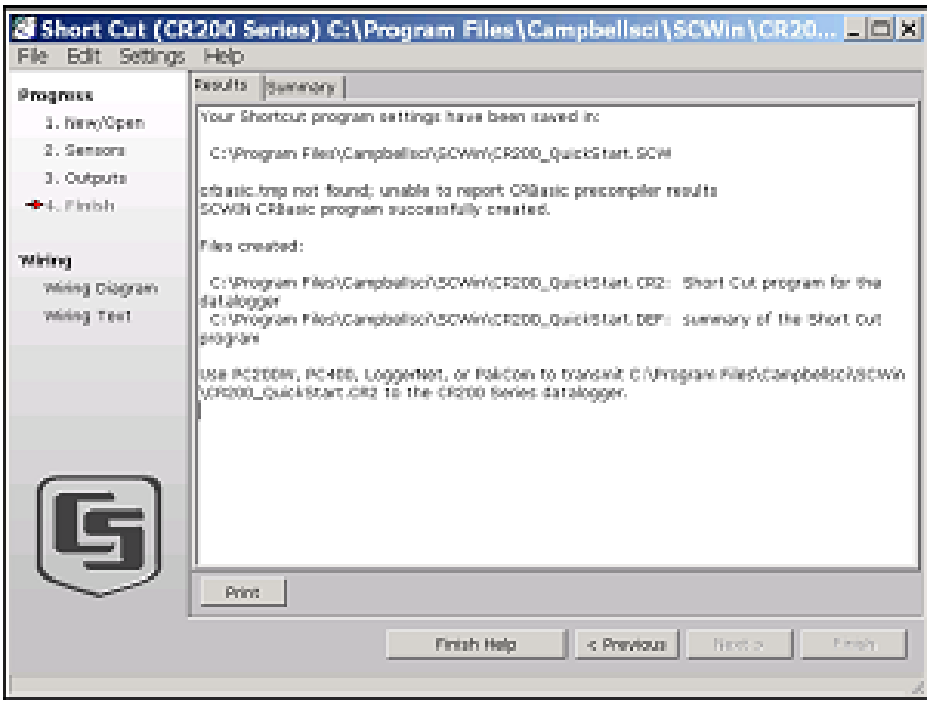
La liste des capteurs sélectionnés se trouve sur la partie gauche de l'écran. Pour ajouter une mesure de capteur à l'intérieur du tableau de sauvegarde, il faut surligner une mesure et cliquer sur un des boutons qui déterminent le type de calcul que l'on veut faire (par exemple la moyenne « Average »). On sélectionnera alors les capteurs appelés « Default » (qui mesure la tension batterie) et on double cliquera sur le bouton « **Minimum** » ; on sélectionnera le capteur « 109 » et on double clique sur « **Average** » ; on sélectionnera enfin l'« ARG100 » et on double cliquera sur le bouton « **Total** » afin d'enregistrer ces valeurs dans le tableau appelé «OneMin».



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

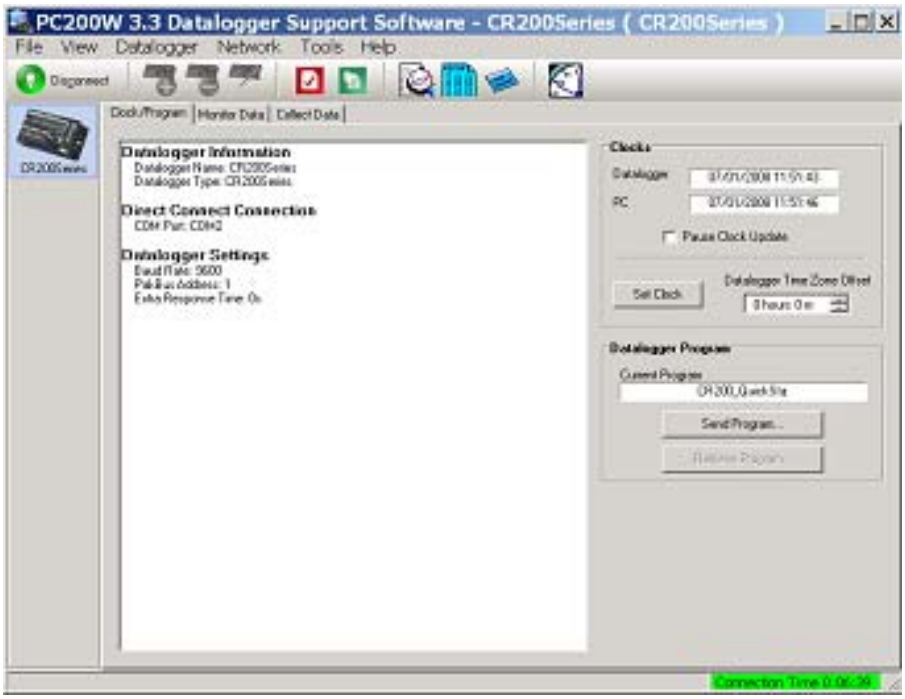
Etape 4 (Step 4): Terminer (Finish)

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 CR200_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 CR200_QuickStart est le fichier programme qui sera envoyé à la CR200, CR200_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).



OV4.3.2 Configuration de l'onglet « Clock / Program »

A partir de l'écran « **Clock / Program** », cliquer sur le bouton « **Connect** » afin d'établir la communication avec la CR200. Lorsque la communication est établie, le texte présent sur le bouton deviendra « **Disconnect** ».



OV4.3.3 Synchronisation de l'horloge

Cliquer sur le bouton « **Set Clock** » afin de synchroniser l'horloge de la centrale avec celle du PC.

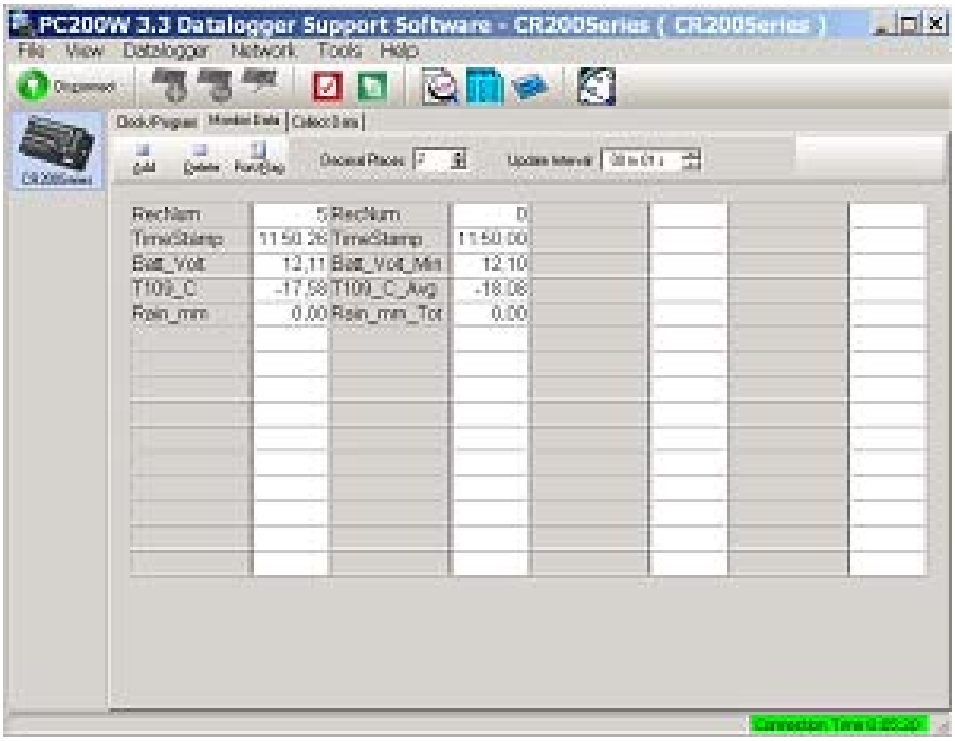
OV4.3.4 Envoyer le programme

Cliquer sur le bouton « **Send Program** ». Naviguer afin de pointer sur le répertoire C:\CampbellSci\SCWin et y sélectionner le fichier appelé CR200_QuickStart.CR2 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.

OV4.3.5 Visualiser les données scrutées

L'onglet « **Monitor Data** » 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 ».

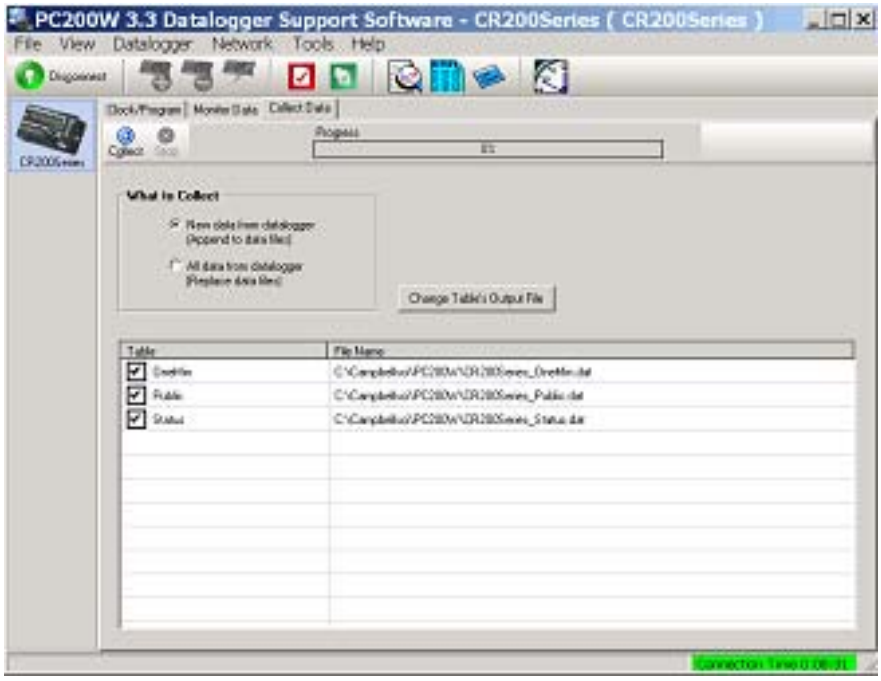
Cliquer sur l'onglet « **Monitor Data** ». Le contenu des variables « Public » est automatiquement sélectionnées et affichées. Pour visualiser les valeurs du tableau « OneMin » il faut cliquer sur le bouton « **Add** », sélectionner le tableau « OneMin » puis cliquer sur le bouton « **Paste** ».



OV4.3.6 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 vous souhaitez les enregistrer sur votre ordinateur.

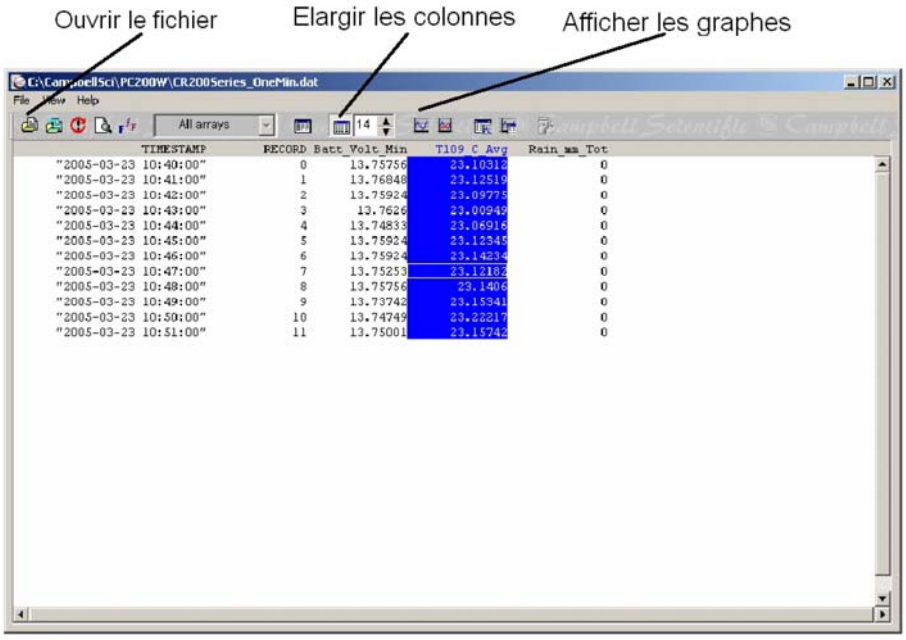
Cliquer sur le tableau appelé « **OneMin** », avec l'option « **New data from datalogger** » de sélectionnée. Cliquer sur le bouton « **Collect** » et une boite de dialogue apparaît, vous demandant un nom de fichier à donner. En cliquant sur « Save » vous utiliserez le nom par défaut prévu pour le fichier, soit CR200_OneMin.dat. Une barre de progression s'affichera, avec à la fin le message « **Collection Complete** ».

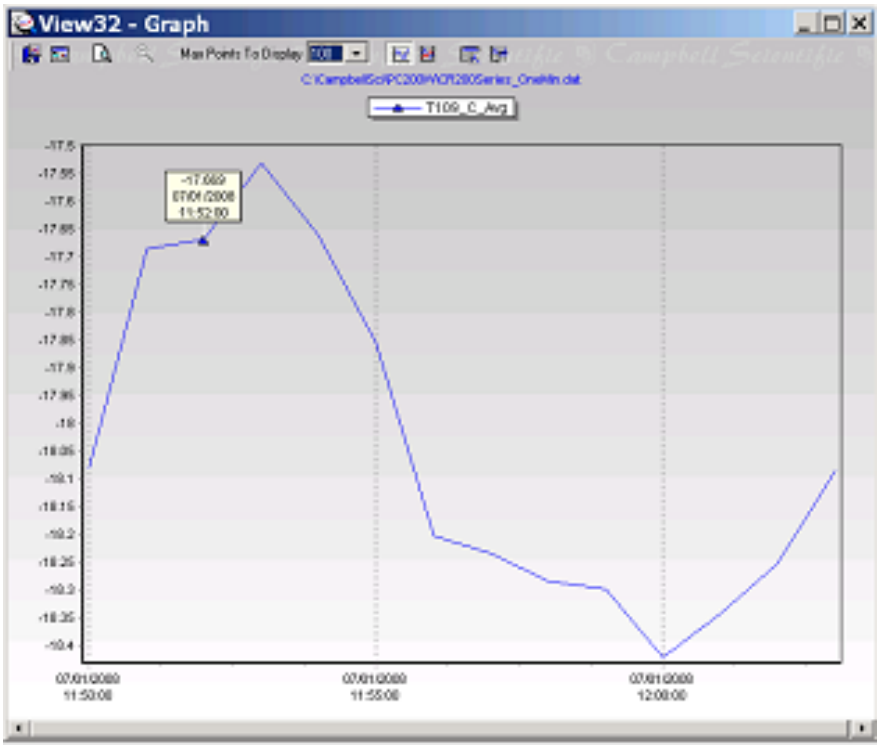


OV4.3.7 Visualiser les données (View Data)

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 CR200_OneMin.dat présent dans le répertoire de PC200W. Cliquer sur « **Expand Tabs** » afin d'afficher les données en colonne avec des en-tête de colonne. Si vous souhaitez afficher les données issues de la thermistance, sous forme graphique, il faut cliquer sur la colonne qui porte le nom « T109_C_Avg », puis cliquer sur l'icône « **Show Graph, 1 Y axis** » qui est présente dans le barre d'outils.





OV4.4 Programmer en utilisant l'éditeur de programme CRBasic

Les utilisateurs qui avaient l'habitude de l'éditeur de programme Edlog, et qui passeraient à CRBasic pour programmer la CR200, 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 CR200_QuickStart.CR2 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 chapitre 4 pour plus de détails au sujet de la programmation CRBasic.

```
'CR200 Series
'Created by Short Cut (2.5)
'Declare Variables and Units
Public Batt_Volt
Public T109_C
Public Rain_mm

Units Batt_Volt=Volts
Units T109_C=Deg C
Units Rain_mm=mm

'Define Data Tables
DataTable(OneMin,True,-1)
    DataInterval(0,60,Min)
    Minimum(1,Batt_Volt,False,False)
    Average(1,T109_C,False)
    Totalize(1,Rain_mm,False)
EndTable

'Main Program
BeginProg
    Scan(10,Sec)
        'Default Datalogger Battery Voltage measurement Batt_Volt:
        Battery(Batt_Volt)
        '109 Temperature Probe (CSL) measurement T109_C:
        Therm109(T109_C,1,1,1,1.0,0.0)
        'ARG100 Tipping Bucket Rain Gauge measurement Rain_mm:
        PulseCount(Rain_mm,P_SW,2,0,0.2,0)
        'Call Data Tables and Store Data
        CallTable(OneMin)
    NextScan
EndProg
```

OV5 Caractéristiques

CPU et stockage

- Convertisseur Analogique / Numérique 12 bits
- Vitesse de scrutation maximale d'un Hz
- Horloge interne sauvegardée par une pile
- 128Ko de mémoire flash pour la mémoire finale (512Ko s'il y a l'étiquette) ; le format de données est de 4 octets par point de mesure (format tableau de données)
- 6,5Ko de mémoire flash pour le stockage du programme
- Des LEDs indiquent si la centrale de mesure est en train de scruter les voies, d'émettre ou de recevoir.

Voies analogiques ; entrées / sorties numériques (SE1 à SE5)

Les voies SE1 à SE5 peuvent être configurées individuellement afin de faire des mesures unipolaires ou un contrôle numérique en entrée / sortie.

Mesures unipolaires

- Etendue de mesure en entrée : 0 à 2,5V CC
- Résolution de mesure : 0,6mV
- Précision de la mesure[1] :
  - Généralement : +/- (0,25% de la valeur lue + 1,2mV d'offset) entre –40 et +50°C
  - Dans le pire des cas : +/- (1% de la valeur lue + 2,4mV d'offset) entre –40 et +50°C

Entrée / sortie numérique

- Etat d'entrée / sortie niveau haut : entre 2,1 et 3,3V CC
- Etat d'entrée / sortie niveau bas : inférieur à 0,9V CC
- Courrant de fuite : 220µA à 2,7V CC
- Tension maximale en entrée : 4V CC

Mesure de demi-pont

- Précision relative à l'excitation.
- Avec l'excitation de 2,5V CC elle est de +/- (0,06% de la valeur lue + 2,4mV)

Mesure de période moyenne

- Tension maxi en entrée : 4V CC
- Etendue de mesure en fréquence : de 0 à 150 kHz
- Tension de seuil : comptage des cycles lors de la transition entre moins de 0,9V CC et plus de 2,1V CC

Voies d'excitation (EX1 & EX2)

- Etendue de mesure : programmable à +2,5 et +5V CC
- Précision[1] : +/- 25mV sur l'étendue de mesure de +2,5V CC, +/- 125mV sur l'étendue de mesure de +5V CC.
- Courant maximum : 25mA sur l'étendue de mesure +2,5V et 10mA sur l'étendue de mesure +5V CC.

