CLASSEMENT PARTIEL DES INSTRUCTIONS PAR CATÉGORIES

1. Avertissement

Ce classement ne fait apparaître que les instructions les plus utilisées. Pour une documentation complète, se référer à la documentation officielle (en anglais).

Cette liste contient des liens vers des fichiers extérieurs (§) ou des compléments ($) en plus des liens internes.

2. Structures de données

Déclarations

2.1. Les types de variable simple §

Les entiers positifs : **Byte** ; **Integer**

Les entiers relatifs : **Long**

Les nombres réels : **Single**

2.2. Les chaines de caractères §

Les mots ou les phrases d'une longueur inférieure à 127 caractères sont considérés comme des ensembles de tableaux comportant un octet par caractère.

String

2.3. Les tableaux §

Arrays

3. Opérations unaires

Decr

Incr

4. Opérations mathématiques et logiques §

Table des **opérateurs**

5. Structures de traitement

5.1. Structures répétitives

Boucle infinie	**Do … Loop**
Répétitives à compteur	**For … Next**
Répétitives à condition de sortie	**Do … Loop While** condition

5.2. Structures alternatives §

Alternative simple	**If … then**
Alternative complète	**if … then … else … endif**
Alternatives imbriquées	**if … then … elseif … endif**
Choix multiple	**Select … case … end select**

5.3. Saut inconditionnel et sous-programmes

Saut inconditionnel	**Goto**
Appel de sous-programme	**Gosub … Return**
Appel de fonction	voir feuilles « sous-programmes » *Compléments* §

6. Les entrées et les sorties d'informations logiques

Sorties	Entrées
output	**input**
out	**in()**
high	**keyin**
low	**keyinh**

7. Conversion analogique numérique

Adin()

Tadin()

Eadin()

8. Conversion numérique analogique

On détourne les instructions suivantes

Pwm

PwmOff

9. Commande des divers types de moteurs

Commande de moteur à courant continu : **Pwm** ; **PwmOff**

Commande de servomoteur : n'existe pas, il faut construire sa propre routine

Commande des moteurs pas à pas : **Steppulse** et dérivées

10. Traitement des impulsions

10.1. Entrée

Certains ports permettent de compter des impulsions. On peut réaliser ainsi :

- un compteur d'évènements
- un fréquencemètre (en association avec une fonction chronomètre)

Compare

Count()

Countreset

10.2. Sortie

Pulsout

11. Traitement des communications

On appelle communication le fait

- d'envoyer des données vers l'extérieur du microcontrôleur
- d'importer des données depuis l'extérieur.

11.1. Bus I²C et assimilés *Compléments*

I2C	SPI
Set I2C	**Spi**
I2Cstart	**Set spi**
I2Cstop	**Shiftin()**
I2Cwrite()	**Shiftout()**
I2Cread()	
I2Creadna()	

11.2. Liaison série RS232 *Compléments* *Utilisation SD*

Opencom

Put et dérivées

Get et dérivées

Blen **Bfree**

Set RS232

On Recv

WaitTx

12. Gestion des interruptions

Une interruption est un événement dont on sait qu'il arrivera mais on ignore quand, on peut donc difficilement de l'inclure dans un programme ordinaire.

Il y a deux sortes d'interruptions :

- les interruptions internes
- les interruptions externes

Tous les CUBLOC peuvent traiter les interruptions internes.

Le CB320 peut traiter des interruptions externes.

Lorsque l'événement survient, le microcontrôleur lance un sous-programme spécifique (écrit par le programmeur).

12.1. Préparation des interruptions

Set Onglobal **Set OnRecv**

Set OnTimer **Set Onint**

12.2. Interruptions internes

Interruption par chronomètre **On Timer()**

Interruption par liaison série **On Recv**

Interruption par le LADDER **On LadderInt Gosub**

12.3. Interruptions externes

On Int

Adin( )

Variable = ADIN (Canal)

Variable : Variable servant à mémoriser le résultat
Canal : Port de conversion Analogique/Numérique (pas le n° de la broche du CUBLOC)

Les CUBLOC™ disposent d'entrées de convertisseurs Analogiques/Numériques avec une résolution de 10 bits et de sorties PWM (MLI) avec une résolution de 16 bits.
La commande ADIN effectue une conversion de la valeur analogique d'une tension et mémorise le résultat dans la variable. En fonction du modèle de module CUBLOC™ utilisé, le n° du canal de conversion « A/N » sera différent. Pour le CUBLOC™ CB280 par exemple, vous disposez de 8 ports de conversion « A/N » réparties entre P24 à P31. Attention les ports d'entrées de conversion « A/N » doivent être déclarés en tant que port d'entrée avant toute utilisation.
Une tension comprise entre 0 et AVREF est convertie en un nombre compris entre 0 et 1023.
* AVREF peut être compris entre +2 et +5 V (la valeur usuelle est de +5 V).
Si vous utilisez une tension de +3 V pour AVREF, alors les tensions comprises entre 0 et +3 V seront converties en un nombre compris entre 0 et 1023.
Note : Pour le CB220, la valeur de AVREF est prédéfinie à +5 Vcc.

Exemple de programme sur un module « CB280 »

```
DIM A AS INTEGER
INPUT 24     ' Configure le n° de broche du CUBLOC en entrée.
A=ADIN(0)    ' Réalise une conversion "A/N" sur le canal 0 et
             ' mémorise le résultat dans la variable A
```

Comme on peut le voir, le paramètre Canal (qui ici a la valeur 0) correspond au n° canal de conversion « Analogique/Numérique » du module CUBLOC™. Ainsi, dans l'exemple ci-dessus, le port de conversion « Analogique/Numérique » n° 0 est le Port 24 du CB280.Les schémas et la table ci-dessous donnent la correspondance entre canal de conversion « Analogique / Numérique » et de n° de port, en fonction des modèles de CUBLOC™.

Tableau de correspondance.

	CB220	CB280	CB290	CT17X0	CB405
Canal "A/N" 0	Port 0	Port 24	Port 8	Port 0	Port 16
Canal "A/N" 1	Port 1	Port 25	Port 9	Port 1	Port 17
Canal "A/N" 2	Port 2	Port 26	Port 10	Port 2	Port 18
Canal "A/N" 3	Port 3	Port 27	Port 11	Port 3	Port 19
Canal "A/N" 4	Port 4	Port 28	Port 12	Port 4	Port 20
Canal "A/N" 5	Port 5	Port 29	Port 13	Port 5	Port 21
Canal "A/N" 6	Port 6	Port 30	Port 14	Port 6	Port 22
Canal "A/N" 7	Port 7	Port 31	Port 15	Port 7	Port 23
Canal "A/N" 8	-	-	-	-	Port 32
Canal "A/N" 9	-	-	-	-	Port 33
Canal "A/N" 10	-	-	-	-	Port 34
Canal "A/N" 11	-	-	-	-	Port 35
Canal "A/N" 12	-	-	-	-	Port 36
Canal "A/N" 13	-	-	-	-	Port 37
Canal "A/N" 14	-	-	-	-	Port 38
Canal "A/N" 15	-	-	-	-	Port 39

La commande ADIN ne réalise qu'une seule conversion, au moment de son l'exécution. Par contre, la commande TADIN retourne la valeur moyenne de 10 conversions. Il conviendra donc d'utiliser la commande TADIN dans le cadre d'applications nécessitant une plus grande précision, pour convertir des tensions variant très lentement.