Batterie commutée (SW BATTERY)

- Commutée sous contrôle du programme
- 300mA de courant disponible au minimum
- Compteurs d'impulsion

Compacts secs (P_SW)

- Taux de comptage maximum : 100Hz
- Temps d'ouverture minimum du contact : 5ms
- Temps de fermeture minimum du contact : 5ms
- Temps maximum de rebond : 4ms

[1] La précision d'une mesure de pont utilisant une excitation de 2,5Vest de +/- (0,06% de la valeur lue + 2,4mV)

Comptage d'impulsion (P_SW, C1 & C2)

- Seuil de tension : compte une transition de moins de 0,9V CC à plus de 2,7V CC
- Fréquence maximum en entrée : 1kHz
- Tension maximum en entrée : 6,5V CC (pour C1 & C2) / 4,0V CC (pour P_SW)

Tension alternative bas niveau (P_LL)

- Tension minimum en entrée : 20mV RMS
- Fréquence maximum : 1kHz
- Seuil de détection : <0,5V ou >2,0V CC
- Tension maximale en entrée : 20V CC

Note : Les voies P_LL, C1 & C2, peuvent être utilisées afin de mesurer des contacts secs en utilisant la tension batterie et une résistance de 20kOhm.

Si l'offset CC est >0,5V alors il faut effectuer une mesure en couplage C.A.

Ports de contrôle (C1 & C2)

Entrée / sortie numérique

- Tension lorsqu'elle sont configurées en tant qu'entrée : inférieur à 0,9V CC (état bas) ou supérieur à 2,7V CC (état haut)
- Tension lorsqu'elles sont configurées en tant que sortie : 0V (état bas), 5V (état haut)
- Niveau logique : TTL
- Courant maximum fourni : 1,5mA à 4,5V.

SDI-12

- Les capteurs SDI-12 se connectent à C1.

Alimentation

- Tension d'alimentation : entre 7 et 16V CC (on peut programmer la centrale de mesure afin qu'elle lise la tension de la batterie)
- Batterie : Besoin d'une batterie rechargeable 12V ; le circuit de charge est intégré au bornier. On peut utiliser des piles alcalines ou un autre type de source de courant non rechargeable, à condition que le circuit de charge ne soit pas utilisé (rien n'est connecté aux bornes de charge).
- Tension acceptée pour le chargeur de batterie : de 16 à 22V CC
- Durée de vie de la pile de sauvegarde de l'heure : 5 ans

Boîtier

- Boîtier en aluminium avec des terminaisons à ressort « spring loaded »
- Dimensions : 140 x 76 x 51mm (terminaisons de connexion comprise)
- Boîtier personnalisé : disponible pour des applications en OEM ; contacter Campbell Scientific.

Communication

- RS-232 supporté en direct, avec ou sans radio.
- Radio à étalement de spectre intégrée au bornier (pour les modèles CR205, CR210 ou CR215 seulement)
  - 916MHz (CR205), 922MHz (CR210) ou 2,4GHz (CR215 & CR216)
  - Transmission à vue à un sur 1 mile avec une antenne radio 1/4 à 0 dBd et les radio à 900MHz
  - Transmission à vue à un sur 1 km (0,6 mile) avec une antenne radio 1/2 à 0 dBd et la radio à 2,4GHz
  - Transmission à vue à plus de 10 miles avec des antennes à plus fort gain

  Note : L'utilisation d'une antenne avec un gain plus élevé, pourrait invalider la conformité d'utilisation dans votre pays. Les antennes à gain élevé ne sont généralement pas autorisées dans les pays Européens.

    - La RF4XX est utilisée en tant que station de base
  - Modes de radio disponibles pour la transmission :
    - Toujours en transmission : contrôlé par le programme
    - Cycles d'émission : 1 ou 8 cycles ; actif pendant 100ms à chaque période ; vérifie s'il y a de la communication entrante
    - Transmission programmable : transmission inactive jusqu'à ce que l'heure soit atteinte
  - Protocole réseau PakBus pour transmission de paquets.

Logiciel

Pour une seule station

- Logiciel gratuit pour PC : PC200W (ou anciennement PakCom)

Pour un réseau

- Logiciel de Campbell Scientific appelé LoggerNet, afin de collecter les données d'un réseau de centrales de mesure qui inclut la CR2XX en tant que capteurs sans fil.

Développement de logiciel

- Possibilité d'obtenir des composants de développement de logiciel avec PakCom Active X (BMP5) pour Windows. Version Windows et Windows CE disponibles.

Programmation

- On programme la centrale à l'aide de SCWin ou de l'éditeur CRBasic
- On télécharge le programme compilé, depuis le PC jusqu'à la centrale. On ne peut pas éditer le programme sur la centrale.

Consommation en courant (à 12V)

- Consommation au repos : sans radio ou radio éteinte : ~0,2mA
- Consommation en fonctionnement :
  Sans radio : ~3mA
  En réception radio : ~20mA (CR206, CR211), ~36mA (CR216)
  En transmission radio : ~75mA (CR206, CR211, CR216)
- Consommation moyenne de courant :
  Radio toujours active :20mA (CR206, CR211), ~36mA (CR216)
  Radio en cycle de 1 seconde : 2,2mA (CR206, CR211), ~4mA (CR216)
  Radio en cycle de 8 secondes : 0,45mA (CR206, CR211), ~0,8mA (CR216)

Conformité CE

La conformité est déclarée pour les normes

IEC 61326:2002

Protection EMI et ESD

Immunité : Conforme ou dépassant les standards suivants :

ESD : IEC 1000-4-2 ; +/- 8kV air, +/- 4kV contact de décharge
RF : IEC 1000-4-3 ; 3V/m, 80-1000 MHz
EFT : IEC 1000-4-4; 1kV d'alimentation, 500V d'entrée/sortie
Fusible : IEC 1000-4-5 ; 1kV d'alimentation et d'entrée/sortie
Conduction : IEC 1000-4-6 ; 3V 150 kHz-80 MHz

Les émissions et les critères d'immunité et de performance, sont disponibles sur demande.

Garantie : Pièces et main d'œuvre pendant un an

.

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 CR200 est faite pour fonctionner de façon optimum entre –40 et +50°C, en atmosphère sans condensation. Lorsque l'humidité devient trop forte, on peut endommager les composants électroniques, 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.

La centrale de mesure doit être mise en place dans un coffret lorsqu'elle est utilisée sur le terrain. Le coffret devra contenir du dessicatif qui sera remplacé ou séché assez souvent, de façon à assurer le contrôle de l'humidité.

1.1.1 Coffret ENC-200E

Le coffret ENC-200E est spécialement dédié pour les centrales d'acquisition CR200. Ce coffret est conçu pour recevoir une CR200, une petite batterie et un petit périphérique supplémentaire, par exemple le modem GSM. Il peut être monté directement sur un mur ou installé sur un mât ou trépied à l'aide du kit de montage optionnel.

Le coffret ENC-200E a une dimension interne de 168 x 175 x 92 mm, fabriqué en poly- carbonate avec un indice de protection IP66. Ce coffret possède une prise de terre et 7 presses étoupes, livrées avec une protection pour permettre le cheminement des câbles de sonde et de communication dans le coffret, qui permettent ainsi de maintenir les propriétés d'étanchéité. En option, les antennes compatibles avec la CR216 ou le modem GSM 900 peuvent être fixées sur le dessus du coffret afin de réduire au maximum la longueur des câbles.

Pour installer le coffret ENC-200E sur un mur, enlever le couvercle et utilisez les trous se situant aux coins du coffret, directement au-dessous des propres vis du couvercle, pour fixer le coffret au mur. Alternativement, la platine interne peut être enlevé pour pouvoir forer les trous avant, permettant d'employer un plus grand choix de taille de vis.

ATTENTION: **Percer les parois du coffret peut entraîner une dégradation des capacités d'étanchéités du coffret à moins que vous utilisiez des vis équipés de tête comportant des joints étanches.**

Pour installer l'ENC-200E sur un mât :

1. Fixez le kit de montage optionnelle à l'arrière du coffret en utilisant les vis et écrous fournis pour aller aux quatre coins du coffret. Les attaches sur le bras de montage doivent être dirigées vers l'arrière du coffret. L'attache avec les deux trous devra dépasser au-dessus de la boîte (l'autre attache sera placé au niveau des presses étoupes vers le bas)
2. Amenez le coffret assemblé jusqu'au poteau et installez l'étrier en « V » fourni autour du poteau, le poussant par les deux trous dans l'attache saillante.
3. Utilisez des rondelles anti-vibration et des écrous pour fixer l'étrier en « V »autour du poteau.

ATTENTION: **Ne pas serrer trop fortement les écrous, vous risqueriez de déformer les attaches.**

Lorsque le coffret est installé en position fixe, vous pouvez compléter l'installation de la façon suivante :

1 Installer la centrale d'acquisition CR200 sur la platine du coffret en utilisant les vis et les fixations en nylon fournis. En complément des bandes Velcro sont fournies pour installer la batterie optionnelle de 0,8 Amp/h dans l'espace au dessus de la centrale d'acquisition et les périphériques en dessous ou sur la droite.

2 Connectez la prise de terre de la centrale d'acquisition au connecteur de terre du coffret.

3 Reliez un câble de terre à la prise de terre extérieure du coffret au piquet de terre de référence ou au tuyau d'eau (voir chapitre 1.6.1 ci dessous)

4 Faites passer les câbles des capteurs par les presses étoupes du coffret. Le presse étoupes le plus large est réservé pour le câble de l'antenne équipée d'un connecteur, pour être installé dans le coffret. Pour permettre au connecteur de passer par le presse étoupe, nous vous suggérons d'enlever l'écrou et le joint en caoutchouc intérieure du presse étoupe et de les faire passer un à un autour du connecteur plutôt que d'enfoncer le connecteur en force à travers le presse étoupe.

1.1.2 Autres coffrets

Toute une gamme de coffret vous est offert par Campbell Scientific pour monter la famille des centrales d'acquisition CR200 avec des périphériques et des alimentations.

Les coffrets ENC 10/12 et ENC 12/14 sont construite en fibre de verre renforcée en polyester avec une charnière et un loquet que vous pouvez fermer par un cadenas. Ces coffrets sont résistant à la pluie, à la poussière et à la corrosion. (Chaque coffret fourni par Campbell Scientific est classifié selon la norme IP68 mais cette norme ne peut pas être appliquée lorsque le coffret est équipé de presses étoupes. Il est de votre responsabilité de vérifier que l'étanchéité des presses étoupes au passage des câbles est correcte.) Veuillez vous référer au *manuel d'installation des coffrets* pour de plus amples informations sur ce type de coffret.

1.2 Besoins en énergie

La CR200 fonctionne a 12V CC nominal. En dessous de 7V ou au dessus de 16V, la CR200 ne fonctionne pas correctement.

La tension d'entrée de la CR200 est protégée contre l'inversion de polarité. Des tensions d'alimentation supérieures à 18V peuvent endommager la CR200 et/ou l'alimentation.

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 CR200 a une consommation moyenne de courant qui est d'environ 3mA (excluant la transmission radio). Soyez sûr d'incorporer la consommation de tous les capteurs alimentés, lorsque vous calculez votre besoin en énergie.

1.3 Alimentations pour CR200

Une batterie acide-plomb de 0,8 Amp/h est disponible pour alimenter la centrale d'acquisition. Cette batterie peut être installée dans le coffret ENC-200E. La capacité de cette batterie est telle que vous pourrez seulement alimenter la centrale d'acquisition dans de nombreuses applications pour 10 jours d'autonomie et de quelques heures avec un modem GSM qui est alimenté en permanence. Cette batterie est à utiliser comme secours dans le cas où vous êtes connecté à une source de courant fiable, comme un chargeur CA par exemple. Dans de nombreux cas cette batterie peut être utilisée avec un panneau solaire dans le cas où votre application consomme peu d'énergie et que l'ensoleillement de votre site d'installation est important.

Une batterie de 7 Amp/h est aussi disponible (PS100E-LA200). Cette batterie est destinée pour les centrales d'acquisition de la famille CR200. La capacité supérieure de cette batterie permettra d'avoir une marge de sécurité plus importante pour les applications qui nécessitent une alimention plus conséquente ou lorsque que l'ensoleillement est faible. Cette batterie peut être rechargé directement via la CR200. Pour le montage d'un tel dispositif, nous vous suggérons un coffret de dimension plus grande tel que l'ENC 10/12.

1.3.1 Installation

Connectez la batterie aux bornes « Battery » de la centrale d'acquisition, veillez à respecter la polarité indiquée sur le bornier. La batterie peut être chargé via le circuit interne de charge si la source de charge est connectée aux bornes « Charge » de la centrale d'acquisition.

NOTE: Le chargeur interne ne chargera la batterie uniquement lorsque la tension de charge sera au-dessus de 16V CC (22V au maximum). Cela limitera le courant maximum de charge à une valeur nominale de 1,2 A. Pour éviter une dissipation excessive de chaleur du chargeur interne de la centrale d'acquisition il faudra éviter de connecter une batterie ayant une capacité plus importante au chargeur interne.

1.4 Panneaux Solaires

Une source d'alimentation auxiliaire de type photovoltaïque peut être utilisée afin de maintenir la charge de la batterie acide-plomb.

Lorsqu'on choisit un panneau solaire, une règle de bon sens est de dire que pendant un jour de mauvais temps, le panneau solaire doit pouvoir fournir assez de courant pour assurer les besoins en courant du système (on considère que par mauvais temps on doit recevoir 10% de la moyenne annuelle en rayonnement solaire). Les informations relatives au site, si elles sont disponibles, peuvent fortement influencer le choix du panneau solaire. Par exemple les effets locaux tels que les ombres des montagnes, le brouillard du aux inversion de température dans la vallée, la neige, la glace, les feuilles mortes, les oiseaux etc., peuvent être pris en compte dans le calcul.

Des conseils sont disponibles auprès de Solarex Corporation dans leur document « DESIGN AIDS FOR SMALL PV POWER SYSTEMS », pour vous aider à choisir son panneau solaire. Ce document donne une méthode de calcul de la taille du panneau solaire en fonction de la situation géographique globale du système, et des besoins en énergie du système. Si vous avez besoin d'aide afin de calculer le besoin en énergie de votre système, vous pouvez contacter Campbell Scientific.

1.5 Connexion directe d'une batterie au bornier de la CR200

N'importe quelle source de 7 à 16V peut être connecté aux bornes « Battery + » et « Battery – »,. Lorsque vous reliez une batterie externe à la CR200, mettre la borne positive de la batterie dans la borne « Battery + », et la borne négative dans le borne « Battery – ».

1.6 Mise à la masse de la CR200

La mise à la terre de la CR200, 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.

1.6.1 Protection ESD

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 CR200 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 CR200.

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 CR200 via un fil de cuivre de diamètre 14SWG (2,04mm). 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 consultant spécialisé 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 automobile, le fil de masse doit être solidement relié au châssis du véhicule via un câble 12 SWG 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.

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

Les mesures unipolaires de faible tension peuvent être problématiques, à cause de problèmes de fluctuation de la masse. Or la CR200 n'est pas assez sensible pour faire des mesures de tensions faibles, donc ceci n'est pas réellement un problème.

1.7 Alimentation des capteurs et des périphériques

La CR200 a deux sources de tension d'excitation commutée, afin de fournir du 2500 ou 5000mV pour des ponts de mesure. Il n'y a aucune borne afin de fournir une tension en continu. De tels capteurs devront être reliés directement à la source d'alimentation, ou à la borne SW-Batt (voir paragraphe 1.9).

Assurez-vous que la source d'alimentation primaire que vous avez choisi pour votre CR200, pourra supporter le besoin en tension, durant le temps nécessaire. Contactez Campbell Scientific afin de déterminer la consommation en énergie dont vous aurez besoin, lorsque vos applications approchent les limites des capacités de l'alimentation. Soyez particulièrement vigilant lorsque vous mettez en place une application avec panneau solaire, surtout si vous prévoyez de faire des visites espacées sur le site, ou que votre application doit subir des températures extrêmes.

1.8 Contrôle de l'alimentation de capteurs et de périphériques

Le contrôle de l'alimentation d'un appareil extérieur, est une utilisation habituelle de la CR200.

Plusieurs appareils peuvent être contrôlés par le SW-Batt (la batterie commutée) de la CR200. Le tableau 1.9-1 donne les quantités de courant disponibles depuis le port SW-Batt.

Les applications qui nécessitent plus de port de contrôle ou une source d'alimentation plus importante que celle disponible par la CR200, peuvent généralement être satisfaites avec l'utilisation du A21REL-12, ou en utilisant les ports de contrôle C1 et C2 comme cela est décrit au paragraphe 1.9.1.

Tableau 1.9-1 Limite de courant disponible

Borne	Limite de source de courant
SWBatt	<900mA @20°C
	<729mA @ 40°C
	<630mA @ 50°C
	<567mA @60°C
	<400mA @ 80°C

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

Chacun des ports de contrôle peut être configuré 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 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 figure 1.9-1 et 1.9-2. Pour de plus amples informations, vous pouvez contacter Campbell Scientific.

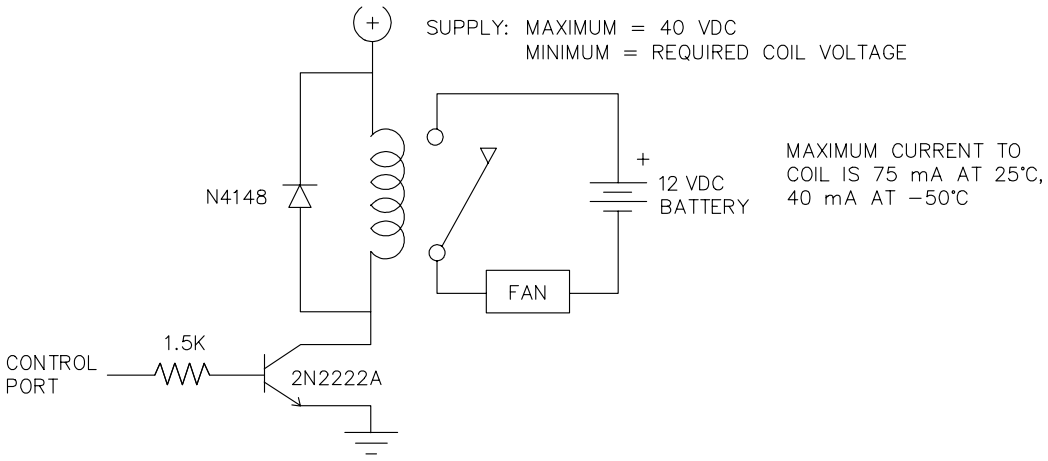


Figure 1.8-1 Circuit de pilotage avec un relais.