Alias

ALIAS NomRegitre = NomAlias

NomRegistre : Nom du Registre tels que P0, M0, T0 (Ne pas utiliser D)
NomAlias : Nom plus représentatif de la fonction du Registre

Les Alias permettent de donner un nom plus explicite aux Registres utilisés dans le LADDER (tels que P0, M0, C0) afin que votre programme soit plus simple à lire.

```
Alias M0 = EtatR
Alias M0 = EtatK
Alias P0 = BPStart
```

Bcd2bin

Variable = BCD2BIN(valeurbcd)

BCD = décimal codé en binaire
Variable : Variable servant à mémoriser le résultat (Retourne un LONG)
valeurbcd : Valeur BCD à convertir en binaire

Cette commande réalise la fonction inverse de BIN2BCD.

```
Dim A As Integer
A=Bcd2bin(&h1234)
Debug  Dec A' Affiche 1234
```

Bclr

BCLR canal, typebuffer

canal : canal RS232 (0 à 3 suivant type de module CUBLOC™)
typebuffer : 0 = Réception, 1 = Transmission, 2 = les 2

Efface le buffer du canal RS-232 sélectionné.

```
Bclr 1,0        ' Efface le canal de réception RS232 n° 1
Bclr 1,1        ' Efface le canal d'émission RS232 n°1
Bclr 1,2        ' Efface les canaux d'émission & de réception RS232 n°1
```

Beep

BEEP Broche, Longueur

Broche : n° de Port (0 à 255)
Longueur : durée de sortie des impulsions (1 à 65535)
La commande BEEP est utilisée pour créer des « bips » sonores. Un buzzer (sans oscillateur) doit être relié sur un des ports du module CUBLOC™. Un bip sonore court sera alors généré. Utilisez cette commande pour, par exemple, confirmer la saisie d'une touche ou réaliser des signaux sonores divers. Cette commande, configure automatiquement la broche en sortie. .

```
BEEP 2, 100 ‘ Génère un Bip sonore sur P2, d'une durée de 100 ms
```

Bfree( )

Variable = BFREE(canal, typebuffer)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single)
canal : canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)
Typebuffer : 0 = Buffer de réception, 1 = Buffer d'émission

Cette commande retourne le nombre d'octets de disponibles dans le buffer de réception ou d'émission. Elle doit être utilisée pour envoyer des données via le port série en évitant un dépassement de la capacité du buffer.

```
DIM A AS BYTE
OPENCOM 1,19200,0, 100, 50 IF BFREE(1,1)>10 THEN
PUT "TECHNOLOGY" END IF
```

Si la taille du buffer est configurée à 50 octets, la valeur maximale retournée sera de 49 car le compte commence à 0 et non à 1.

Bin2bcd

Variable = BIN2BCD(valeurbinaire)

Variable : Variable servant à mémoriser le résultat (Retourne un LONG) valeurbinaire : Valeur binaire à convertir
La commande BIN2BCD convertit un nombre binaire en valeur BCD (afin d'en avoir une expression décimale).
Par exemple: 3451 sera représenté comme suit en binaire et en hexadécimal:

```
          3 4 5 1
0 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1
          0D7B
```

L'expression ci-dessous montre la valeur binaire 3451 convertie en valeur BCD. Comme vous pouvez le voir, chaque groupe de 4 bits représente un chiffre du nombre décimal.

```
          3 4 5 1
0 0 1 1 0 1 0 0 0 1 0 1 0 0 0 1
    3       4       5       1
```

Cette commande permet de convertir une variable binaire afin qu'elle puisse être représentée sur un afficheur 7 segments à Leds.

```
I = 123456
j = bin2bcd(i)
Debug Hex J  ' Affiche 123456
```

Blen ( )

Variable = BLEN(canal, typebuffer)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single) canal : canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)
typebuffer: 0 = Buffer de réception , 1 = Buffer d'émission
La commande Blen() retourne le nombre d'octets disponibles dans le buffer RS232 spécifié. Si le buffer est vide, le nombre 0 sera retourné. Lorsque vous recevez des données dans le buffer, cette commande peut être utilisée pour savoir combien de données ont été reçues avant de les récupérer avec les commandes GET ou GETSTR.
Si le buffer de réception est plein, il ne vous sera plus possible de recevoir d'autres données. Pour éviter cette situation, utilisez les interruptions en cas de réception de données ou augmentez la taille du buffer de réception.

```
Dim A As Byte
Opencom 1,19200,0,100,50
On Recv1 DATARECV_RTN ' Lorsque des données sont reçue sur la
      ' RS232, continuer programme à DATARECV_RTN

Do    ' Boucle sans fin

Loop
DATARECV_RTN:
If Blen(1 ,0) > 0 Then A = Get(1)     ' S'il y a au moins 1 octet de présent…
                                      ' Récupérer 1 octet
End If
Return                                ' Fin de la routine d'interruption
```

Bytein( )

Variable = BYTEIN(Bloc)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single)
Bloc : n° du Bloc d'« E/S » (0 à 15 suivant le type de CUBLOC™ utilisé)

Effectue la lecture de l'état en cours, d'un Bloc d'E/S. Huit broches d'E/S (ports) sont associées à ce qu'on appelle un Bloc. Les ports 0 à 7 correspondent au Bloc 0 et les ports 8 à 15 au Bloc 1. En fonction du modèle de CUBLOC™, le nombre de Blocs peut varier. Lorsque vous utilisez cette commande, toutes les broches d'E/S du Bloc sont configurées en entrées et l'état de ces dernières est mémorisé dans la Variable.

```
DIM A AS BYTE
A = BYTEIN(0)        ' Sauvegarde l'état des ports du Bloc 0 dans la variable A.
```

Les schémas ci-dessous représentent la répartition des blocs en fonction du modèle de CUBLOC.

Byteout

BYTEOUT Bloc, valeur

Bloc : n° du Bloc d'« E/S » (0 à 15 suivant le type de CUBLOC™ utilisé)
valeur : Valeur à appliquer sur les sorties (comprise entre 0 et 255).