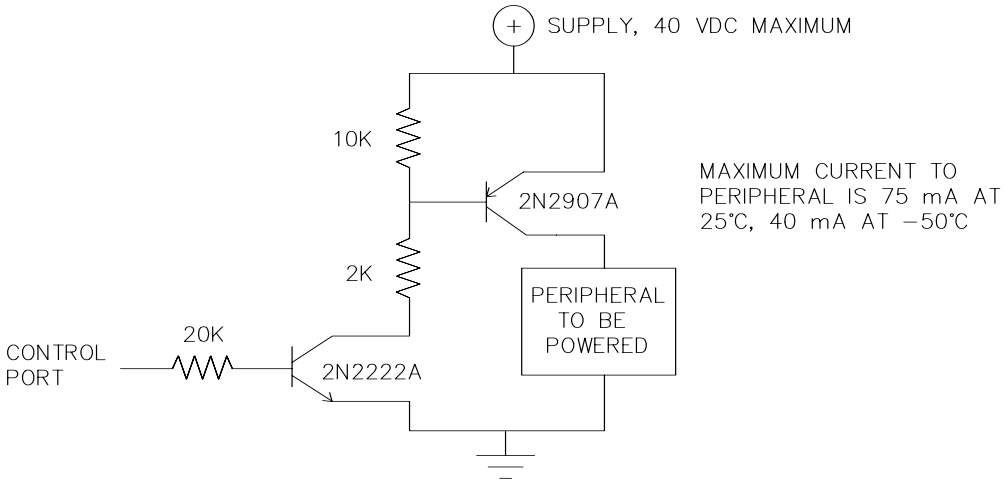


Figure 1.8-2 Circuit de pilotage sans relais.

1.9 Entretien, maintenance

Les alimentations de CR200 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.

1.9.1 Dessiccatif

Afin d'éviter des problèmes de corrosion dans des atmosphères incontrôlées, la CR200 doit être placée à l'intérieur d'un coffret de protection envers le climat, comprenant du dessiccatif dans le coffret.

Chapitre 2. Stockage et récupération des données

La CR200 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 SCwin (voir la présentation), ou lorsqu'on utilise un programme d'édition direct (voir chapitres de 4 à 9).

2.1 Enregistrement de données sur la CR200

Dans le programme CRBASIC, l'instruction Data Table fixe la taille du tableau de données. On peut créer un maximum de 4 tableaux par programme.

Les tableaux de donnés sont stockés dans la mémoire Flash EEPROM interne. Les données restent en mémoire lorsque la CR200 n'est plus alimentée. Les données sont effacées lorsqu'un nouveau programme est chargé et exécuté. La flash EEPROM peut être utilisée pour plus de 50 000 cycles.

ATTENTION: Si une erreur de mémoire EEPROM est détectée, la centrale de mesure arrête d'exécuter le programme, et la LED rouge clignote deux fois à chaque intervalle de scrutation supposé. Le code « Trap Code » du tableau d'état aura la valeur 16. La centrale de mesure doit être renvoyée à Campbell Scientific afin de remplacer sa mémoire série flash EEPROM.

La CR200 stocke les données sauvegardées dans la flash EEPROM. La mémoire série flash EEPROM est aussi l'emplacement mémoire où est stocké le fichier appelé « Table Definition File » (TDF). Lorsqu'un fichier programme en CRBasic est envoyé à la CR200, le fichier TDF est extrait à partir de la version compilée du programme en CRBasic ; ce fichier TDF est stocké dans la mémoire série flash EEPROM. Toute la place restante peut alors être utilisée pour stocker des données. Sur la CR200, il y a 128Ko (ou 512Ko) de Flash EEPROM disponible pour les tableaux de données. Jusqu'à 5.12Ko de cette mémoire, peuvent être utilisés pour le fichier TDF (si la taille du fichier TDF dépasse cette limite, la compilation échoue). Si l'allocation de la taille mémoire pour un ou plusieurs tableaux de données, utilise trop de mémoire, il N'Y AURA PAS d'erreur de compilation. La CR200 modifiera automatiquement la taille allouée ; la CR200 fera la somme des tailles allouées, et multipliera chaque taille allouée par 90%, jusqu'à ce que la somme des tailles obtenues soit inférieure à la place disponible sur la mémoire série flash EEPROM.

2.2 Format de stockage interne

Tableau 2.2-1 Données IEEE4 de la CR200

Taille	Etendue	Résolution
4 octets	De $1,8^{E-38}$ à $1,7^{E38}$	24 bits (environ 7 digits)

Les calculs sont effectués et les données sont enregistrées directement au format IEEE à 4 octets et virgule flottante, au format binaire. Le tableau 2.2-1 liste l'étendue et la résolution des données au format IEEE4. Le temps est enregistré en tant qu'un nombre entier de secondes depuis minuit avec comme origine 1990, sous la forme d'un nombre à 4 octets.

2.3 Récupération des données

Les données peuvent être transférées vers un ordinateur via un lien de communication.

2.4 Format des données sur l'ordinateur

Le format de stockage des données sur l'ordinateur, peut être aussi bien de l'ASCII ou du binaire, selon le type de fichier demandé dans la boite de dialogue de collecte des données. Les données récupérées à partir d'une fenêtre d'affichage en temps réel, sont toujours enregistrées au format ASCII.

2.4.1 Informations de l'en-tête

Chaque fichier de données stocké sur le disque, a une en-tête ASCII qui est présente au début. 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 entrées sont décrites en suivant cette figure.

"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 Informations de l'en-tête

Format du fichier (File Format)

Le format du fichier sur le disque. Le TOA5 est un format ASCII. Le TOB1 est un format binaire. Cette information est utilisée par les fonctions historiques de graphisme et de conversion des fonctions, issues de PC9000.

Nom de la station (Station Name)

Adresse PakBus de la station depuis 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. CR2XX est utilisé pour CR200, CR205, CR210 et CR215.

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

Non utilisé avec la CR200.

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 programme (Program File)

Le nom du fichier programme qui était en train de tourner lorsque les données ont été créées.

Signature DLD (DLD Signature)

Signature du programme qui a crée 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 CR200 en ajoutant un tiret bas (_) et trois caractères mnémoniques décrivant le traitement des données.

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, par rapport 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 champ (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","1","CR2XX","","v0.1.06","EXPLS4.CR2","45828","AvgTemp"
"TMSTAMP","RECNBR","SoilT_Avg(1)","SoilT_Avg(2)","SoilT_Avg(3)","SoilT_Avg(4)"
"TS","RN","DegC","DegC","DegC","DegC"
"","","Avg","Avg","Avg","Avg"
"2002-03-20 11:00:00",1,15.498,15.9926,18.516,19.5019
"2002-03-20 12:00:00",2,15.4996,15.9993,18.5069,19.502
"2002-03-20 13:00:00",3,15.4963,16.0042,18.4975,19.496
```

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

TOA5	1	CR2XX		v1.0	EXPLS4.CR2	45828	AvgTemp
TMSTAMP	RECNBR	SoilT_Avg(1)	SoilT_Avg(2)	SoilT_Avg(3)	SoilT_Avg(4)		
TS	RN	DegC	DegC	DegC	DegC		
		Avg	Avg	Avg	Avg		
3/20/02 11:00	1	15.498	15.9926	18.516	19.5019		
3/20/02 12:00	2	15.4996	15.9993	18.5069	19.502		
3/20/02 13:00	3	15.4963	16.0042	18.4975	19.496		

2.4.3 Format de fichier binaire TOB1

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

```
"TOB1","1","CR2XX","","v0.1.06","EXPLS4.CR2","45828","AvgTemp"
"TMSTAMP","RECNBR","SoilT_Avg(1)","SoilT_Avg(2)","SoilT_Avg(3)","SoilT_Avg(4)"
"TS","RN","DegC","DegC","DegC","DegC"
"","","Avg","Avg","Avg","Avg"
"UINT4","UNIT4","IEEE4"," IEEE4"," IEEE4"," IEEE4"
```

(les lignes de données sont au format binaire et ne sont pas directement lisibles)

2.4.4 Format de fichier binaire TOB2

Le format binaire TOB2, a le même type d'en-tête que les autres formats. Les données TOB2 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 un intervalle de temps entre des enregistrements à intervalle de temps périodique, et que celui-ci n'apparaît pas sur une limite de 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 des données.

Quand les fichiers au format TOB2 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 aucun intervalle ne s'est produit. Si il se produit plus d'intervalle que ce qui a été prévu, il pourra y avoir moins de données dans le fichier, que ce qui avait été prévu.

Chapitre 3. Détails sur les mesures de la CR200

3.1 Séquence de mesures de tension analogique

La première étape avant de faire une mesure de tension analogique, est d'effectuer un étalonnage afin de mesurer l'offset dû à la prise de terre. L'étalonnage est effectué une fois, pour chaque mesure de tension analogique. La CR200 mesure une tension analogique à l'aide d'un échantillon, et effectue ensuite une conversion Analogique / Numérique. La conversion A / N est effectuée avec une technique d'approximation successive à 12 bits, qui donne une résolution de 1 partie parmi 4096, sur l'étendue de mesure de 2,5V (ce qui donne environ 0,6 mV).

Afin de réduire le bruit, 10 mesures rapides sont effectuées et moyennées afin de créer le résultat. Les mesures qui sont prises pour effectuer la moyenne, prennent chacune environ 26 microsecondes.

3.1.1 Etendue de mesure en tension

La CR200 n'a qu'une étendue de mesure, qui est de 0 à 2,5V. La résolution d'une mesure unipolaire est de 0,6 mV après conversion A/N.

3.1.2 Intégration : Moyenner un nombre de conversions A/N

L'intégration est utilisée pour réduire le bruit inclut dans une mesure. La CR200 utilise une certaine forme d'intégration numérique ; elle effectue 10 conversions A/N, les moyenne et retourne un seul résultat. Les conversions A/N sont effectuées toutes les 26 microsecondes.

Le fait de moyenner les valeurs réduira aussi le bruit du signal (par exemple en ce qui concerne une mesure de transducteur de pression qui varie légèrement à cause de fluctuations de pression dues au vent.

Le fait de faire des moyennes, a aussi une influence sur la résolution. La résolution vue sur le résultat numérique, est la résolution d'une seule conversion A/N (0,6mV) divisée par le nombre de conversions A/N (10).

3.2 Mesures de tension unipolaire

Une mesure de tension unipolaire est effectué sur une seule voie de mesure, par rapport à la terre. La CR200 ne fait pas de mesure différentielle entre deux voies de mesure en entrée.

3.3 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).

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'1ms a un résolution de 1000Hz.). 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 3.3-1. Si la mesure d'impulsion est moyennée, le résultat sera la valeur correcte.

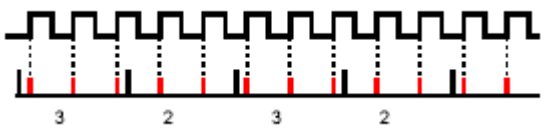


Figure 3.3-1 Nombre de comptage variant à l'intérieur d'un intervalle de scrutation.

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 100 Hz (2000 TPM x (1min / 60 sec) x 30 = 100). 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 1ms, la résolution serait de 2000 TPM. Avec un tel intervalle de scrutation, si tout était parfait, à chaque intervalle il devrait y avoir un comptage. Si cependant il y avait une légère variation à l'intérieur de l'intervalle, cela pourrait donner 2 comptages dans un intervalle, et aucun comptage dans le suivant , avec un résultat variant entre 0 et 4000 TPM !

3.4 Auto-calibrage

Un calibrage de l'offset dû à la terre, est effectué au début de chaque instruction de mesure qui comprend une mesure de tension. Le calibrage prend environ 400 microsecondes. Un seul calibrage est effectué, même si l'instruction de mesure comprend plusieurs répétitions.

La mesure de le tension batterie est vérifiée toutes les 8 secondes afin de s'assurer qu'elle est dans l'étendue de mesure permise pour un bon fonctionnement.

Chapitre 4. Langage de programmation – CRBasic

La CR200 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 des données. Le résultat de toutes les mesures sont 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 Format des introductions

4.1.1 Opérations mathématiques

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

TempF = TempC * 1.8 + 32

Avec la CR200 il peut y avoir 1 à 6 mesures de température (ou autre type de mesure). Au lieu d'avoir 6 noms de variables différents, une ligne de variable, avec un nom et 6 éléments, pourrait être utilisée. Une température de thermistance pourrait être appelée Temp. Avec une ligne de 6 éléments, le nom de chacune des températures seraient Temp(1) à Temp(6). La notion de ligne permet de compacter le code afin d'effectuer des opérations sur toutes les variables. Par exemple, pour convertir six températures d'une ligne variable, de °C à °F, on a :

```
For I=1 to 6
    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 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 CR200 :

Battery (*Dest*)

Battery est le nom que l'on entre au clavier/ mot clé (*keyword*), pour cette instruction. Le paramètre associé à l'instruction Battery, est la *Dest*ination, qui est le nom de la variable dans laquelle sera mise la tension. Si vous voulez mettre la valeur de la tension batterie dans la variable appelée BattVolt, vous devrez entrer le code suivant :

Battery (BattVolt)

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 CR200. Quand le compilateur de la CR200 voit un « ' », il ignore le reste de la ligne.

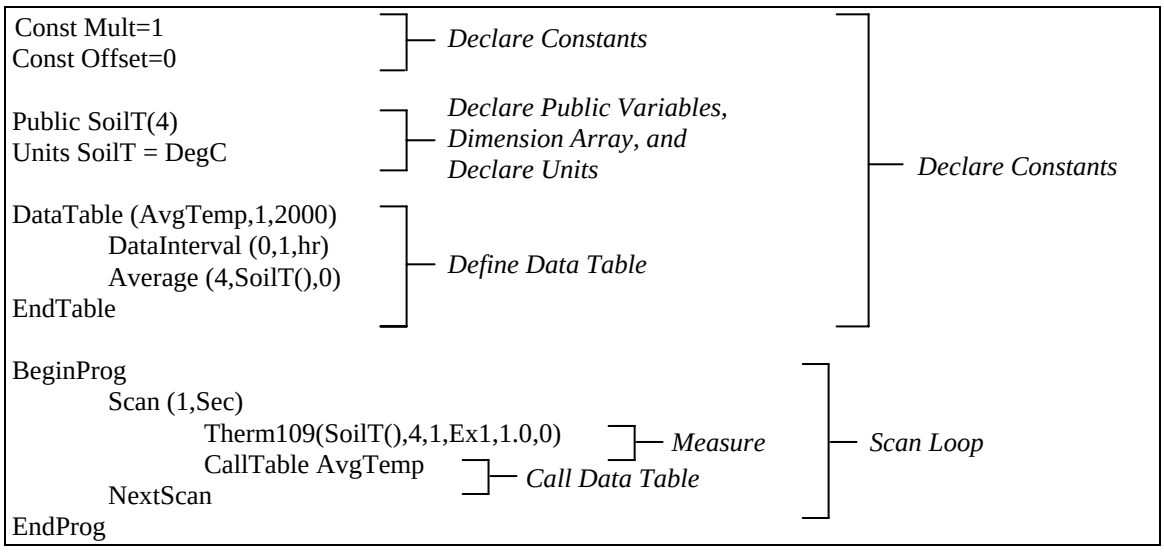
```
' La déclaration des variables débute ici
Public Start(6)    'Déclare la ligne de début de temps.
```

4.2 Séquence de programmation

Le tableau suivant décrit la structure typique d'un programme de CR200 :

Declaration	*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 valeurs que l'utilisateur peut visualiser lorsque le programme est en cours d'exécution*
Dimension variables	*Le nombre de chaque mesure qui sera effectuée,*
Define Alises	*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 CR200*
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

TOA5	1	CR2XX		v1.0	EXPLS4.CR2	45828	AvgTemp
TMSTAMP	RECNBR	SoilT_Avg(1)	SoilT_Avg(2)	SoilT_Avg(3)	SoilT_Avg(4)		
TS	RN	DegC	DegC	DegC	DegC		
		Avg	Avg	Avg	Avg		
3/20/02 11:00	1	15.498	15.9926	18.516	19.5019		
3/20/02 12:00	2	15.4996	15.9993	18.5069	19.502		
3/20/02 13:00	3	15.4963	16.0042	18.4975	19.496		

Le stockage des données suit une structure fixe dans la CR200, afin d'optimiser le temps et l'espace nécessaire. Les données sont enregistrées dans des tableaux tels que :

La CR200 peut être programmée avec jusqu'à 4 tableaux de données définis par l'utilisateur. Le programme de l'utilisateur détermine les valeurs qui seront enregistrées, et l'ordre dans lequel ce sera fait. La CR200 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 CR200 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 CR200 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 (AvgTemp,1,2000)
DataInterval(0,1,hr)
Average(4,SoilT(),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 de la CR200. La condition de basculement peut être une variable, une expression, ou une constante. La condition de basculement est vraie si elle n'est pas égal 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.

DataInterval(TintoInt,Interval,Units)
DataInterval(0,1,hr)

L'instruction DataInterval est une instruction qui modifie la condition pour laquelle les données seront stockées. Les trois paramètres sont le temps à l'intérieur de l'intervalle de temps, la durée de l'intervalle de temps auquel les données sont stockées, et l'unité de temps de l'intervalle. L'exemple donné enregistrera des valeurs à chaque valeur du temps « 0 » dans l'intervalle de temps faisant « 1 » heure, avec la valeur du 0 calé par rapport à l'horloge interne de la centrale de mesure.

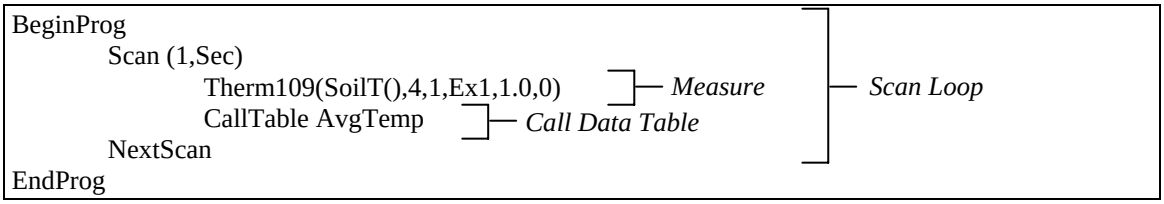
Les instructions de sauvegarde comprises dans la définition du tableau de données, déterminent les valeurs qui seront enregistrées dans le tableau. Le tableau doit être appelé par le programme si l'on veut que les instructions de sauvegarde soient exécutées. Ceci se produira à chaque fois que de nouvelles mesures sont effectuées et que le tableau de données est appelé. 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, DisableVar)
Average(4,SoilT(),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, 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 qui suit a « 0 » comme valeur pour le paramètre de « passage outre », ainsi toutes les valeurs seront prises en compte dans le calcul de la moyenne.

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.



L'instruction « Scan », détermine à quelle fréquence les mesures comprises dans la boucle, sont effectuées :

Scan(Intervalle,Unité)
Scan(1,SEC)

L'instruction « Scan » a deux paramètres. L'*intervalle* est l'intervalle de temps entre deux scrutations. L'*Unité*, est l'unité de temps à utiliser pour l'intervalle. L'intervalle des temps maximum que l'on peut donner à l'intervalle est d'une minute. Dans l'exemple, il est d'une seconde.

4.4 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 4.4-1).

Tableau 4.4-1 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

4.5 Evaluation des expression logiques

4.5.1 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 », ou « haut » et « bas » 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 4.5-1. La CR200 évalue le test ou le paramètre en tant que numéro ; 0 si le résultat est faux, différent de 0 si c'est vrai.

Tableau 4.5-1 Synonymes pour « vrai » ou « faux »

Constante prédéfinie	Vrai (-1)	Faux (0)
Synonyme	Haut	Bas
Synonyme	Activé	Désactivé
Synonyme	Oui	Non
Synonyme	Basculement	Pas de basculement
Nombre	≠ 0	0
Port numérique	5 Volts	0 Volts

4.5.2 Evaluation de l'expression

Les test conditionnels nécessitent à la CR200 d'évaluer une expression, et de suivre une voie si la condition est vraie, ou une autre 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 CR200 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 8. Une condition peut prendre en compte plusieurs « And » et plusieurs « Or ».