Permet d'appliquer une valeur sur un Bloc. Huit broches d'E/S (ports) sont associées à ce qu'on appelle un Bloc. Les ports 0 à 7 correspondent au Bloc 0 et les ports 8 à 15 au Bloc 1. En fonction du modèle de CUBLOC™, le nombre de Blocs peut varier. Lorsque vous utilisez cette commande, toutes les broches d'E/S du Bloc sont configurées en sortie.

```
BYTEOUT 1,255      ' Positionne le Bloc 1 à 255.
                   ' les ports 8 à 15 sont au niveau logique HAUT.
```

* Le port P1 ne pouvant être utilisé qu'en entrée, il en résulte que la commande BYTEOUT 0 ne vous permettra pas d'utiliser le port P1 en sortie.

CheckBf( )

Variable = CheckBf(canal)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single)
canal : Canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)

Cette commande vous permettra de lire l'octet présent dans le buffer de réception RS232 sans l'effacer (contrairement à la commande GET).

```
A = Checkbf(1)          ' Lit la donnée dans le buffer de réception sans modification
```

Compare

Compare Canal, Cible#, Port, Etatcible

Canal : Canal compteur rapide
Cible# : Cible# d'impulsions (CH0 : 0 à 65535, CH1 : 0 à 255)
Port : Port de sortie (Ne pas utiliser les ports d'entrée seul)
Etatcible : Etat Port Cible de sortie

Lorsqu'un compteur rapide atteint la valeur prédéfinie (Cible#), le CUBLOC™ applique un état logique (Etatcible) sur un Port donné.
Si Etatcible = 1 et que le compteur atteint la valeur Cible# alors le Port prendra le niveau logique haut « 1 ». A l'inverse, si Etatcible = 0 et que le compteur atteint la valeur Cible# alors le Port prendra le niveau logique bas « 0 ».

Canal	Gamme de comparaison
Canal 0 (compteur rapide)	0 ~ 255
Canal 1 (compteur rapide)	0 ~ 65535

Bien que le compteur rapide du CUBLOC™ puisse fonctionner sur 32 bits, la commande COMPARE est limitée afin que le « système d'exploitation » du CUBLOC™ ne soit pas affecté dans sa gestion « multitâches BASIC / Ladder ».
Note : Pour le canal 0, utilisez la commande Set Count0 On avant d'utiliser la commande Compare.

```
Dim i As Integer
Set Count0 On
Compare 0,10,61,1

Do
i = Count(0)
Debug Goxy,0,0,dec4 i,Cr
Delay 100
Loop
```

Cet exemple de programme utilise le compteur rapide 0 avec une Cible# de 10 impulsions. Lorsque le compteur atteint 11 impulsions, le port P61 passe au niveau logique « haut ».

Count( )

Variable = COUNT(canal)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single)
Canal : n° du Canal Compteur (0 ou 1).

Retourne la valeur du canal compteur spécifié. Il vous faut au préalable configurer le port en entrée avant d'utiliser cette commande.
Le comptage peut s'effectuer sur 32 bits (Byte, Integer, Long). La fréquence maximum est de l'ordre de 500 KHz.
Les compteurs des modules CUBLOC™ sont gérés de façon matérielle (c'est à dire qu'ils fonctionnent de façon indépendante de l'exécution du programme principal). Ils seront ainsi capables d'effectuer un comptage en « temps réel » (quelque soit l'état d'occupation du processeur du module CUBLOC™).
Les modules CUBLOC™ disposent de 2 Compteurs. Le compteur du Canal 0 utilise les mêmes ressources que les fonctions PWM0, 1, 2 et ne pourra donc pas être utilisé en même temps que ceux-ci. Toutefois le compteur du Canal 1 pourra être utilisé librement.
Pour exploiter le compteur du Canal 0, la commande SET COUNT0 ON devra être utilisée au préalable. L'exploitation du compteur du Canal 1 ne nécessite aucune déclaration préalable.

```
Dim R As Integer
Input 15          ' Configure le port P15 en entrée (compteur du Canal 1 ).
R = Count(1)      ' Lecture de la valeur du compteur.
Set Count0 On     ' Active le compteur du Canal 0
                  ' (les fonctions PWM 0,1,2 deviennent inutilisables).
Input 14          ' Configure le port P15 en entrée (compteur du Canal 0).
R = Count(0)      ' Lecture de la valeur du compteur.
```

Du fait que le compteur du Canal 0 utilise les mêmes ressources que les fonctions PWM (comme indiqué ci-dessous), gardez à l'esprit qu'il ne sera pas possible d'utiliser toutes ces fonctions en même temps.

```
' Exemple de mesure de la fréquence des impulsions de sortie canal PWM 0
Const Device = CB280
Dim A as Integer
Input 15
Low 5
Freqout 0,2000
Low 0
On Timer(100) Gosub GetFreq
Do
Loop

GetFreq:
A=Count(1) Debug goxy,10,2
Debug dec5 A Countreset 1 Reverse 0
Return
```

Countreset

COUNTRESET canal

Canal : n° du Canal Compteur (0 ou 1)
Remet à 0 le Compteur du Canal spécifié.

```
COUNTRESET 0      ' Reset le compteur du Canal 0
COUNTRESET 1      ' Reset le compteur du Canal 1
```

Dcd

Variable = Source DCD

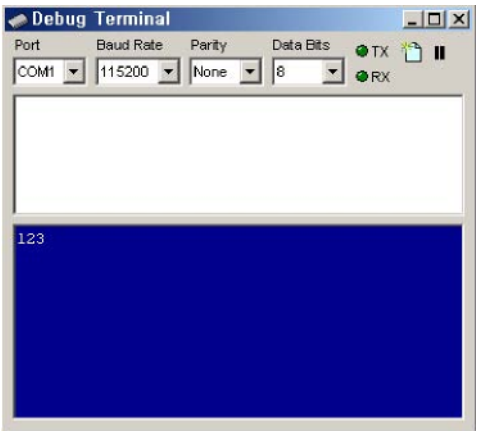
Variable : Variable servant à mémoriser le résultat (ni String, ni Single).
Source : Valeur source
La commande DCD réalise l'inverse de la commande NCD.
Cette commande retourne la position du bit à 1 de poids le plus fort (en partant du bit de poids faible, LSB 0).

```
I = DCD 15     ' Le résultat est 3 car 15 = 0b00001111
```

Debug

DEBUG donnée

donnée : donnée à envoyer au PC
Les modules CUBLOC™ disposent d'une commande de DEBUG qui pourra être insérée à plusieurs reprises où vous voulez dans votre programme. Le résultat de cette commande sera affiché dans la fenêtre du terminal de DEBUG du PC lorsque le programme sera exécuté et qu'il arrivera sur une commande DEBUG.




```
DIM A AS INTEGER
A = 123
DEBUG DEC A
```

Utilisez DEC ou HEX pour afficher des nombres. Si vous n'utilisez ni DEC, ni HEX, les nombres seront représentés par leur code ASCII. Utilisez également DEC ou HEX pour afficher l'état des variables.
Si vous insérez un point d'interrogation (?) après DEC ou HEX, le nom de la variable sera affiché en même temps que sa valeur. Cr est le symbole du retour chariot.

DEBUG DEC? A,CR DEBUG HEX? A,CR

Vous pouvez également limiter le nombre de chiffres à afficher. DEBUG HEX8 A
1 à 8 peuvent être utilisés avec HEX.
HEX8 affichera un nombre héxadécimal à 8 chiffres. 1 à 10 peuvent être utilisés avec DEC.
Vous êtes libre de mixer les variables de type strings, nombre, etc… DEBUG "CHECK VALUE " HEX? A, CR
Les commandes de DEBUG sont utiles pour trouver rapidement et facilement vos erreurs de programmation. Vous pourrez vous assurer que le programme s'exécute à un endroit particulier (celui où vous avez placé la commande DEBUG) et d'autre part, de suivre l'évolution de vos variables durant l'exécution de votre programme.
Si vous saisissez des caractères dans la partie blanche de la fenêtre du Terminal de DEBUG, ces derniers sont envoyés vers le port de téléchargement du module CUBLOC™. Cette fonction est réservée à un usage futur .
Attention : Les commandes DEBUG du BASIC et le mode MONITORING du LADDER ne doivent jamais être utilisées en même temps
Le tableau ci-dessous montre les différentes possibilités offertes par les commandes DEBUG (vous retrouvez de grandes similitudes avec les commandes d'afficheurs LCD).

Commande	Code	Explications	Exemple
CLR	0	Efface l'écran de DEBUG	Debug CLR
HOME	1	Déplace le curseur en haut à gauche de la fenêtre de DEBUG	Debug HOME
GOXY	2	Déplace le curseur à la position X, Y	Debug GOXY, 4, 3
CSLE	3	Déplace le curseur d'une position vers la gauche	
CSRI	4	Déplace le curseur d'une position vers la droite	
CSUP	5	Déplace le curseur d'une position vers le haut	
CSDN	6	Déplace le curseur d'une position vers le bas	
BELL	7	Génére un "bip sonore" sur le PC	
BKSP	8	Equivalent de la barre ESPACE	
LF	10	Passage à la ligne	Debug "ABC",LF
CLRRI	11	Efface tous les caractères à droite du curseur jusqu'à la fin de la ligne.	
CLRDN	12	Efface tous les caractères en bas du curseur	
CR	13, 10	Equivalent touche "Return" (va à la ligne suivante)	Debug, "ABC",CR

Vous pouvez utiliser plusieurs variantes de la commande DEBUG.
DEBUG GOXY,5,5,DEC I
DEBUG CLR,"TEST PROGRAM"

Decr

DECR variable

Variable : Variable à décrémenter (ni String, ni Single).
Décrémente la variable d'une unité (similaire à "A - -" en langage « C ») DECR A ‘
Décrémente A d'une unité.

Delay

DELAY Durée

Durée : variable ou constante (de type Long ou inférieur)
Réalise une temporisation exprimée en millisecondes. La commande Delay ne doit être utilisée que pour générer des temporisations de faible durée. Il n'est pas conseillé d'avoir recours à cette commande pour réaliser des mesures de durées ou une gestion temporelle où la précision est prépondérante. Le microcontrôleur est indisponible pendant la temporisation.

```
DELAY 10     ‘ Temporisation d'environ 10 ms.
DELAY 200    ‘ Temporisation d'environ 200 ms
```

Do...Loop

La commande DO...LOOP réalise une structure répétitive infinie.
Les commandes DO WHILE ou DO UNTIL réalisent une structure répétitive associée à une condition de sortie. La commande EXIT DO force la sortie d'une structure de type DO... LOOP.

```
Dim K As Integer
Do
K=Adin(0)      ‘ Lecture de l'entrée « A/N » du canal 0
Debug Dec K,Cr
Delay 1000
Loop
```

Dans l'exemple ci-après, le programme effectuera une boucle entre DO et LOOP. Les commandes EXIT DO ou GOTO permettent de sortir d'une boucle.

Les instructions DO … WHILE et DO … UNTIL ont deux formes, évaluation de la condition en entrée ou en sortie de la boucle.

Do While [Condition]
Commands
[Exit Do]
Loop

Do
Commands
[Exit Do]
Loop While [Condition]

DO. .WHILE réalisera une répétition tant que [Condition] est vraie.

Do Until [Condition]
Commands
[Exit Do]
Loop

Do
Commands
[Exit Do]
Loop Until [Condition]

DO. .UNTIL réalisera une boucle jusqu'à ce que [Condition] soit vraie.

Les conditions de sortie pour WHILE et pour UNTIL sont complémentaires

Dtzero

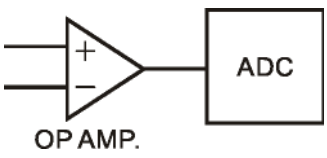
DTZERO variable

Variable : Variable à décrémenter (ni String, ni Single).
Décrémente la variable d'une unité. Lorsqu'elle arrive à 0, elle n'est plus décrémentée.
DTZERO A ' Décrémente A d'une unité.

Eadin( )

Variable = EADIN (mux)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single).
mux : Combinaison Port d'entrée de conversion « A/N » (0 à 21)



Cette commande réalise la conversion « A/N » de la différence de deux tensions. On réalise ainsi un effet de loupe au-dessus de la tension de référence (celle sur OPAMP-) Les modules CUBLOC™ disposent d'un OPAMP intégré. Avec la commande ADIN, cet OPAMP n'est pas exploité.