4.5.3 Résultats numériques de l'évaluation de l'expression

La fonction d'évaluation de la CR200, évalue une expression, et donne un chiffre en résultat. Une expression conditionnelle utilise le chiffre afin de décider quel chemin prendre. L'expression 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 CR200, 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 CR200 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 8). 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 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.

4.6 Les drapeaux (Flags)

N'importe quelle variable peut être utilisé en tant que « drapeau », pour autant que des tests logiques soient utilisés avec le CRBasic. Si la valeur de la variable est différent 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 (paragraphe 4.5).

4.7 Les types de paramètre

Les paramètres des instructions permettent d'entrer différent types de choses en entrée ; ces différents types d'entrée sont listées ci-dessous, et sont repris dans les chapitres suivant, 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 4.7-1 donne la liste des longueurs maximales et des caractères autorisés pour ce qui est du nom des variables, des lignes de constante etc.

Tableau 4.7-1 Règles pour l'établissement des noms

Nom pour	Longueur maximum (nombre de caractères)	Caractères admis
Variable ou Ligne de données	16	Les lettre 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	

4.7.1 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 4.5.3). 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.

4.7.2 Lignes de données de multiplicateurs et d'offsets pour la calibrage 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 calibrés 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.

VoltSE(Dest,Reps,SEChan,Mult,Offset)

```
'Facteurs de calibrage :
Mult(1)=0.123 : Offset(1)=0.23
Mult(2)=0.115 : Offset(2)=0.234
Mult(3)=0.114 : Offset(3)=0.224
VoltSE(Pressure(),3,1,Mult(),Offset())
```

4.8 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 champs Fieldname est toujours une ligne de donnée, même s'il ne 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=AvgTemp.SoilT(1,1)-AvgTemp.SoilT(1,2)

pourrait être utilisée dans l'exemple de programme (paragraphe 4.3) afin de calculer le changement de température sur la première thermistance, entre la moyenne la plus récente et celle qui a été enregistrée pour l'heure précédente.

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 :

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

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 appliquée 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
```

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 * Rayon    'Calcul de la circonférence
Aire = PI * ( Rayon ^ 2)    'Calcul de l'aire
```

Dim

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

Dans la CR200 il y a de la place pour 60 variables déclarées en tant que **Dim** *et/ou* **Public** *et* les instructions de sauvegarde qui créent des espaces mémoires « *scratch* » pour le calcul intermédiaire.

Syntaxe

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.

Les argument des « *subscripts* » ont la synthaxe suivante :

Taille [taille, taille]

En CRBasic l'option de base est toujours 1.
Cela signifie que plus petit nombre de dimension est 1 et non 0.

```
Dim A(3)
```

La CR200 ne permet d'utiliser des ligne de données qu'à une seule dimension. 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.

Public

Déclare une variable en tant que variable publique, ce qui la rend accessible à la consultation dans le tableau « Public » de la CR200.

Dans la CR200 il y a de la place pour 60 variables déclarées en tant que **Dim** *et/ou* **Public** *et* les instructions de sauvegarde qui créent des espaces mémoires « *scratch* » pour le calcul intermédiaire.

Syntaxe

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 » :

```
Public x, y, z(3)
Public w
```

Units

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

Syntaxe

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 contenue dans un champs 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 CR200.

Exemple

Dim Temp(1)

Units Temp(1) = Deg_C

Sub, Exit Sub, End Sub

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

Syntaxe

Sub *Subname*
[*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. Etant donné que les noms des sous-programmes sont reconnus par toutes les procédures et tous les modules, le nom du sous-programme (*subname*), ne peut pas être le même que celui d'une autre variable globalement reconnue dans le programme.
Statementblock	N'importe quel groupement d'instructions qui est exécuté à l'intérieur 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.

Les sous-programmes sont des procédures qui peuvent 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. Reportez-vous à l'instruction « Call » pour voir comment appeler un sous-programme.

La CR200 ne permet pas de passer des variables dans le sous-programme. Les sous-programmes peuvent fonctionner avec les variables, Dim ou Public, déclarées dans le programme général.

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 d'étonnant résultat

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, l'appareil de sauvegarde vers lequel on envoie les données, 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</td><td>Ne pas effectuer de basculement</td></tr>
<tr><td>≠ 0</td><td>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 le langage de programmation au chapitre 4.

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)

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. 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.

<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 minutes ou en heures, dans le paramètre d'Unités. On entre la valeur 0 si on souhaite que la sauvegarde s'effectue à chaque intervalle de scrutations. L'intervalle de temps minimum est la minute.</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>Code numérique</th><th>Unité</th></tr>
<tr><td>MIN</td><td>3</td><td>Minutes</td></tr>
<tr><td>HR</td><td>4</td><td>Heures</td></tr>
</table></td></tr>
</table>

6.3 Instructions de sauvegarde

Average (Reps, Source, DisableVar)

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

<table>
<tr><th>Paramètres
& type de donnée</th><th>Entrée</th></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>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 ou <>0), la valeur du moment ne sera pas incluse 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 prend en compte la valeur actuelle de la variable</td></tr>
<tr><td>≠ 0</td><td>On ne prend pas en compte la valeur actuelle de la variable</td></tr>
</table>
</td></tr>
</table>

ETo (Temp, RH, u2, Rs, Longitude, Latitude, Altitude, DisableVar)

L'instruction ETo permet de calculer l'évapotranspiration, à partir de mesures effectuées par l'utilisateur, et de constantes définies dans le programme. D'autres données météorologiques sont aussi récupérées. Cette instruction est plus particulièrement adaptée aux intervalles de sauvegarde courts (inférieurs à 24 heures). Pour des sauvegardes journalières, on utilisera « EtoDaily ».

Cette instruction donne comme résultat 5 valeurs, qui sont calculées pour la période de sauvegarde définie. Les valeurs sont : la température moyenne (en °C), l'humidité relative moyenne (en %), la vitesse du vent moyenne (en m/sec), le rayonnement solaire moyen (en W/m²), et l'évapotranspiration (en mm). Cette instruction utilise l'équation d'évapotranspiration de référence standardisée de type ASCE (*ASCE Standardized Reference Evapotranspiration equation*).

EtoDaily (Temp, RH, u2, Rs, Longitude, Latitude, Altitude, DisableVar)

L'instruction EtoDaily permet de calculer l'évapotranspiration, à partir de mesures effectuées et de constantes définies par l'utilisateur dans le programme. D'autres données météorologiques sont aussi récupérées. Cette instruction est plus particulièrement adaptée aux intervalles de sauvegarde importants (par exemple 24 heures). Pour des intervalles de sauvegarde plus courts, on utilisera « ETo ».

Cette instruction donne comme résultat 9 valeurs, qui sont calculées pour la période de sauvegarde définie. Les valeurs sont : la température maximum (en °C), le moment où a eu lieu le température moyenne (mois/jour/année heure:minute), la température minimum, le moment où a eu lieu la température minimum (mois/jour/année heure:minute), l'humidité relative maximum (en %), l'humidité relative minimum (en %), la vitesse moyenne du vent (en m/sec), le rayonnement solaire moyen (en MJ/m²), et l'évapotranspiration Eto (en mm). Cette instruction utilise l'équation d'évapotranspiration de référence standardisée de type ASCE (*ASCE Standardized Reference Evapotranspiration equation*).

Paramètres *& type de donnée*	**Entrée**
Temp *Variable*	La variable dans laquelle la mesure de température est stockée, et qui sera utilisée dans le calcul de l'ET0. La température doit être en °C.
RH *Variable*	La variable dans laquelle la mesure d'Humidité Relative est stockée, et qui sera utilisée dans le calcul de l'ET0. La mesure d'HR doit être en %.
u2 *Variable*	La variable dans laquelle la mesure de vitesse du vent est stockée, et qui sera utilisée dans le calcul de l'ET0. Elle doit être en m/sec et est supposée être prise à 2m de hauteur par rapport au sol.
Rs *Variable*	La variable dans laquelle la mesure de rayonnement solaire est stockée, et qui sera utilisée dans le calcul de l'ET0. Elle doit être en W/m².
Longitude *Variable ou constante*	La variable ou la constante qui fournit la longitude de la station météo. La longitude doit être comprise entre –180 et + 180.
Latitude *Variable ou constante*	La variable ou la constante qui fournit la latitude de la station météo. La latitude doit être comprise entre –90 et + 90.
Altitude *Variable ou constante*	La valeur de la hauteur où se trouve la station météo, en mètres.
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. Par exemple, si la variable de désactivation est différente de 0, on ne cherchera pas à trouver une nouvelle valeur minimum sur l'intervalle de sauvegarde. Ainsi la valeur minimum qui est éventuellement sauvegardée, est la valeur minimum des valeurs en entrée, qui se sont produites lorsque la variable de désactivation portait la valeur 0. **Valeur** | **Résultat** 0 | On prend en compte la valeur actuelle dans les calculs ≠ 0 | On ne prend pas en compte la valeur actuelle dans les calculs

FieldNames « list of fieldnames »

L'instruction « FieldNames » peut être utilisée afin de remplacer le nom des champs qui sont crées par la CR200, et attribués aux colonnes des tableaux de sauvegarde. L'instruction **Fieldname** doit être positionnée immédiatement après l'instruction de sauvegarde qui crée le tableau de sauvegarde. Le nom des champs est limité à 19 caractères. Des dénominations individuelles peuvent être entrées pour chaque résultat généré par l'instruction de sauvegarde précédente, ou bien on peut utiliser une ligne de données afin de donner un nom à plusieurs champs. Lorsque le programme est compilé, le compilateur déterminera combien de champs sont créés. Si la liste de noms est supérieure au nombre de champs, les noms supplémentaires seront 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))
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))
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))
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).

Maximum (Reps, Source, DisableVar, Time)

L'instruction de maximum est une instruction qui enregistre le maximum de la valeur indiquée dans la variable source, durant l'intervalle de sauvegarde. 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 ».

Paramètres *& type de donnée*	**Entrée**
Reps *Constante*	Le nombre de maximum 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.
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 du moment ne sera pas prise 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. **Valeur** | **Résultat** 0 | On prend en compte la valeur actuelle dans le calcul ≠ 0 | On ne prend pas en compte la valeur actuelle dans le calcul
Time *Constante*	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. **Valeur** | **Résultat** 0 | On n'enregistre pas l'heure 1 | On enregistre l'heure

Minimum (Reps, Source, DisableVar, Time)

L'instruction de minimum est une instruction qui enregistre le minimum de la valeur indiquée dans la variable source, durant l'intervalle de sauvegarde. 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 ».

Paramètres *& type de donnée*	**Entrée**
Reps *Constante*	Le nombre de minimum 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.
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 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. **Valeur** | **Résultat** 0 | On prend en compte la valeur actuelle dans le calcul ≠ 0 | On ne prend pas en compte la valeur actuelle dans le calcul
Time *Constante*	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. **Valeur** | **Résultat** 0 | On n'enregistre pas l'heure 1 | On enregistre l'heure

Sample (Reps, Source)

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.

StdDev (Reps, Source, 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.
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 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 actuelle dans le calcul
≠ 0	On ne prend pas en compte la valeur actuelle dans le calcul

Totalize (Reps, Source, DisableVar)

Cette instruction enregistre le total de la (les) valeur(s) source, sur la durée de l'intervalle de sauvegarde.

Paramètres *& type de donnée*	**Entrée**
Reps *Constante*	Le nombre de totaux 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.
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 du moment 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.

Valeur	**Résultat**
0	On prend en compte la valeur actuelle dans le calcul
≠ 0	On ne prend pas en compte la valeur actuelle dans le calcul

WindVector (Speed/East, Direction/North, DisableVar, 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, 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><th>Paramètres
& type de donnée</th><th>Entrée</th></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 variables contenant des éléments pour chacune des répétitions.</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 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 actuelle dans le calcul</td></tr>
<tr><td>≠ 0</td><td>On ne prend pas en compte la valeur actuelle dans le calcul</td></tr>
</table></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</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>

Lorsqu'elle est utilisée avec des capteurs polaires, l'instruction fait un modulo par 360, de la direction du vent, ce qui permet à la direction du vent (en degrés) de varier entre 0 et 360, 0 et 540, moins de 0, ou plus de 540. Il est utile de pouvoir traiter facilement des valeurs négatives, dans le cas où il est difficile d'atteindre la girouette qui n'est pas orientée correctement. Si par exemple la lecture de la girouette est de 0 degrés au lieu de la valeur réelle de 340 degrés, la solution la plus simple est d'entrer un offset de –20 dans l'instruction qui mesure la girouette, et le résultat sera la valeur réelle en degrés de 0 à 360, après le modulo.

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.

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 :

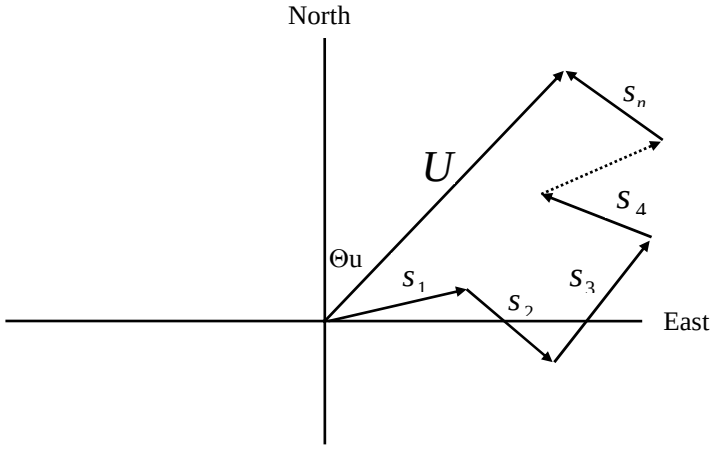


Figure 6.3-1 Vecteurs échantillonnés en entrée.

Sur la figure 6.3-1, 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 $\overline{U} = U / N$.

Scalaire de la vitesse moyenne 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 de la direction moyenne du vent, Θ_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 :

$Ux=(\Sigma(Ue_i/U_i))/N$

$Uy=(\Sigma(Un_i/U_i))/N$

où $U_i=(Ue_i^2+Un_i^2)^{1/2}$

Ecart type de la direction du vent, $\sigma(\Theta 1)$, équation 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.

Résultante de la vitesse moyenne du vent horizontal, $\bar{U}$:

$\bar{U} =(Ue^2+Un^2)^{1/2}$

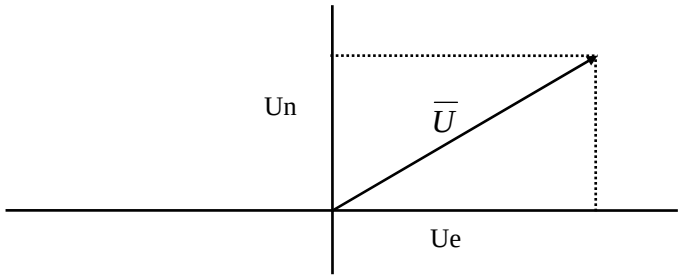


Figure 6.3-2 Vecteur vent moyen

et 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 :

$Ue=(\Sigma Ue_i)/N$

$Un=(\Sigma Un_i)/N$

Résultante de la direction moyenne 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)

$\text{Cos}\ (\Theta_i') = U_i/s_i$; où $\Theta_i' = \Theta_i-\Theta_u$

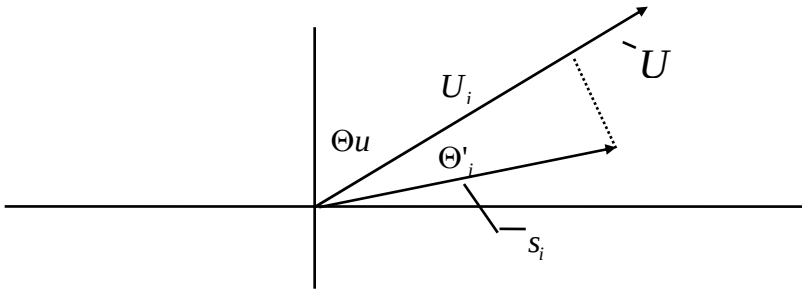


Figure 6.3-3 Ecart type pour la direction.

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.

$$(\sigma(\Theta u'))^2=\sum_{i=1}^{N} (\Theta_i')^2/N = 2(1-\bar{U}/S)- \sum_{i=1}^{N}(((\Theta_i')^2 s_i')/NS$$

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 données de vent provenant de différents sites, par Campbell Scientific aux USA. 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.

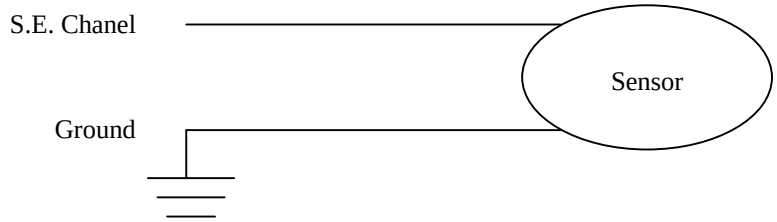
On parvient à la formule finale en convertissant les radians en degrés (57.296 degrés / radian).

$$\sigma(\Theta u) = (2(1- \bar{U}/S))^{1/2} = 81(1- \bar{U}/S)^{1/2}$$

Chapitre 7. Instructions de mesure

7.1 Mesures de tension

VoltSE (Dest, Reps, SEChan, Mult, Offset)

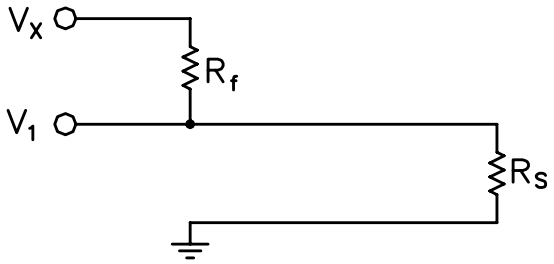


Cette instruction mesure la tension d'une entrée unipolaire, par rapport à la terre.Avec un multiplicateur égal à 1, et un offset égal à 0, le résultat est en millivolts.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou Ligne de données*	Les variables dans lesquelles on stocke le résultat des instructions. Quand on utilise des répétitions, les données sont sauvegardées dans des lignes de donnée ayant le nom de la variable. Une ligne de donnée doit être dimensionnée afin d'avoir des éléments pour chaque répétition.
Reps *Constante*	Le nombre de répétitions pour la mesure ou l'instruction.
SEChan *Constante*	Le numéro de la voie unipolaire sur laquelle on effectue la première mesure. Lorsque des répétitions sont utilisées, des mesures seront effectuées sur les voies de mesure consécutive à celle indiquée ici.
Mult, Offset *Constante, Variable, Ligne de donnée ou Expression*	Un multiplicateur et un offset qu'on utilise pour mettre à l'échelle les résultats de la 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 afin d'obtenir une valeur brute pour résultat.

7.2 Demi pont

ExDelSE (Dest, Reps, SEChan, ExChan, ExmV, Delay, Mult, Offset)



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 la tension mesurée en millivolts.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou Ligne de données*	Les variables dans lesquelles on stocke le résultat des instructions. Quand on utilise des répétitions, les données sont sauvegardées dans des lignes de donnée ayant le nom de la variable. Une ligne de donnée doit être dimensionnée afin d'avoir des éléments pour chaque répétition.
Reps *Constante*	Le nombre de répétitions pour la mesure ou l'instruction.
SEChan *Constante*	Le numéro de la voie unipolaire sur laquelle on effectue la première mesure. Lorsque des répétitions sont utilisées, des mesures seront effectuées sur les voies de mesure consécutive à celle indiquée ici.
ExChan *Constante*	Le numéro de la voie d'excitation où effectuer la première mesure. **Code lettre** / **Code / n° de voie** / **Résultat** EX1 / 1 EX2 / 2 Les voies d'excitation commutée sont activées pendant les mesures / désactivées entre les mesures.
ExmV *Constante*	La tension d'excitation en mV. **Code** / **Résultat** mV2500 / Donne une excitation de 2500mV mV5000 / Donne une excitation de 5000mV
Delay *Constante*	Le temps, en microsecondes, est le délai après la stabilisation de l'excitation et avant de faire la mesure de tension. Le délai minimum est de 500 microsecondes ; toute valeur inférieure à celle-ci causera une erreur à la compilation.
Mult, Offset *Constante, Variable, Ligne de donnée ou Expression*	Un multiplicateur et un offset qu'on utilise pour mettre à l'échelle les résultats de la mesure brute.

Therm109 (Dest, Reps, SEChan, ExChan, Mult, Offset)

L'instruction Therm109 est utilisée afin de mesurer la thermistance 109. Cette instruction effectue une mesure de demi pont, puis en déduit le résultat en utilisant l'équation de Steinhart-Hart afin de calculer la température de la thermistance à partir de la résistance de la thermistance. Le résultat de la mesure effectuée avec un multiplicateur égal à 1 et un offset égal à 0, est la température en °C.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou Ligne de données*	Les variables dans lesquelles on stocke le résultat des instructions. Quand on utilise des répétitions, les données sont sauvegardées dans des lignes de donnée ayant le nom de la variable. Une ligne de donnée doit être dimensionnée afin d'avoir des éléments pour chaque répétition.
Reps *Constante*	Le nombre de répétitions pour la mesure ou l'instruction.
SEChan *Constante*	Le numéro de la voie unipolaire sur laquelle on effectue la première mesure. Lorsque des répétitions sont utilisées, des mesures seront effectuées sur les voies de mesure consécutive à celle indiquée ici.
ExChan *Constante*	Le numéro de la voie d'excitation où effectuer la première mesure. **Code lettre** / **Code / n° de voie** / **Résultat** EX1 / 1 EX2 / 2 Les voies d'excitation commutée sont activées pendant les mesures / désactivées entre les mesures.
Mult, Offset *Constante, Variable, Ligne de donnée ou Expression*	Un multiplicateur et un offset qu'on utilise pour mettre à l'échelle les résultats de la mesure brute. Avec un multiplicateur de 1 et un offset de 0, la température est mesurée en °C. Avec un multiplicateur de 1,8 et un offset de 32, elle est mesurée en °F.

7.3 Excitation / Sortie de tension en continu

ExciteV (ExChan, ExmV,)

Cette instruction applique l'excitation spécifiée à la voie d'excitation commutée indiquée. Une fois que la voie d'excitation est mise sous tension de 2500 ou 5000mV par l'instruction ExciteV, la voie d'excitation reste dans cet état jusqu'à ce qu'une autre instruction affecte une nouvelle tension d'excitation. Le fait de laisser les voies d'excitation en position activée, accroît la consommation de la CR200 d'environ 300mA, en plus du courant fourni par la voie d'excitation. Si une tension d'excitation continue n'est pas nécessaire, il faut utiliser une autre instruction qu'ExciteV afin de ramener la tension à 0mV avant la fin de l'exécution du programme, afin de réduire la consommation en courant.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>ExChan
Constante</td><td>Le numéro de la voie d'excitation où effectuer la première mesure.
<table>
<tr><th>Code lettre</th><th>Code / n° de voie</th><th>Résultat</th></tr>
<tr><td>EX1
EX2</td><td>1
2</td><td>Les voies d'excitation commutée sont activées pendant les mesures / désactivées entre les mesures.</td></tr>
</table></td></tr>
<tr><td>ExmV
Constante</td><td>La tension d'excitation en mV.
<table>
<tr><th>Code</th><th>Résultat</th></tr>
<tr><td>mV0</td><td>Donne une excitation de 0mV</td></tr>
<tr><td>mV2500</td><td>Donne une excitation de 2500mV</td></tr>
<tr><td>mV5000</td><td>Donne une excitation de 5000mV</td></tr>
</table></td></tr>
</table>

7.4 Mesure ne nécessitant pas de capteur

Battery (Dest)

Cette instruction lit l'état de la tension de la batterie, et stocke le résultat dans une variable de destination. L'unité de mesure de la batterie est le Volt.

7.5 Port de contrôle E/S

AnalogPortGet (Dest, SEChan)

La CR200 a la capacité d'utiliser des voies de mesure analogiques en tant qu'entrée numériques (AnalogPortGet) ou en tant que sortie (AnalogPortSet). Lorsqu'elles sont utilisées en tant qu'entrées, une résistance de 100 Kohm doit être connectée entre la voie de mesure et la masse, afin de s'assurer que la voie est à l'état bas lorsqu'elle n'est pas commandée par un signal extérieur. Le fait d'exécuter l'instruction AnalogPortGet, configure la voie en tant qu'entrée (c'est à dire que la voie n'est pas mise en état haut ni en état bas) ; les instructions AnalogPortSet précédentes, deviendront inactives.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Variable ou Ligne de données</td><td>La variable qui contient le code résultat pour une voie spécifiée. Les codes sont :
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>La voie est à l'état bas (Low)</td></tr>
<tr><td>1</td><td>La voie est à l'état haut (High)</td></tr>
</table></td></tr>
<tr><td>SEChan
Constante</td><td>Le numéro de la voie unipolaire à tester (entre 1 et 5).</td></tr>
</table>

AnalogPortSet (SEChan, State)

La CR200 a la possibilité d'utiliser les voies analogiques en tant qu'entrées numériques (AnalogPortGet) ou en tant que sorties (AnalogPortSet).

Paramètres *& type de donnée*	**Entrée**
SEChan *Constante*	La voie unipolaire à tester (entre 1 et 6 ; 6 est la voie marquée P_SW).
State *Constante, Variable ou Expression*	La variable qui contient le code de résultat pour la voie spécifiée. Les codes sont : **Valeur** / **Résultat** 0 — La voie est à l'état bas (*Low*) ≠ 0 — La voie est à l'état haut (*High*)

PeriodAvg (Dest, SEChan, PAOption, Cycles, Timeout, Port, Mult, Offset)

Cette instruction mesure la période d'un signal, sur une voie unipolaire. Le nombre de cycles spécifiés, sont espacés dans le temps avec une résolution de 136ns, ce qui donne une résolution de la mesure de période égale à 136ns, divisé par le nombre de cycles choisis.

Le signal doit avoir une transition entre moins de 0,9 volt et plus de 2,1 volts, afin d'être comptabilisée. Un genre de signal typiquement mesurable, et un signal carré 0 à 2,5V.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou Ligne de données*	Les variables dans lesquelles on stocke le résultat des instructions. Quand on utilise des répétitions, les données sont sauvegardées dans des lignes de donnée ayant le nom de la variable. Une ligne de donnée doit être dimensionnée afin d'avoir des éléments pour chaque répétition.
SEChan *Constante*	Le numéro de la voie unipolaire sur laquelle on effectue la première mesure. Lorsque la voie est indiquée avec un signe négatif, toutes les répétitions sont effectuées sur cette même voie.
PAoption	Indique si on prend la période en µs ou la fréquence en kHz. **Code numérique** / **Résultat retourné** 0 — La période du signal 1 — La fréquence du signal
Cycles *Constante*	Le nombre de cycles à mesurer pour effectuer le calcul de la moyenne.
Timeout *Constante*	La durée maximum (en msec) qu'attendra la centrale afin de mesurer le nombre de cycles nécessaires au calcul de la moyenne.
Port *Constante*	Le port de contrôle ou le voie analogique qu'il faut mettre à l'état haut au début de la mesure, et à l'état bas une fois qu'elle est effectuée. Le port peut être utilisé afin de commuter l'alimentation d'un capteur tel que le réflectomètre de teneur en eau CS625 **Code** / **Description** 0 — Aucune C1 — Port de contrôle n° 1 C2 — Port de contrôle n° 2 3 — Voie de mesure analogique n° 3 4 — Voie de mesure analogique n° 4 5 — Voie de mesure analogique n° 5 P_SW — Voie de mesure analogique n° 6 / P_SW
Mult, Offset *Constante, Variable, Ligne de donnée ou Expression*	Un multiplicateur et un offset qu'on utilise pour mettre à l'échelle les résultats de la mesure brute.

PortGet (Dest, Port)

La fonction PortGet est utilisée afin de lire l'état d'un ou de deux des ports de contrôle.

Remarques :

Cette instruction lit l'état du port de contrôle spécifié, et place le résultat dans une variable de destination. Le fait d'exécuter l'instruction PortGet, configure le port en tant qu'entrée (c'est à dire qu'il n'est pas mis en position haute ou basse) ; les instructions PortSet précédentes, deviendront inactives.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable*	C'est la variable dans laquelle est envoyée le résultat de l'instruction. Un « 1 » est indiqué si le port est à l'état haut ; « 0 » s'il est à l'état bas.
Port *Constante*	Le port de contrôle (1 ou 2) pour lequel on cherche à connaître l'état.

PortSet (Dest, State)

Cette fonction mettra le port de contrôle spécifié à l'état haut ou bas.

Paramètres *& type de donnée*	**Entrée**
Port *Constante*	Le port de contrôle (1 ou 2) pour lequel on cherche à connaître l'état.
State *Constante, Variable, Ligne de donnée ou Expression*	L'état dans lequel on va mettre le port de contrôle.

Code	**Valeur**	**Etat**
Vrai	0	Bas (*Low*)
Faux	≠0	Haut (*High*)

PulseCount (Dest, Pchan, PConfig, POption, Mult, Offset)

L'instruction PulseCount est utilisée afin de mesurer les comptages d'impulsions ou la fréquence sur l'un des ports de contrôle ou port de comptage d'impulsion.

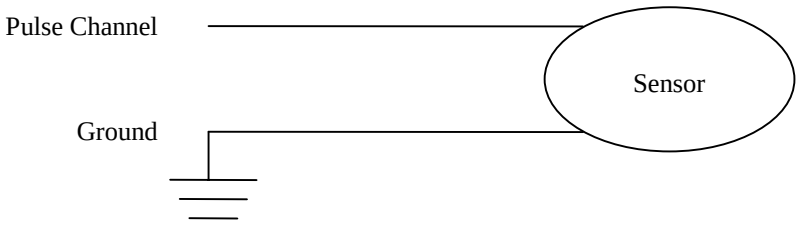


Schéma de câblage pour un capteur à comptage d'impulsion.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou ligne de donnée*	C'est la variable dans laquelle est envoyée le résultat de l'instruction. Lorsque le nombre de répétitions est différent de 1, les résultats sont stockés dans des lignes de donnée, avec le nom de la variable. Il faut alors dimensionner la ligne de donnée afin d'avoir de la place pour chaque répétition.
PChan *Constante*	La voie utilisée pour la mesure

Code alphabétique	**Code numérique**	**Voie**
P_SW	6	P_SW
P_LL	0	P_LL
C1	1	C1
C2	2	C2

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>PConfig
Constante</td><td>Ce code spécifie le type d'impulsion mesurée en entrée.
<table>
<tr><th>Code</th><th>Configuration de l'entrée</th></tr>
<tr><td>0</td><td>Impulsion de tensions</td></tr>
<tr><td>1</td><td>Courant alternatif bas niveau (seulement sur P_LL)</td></tr>
<tr><td>2</td><td>Contact sec (mettre la résistance nécessaire sur C1, C2 ou P_LL)</td></tr>
</table></td></tr>
<tr><td>POption
Conctante</td><td>Le code qui détermine si le résultat brut (multiplicateur de 1, offset de 0) est affiché en tant que comptage ou fréquence. La moyenne glissante peut être utilisée afin de lisser les valeurs lorsque la fréquence est faible par rapport à la vitesse de scrutation, et que la différence de fréquence entre les scrutations (scan) peut être importante.
<table>
<tr><th>Code</th><th>Résultat</th></tr>
<tr><td>0</td><td>Comptages</td></tr>
<tr><td>1</td><td>Fréquence (Hz) = comptages / intervalle scanné</td></tr>
<tr><td>>1</td><td>Moyenne glissante sur la fréquence. Le nombre entré est la période de temps sur laquelle la fréquence est moyennée, en milisecondes.</td></tr>
</table></td></tr>
<tr><td>Mult, Offset
Constante, Variable, Ligne de donnée ou Expression</td><td>Un multiplicateur et un offset qu'on utilise pour mettre à l'échelle les résultats de la mesure brute. Voir la description de la mesure afin de connaître l'unité de la mesure brute ; un multiplicateur égal à 1 et un offset égal à 0, sont nécessaires afin de fournir un résultat en unité de valeur brute.</td></tr>
</table>

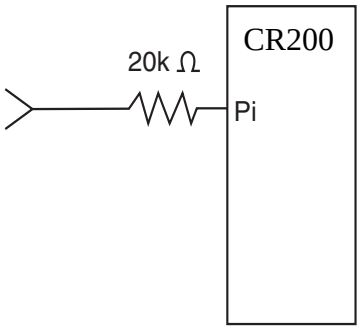


Figure 7.5-1 Conditionnement pour des impulsions (forte) tension.

La tension maximum d'entrée sur la voir P_LL est de + /- 20V. Elle est de 6,5V CC pour C1 & C2, et 4V CC pour P_SW. Reportez-vous à la figure 7.5-1 si vous devez réduire la tension en entrée pour C1, C2 ou P_SW.

Voltage Pulse (P_LL, C1, C2), impulsions en tension

Durée minimum de l'impulsion : 400 microsecondes
Fréquence maximum : 1KHz (50% du cycle)
Seuil bas : 1,5V*
Seuil Haut : 3,5V*

Low Level AC (P_LL seulement), Courant Alternatif bas niveau

Hystérésis en entrée : 15mV
Tension maximum en entrée : 20 V de pic à pic
Tension d'entrée et étendue de fréquence :
De 20mV à 2 V CA pic à pic, de 1.0 Hz à 1 KHz.

Switch closure, contact sec

Un contact sec est relié entre P_SW, P_LL, C1 ou C2, et une masse analogique. P_SW à une résistance interne de 100Kohm, qui met le signal vers le haut quand le contact est ouvert. Si l'on veut utiliser une des autres voies (autre que P_SW), afin de faire une mesure de contact sec, une résistance de 100Kohm doit être reliée entre la voie utilisée et la borne « Batt + » de la centrale. Quand le contact sera fermé, la voie d'impulsion sera alors mise à la masse. Le comptage est incrémenté lorsque le contact s'ouvre à nouveau.

* L'offset CC doit être inférieur à 0,5 V

Fréquence maximum : 100 Hz
Durée minimum du contact sec en état fermé : 5 ms
Durée minimum du contact sec en état ouvert : 5 ms
Durée maximum de rebond : 1 ms d'état ouvert sans être comptabilisé

SDI12Recorder (Dest, OutString, Multiplier, Offset)

L'instruction SDI12Recorder est utilisée afin de récupérer les données issues d'un capteur SDI12.

Chaque fois que l'instruction SDI12Recorder est exécutée, elle envoie les instructions suivantes au capteur, afin qu'il effectue la mesure : (adresse)M ! suivi de (adresse)D0 !. La commande « M! » permet au capteur d'effectuer le mesure. « D0 ! » est celle qui demande les données. L'instruction SDI12Recorder peut aussi être utilisée afin d'envoyer au capteur, des commandes étendues, comme des variables de la table « public » de la centrale d'acquisition.

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou ligne de donnée*	C'est la variable ou la ligne de donnée de variables dans laquelle est envoyée le résultat de la mesure. Lorsqu'on demande les données à un capteur, la Destination doit être dimensionnée correctement afin de pouvoir recevoir toutes les données, sinon une erreur de « variable hors gamme » (*variable out of range*), sera visible pendant l'exécution de l'instruction. Si cette instruction est utilisée afin d'envoyer des commandes étendues au capteur, le paramètre de Destination stocke le résultat de la commande. 0 indique que la commande a fonctionné ; NaN indique qu'elle a échoué.
OutString	L'adresse du capteur SDI-12 et les chaînes de commande à envoyer aux capteurs. Les valeurs de la table Public peuvent être envoyées au capteur en utilisant la commande suivante : « OXD%variable_name% ! » De multiples valeurs d'un tableau ou d'une ligne peuvent être envoyées en faisant précéder le nom du tableau ou de la ligne par le nombre de valeurs envoyées (exemple : « 0XD%3variable% ! » cette commande enverra la variable(1), la variable (2) et la variable (3).
Mult, Offset *Constante, Variable, Ligne de donnée ou Expression*	Un multiplicateur et un offset qu'on utilise, sous forme de constante, de variable, de ligne de données ou d'expression, afin de mettre à l'échelle les résultats de la mesure brute.

SwBatt

L'instruction SwBatt est utilisée afin de mettre le port 12V commuté à l'état haut ou à l'état bas.

Syntaxe

SwBatt(état)

Remarque

La centrale de mesure a une borne de connexion pour une batterie commutée. Cette batterie commutée peut être contrôlée par l'instruction SwBatt, afin de fournir la tension de la batterie à un périphérique externe. A température ambiante, il y a 900mA de disponible entre la borne SwBatt et la masse. L'indicateur d'état indique si la batterie commutée est active (différent de zéro) ou inactive (0).

NOTE: La borne SW-Batt est non régulée, et peut fournir jusqu'à 900mA à 20°C, et jusqu'à 630mA à 50°C. Un fusible en polymère, réinitialisable, protège la centrale contre les court circuits. La réinitialisation est accomplie en supprimant la charge ou en désactivant le port SW-Batt durant quelques secondes.

Opérateurs

^	Mettre à la puissance
*	Multiplier
/	Diviser
+	Ajouter
-	Soustraire
=	Est égal à
<>	Est différent de
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à

ABS (source)

Cette instruction renvoie la valeur absolue d'un nombre.

Syntaxe

X = ABS (*source*)

Remarques

La source peut être une quelconque expression numérique valide. La valeur absolue du nombre est sa magnitude, sans le signe. Par exemple, **ABS**(-1) et **ABS**(1) ont tout les deux pour résultat « 1 ».

Exemple de fonction ABS

L'exemple suivant trouve la valeur approximative d'une racine carrée. Il utilise la fonction ABS afin de trouver la différence absolue entre deux nombres.

```
Dim Précision, Valeur, X, X1, X2      'Déclaration des variables
Précision = 0,00000000000001
Valeur = Volt(3)                      'Volt(3) sera évalué
X1 = 0 :X2 = Valeur                   'On fait les 2 premières suppositions
'on fait une boucle jusqu'à ce que la différence entre les deux suppositions, soit inférieure à la précision
Do Until Abs(X1 – X2) < Précision
X = X1 + X2) / 2
If X * X * X – Valeur < 0 Then        'On ajuste la supposition
      X1 = X
Else
      X2 = X
End If
Loop
' X devient la racine cubique de Volt(3)
```

ACOS (source)

La fonction ACOS donne comme résultat l'arc cosinus du nombre.