Sélectionnez la valeur de « mux » en fonction du tableau ci-dessous:

MUX	OPAMP +	OPAMP -	Multiplicateur	Résolution
0	ADC0	ADC0	10	8 Bits
1	ADC1	ADC0	10	8 Bits
2	ADC0	ADC0	200	7 Bits
3	ADC1	ADC0	200	7 Bits
4	ADC2	ADC2	10	8 Bits
5	ADC3	ADC2	10	8 Bits
6	ADC2	ADC2	200	7 Bits
7	ADC3	ADC2	200	7 Bits
8	ADC0	ADC1	1	8 Bits
9	ADC1	ADC1	1	8 Bits
10	ADC2	ADC1	1	8 Bits
11	ADC3	ADC1	1	8 Bits
12	ADC4	ADC1	1	8 Bits
13	ADC5	ADC1	1	8 Bits
14	ADC6	ADC1	1	8 Bits
15	ADC7	ADC1	1	8 Bits
16	ADC0	ADC2	1	8 Bits
17	ADC1	ADC2	1	8 Bits
18	ADC2	ADC2	1	8 Bits
19	ADC3	ADC2	1	8 Bits
20	ADC4	ADC2	1	8 Bits
21	ADC5	ADC2	1	8 Bits

Le port EADIN doit être configuré en entrée avant toute utilisation. En exploitant l'OPAMP une mesure plus précise (liée à un filtrage du « bruit ») sera obtenue.

```
Dim J As Long
Input 24          ' Configure le port en entrée (Utiliser les ports 24 et 25 pour le CB280)
Input 25
Do
j = Eadin(8)      ' Conversion « A/N » depuis AD0 et Ad1, via OPAMP, 1
Locate 0,0
Print hex5 J,cr   ' Affiche résultat sur le LCD
Delay2 500        ' Petite temporisation
Loop
End

Sub Delay2(DL As Integer)
Dim I As Integer
For I = 0 To DL
Next
End Sub
```

La commande EADIN n'exploite pas la résolution maximale de 10 bits obtenue avec la commande ADIN. Lorsque vous utilisez les multiplicateurs X1 et X10 vous travaillez avec une résolution de 8 bits. Lorsque vous utilisez le multiplicateur x200 vous travaillez avec une résolution de 7 bits.
ATTENTION : L'OPAMP dispose de caractéristiques qui ne lui permettent une lecture qu'entre 0,5 V et 4,5 V. Avec le module CUBLOC™ « CB405 », la commande EADIN ne peut être exploitée qu'avec les canaux 0 à 7.
Consultez le tableau ci-dessous pour obtenir la correspondance entre les canaux de conversion « A/N » et les n° de port de votre module CUBLOC™ ou CUTOUCH.

Canal	CB220	CB280	CB290	CT17X0	CB405
ADC0	PORT 0	PORT 24	PORT 8	PORT 0	PORT 16
ADC1	PORT 1	PORT 25	PORT 9	PORT 1	PORT 17
ADC2	PORT 2	PORT 26	PORT 10	PORT 2	PORT 18
ADC3	PORT 3	PORT 27	PORT 11	PORT 3	PORT 19
ADC4	PORT 4	PORT 28	PORT 12	PORT 4	PORT 20
ADC5	PORT 5	PORT 29	PORT 13	PORT 5	PORT 21
ADC6	PORT 6	PORT 30	PORT 14	PORT 6	PORT 22
ADC7	PORT 7	PORT 31	PORT 15	PORT 7	PORT 23

Eeread()

Variable = EEREAD (Addresse, NbOctet)

Variable : Variable servant à mémoriser le résultat (Ni String, ni Single). Addresse : 0 à 4095
NbOctet : Nombre d'octets à lire (1 à 4)
Lecture de données depuis une adresse en EEPROM.
Voir la documentation officielle §

Ekeypad()

Variable = EKEYPAD(BlockIn, BlockOut)

Variable : Variable servant à mémoriser le résultat (Retourne un Byte) BlockIn : Bloc devant recevoir les entrées (0 à 15)
BlockOut : Bloc devant recevoir les sorties (0 à 15)

La commande EKEYPAD permet d'étendre les possibilités de la commande KEYPAD afin de lire jusqu'à 64 touches. Deux blocs de ports devront être employés. Le bloc de ports d'entrées (avec résistances de tirage à ramener au +5 V) et le bloc de ports de sorties devront être sélectionnés indépendamment.
Si le clavier dispose de moins de 64 touches :
Il vous faudra toujours ajouter une résistance de tirage (à ramener au +5 V) sur les ports du Bloc d'entrée non utilisés. De plus ces broches ne devront en aucun cas être utilisées pour une autre fonction.
De même, les ports non utilisés du bloc de sortie devront rester « en l'air » et ne devront en aucun cas être utilisés pour une autre fonction.
L'exemple ci-dessous montre comment on utilise le Bloc 0 en entrée et le Bloc 1 en sortie.

Si aucune touche n'est sollicitée, le nombre 255 sera retourné. A l'inverse le code de la touche sollicitée sera retourné.

For...Next

FOR…NEXT

réalise une structure répétitive à compteur.

For Variable = Valeur début To Valeur fin [Pas Incrémental]
Commandes
[Exit For]
Next

Dans l'exemple ci-dessous, l'option de pas incrémental n'est pas utilisée. Le compteur de la boucle FOR…NEXT s'incrémente par défaut d'une unité.

```
Dim K As Long
For K=0 To 10
Debug Dp(K),CR Next
For K=10 To 0 Step –1          ' Décrémentation de 10 jusqu'à 0. Debug Dp(K),CR
Next
```

La commande EXIT FOR permet de sortie d'une boucle FOR…NEXT à n'importe quel moment.

```
For K=0 To 10
Debug Dp(K),CR
If K=8 Then Exit For   ' Si K = 8 alors on sort de la boucle FOR…NEXT.
Next
```

Lorsque vous utilisez une variable dans une boucle de type FOR…NEXT, vérifiez impérativement qu'elle soit capable de couvrir la plage complète des valeurs demandées. Par exemple les variables de type Byte ne pourront aller que de 0 à 255. Pour des nombres plus grands, une variable avec une plus grande gamme de variation devra être choisie.

```
Dim K As Byte
For K=0 To 255
Debug Dp(K),CR
Next
```

Lorsque vous utilisez une option Incrémentale négative (STEP –xx), pensez impérativement à choisir une variable de type LONG afin de traiter les nombres négatifs.

```
Dim LK As Long
For LK=255 To 0 Step –1 ' Le nombre -1 sera atteint en fin de boucle
Debug Dp(LK),CR
Next
```

Exemples de programmes :

```
Const Device = CB280 Dim A As Integer
For A=1 To 9
Debug "3 * "
Debug Dec A
Debug " = "
Debug Dec 3*A,Cr Next
```

```
Const Device = CB280
Dim A As Integer, B As Integer For A=2 To 9
For B=1 To 9
Debug Dec A," * "
Debug Dec B
Debug " = "
Debug Dec A*B,Cr
Next
Debug Cr
Next
```

Freepin

FREEPIN Port

Port : n° du port du CUBLOC™

Permet la libre configuration du Port en Ladder en utilisant la commande Usepin en BASIC.