Syntaxe

x = ACOS (Source)

Remarques
La source peur être n'importe quelle expression numérique valide ayant une valeur comprise entre 1 et –1 inclus.

Une fonction ACOS prend le rapport de deux côtés d'un triangle droit, et donne comme résultat l'angle correspondant. Le rapport est la longueur du côté adjacent de l'angle, divisé par la longueur de l'hypoténuse. Le résultat est exprimé en radians, et est compris entre $-\pi/2$ et $\pi/2$ radians.

Pour convertir des degrés en radians, il faut multiplier les degrés par $\pi/180$. Pour convertir des radians en degrés, il faut multiplier les radians par $180/\pi$.

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 $\pi/2$ radians (90 Degrés).

```
Public Pi                 'Déclaration de la variable
Pi = 2 * ACOS (0)         'Calcule la valeur de Pi
```

AND

Cette instruction est utilisée afin d'effectuer une comparaison de bits entre deux nombres.

Syntaxe

Résultat = nombre 1 **And** nombre 2

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 :

Si le bit dans le Nombre 1 est	**Et le bit dans le Nombre 2 est**	**Alors le résultat est**
0	0	0
0	1	0
1	0	0
1	1	1

Bien que l'opérateur **And**, soit un opérateur de comparaison de bits, il est souvent utilisé pour des test Booléens (vrai / faux). La CR200 décide que quelque chose est vrai ou faux avec le critère que 0 est faux, et que tout autre nombre différent de 0 est vrai (voir paragraphe 4.5). Du fait que **And**, soit un opérateur de comparaison de bits, il est possible d' « additionner » des chiffres différents de zéro (2 **AND** 4, par exemple), et d'obtenir comme résultat « 0 » ! La représentation binaire de « -1 » a tous les bits égaux à « 1 ». Ainsi n'importe quelle nombre **And** « -1 », donne comme résultat le nombre initial. C'est pourquoi on a une constante pré-définie, Vrai = -1.

Constante pré-définie vraie = -1
Constante pré-définie fausse = 0

Si le nombre 1 est	**And** le nombre 2 est	Alors le résultat est :
-1	N'importe quel nombre	Nombre 2
-1	Autre chose qu'un nombre	Autre chose qu'un nombre
0	N'importe quel nombre	0
0	Autre chose qu'un nombre	Autre chose qu'un nombre

Les expressions sont évaluées par rapport à un nombre, et peuvent être utilisée à la place d'un ou de plusieurs nombres. Les expressions de comparaison évaluent le résultat en terme de vrai (-1) ou faux (0), par exemple.

```
If Temp(1) > 50 AND Temp(3) < 20 Then
X = True
Else
X = False
Endif
```

Et

```
X = Temp(1) > 50 AND Temp(3) < 20
```

Les deux façons d'écrire l'expression, ont le même effet : X aura la valeur « -1 » si Temp(1) est supérieur à 50, et que Temp(3) est inférieure à 20. X aura la valeur « 0 » si l'une des expressions est fausse.

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 des variables
A = 10: B = 8: C = 6            ‘On donne les valeurs aux variables
If A > B And B > C Then         ‘On évalue l'expression
   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 $\pi/4$ radians (45 Degrés).

```
Dim Pi                  'Déclaration de la variable
Pi = 4 * Atn (1)        'Calcule la valeur de Pi
```

ATN2 ()

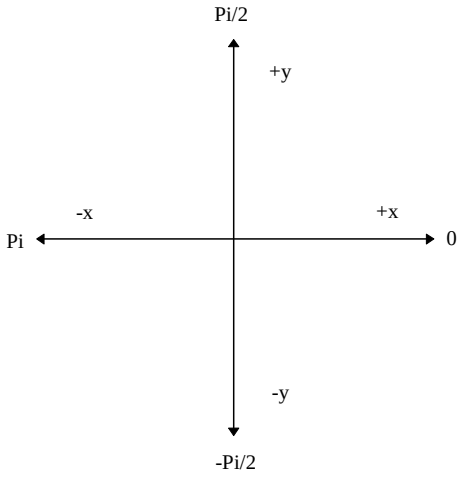
Cette fonction ATN2 donne pour résultat l'arc tangente d'un rapport y/x.

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 dans lequel 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 $\pi/180$. Pour convertir des radians en degrés, il faut multiplier les radians par $180/\pi$.



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 $\pi/4$ 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))
```

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 π/180. Pour convertir des radians en degrés, il faut multiplier les radians par 180/π.

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
```

GetFSValue (Dest, TableName, FieldName, RecsBack)

La fonction GetFSvalue relie un champs à une table de donnée. Son fonctionnement est similaire à celui de la syntaxe Tablenam.fieldname décrite au paragraphe 4.8.

GetFSValue (Dest, AvgDat, Temp_Avg(2), 1)

Charge alors la valeur la plus récente de Temp_avg(2) et présente dans la table de sauvegarde AvgDat, dans la variable Dest. La même opération peut être accomplie avec l'écriture suivante :

Dest = AvgDat.Temp_Avg(2,1)

Paramètres *& type de donnée*	**Entrée**
Dest *Variable*	La variable ou la ligne de données dans laquelle on stocke la valeur prise dans la table de données.
TableName *Nom*	Le nom de la table de données depuis laquelle on récupère les données.
FieldName *Nom*	Le nom du champs depuis lequel on récupère la donnée.
RecsBack *Constante*	Le nombre d'enregistrement avant le plus récent, pour lequel on va récupérer la donnée. « 1 » indique la valeur la plus récente.

CovSpa (Dest, NumOfCov, SizeOfSets, CoreArray, DatArray)

L'instruction CovSpa calcule la covariance de groupes de données qui sont chargées dans des lignes de données.

CovSpa calcule la covariance (les covariances) entre les données dans la ligne de données « CoreArray », et un ou plusieurs jeux de données dans la variable « DatArray ». La covariance des jeux de données X et Y est calculée de la façon suivante :

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

où « n » est le nombre de valeurs dans chaque jeu de données (SizeOfSets). X_i et Y_i sont les valeurs individuelles de X et de Y.

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 l'instruction. Quand plusieurs covariances sont calculées, les résultats sont stockés dans une ligne de données ayant le nom de la variable. La ligne de données doit être dimensionnée de façon à avoir au minimum la taille de « NumOfCov ».
NumOfCov *Constante*	Le nombre de covariances à être calculées. Si 4 jeux de données doivent être comparés à un 5ème jeu, le nombre de covariances devra être fixé à la valeur de 4.
SizeOfSets *Constante*	Le nombre de valeurs des jeux de données, pour le calcul de la covariance.
CorArray *Ligne de données*	La ligne de données qui contient le jeu de données du « cœur ». La covariance du « cœur » du jeu de données est calculée pour tous les autres jeux de données de façon indépendante. Les données ont besoin d'être placées à la suite dans des lignes de données. Si le premier point de donnée n'est pas le premier point de la ligne de données, le premier point du jeu de données doit être mentionné dans ce paramètre.
DatArray *Ligne de données*	La ou les ligne(s) de données qui contient / contiennent le ou les jeu(x) de donnée(s) qui sont nécessaire pour calculer la covariance par rapport au « cœur » du jeu de données. Lorsque plusieurs covariances sont calculées, les jeux de données doivent être chargés de façon consécutive dans une ligne de données. La ligne de données doit avoir la dimension minimale égale à NumOfCov multiplié par SizeOfSets. Par exemple, si chaque jeu de donnée comprend 100 éléments (SizeOfSet) et qu'il y a 4 covariances (NumOfCov) à être calculées, alors la valeur de DataArray doit être égale à 4 x 100 = 400. Si la première valeur du premier jeu de données, n'est pas le premier point du jeu de données, alors le premier point du jeu de données doit être spécifié dans ce paramètre.

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
```

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.

Exemple d'utilisation de la fonction Frac

L'exemple utilise la fonction Frac.

ItTime

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)

L'instruction IfTime donne comme résultat la valeur (-1) quand le résultat du test est vrai, et la valeur (0) quand le résultat du test est faut. Le test est effectué par comparaison avec le temps courant (l'horloge de la centrale de mesure). L'horloge de la centrale est gardée à l'intérieur de la centrale de mesure, et est le temps écoulé depuis le 1er janvier 1990 à 00h00min00sec. L'intervalle est synchronisé avec le temps écoulé (c'est à dire que l'intervalle a la valeur « vraie » lorsque l'intervalle est un multiple de temps écoulé). Le paramètre de « Timeout » permet de définir un offset à l'intérieur de l'intervalle. L'instruction IfTime peut être utilisée afin de changer la valeur d'une variable, ou afin de servir d'expression pour une condition.

L'heure que l'instruction IfTime vérifie, a la résolution de l'intervalle de mesure du programme (c'est à dire qu'il reste fixe durant la durée entière de l'intervalle de scrutation, et s'incrémente lors de la scrutation suivante). L'instruction IfTime doit être insérée à l'intérieure d'une scrutation, afin qu'elle fonctionne.

La fenêtre de temps durant l'instruction IfTime est vraie, est de 1 fois l'unité spécifiée dans le champs d'unité. Par exemple, si l'instruction IfTime spécifie un moment de 0 dans un intervalle de 10 minutes, elle pourrait être vraie pendant la première minute de l'intervalle. Avec un moment de 0 dans un intervalle de 600 secondes, l'intervalle aura toujours une durée de 10 minutes, mais l'instruction IfTime ne sera vraie que durant la première seconde de l'intervalle.

```
IfTime (0,60, Min) then
' instructions à effectuer une fois par heure
EndIf
```

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.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>TintoInt
Constante</td><td>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.</td></tr>
<tr><td>Interval
Constante</td><td>C’est la périodicité à laquelle l’instruction IfTime sera vraie.</td></tr>
<tr><td>Units
Constante</td><td>C’est l’unité qui est commune à TintoInt et Interval
<table>
<tr><th>Code alphabétique</th><th>Code numérique</th><th>Unité</th></tr>
<tr><td>Sec</td><td>2</td><td>Secondes (seconds)</td></tr>
<tr><td>Min</td><td>3</td><td>Minutes (minutes)</td></tr>
<tr><td>Hr</td><td>4</td><td>Heures (hours)</td></tr>
<tr><td>Day</td><td>5</td><td>Jours (days)</td></tr>
</table>
</td></tr>
</table>

IIF (Expression, TrueValue, FalseValue)

Cette instruction est utilisée afin d’évaluer une expression, et de donner comme résultat l’une ou l’autre des valeurs mentionnées, en fonction de l’évaluation.

L’expression qui doit être évaluée, est définie au paramètre 1. Si l’expression est vraie, la valeur vraie (TrueValue, paramètre 2) est donnée comme résultat. Si l’expression est fausse, la valeur fausse (FalseValue, paramètre 3) est donnée comme résultat. La fonction IIF évalue toujours les arguments vrai et faux à la fois, même si elle ne donne que l’un d’eux comme résultat. Cela pourrait mener à des erreurs mathématiques, même lorsque l’expression retournée est valide, alors que le résultat n’aurait pas été valide si on avait utilisé l’argument alternatif.

Int, Fix

Donne comme résultat la partie entière d’un nombre.

Syntaxe

Int (*source*)

Fix (*source*)

Remarques

La source peut être une quelconque valeur numérique ou expression valide. Les fonctions **Int** et **Fix** soustraient la partie fractionnelle de la source, et donnent comme résultat une valeur entière.

Si l’expression numérique n’est pas un nombre, les fonctions **Int** ou **Fix** donnent comme résultat une valeur qui n’est pas un nombre.

La différence entre la valeur **Int** et **Fix** est la suivante : si le *nombre* est négatif, la fonction **Int** donne comme résultat le premier entier négatif inférieur ou égal au *nombre*, alors que la fonction **Fix** donne comme résultat la valeur négative supérieure ou égale au *nombre*. Par exemple, la fonction **Int** convertit -8,4 en –9 et la fonction **Fix** convertit –8,4 en –8.

Exemples d'utilisation des fonctions Int et Fix

Ces exemple illustrent l'utilisation des fonctions **Fix** et **Int.**

```
Dim A, B, C, D                ‘Déclaration des variables
BeginProg
A = Int(-99.8)                ‘Résultat –100
B = Fix(-99.8)                ‘Résultat –99
C = Int(99.8)                 ‘Résultat 99
D = Fix(99.8)                 ‘Résultat 99
EndProg
```

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 (nombre)

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. La fenêtre (*Swath*) est le nombre d'éléments sur lesquels on va chercher le maximum. Deux résultats sont stockés : la valeur maximum, et la place à l'intérieur de la ligne de données, à laquelle se trouve cette valeur (1.swath).

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. Deux résultats sont stockés : la valeur minimum, et la place à l'intérieur de la ligne de données, à laquelle se trouve cette valeur (1.swath).

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
```

NOT

La fonction NOT est utilisée afin d'effectuer une négation de bit sur un nombre.

Syntaxe

Résultat = NOT (nombre)

L'opérateur NOT inverse la valeurs des bits de n'importe quelle variable et donne au bit correspondant le résultat correspondant au tableau ci-dessous :

Si le bit dans l'expression est égal à :	Alors le résultat est égal à :
0	1
1	0

Si l'opérateur NOT est un opérateur de conversion de bits, il est aussi souvent utilisé afin de tester des conditions Booléennes (Vrai / Faux). La CR200 décide que quelque chose est vrai ou faux, avec comme principe que 0 est faux, et que toute autre nombre différent de 0 est vrai (voir chapitre 4.5). Parce que l'opérateur NOT est un opérateur de comparaison de bits, le seul nombre différent de 0 pour lequel l'opérateur NOT peut donner comme résultat la valeur 0, est le nombre « -1 ». Le représentant binaire de « -1 » a tous les bits égaux à 1. C'est pour cela que la constante pré-définie est : Vrai = -1.

Constante pré-définie Vrai = -1
Constante pré-définie Faux = 0

NOT (-1) = 0
NOT (0) = -1
NOT (NAN) = NAN

(*NAN = Not A Number*; valeur différente d'un nombre)

Or

Cet opérateur est utilisé afin d'effectuer une comparaison de bits entre deux nombres.

Syntaxe

Résultat = *nombre1* **Or** *nombre2*

L'opérateur **Or** effectue une comparaison de bit sur des bits positionnés de façon identique à l'intérieur de deux expressions numériques, et donne la correspondance de bits suivante pour résultat :

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	1

Bien que l'opérateur **Or** soit une comparateur de bits, il est souvent utilisé afin de tester des Booléens (vrai / faux). La CR200 décide si quelque chose est vrai ou faux en assumant que 0 est faux et que n'importe quelle valeur numérique différente de 0 est vraie (paragraphe 4.5). Pour la CR200, une constante pré-définie est : Vrai = -1. La représentation binaire de -1 a tous les bits qui sont égaux à 1. Ainsi, n'importe quel nombre **Or** -1, donne comme résultat -1. N'importe quel nombre **And** -1 donne comme résultat le nombre de départ.

Constante pré-définie Vrai = -1
Constante pré-définie Faux = 0

Si l'expression 1 est :	Si l'expression 2 est :	Alors le résultat est :
-1	N'importe quel nombre	-1
-1	Pas un nombre (NAN)	NAN
0	N'importe quel nombre	Nombre 2
0	NAN	NAN

Les expressions sont évaluées comparativement à un nombre (voir paragraphe 4.5), et peuvent être utilisées à la place de l'un ou des deux nombres. La comparaison évaluent les expressions en tant que Vrai (-1) ou Faux (0), par exemple :

```
If Temp(1) > 50 Or Temp(3) < 20 Then
    X = True
Else
    X = False
EndIf
```

Et

```
X = Temp(1) > 50 OR Temp(3) < 20
```

Sont des expressions qui ont le même effet, à savoir donner la valeur –1 à la variable X si Temp(1) est supérieur à 50 ou si Temp(3) est inférieur à 20. X aura la valeur 0 si les deux expressions sont fausses.

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 CR200.

Syntaxe

RealTime (Dest, Value)

Exemple d'instruction RealTime

Le paramètre de destination (*Dest*) est la variable, ou la ligne de données, dans laquelle le résultat de l'instruction RealTime, est stockée. Le paramètre de valeur (*Value*) est utilisé afin d'indiquer si tous les paramètres ou seulement un, doivent être donnés comme résultat. Si la valeur est mise à 0, toutes les options de temps seront renvoyées, et dans l'ordre suivant : (1) année, (2) mois, (3) jour du mois, (4) heure du jour, (5) minutes de l'heure, (6) secondes de la minute, (7) jour de la semaine (allant de 1 à 7, avec 1 = Dimanche), et (8) jour à l'intérieur de l'année. La variable de destination pour la ligne de données doit avoir une dimension de 8. Si vous donnez une valeur (entre 1 et 8) au paramètre *Value*, votre instruction RealTime ne donnera comme résultat, que la valeur de temps demandée par le paramètre.

```
Public rTime(8)                          'Déclaration des variables
Alias rTime(1) = Annee                   'Assigne l'alias « Annee » à rTime(1)
Alias rTime(2) = Mois                    'Assigne l'alias « Mois » à rTime(2)
Alias rTime(3) = Jour                    'Assigne l'alias « Jour » à rTime(3)
Alias rTime(4) = Heure                   'Assigne l'alias « Heure » à rTime(4)
Alias rTime(5) = Minute                  'Assigne l'alias « Minute » à rTime(5)
Alias rTime(6) = Seconde                 'Assigne l'alias « Seconde » à rTime(6)
Alias rTime(7) = Jour_de_la_semaine      'Assigne l'alias « Jour_de_la_semaine» à rTime(7)
Alias rTime(8) = Jour_dans_l_année       'Assigne l'alias « Jour_dans_l_année» à rTime(8)
DataTable (VALUES, 1, 100)               'Définit le tableau de données
'Sample (1, Année, IEEE4)                'met l'année dans le tableau VALUES
'Sample (1, Mois, IEEE4)                 'met le mois dans le tableau VALUES
'Sample (1, Jour, IEEE4)                 'met le jour dans le tableau VALUES
'Sample (1, Heure, IEEE4)                'met l'heure dans le tableau VALUES
'Sample (1, Minute, IEEE4)               'met la minute dans le tableau VALUES
'Sample (1, Seconde, IEEE4)              'met la seconde dans le tableau VALUES
'Sample (1, Jour_de_la_semaine, IEEE4)   'met le jour de la semaine dans le tableau VALUES
'Sample (1, Jour_dans_l_année, IEEE4)    'met le jour à l'intérieur de l'année dans le 'tableau VALUES
 Sample (8, rTime(), IEEE4)              'met les 8 données dans le tableau VALUES
EndTable
```

```
BeginProg
  Scan (1, Sec)
    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 la 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 carré des valeurs contenues dans une ligne de données.

$$Dest = \sqrt{\frac{\sum_{i=j}^{i=j+swath} \left(X(i)\right)^2}{swath}}$$

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

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. 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.

Sgn (Source)

Cette instruction est utilisée afin de connaître la valeur du signe d'un nombre.

Syntaxe

x = Sng (source**)**

Remarques
Cette instruction donne comme résultat un entier, qui représente le signe du nombre.

L'argument du nombre peut être n'importe quelle expression numérique valide. Son signe détermine 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 Sgn afin de déterminer le signe d'un nombre.