Freqout

FREQOUT Canal, ValFreq

Canal : Canal PWM (0 à 15)
ValFreq : Frequence comprise entre 1 et 65535

La commande FREQOUT permet de générer une fréquence sur un canal PWM. Veuillez vous assurer d'indiquer sur quel canal PWM vous voulez travailler (n'indiquez pas le n° de la broche). Pour le module CUBLOC™ CB220 et CB280, les ports 5, 6 et 7 sont respectivement attribués aux canaux PWM 0, 1 et 2.
Les tableaux ci-dessous donnent une indication de la correspondance entre le paramètre ValFreq et la fréquence obtenue. Une valeur de 1 pour ValFreq correspondra à la fréquence la plus élevée tandis que la valeur de 65535 à la fréquence la plus basse. Une valeur de 0 pour ValFreq ne générera aucune fréquence

Il vous est également possible de calculer la fréquence en fonction de la valeur de ValFreq grâce à la formule ci-dessous.
ValFreq = 2304000 / fréquence désirée

Avant de générer une fréquence, il faut configurer le port PWM en sortie. Pour stopper le signal, vous devez utiliser la commande PWMOFF.
Voir l'exemple ci-dessous:

```
Const Device = cb280
Dim i As Integer
Low 5           ' Configure la broche 5 en sortie (et au niveau bas).
Freqout 0,10    ' Génère une fréquence de 209.3 Khz
Do              ' Boucle infinie
Loop
```

Du fait que la commande FREQUOUT utilise les mêmes ressources que les signaux PWM, il y a certaines restrictions qu'il faut connaître.
Les canaux PWM 0,1 et 2 utilisent les mêmes timers. Si le canal PWM 0 est utilisé pour la commande FREQUOUT, il ne sera plus possible d'utiliser les canaux 0,1 et 2 pour générer des signaux PWM.
Il en sera de même avec les canaux PWM 3, 4 et 5 qui seront inutilisables pour la génération de signaux PWM si vous utilisez la commande FREQOUT pour générer une fréquence sur le canal PWM 3.
A noter qu'il est possible de générer des fréquences différentes sur les canaux PWM 0 et 3.
Pour résumer, vous pourrez générer 2 fréquences différentes à la fois à l'aide de la commande FREQUOUT mais dès lors, il ne vous sera plus possible de générer de signaux PWM.
La commande FREQUOUT pourra également générer des notes de musiques. Le tableau ci-dessous donne une correspondance entre la valeur du paramètre ValFreq et les notes de musique.

ValFreq	Fréquence
1	1152 KHz
2	768 kHz
3	576 KHz
4	460.8KHz
5	384 KHz
10	209.3 KHz
20	109.7 KHz
30	74.4 KHz
100	22.83 KHz

ValFreq	Fréquence
200	11.52 KHz
1000	2.3 KHz
2000	1.15 KHz
3000	768 Hz
4000	576 Hz
10000	230 Hz
20000	115.2 Hz
30000	76.8 Hz
65535	35.16 Hz

Note	Octave 2	Octave 3	Octave 4	Octave 5
A	20945	10473	5236	2618
Bb	19770	9885	4942	2471
B	18660	9330	4665	2333
C	17613	8806	4403	2202
Db	16624	8312	4156	2078
D	15691	7846	3923	1961
Eb	14811	7405	3703	1851
E	13979	6990	3495	1747
F	13195	6597	3299	1649
Gb	12454	6227	3114	1557
G	11755	5878	2939	1469
Ab	11095	5548	2774	1387

```
Freqout 0,5236          ‘ Note A en Octave 4(440Hz)
Freqout 0,1469          ‘ Note G en Octave 5
```

Get()

Variable = GET(canal, NbData)

Variable : Variable servant à mémoriser le résultat (non String, ni Single)
canal : Canal RS232 (0 à 3 suivant modèle de CUBLOC™ utilisé)
NbData : Nombre de données à recevoir (1 à 4)

Lecture des données depuis le port RS232. La commande GET() effectue une lecture des données depuis le buffer de réception RS232. Si aucune donnée n’est présente dans le buffer de réception, la commande n’est pas effectuée (vous pourrez également utiliser la commande BLEN() pour vous assurer au préalable de la présence de données dans le buffer avant d’essayer de les récupérer). Le nombre de données à lire devra être compris entre 1 et 4. Pour recevoir un seul octet, le paramètre NbData devra être 1. Pour recevoir une donnée de type Long, le paramètre NbData devra donc être 4. Pour recevoir une plus grande quantité de données, utilisez la commande GETSTR().

Astuce : Utilisez la commande SYS(1) après GET() ou GETSTR() pour vérifier combien de données ont été réellement lues. Si 5 octets ont été reçus et seulement 4 vérifiés… 1 octet aura été perdu.

```
Const Device = cb280 Dim A as Byte
Opencom 1,115200,3,50,10
On Recv1 Gosub GOTDATA
Do
   Do while In(0) = 0
   Loop             ' Attend jusqu’à ce qu’un BP connecté sur P0 soit sollicité
   Put 1,asc("H"), 1
   Put 1,asc("E"), 1
   Put 1,asc("L"), 1
   Put 1,asc("L"), 1
   Put 1,asc("O"),1
   Put 1,13,1       ' HELLO + Chr (13) + Chr (10)
   Put 1,10,1
   Do while In(0) = 1
   Loop
Loop
```

Geta

GETA canal, NomTableau, NbData

canal : RS232 Canal (0 à 3 suivant modèle de CUBLOC™ utilisé)
NomTableau : Tableau servant à mémoriser les données
NbData : Nombre de données à mémoriser (1 à 65535)

La commande GETA peut être utilisée pour recevoir des octets depuis le port RS232 , ces données seront stockées dans un tableau (le stockage débute depuis le premier élément du tableau). Vérifiez au préalable la présence de données dans le buffer RS232 à l'aide de la commande BLEN() afin de ne pas remplir le tableau avec des données non désirées.

```
Const Device = cb280
Dim A(10) As Byte
Opencom 1,115200,3,50,10
Set Until 1,8
On Recv1 Gosub GOTDATA
Do
   Do While In(0) = 0
   Loop        ' Attend jusqu'à ce qu'un BP connecté sur P0 soit sollicité
   Putstr 1,"CUBLOC",Cr
   Do While In(0) = 1
   Loop
Loop
GOTDATA:
Geta 1,A,8
Debug A(0),A( 1),A(2),A(3),A(4),A(5),A(6),A(7)
Return
```

Geta2

GETA2 canal, NomTableau, NbData, CarStop

canal : RS232 Canal (0 à 3 suivant modèle de CUBLOC™ utilisé)
NomTableau : Tableau servant à mémoriser les données
NbData : Nombre de données à mémoriser (1 à 65535)
CarStop : caractère ASCII d'arrêt

La commande GETA2 s'utilise de la même façon que la commande GETA à l'exception que la lecture des données s'arrêtera au caractère ASCII CarStop (même si d'autres
données restent à lire). Si le caractère CarStop n'est pas trouvé alors cette commande réagit comme la commande GETA.
CarStop sera stocké dans une variable de type String. Vous pouvez utiliser la commande SYS(1) pour connaître le nombre d'octets lus.

```
Dim A(10) As Byte
Opencom 1,19200,0,50,10
Geta2 1,A,20, 10      ' Lecture jusqu'à ce que le caractère ASCI I 10 soit trouvé ou que
                      ' 20 octets aient été reçus.
```

Getcrc

GETCRC variable, NomTableau, NbData

Variable : variable String pour mémoriser le résultat (type Integer) NomTableau : Tableau avec données (tableau de type Byte) NbData : Nombre de données pour calculer le CRC
Cette commande est dédiée au calcul de CRC lorsque vous utilisez le MODBUS RTU en mode maître.
Voir la documentation officielle. §

Getstr()

Variable = G ETSTR(canal, N b Data)

Variable : Variable (de type string) servant à mémoriser le résultat
canal : Canal RS232 Canal
NbData : Nombre de données à recevoir

Identique à Get() mis à part que la variable servant à recevoir les données est du type String.

```
Const Device = cb280

Dim A As String * 10
Opencom 1,115200,3,50,10
Set Until 1,8
On Recv1 Gosub GOTDATA
Do
   Do While In(0) = 0          ' Attend jusqu'à ce qu'un BP connecté sur P0 soit sollicité
   Loop
   Putstr 1,"CUBLOC",Cr

   Do While In(0) = 1
   Loop
Loop

GOTDATA:
A=Getstr(1 ,8)
Debug A
Return
```

Getstr2()

Variable = GETSTR2(canal, NbData, CarStop)

Variable : Variable (de type string) servant à mémoriser le résultat
canal : Canal RS232 Canal
NbData : Nombre de données à recevoir
CarStop : caractère ASCII d'arrêt (0x1A pour ctrl Z)

La commande GETSTR2 s'utilise de la même façon que la commande GETSTR à l'exception que la lecture des données sera arrêtée à la détection du caractère ASCII CarStop (même si d'autres données restent à lire). Si le caractère CarStop n'est pas trouvé alors cette commande réagit comme GETSTR.

Gosub … Return

La commande GOSUB est destinée à appeler une sous-routine. La commande RETURN doit être utilisée pour terminer la sous-routine.

```
GOSUB ADD_VALUE

ADD_VALUE: A=A+1
RETURN
```

Goto

La commande GOTO permet au programme de continuer son exécution à un autre endroit spécifié (repéré par une étiquette). Cette commande est présente dans la plupart des langages BASIC usuels.

```
If I = 2 Then
Goto LAB1
End If
LAB1:
I = 3
```

A propos des « étiquettes »…
Une « étiquette » sera définie en ajoutant le caractère ':' à la fin d'un mot. Dès lors, il vous sera possible d'y faire référence avec les instructions GOTO ou GOSUB afin que votre programme puisse « s'y rendre ».

```
ADD_VALUE:
LINKPOINT:
```

Une « étiquette » ne doit pas utiliser de mots réservés, de constantes, d'espace, ou commencer par un chiffre.
Ci-dessous quelques exemples «d'étiquettes » qu'il ne faut pas utiliser:

```
Ladder:        ' Constante réservée
123:   ' Etiquette avec des chiffres.
Aboot 10:      ' Etiquette avec un espace.
```

High

HIGH Port

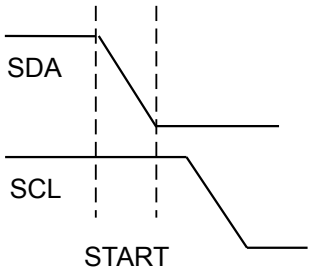
Port : n° de Port du module CUBLOC™
Cette commande applique un niveau logique HAUT (+5 V) sur le Port du module CUBLOC™. Elle configure le Port en sortie et y applique un niveau logique HAUT (+5 V).

```
OUTPUT 8    ' Configure le Port 8 en sortie.
HIGH 8      ' Applique un niveau logique HAUT (+5 V) sur le Port 8.
```

Lorsque qu'un Port est activé par la commande HIGH, ce dernier se retrouve connecté à VDD (lorsqu'il est configuré par LOW, ce Port est connecté à VSS).

I2Cstart

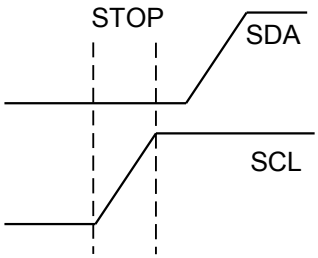
I2CSTART



Génère (via les signaux SDA et SCL) une condition « Start » sur le bus I2C™ . Après cette commande les signaux SDA et SCL sont au niveau logique BAS.

I2Cstop

I2CSTOP



Génère (via les signaux SDA et SCL) une condition « Stop » sur le bus I2C™ . Après cette commande les signaux SDA et SCL sont au niveau logique HAUT.

I2Cread( )

Variable = I2CREAD()

Variable : Variable servant à mémoriser le résultat (ni String, ni Single).
nimportequoi : valeur quelconque non prise en compte.

Lecture d'un octet depuis le bus I2C™ (initialisé à l'aide de la commande SET I2C). Utilisez n'importe quelle valeur pour le paramètre dummy.

A = I2CREAD(0)

Cette commande renverra un signal d'acknowledge « ACK » au composant esclave I2C™ que vous piloterez. Ainsi après la lecture d'un octet, une impulsion sur « SCL » sera envoyée tandis que SDA sera maintenu au niveau logique « BAS ». Ceci sera interprété comme un signal d'acknowledge par le composant esclave I2C™.

I2Creadna( )

Variable = I2CREADNA(nimportequoi)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single).
nimportequoi : valeur quelconque.

Cette commande s'utilise exactement comme la précédente, mis à part que le module CUBLOC™ ne génèrera aucun signal d'acknowledge après la lecture.

I2Cwrite

Variable = I2CWRITE (donnée)

Variable : Acknowledge (0 = Acquittement reçu , 1 = Acquittement non reçu)
donnée : Donnée à envoyer

Envoie un octet sur le bus I2C™. Cette commande vérifie l'acquittement envoyé par le récepteur Elle retourne la valeur 0 si l'acquittement du récepteur est reçu et 1 dans le cas contraire. L'absence de signal d'acquittement peut être dûe à :

- une adresse du composant I2C™ mal configurée,
- un mauvais raccordement des signaux SDA et SCL,
- un problème d'alimentation, un défaut du composant I2C™, etc…

Il est intéressant de prévoir une vérification de la bonne communication I2C™
(voir exemple ci-dessous) :
IF I2CWRITE(DATA) = 1 THEN GOTO ERR_PROC
La transmission d'un octet nécessite environ 60 µs

If…Then…Elseif...Else…EndIf

Ces commandes réalisent les différentes structures alternatives.