```
Dim Msg, Nombre                 'Déclaration des variables
Nombre = Volt(1)                'On prend une valeur en Entrée
Select Case Sgn(Nombre)         'On évalue le nombre
  Case 0                        'La valeur est 0
    Msg = 0
  Case 1                        'La valeur est positive
    Msg = 1
  Case -1                       'La valeur est négative
    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 $\pi/180$. Pour convertir des radians en degrés, on multiplie par $180/\pi$.

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
```

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
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
```

XOR

La fonction XOR est utilisée afin d'effectuer une exclusion logique sur deux nombres.

Syntaxe

Résultat = nombre1 XOR nombre2

L'opérateur XOR effectue une comparaison de bits sur des bits positionnés à l'identique dans 2 nombres, et met la valeur fixe 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

Bien que l'opérateur XOR soit un opérateur de comparaison de bits, il peut être utilisé afin d'effectuer des tests sur ces conditions Booléennes (Vrai / Faux). La CR200 décide si quelque chose est vrai ou faux en assumant que 0 est faux et que n'importe quelle valeur numérique différente de 0 est vraie (paragraphe 4.5). Puisque XOR est une opération de comparaison de bits, il est possible d'effectuer une opération XOR sur deux nombres différents de zéro (par exemple 2 et 4), et obtenir un nombre différent de zéro. (XOR ne fonctionnera qu'avec deux nombres différents de zéro, et donner zéro comme résultat si les nombres de départ sont égaux.)

Constante pré-définie Vrai = -1

Constante pré-définie Faux = 0

Si l'expression 1 est :	Si l'expression 2 est :	Alors le résultat est :
-1	N'importe quel nombre	-1
-1	Pas un nombre (NAN)	NAN
0	N'importe quel nombre	Nombre 2
0	NAN	NAN

Les expressions sont évaluées par rapport à un nombre (paragraphe 4.5), et peuvent être utilisées à la place de chacun des deux nombres. Les expressions de comparaison sont évaluées en tant que Vrai (61) ou Faux (0). Par exemple :

```
If Temp(1) > 50 XOR Temp(3) < 20 Then
    X = True
Else
    X = False
EndIf
```

Et

```
X = Temp(1) > 50 XOR Temp(3) < 20
```

Ont le même effet : X aura la valeur -1 si seule une des expressions « Temp(1) > 50 » ou « Temp(3) < 20 » est vraie. X aura la valeur 0 si les deux expressions sont vraies ou si les deux expressions sont fausses.

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 CRBasic :**
Secant	Sec = 1 / Cos(X)
Cosecant	Cosec = 1 / Sin(X)
Cotangente	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

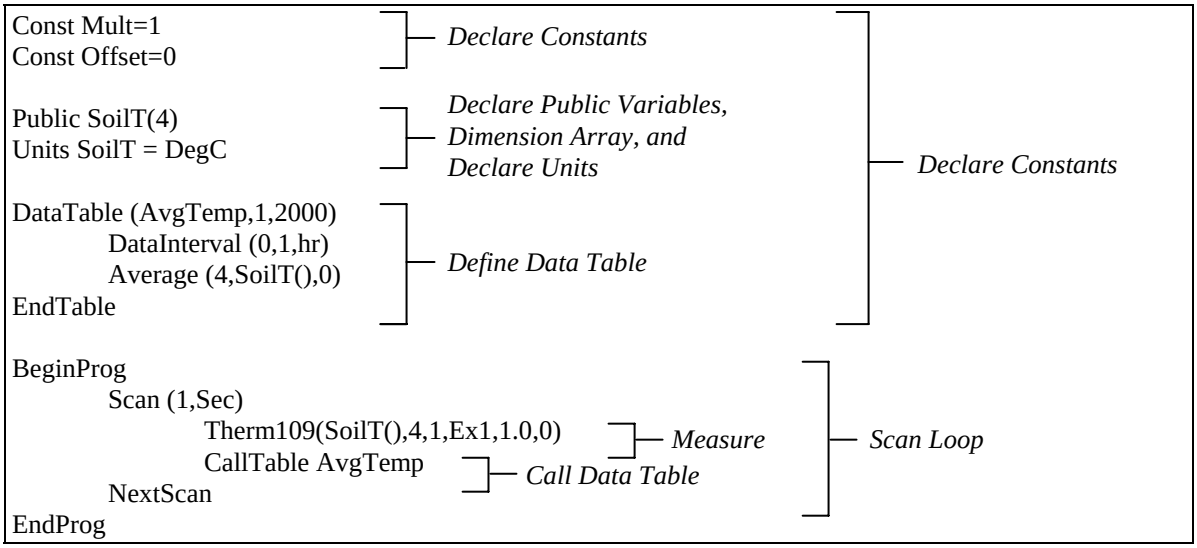
…

…

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. Les balises BeginProg / EndProg ne sont pas nécessaires s'il n'y a ni sous-programmes ni tables de données (DataTables).



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 une mot optionnel, afin de transférer le 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 « 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 une table de données (DataTable). Les tables de données sont listées dans la partie « Déclaration » du programme, avant la balise « BeginProg ». Lorsque la table de données est appelée, elle effectuera un traitement des données de la façon indiquée dans le programme, et elle 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

ClockSet (Source)

Cette instruction permet de fixer l'heure de la CR200 à partir des valeurs contenues dans une ligne de données. L'utilisation la plus probable de cette instruction sera lorsque la CR200 utilise une source de synchronisation temporelle qui est plus précise que celle de la CR200, 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 sera utilisée afin de fixer l'heure de la CR200.

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 (Delay, Units)

Cette instruction est utilisée afin de donner un délai au programme.

Syntaxe

Delay(Delay, Units)

Remarques

L'instruction de délai est utilisée afin d'effectuer une pose à l'intérieur du programme, pendant une durée spécifiée par les valeurs « Delay » et « Units ».

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.

Paramètres *& type de donnée*	**Entrée**
Delay *Constante*	La valeur numérique du délai.
Units *Constante*	L'unité de mesure du délai

Code alphabétique	**Code numérique**	**Unité**
USEC	0	Microsecondes
MSEC	1	Millisecondes
SEC	2	Secondes
MIN	3	Minutes

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 répété 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
```

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*

Partie	Description
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 écritjuste 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
```

LoggerIdentify

L'instruction « LoggerIdentify » est utilisée afin de donner une valeur à une chaîne d'identification présente dans la centrale de mesure, et qui sera envoyée en retour lorsqu'une autre chaîne de caractère sera envoyée à la centrale de mesure.

Syntaxe

LoggerIdentify ("RequestString", "ReturnString")

Remarques

Cette instruction n'est disponible que dans certains systèmes d'exploitation.

L'instruction "LoggerIdentify" est écrite à n'importe quell emplacement du programme. Lorsque la chaîne RequestString est reçue par la centrale de mesure, elle répondra par la chaîne ReturnString. La chaîne RequestString ne peut avoir plus de 4 caractères. La chaîne ReturnString est limitée à 29 caractères.

Si la centrale de mesure est en émulation de terminal et que la chaîne RequestString est reçue, la centrale de mesure répondra avec la valeur « Invalid ». La centrale devra sortir du mode d'émulateur de terminal (après son délai de temps approprié), ou bien vous devrez entrer la chaîne RequestString avant que la centrale de mesure ne soit en mode d'émulation de terminal (la CR200 est mise en mode d'émulation de terminal si on lui envoie 4 retours chariot (*carriage return*).

Exemple

Dans l'exemple suivant, lorsque la CR200 reçoit la commande « AMXB », la centrale de mesure répondra par « MyCR200Logger ». A noter que l'instruction *LoggerIdentify* n'est disponible que dans des systèmes d'exploitation (*Operation Systems*) spéciaux.

```
Public Battery

DataTable (Table,True,1000)
    DataInterval (0,1,min)
    Sample (1,Battery)
EndTable

BeginProg
LoggerIdentify ("AMXB","MyCR200Logger")
    Scan (1,sec)
      Battery (Battery)
      CallTable (Table)
    NextScan
EndProg
```

Print (Port, BaudRate, PrintParams)

L'instruction « Print » est utilisée pour envoyer les valeurs du programme ou d'autres caractères vers le port de communication.

Cette instruction est utilisée souvent comme outil afin de mettre au point « déboguer » un programme. Les valeurs spécifiées pour la variable, sont envoyées en tant que caractère ASCII, vers le port RS-232 ; la liste des paramètres à envoyer ne peut excéder 5 éléments. (chaque élément est séparé par une virgule)

Paramètres *& type de donnée*	**Entrée**
Port *Constante*	Le paramétrage du port de communication est utilisé pour spécifié le port COM qui devra être utilisé par l'instruction. Le bouton droit de la sourie fera apparaître liste. **Code** : **Port de communication** 0 : RF 1 : RS-232 3 : C1
BaudRate *Vitesse des bauds*	Le paramètre BaudRate est la vitesse de communication à utiliser pour le port séléctionné dans le premier paramètre ; les options sont 1200 et 9600 bps. Si vous utilisez la radio, le vitesse de transmission doit être 9600.
PrintList *Variables, caractères ou chaînes de caractères*	PrintList est une liste d'éléments à imprimer, de longueur variable. Elle peut contenir des chaînes de caractères ou de variables (avec une syntaxe de type CHR$(code ascii)). Les éléments sont séparés par des virgules.

Scan

Cette instruction est utilisée afin de donner le temps de scrutation du programme.

Syntaxe

Scan (Interval, Units)

…

… [**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.

Paramètres *& type de donnée*	**Entrée**
Interval *Constante*	On entre l'intervalle de temps auquel la scrutation sera effectuée. L'intervalle peut être en secondes ou en minutes, comme cela est sélectionné dans le paramètre des unités.
Units *Constante*	L'unité pour le paramètre de temps **Code alphabétique** : **Code numérique** : **Unité** SEC : 2 : Secondes MIN : 3 : Minutes

ScanLEDOff

Cette instruction désactive les LEDs du bornier de la centrale de mesure, LEDs qui indiquent que le programme est en cours d'exécution.

Syntaxe

ScanLEDOff

Remarques

Cette instruction n'a pas de paramètres. Quand cette instruction est présente dans un programme, la LED qui indique si le programme est exécuté ou non, est alors éteinte jusqu'à la fin de l'exécution présente du programme. Lors des scrutations suivantes, la LED sera de nouveau active jusqu'à ce que le programme lise l'instruction ScanLEDOff. Cette instruction est donc typiquement utilisée afin d'économiser de l'énergie lorsque les programmes sont longs et que la LED reste allumée pendant longtemps si on n'utilisait pas l'instruction ScanLEDOff.

Select Case

Cette instruction exécute les commandes présentes à l'intérieur du bloc de commandes, en fonction de la valeur d'une expression.

Syntaxe

Select Case expression_de_test
[**Case** liste_d_expression_1
    [bloc_d_instruction_1]]
[**Case** liste_d_expression_2
    [bloc_d_instruction_2]]
[**Case Else** liste_d_expression_n
    [bloc_d_instruction_n]]
End Select

La syntaxe de l'expression Select Case contient ces parties :

Partie	Description
Select Case	Ceci débute la structure de contrôle **Select Case**. 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_dinstruction_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…
```

SetStatus()

L'instruction SetStatus est utilisée afin de mettre une valeur à l'intérieur du tableau d'état (*Status table*) de la centrale de mesure.

Syntaxe

SetStatus (FieldName, Value)

Remarques

L'instruction SetStatus permet à l'utilisateur de changer une valeur à l'intérieur du tableau d'état de la centrale de mesure. Tous les champs de ce tableau d'état peuvent être modifiés, excepté le champs OSVersion (version du système d'exploitation), OSDate (date du système d'exploitation), ProgName (nom du programme), ProgSig (signature du programme), et CalOffset (offset de calibration). L'instruction peut aussi être utilisée afin de remettre à zéro (ré-initialiser) tous les tableaux de données de la centrale de mesure.

FieldName	L'argument de FieldName est utilisé afin de spécifier le paramètre du tableau d'état, qui doit être modifié. Cliquer sur le bouton droit de la souris afin de visualiser dans des fenêtres, les noms de champs valides que l'on peut afficher. Les champs valides sont :

<table>
<tr><th>Nom du champs (FieldName)</th><th>Description</th></tr>
<tr><td>PakBusAddress</td><td>Adresse PakBus de la centrale de mesure</td></tr>
<tr><td>RfNetAddr</td><td>L’adresse du réseau RF. Les adresses valides sont entre 0 et 63.</td></tr>
<tr><td>RfAddress</td><td>L’adresse de la radio RF. Les adresses valides sont entre 0 et 1023.</td></tr>
<tr><td>RfHopSeq</td><td>La séquence de saut de fréquence RF sur le réseau. Les modes valides sont entre 0 et 6.</td></tr>
<tr><td>RfPowrMode</td><td>Le mode de configuration de l’alimentation des radio. Une constante prédéfinie est utilisée. Un click sur le bouton droit de la souris permet de choisir entre :
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>RF_ON</td><td>La radio est toujours active.</td></tr>
<tr><td>RfpinEn</td><td>La radio est activée par un appareil via une pin.</td></tr>
<tr><td>FR1_Sec</td><td>La radio fonctionne selon un cycle de 1 seconde.</td></tr>
<tr><td>RF8_Sec</td><td>La radio fonctionne selon un cycle de 8 secondes.</td></tr>
<tr><td>RF1S_LH</td><td>La radio fonctionne selon un cycle de 1 seconde et transmet de longues en-têtes.</td></tr>
<tr><td>RF8S_LH</td><td>La radio fonctionne selon un cycle de 8 secondes et transmet de longues en-têtes.</td></tr>
</table></td></tr>
<tr><td>Rf_ForceOn</td><td>N’importe quelle valeur différente de zéro modifiera le mode d’alimentation actuel, et sera pris en compte.</td></tr>
<tr><td>Rf_Protocol</td><td>1 = Mode Transparent (Transparent Mode). Ce mode est compatible avec les anciens systèmes d’exploitation des CR205, CR210, CR215, RF400, RF410 et RF415.
2 = PakBus (PakBus Aware Mode). Ce mode peut être utilisé avec des réseaux utilisant des RF401/RF411/RF416 ou des CR206/CR211/CR216, en utilisant les capacités de rappel inhérentes aux radios Maxtream. Ce mode n’est pas compatible avec les anciennes radios.</td></tr>
<tr><td>VarOutOfBounds</td><td>Le nombre de variables qui sont sous-dimensionnées ; c’est à dire que la taille de la ligne de données est insuffisante pour contenir la variable.</td></tr>
<tr><td>SkipScan</td><td>Le nombre de scrutations qui ont été manquées du fait que la scrutation précédente n’était pas finie.</td></tr>
<tr><td>TrapCode</td><td></td></tr>
<tr><td>WatchDogCnt</td><td>Le nombre de fois où le processeur a été ré-initialisé par le compteur de chien de garde.</td></tr>
<tr><td>ResetTables</td><td>Cette commande est utilisée afin de ré-initialiser toutes les données de la centrale de mesure. Pour effectuer l’initialisation, donner la valeur 8888 à la variable ResetTables, soit écrire : SetStatus(ResetTables, 8888)</td></tr>
<tr><td>Value</td><td>La valeur qui sera envoyée dans le champ spécifié par l’instruction, dans la tableau d’état.</td></tr>
</table>

Ticker250ms

L’instruction Ticker250ms stocke la valeur totale de 250ms dans une variable.

Le système d’horloge de 250ms est indépendant de l’horloge de la centrale d’acquisition et n’est pas affecté par un changement de l’horloge. Cependant, cette instruction peut être utilisée pour exécuter une ou plusieurs instruction basée sur le temps, ou basé sur le temps de l’horloge de la centrale d’acquisition (qui peut être changé par l’utilisateur) pour affecter les sorties. C’est une horloge de 24 bits, qui une fois démarrée effectuera un cycle tout les 48,5 jours. Cette instruction commence lorsque le programme démarre. Elle ne peut être remise à zéro.

L’instruction Ticker250ms a un paramètre – la destination dans laquelle la valeur de l’houloge sera stockée.

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, de 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. Instructions pour communiquer d'une centrale de mesure à : une autre centrale / une radio / un capteur déporté

La CR200 est souvent utilisée en tant qu'interface pour capteur déporté, pour une centrale d'acquisition « Hôte ». Généralement, la centrale de mesure Hôte et les capteurs (CR200) auront des programmes qui permettront aux capteurs de fonctionner avec un minimum de consommation électrique (110-µA). Ces programmes synchronisent les capteurs CR200 afin qu'ils reportent les données à des intervalles de temps spécifiés.

Pour des applications qui demandent des communications plus fréquentes, l'instruction de récupération de la variable (get/set variable) dans la centrale d'acquisition hôte, peut être utilisée avec un capteur CR200 qui est configuré pour une consommation en courant plus importante (250-µA pour une réponse toutes les 8 Secondes ; 20-mA pour une réponse toutes les secondes).

Dans le mode synchronisé demandant le moins d'alimentation, la CR200 initie toujours la communication. A l'exception du moment où la centrale sort de l'état de veille pour émettre, sa radio est éteinte et ne consomme pas de courant. La radio de la centrale de mesure hôte sera en mode entièrement actif durant la période où elle attend une communication en provenance du capteur, prête à recevoir et à répondre instantanément.

Exemple de programme de faible consommation pour capteur / contrôleur :

```
'Exemple de programme pour capteur sans fil sur une CR200
Const MT = 20                    'Temps de mesure (secs) du SDI-12; ici cela prend 20 'secondes
Const Port = 0                   'Envoi des données vers : 0 = Radio, 1 = RS-232
Const RouterAddr = 1             'Adresse PakBus d'un router, ici c'est la même adresse que 'l'hôte
Const HostAddr = 1               'Adresse PakBus de la centrale de mesure hôte
Const NumVals = 8                '8 mesures
Const Security = 0               'utiliser une valeur différente de zéro si l'hôte a un code de 'sécurité qui est
                                 activé
Const NumControl = 4
Public Measurements(NumVals)     'Valeurs du capteur, à envoyer
Public Control(NumControl)       'Valeurs de contrôle renvoyées par l'hôte
Public Response

BeginProg
  Scan(1, sec)
    If TimeUntilTransmit(Port) = MT then   'Si c'est le moment pour effectuer les 'mesures de début, les
                                           mesures 'applicables qui prennent MT secondes, sont 'insérées ici
    Endif
    If TimeUntilTransmit(Port) = 0 then    'c'est le moment pour communiquer
      SendGetData(Response,Control(),Measurements(),Port,RouterAddr,HostAddr,Security)
    Endif
  NextScan
EndProg
```

TimeUntilTransmit(port) est une fonction qui donne comme résultat le nombre de secondes avant que ce soit le moment pour communiquer. Elle utilisera les informations reçues par la centrale de mesure afin de déterminer le temps qu'il reste avant la prochaine tentative de communication (via le marqueur de temps, *time slot*).

SendGetData(..) permettra d'envoyer une ligne de données vers la centrale de mesure hôte et de recevoir de cette centrale un marqueur de temps, une configuration de l'heure, et en option une ligne de données provenant de la centrale hôte. La ligne de donnée en entrée, en sortie ou bien les deux, peuvent être mises à la valeur « 0 », ce qui implique que le flux de données dans la direction associée, soit annulé.

Le paramètre HostAddr est l'adresse PakBus du « maître » (*master*) du réseau, l'endroit où les données du capteur sont envoyées. L'adresse du Router (RouterAddr) est l'adresse PakBus du router, si la distance est trop importante pour rejoindre l'hôte directement. Généralement il n'y a pas de router, donc cette adresse est identique à celle de l'hôte. Si cette adresse est fixée à 0, alors le système cherchera automatiquement un voisin par lequel il puisse rejoindre l'hôte. (cela n'était pas programmé lors de la première version de PakBus en Août 2002, donc pour l'instant cela doit être vérifié avant utilisation).

A noter que toutes les communications avec la centrale de mesure hôte sont décrites dans l'instruction **SendGetData**. La fonction **TimeUntilWireless**() aura déterminé un marqueur de temps pour son exécution. Cette communication fixera les valeurs des variables dans l'hôte, et récupèrera une ligne de données provenant de l'hôte, ainsi qu'un marqueur de temps et une vérification de l'heure.

GetValue (ResponseDest, Dest, Swath, RemoteVar, Port, RemoteAddr, RepeatAddr, Security)

L'instruction GetValue est utilisée afin de recevoir une ou plusieurs valeurs en provenance de variables faisant partie du tableau de valeurs publiques (*public table*) d'une centrale de mesure distante, et de les placer dans des variables de la CR200.

Les paramètres RemoteVar et Swath sont utilisés afin de déterminer quelles valeurs seront rapatriées depuis la centrale de mesure hôte. RemoteVar est le nom de la variable sur la centrale de mesure distante, et le nombre de valeurs est spécifié dans le paramètre Swath. Si le paramètre Swath est supérieur à la valeur « 1 », RemoteVar doit être une ligne de données.

<table>
<tr><th>Paramètres
& type de donnée</th><th>Entrée</th></tr>
<tr><td>ResponseDest
Variable ou ligne de données</td><td>C'est la variable dans laquelle sera stocké le code de réponse décrivant la transmission. Ce code indique si la transmission a eu lieu avec succès ou non.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Transmission réussie</td></tr>
<tr><td>-1</td><td>Réponse reçue, mais permission refusée</td></tr>
<tr><td>-16</td><td>Réponse reçue, mais RemoteVar n'est pas « public »</td></tr>
<tr><td>-17</td><td>Réponse reçue, mais conversion de données impossible</td></tr>
<tr><td>1, 2.. n</td><td>Nombre de « timeouts » avant d'avoir la réponse</td></tr>
</table></td></tr>
<tr><td>Dest
Variable ou ligne de données</td><td>La ligne de données dans laquelle stocker la variable reçue depuis la centrale de mesure distante. Le paramètre Dest doit avoir une taille supérieure ou égale au paramètre Swath.</td></tr>
<tr><td>Swath
Constante</td><td>Le nombre de valeur à récupérer à partir de la centrale distante.</td></tr>
<tr><td>RemoteVar
Variable ou ligne de données</td><td>La variable ou la ligne de données de la table « public » de la centrale de mesure distante, à partir desquelles on va prendre les valeurs. Si Swath est supérieur à 1, la variable doit être une ligne de données de dimension supérieure ou égale à celle de Swath.</td></tr>
<tr><td>Port
Constante</td><td>Le port de communication utilisé pour communiquer avec l'appareil distant
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>1</td><td>RF</td></tr>
<tr><td>2</td><td>RS-232</td></tr>
</table></td></tr>
<tr><td>RemoteAddr
Constante</td><td>L'adresse Pakbus de la centrale distante.</td></tr>
<tr><td>RepeatAddr
Constante</td><td>Adresse de la première centrale de mesure via laquelle la centrale de mesure doit passer, afin de communiquer avec le reste du réseau. S'il n'y a pas de « Repeater », on entre l'adresse de RemoteAddr.</td></tr>
<tr><td>Security
Constante</td><td>Code de sécurité pour le réseau Pakbus.</td></tr>
</table>

ReadSendGetInfo (Dest, Port)

L'instruction ReadSendGetInfo retourne l'intervalle et l'Offset de l'instruction SendGetData .

La centrale d'acquisition hôte (typiquement une CR10X-PB, CR510-PB ou une CR23X-PB) procure le temps de transmission pendant lesquels la CR200 transmet les données.

Vous devez vous référer à l'instruction 193 dans le menu d'aide de Edlog pour de plus amples détails.

L'instruction ReadSendGetInfo a les paramètres suivants :

Paramètres *& type de donnée*	**Entrée**
Dest *Variable ou*	Le paramètre de Dest est un champ qui contient les résultats de l'instruction. Il doit être dimensionner en 2 pour contenir l'intervalle et l'offset de l'instruction SendGetData. Le résultat de l'intervalle est contenu dans Dest(1) et l'Offset dans Dest(2).
Port *Constante*	Les ports de communication qui doivent être utilisés avec le dispositif à distance **Code** / **Description** 1 – RF 2 – RS-232

SendGetData (ResponseDest, Control, Measurement, Port, HostAddr, RepeatAddr, Security)

L'instruction SendGetData est utilisée sur une centrale d'acquisition distante, afin de d'envoyer une ligne de donnée à une centrale de mesure hôte, et/ou afin de recevoir une ligne de données depuis cette centrale.

L'instruction SendGetData enverra une ligne de données de mesure vers la centrale de mesure hôte et recevra un marqueur de temps, une configuration de l'horloge et si cela est demandé (en option) une ligne de données de valeurs provenant de l'hôte. La ligne de donnée **Control** ou **Measurement** (ou les deux) peut être mise à « 0 », indiquant qu'il n'y aura pas de flux de données dans le sens indiqué.

Le paramètre de HostAddr, est l'adresse Pakbus du « maître » (*master*) du réseau, là où les données du capteur sont envoyées. L'adresse RouterAddr, est l'adresse Pakbus du router, dans le cas où la distance est trop importante pour relier la centrale de mesure hôte directement. Le plus souvent il n'y a pas de router, et l'adresse sera identique à HostAddr. Si la valeur de l'adresse est 0, alors le système trouvera automatiquement un voisin par lequel il atteindra la centrale hôte. (cela n'était pas programmé lors de la version PakBus en juillet 2007, donc pour l'instant cela doit être vérifié avant utilisation).

On note que toutes les communications avec la centrale de mesure hôte, sont décrites dans cette instruction. L'instruction TimeUntilTransmit a déterminé un marqueur de temps (*time slot*) pour elle. Cette communication fixera une variable à l'intérieure de l'hôte et récupèrera une ligne de données de variables depuis l'hôte, ainsi qu'un marqueur de temps et une configuration de l'horloge.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>ResponseDest
Variable ou ligne de données</td><td>C’est la variable dans laquelle sera stocké le code de réponse décrivant la transmission. Ce code indique si la transmission a eu lieu avec succès ou non.
<table><tr><th>Code</th><th>Description</th></tr><tr><td>0</td><td>Transmission réussie</td></tr><tr><td>-1</td><td>Réponse reçue, mais permission refusée</td></tr><tr><td>-2</td><td>Réponse reçue, mais pas assez de ressources disponibles</td></tr><tr><td>1, 2.. n</td><td>Nombre de « timeouts » avant d’avoir la réponse</td></tr></table></td></tr>
<tr><td>Control
Variable ou ligne de données</td><td>Le paramètre de Control, est une variable ou la ligne de données variable dans lequel la ligne de données sera stocké dans la centrale de mesure hôte,. Si on le met à 0, aucune valeur ne sera stockée.</td></tr>
<tr><td>Measurement
Variable ou ligne de données</td><td>Le paramètre Measurement est la variable ou la ligne de données de variables qui sera transmise à la centrale de mesure hôte. Si on lui donne la valeur 0, aucune valeur n’est transmise.</td></tr>
<tr><td>Port
Constante</td><td>Le port de communication utilisé pour communiquer avec l’appareil distant
<table><tr><th>Code</th><th>Description</th></tr><tr><td>1</td><td>RF</td></tr><tr><td>2</td><td>RS-232</td></tr></table></td></tr>
<tr><td>HostAddr</td><td>L’adresse Pakbus de la centrale de mesure hôte.</td></tr>
<tr><td>RouterAddr</td><td>Cette adresse est utilisée afin de spécifier l’adresse d’un router sur le réseau Pakbus, par laquelle la centrale de mesure doit passer, afin de communiquer avec le reste du réseau. S’il n’y a pas de router, on entre alors l’adresse de HostAddr.</td></tr>
<tr><td>Security
Constante</td><td>Code de sécurité pour le réseau Pakbus.</td></tr>
</table>

SetValue (ResponseDest, Source, Swath, RemoteVar, Port, RemoteAddr, RepeatAddr, Security)

L’instruction SetValue est utilisée afin de fixer les valeurs de une ou plusieurs variables, à l’intérieur du tableau des valeurs publiques d’une centrale de mesure distante.

Les paramètres source et Swath, sont utilisés afin de déterminer quelles valeurs seront envoyées à la centrale de mesure distante. La source est la ligne de données variable de la CR200, qui contient les variables, et le nombre de valeurs est spécifié dans le paramètre Swath. Si le paramètre Swath est supérieur à 1, RemoteVar doit être une ligne de données.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>ResponseDest
Variable ou ligne de données</td><td>C'est la variable dans laquelle sera stocké le code de réponse décrivant la transmission. Ce code indique si la transmission a eu lieu avec succès ou non.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Transmission réussie</td></tr>
<tr><td>- 1</td><td>Réponse reçue, mais permission refusée</td></tr>
<tr><td>-16</td><td>Réponse reçue, mais RemoteVar n'existe pas</td></tr>
<tr><td>-17</td><td>Réponse reçue, mais conversion de données impossible</td></tr>
<tr><td>1, 2.. n</td><td>Nombre de « timeouts » avant d'avoir la réponse</td></tr>
</table></td></tr>
<tr><td>Source
Variable ou ligne de données</td><td>Le paramètre de Source, est la ligne de données variable dans laquelle seront stockées les valeurs envoyées à la centrale de mesure distante.</td></tr>
<tr><td>Swath
Constante</td><td>Le paramètre Swath indique le nombre de valeurs qui seront transmises à la centrale de mesure distante.</td></tr>
<tr><td>RemoteVar
Variable ou ligne de données</td><td>RemoteVar est utilisée afin de spécifier la variable (ligne de données) de la centrale de mesure distante, dans laquelle on enverra les valeurs.</td></tr>
<tr><td>Port
Constante</td><td>Le port de communication utilisé pour communiquer avec l'appareil distant ; on entre un code numérique :
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>1</td><td>RF</td></tr>
<tr><td>2</td><td>RS-232</td></tr>
</table></td></tr>
<tr><td>RemoteAddr
Constante</td><td>L'adresse Pakbus de la centrale de mesure distante.</td></tr>
<tr><td>RepeatAddr
Constante</td><td>Cette adresse est utilisée afin de spécifier l'adresse d'un «relais » (repeater) sur le réseau Pakbus, par lequel la centrale de mesure doit passer, afin de communiquer avec la centrale de mesure distante. S'il n'y a pas de relais, on entre alors l'adresse de RemoteAddr.</td></tr>
<tr><td>Security
Constante</td><td>Code de sécurité pour le réseau Pakbus.</td></tr>
</table>

TimeUntilTransmit(port)

La fonction TimeUntilTransmit donne comme résultat le nombre de secondes qu'il reste avant la prochaine communication. Elle utilise des informations reçues par la centrale de mesure hôte, afin de déterminer le temps qu'il reste avant le marqueur de temps qui est défini pour la communication.

<table>
<tr><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Port
Constante</td><td>C'est le port de communication sur lequel la communication est programmée
<table>
<tr><th>Code</th><th>Port de communication</th></tr>
<tr><td>1</td><td>RF</td></tr>
<tr><td>2</td><td>RS-232</td></tr>
</table></td></tr>
</table>

Annexe A. Mode terminal pour CR200/SDI12

La CR200 dispose d'une interface de terminale simple, à laquelle on peut accéder à partir d'un PC et d'un émulateur de terminal tel que ; ceux que l'on trouve avec PC200W, PC400 ou LoggerNet. Une des options du mode terminal de la CR200 est celle qui permet d'entrer en mode SDI12 transparent, ce qui permet de passer des commandes entrées au clavier de l'ordinateur, directement vers l'appareil SDI12 connecté à la CR200.

Pour effectuer cela on connecte un câble série entre l'ordinateur/terminal et la CR200. On configure le terminal pour qu'il communique sur le port COM à 9600 baud, puis on appuie 4 fois sur la touche entrée (c*arriage return*) afin d'entrer en mode terminal. La CR200 répondra par :

^ ^ ^ ^

CR2xx>

La CR200 sortira du mode terminal si elle ne reçoit pas de commandes dans les 12 secondes qui suivent l'entrée en mode terminal.

Pour entrer une commande il faut entrer la commande puis taper sur entrée (*carriage return*). Les commandes sont les suivantes :

Commandes	**Action (ce qui se produit)**
A	Informations sur le paramètre *Trap Code 16* « Done » = mémoire OK « 2005-7-14 13 :56 :6=13 » veut dire qu'il y a eu un problème détecté sur la mémoire (code 16), à la date mentionnée. *NOTE* : Lorsqu'un code 16 est obtenu pour *Trap Code*, la centrale de mesure arrête d'exécuter le programme et la LED rouge clignote deux fois à chaque intervalle de scrutation supposé. Lorsqu'il y a un code 16 en *Trap Code*, il faut remplacer la mémoire de la CR200 (en usine).
1	Affiche l'horloge (l'heure actuelle)
2	Met à jour l'heure (après la mise à jour on sort du mode terminal)
3	Codes d'état (*Status Codes*)
4	Tableau d'état (*Status Table*)
5	Tableau des valeurs *Public* (valeurs en temps réel)
6	Données les plus récentes du tableau 1
7	Données les plus récentes du tableau 2
8	Données les plus récentes du tableau 3
9	Données les plus récentes du tableau 4
SDI12	Entre en mode terminal SDI12 et garde ce mode actif

Annexe B. Centrale de mesure CR295

Le jeu d'instructions pour le CR295 ajoute les instructions CRBasic suivantes, afin de communiquer avec le transmetteur HDR GOES de Campbell Scientific. La CR295 ne permet pas de faire de la communication radio, ni le calcul de l'évapotranspiration. Elle n'est pas conforme aux normes CE (et GOES n'est accessible que sur la zone Américaine).

Figure B-1 Centrale CR295.

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 ou le transmetteur satellite TX312. 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
Constante</td><td>Le paramètre DataFormat spécifie le format de la donnée envoyée au transmetteur
<table>
<tr><th>Code</th><th colspan="2">Description</th></tr>
<tr><td>0</td><td colspan="2">Format FP2 de CSI; 3 octets (bytes) par point de donnée</td></tr>
<tr><td>1</td><td colspan="2">ASCII à virgule flottante; 7 octets (bytes) par point de donnée</td></tr>
<tr><td>2</td><td colspan="2">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 colspan="2">RAWS7; 7 points de donnée :</td></tr>
<tr><td></td><td>Points de donnée</td><td>Description</td></tr>
<tr><td></td><td>1</td><td>Pluviométrie totale en “inches”, format = xx.xxx</td></tr>
<tr><td></td><td>2</td><td>Vitesse du vent en “MPH”, format = xxx</td></tr>
<tr><td></td><td>3</td><td>Vecteur moyen de la direction du vent en degrés, format = xxx</td></tr>
<tr><td></td><td>4</td><td>Température de l’air en °F, format = xxx</td></tr>
<tr><td></td><td>5</td><td>Pourcentage d’HR, format = xxx</td></tr>
<tr><td></td><td>6</td><td>Température du combustible en °F , format = xxx</td></tr>
<tr><td></td><td>7</td><td>Tension batterie en V CC, format = xx.x</td></tr>
<tr><td>4</td><td colspan="2">ASCII à décimale pré-positionnée de type xxx.x</td></tr>
<tr><td>5</td><td colspan="2">ASCII à décimale pré-positionnée de type xx.xx</td></tr>
<tr><td>6</td><td colspan="2">ASCII à décimale pré-positionnée de type x.xxx</td></tr>
<tr><td>7</td><td colspan="2">ASCII à décimale pré-positionnée de type xxx</td></tr>
<tr><td>8</td><td colspan="2">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><td>Paramètres
& type de donnée</td><td>Entrée</td></tr>
<tr><td>Dest
Ligne de données</td><td>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.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0</td><td>Commande exécutée avec succès</td></tr>
<tr><td>1</td><td>Erreur de Checksum dans la réponse</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 (différent de STX) reçu après l’adressage SDC</td></tr>
<tr><td>4</td><td>En train de recevoir un « NAK »</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)</td></tr>
<tr><td>9</td><td>Commande invalide</td></tr>
</table></td></tr>
<tr><td>StatusCommand
Constante</td><td>StatusCommand spécifie le type d’information demandée par le transmetteur.
<table>
<tr><th>Code</th><th>Description</th><th>Taille de la ligne de donnée nécessaire</th></tr>
<tr><td>0</td><td>Temps réel - Read time</td><td>4</td></tr>
<tr><td>1</td><td>Etat - Status</td><td>13</td></tr>
<tr><td>2</td><td>Etat du dernier message</td><td>14</td></tr>
<tr><td>3</td><td>Transmet un message aléatoire</td><td>1</td></tr>
<tr><td>4</td><td>Lire le registre des erreurs</td><td>10</td></tr>
<tr><td>5</td><td>Ré-initialise le registre des erreurs</td><td>1</td></tr>
<tr><td>6</td><td>Remet le transmetteur en mode “en ligne”</td><td>1</td></tr>
</table></td></tr>
</table>

Annexe C. Mise en garde de conformité sur les émissions radio

Les changements effectuée aux systèmes radio de la CR200 sans l'accord préalable de Campbell Scientific, peuvent engendrer la non conformité à l'utilisation du produit par l'utilisateur.

Merci de vous conformer aux réglementations locales pour les télécommunications (ART en France, ou FCC aux USA).

LISTE DES AGENCES CAMPBELL SCIENTIFIC DANS LE MONDE

Campbell Scientific, Inc.(CSI)
815 West 1800 North
Logan, Utah 84321
ETATS UNIS
www.campbellsci.com
info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
AFRIQUE DU SUD
www.csafrica.co.za
sales@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 444
Thuringowa Central
QLD 4812 AUSTRALIE
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 BREZIL
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 Hatern Road
Shepshed, Loughborough LE12 9GX
GRANDE BRETAGNE
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
contact@campbellsci.fr

Campbell Scientific Spain, S. L.
Psg. Font 14, local 8
08013 Barcelona
Espagne
www.campbellsci.es
info@campbellsci.es

Campbell Scientific Ltd. (Allemagne)
Fahrenheistrasse1, D-28359 Bremen
Allemagne
www.campbellsci.de
info@campbellsci.de