If Condition1 Then [Expression1]
[Expression2] [Elseif Condition2 Then
[Expression3]] [Else
[Expression4]] [End If]

Exemple n° 1 de « rédaction » possible
If A<10 Then B=1

Exemple n° 2 de « rédaction » possible
If A<10 Then B=1 Else C=1

Exemple n° 3 de « rédaction » possible
If A<10 Then '* Lorsque vous utilisez plus d'une ligne,
B=1 '* ne mettez pas d'expression sur la ligne du Then. End If

Exemple n° 4 de « rédaction » possible
If A<10 Then
B=1 Else
C=1 End If

Exemple n° 5 de « rédaction » possible If A<10 Then
B=1
Elseif A<20 Then
C=1
End If

In( )

Variable = IN(Port)

Variable : Variable servant à mémoriser le résultat
Port : Port du module CUBLOC (0 à 255)

Cette commande sauvegarde le niveau logique présent sur un Port du module CUBLOC™ dans la variable. L'exécution de cette commande a également pour conséquence de configurer automatiquement le Port en entrée (sans que vous ayez besoin de l'indiquer au préalable au module CUBLOC™).

```
DIM A AS BYTE
A = IN(8)       ' lecture du niveau logique du Port 8 et mémorise le
                ' résultat (0 ou 1) dans la variable A
```

Rappel
Tous les CUBLOC™ disposent de Ports pouvant être configurés en « E/S ». Vous disposez de nombreuses options pour configurer le rôle (Entrée ou Sortie) de chaque Port. Par défaut, tous les Ports d'« E/S » sont configurés en haute impédance (HIGH-Z) à la mise sous tension.
Lorsqu'un Port est configuré en sortie, il pourra alors être amené à prendre un niveau logique « BAS » (0 V) ou « HAUT » (+ 5 V).

Incr

INCR variable

Variable : Variable à incrémenter.
Ajoute 1 à la variable.

```
INCR A        ' Incrémente A d'une unité.
```

Input

INPUT Port

Port : Port du module CUBLOC (0 à 255)

Configure le Port spécifié en entrée, il est en haute impédance (High-Z).
Toutes les entrées des CUBLOC™, sont configurées en entrée à la mise sous tension du module. Ceci signifie que le port n'impose aucune tension au circuit extérieur, ni un niveau HAUT, ni un niveau BAS, il ne peut donc y avoir de court-circuit entre un HAUT extérieur et un BAS interne.
INPUT 8 Configure le Port 8 en entrée.

Keyin

Variable = KEYIN(Port, dureeantirebond)

Variable : Variable servant à mémoriser le résultat
Port : Port du module Cubloc™
dureeantirebond : Durée anti-rebond

La commande KEYIN permet d'éliminer auant que faire se peut, les effets de « rebonds » que certaines touches ou boutons-poussoirs peuvent génerer. Cette commande ne peut être utilisée que lorsque votre bouton-poussoir met une entrée A LA MASSE. Si le bouton poussoir est câblé pour mettre l'entrée au NIVEAU HAUT, utilisez alors KEYINH. Lorsque la touche est sollicitée, la commande KEYIN retourne 0 et 1 lorsque la touche n'est pas sollicitée.
Si vous utilisez une valeur de 10 pour le paramètre « dureeantirebond » alors le module CUBLOC™ vérifiera qu'un état soit stable à l'entrée de la broche pendant environ 10 ms (cette durée est la valeur typique à utiliser).

```
A = KEYIN(1,10)       ' Lecture d'une entrée avec filtrage anti-rebond.
```

Keyinh

Variable = KEYINH(Port, dureeantirebond)

Variable : Variable servant à mémoriser le résultat
Port : Port du module Cubloc™
dureeantirebond : Durée anti-rebond

La commande KEYINH permet d'éliminer autant que faire se peut, les effets de « rebonds » que certaines touches ou boutons-poussoirs peuvent génerer. Cette commande ne peut être utilisée que lorsque votre bouton-poussoir met une entrée AU NIVEAU HAUT. Si le bouton poussoir est câblé pour mettre l'entrée à la MASSE, utilisez alors KEYIN. Lorsque la touche est sollicitée, la commande KEYIN retourne 1 et 0 lorsque la touche n'est pas sollicitée.
Si vous utilisez une valeur de 10 pour le paramètre « dureeantirebond » alors le module CUBLOC™ vérifiera qu'un état soit stable à l'entrée de la broche pendant environ 10 ms (cette durée est la valeur typique à utiliser).

```
A = KEYINH(1,10)      ' Lecture d'une entrée avec filtrage anti-rebond.
```

Keypad

Variable = KEYPAD(BlocPort)

Variable : Variable servant à mémoriser le résultat (Retourne un octet)
BlocPort : Bloc Port

La commande KEYPAD permet de lire l'état d'un clavier matricé de 16 touches max. via un bloc d'entrées. Les 4 bits de poids faibles du bloc devront être associés aux entrées du clavier (avec des résistances de tirage à ramener au +5 V) et les 4 bits de poids forts serviront en tant que sorties pour piloter le clavier matricé (voir schéma ci-dessous).
Si le clavier dispose de moins de 16 touches : Il vous faudra toujours ajouter une résistance de tirage (à ramener au +5 V) sur les ports d'entrée non utilisés du Bloc. De plus ces broches ne devront en aucun cas être utilisées pour une autre fonction. De même, les ports de sortie non utilisés du bloc ne devront en aucun cas être utilisés pour une autre fonction.

```
A = KEYPAD(0)        ‘ Lecture de l'état du clavier via le bloc 0
```

Si aucune touche n'est sollicitée, le nombre 255 sera retourné. A l'inverse, le code de la touche sollicitée sera retourné.

Ladderscan

LADDERSCAN

La commande LADDERSCAN lance une scrutation du programme LADDER
Lorsque la commande est placée dans une boucle sans fin telle que DO…Loop, il en résulte une accélération de traitement du LADDER.

```
Const Device = CB280          ' Déclaration du type de CUBLOC utilisé
Usepin 0,In,START ' Déclaration des Ports Usepin 1 ,In,RESETKEY
Usepin 2,In,BKEY
Usepin 3,Out,MOTOR
Alias M0=RELAYSTATE       ' Alias
Alias M1=MAINSTATE
Do
LadderScan
Loop
```

Low

LOW Port

Port: n° du Port du Cubloc (0 à 255)
Met un Port au niveau logique bas (0 V) – Cette commande configure le Port en sortie et y niveau logique « BAS » (0 V)

```
OUTPUT 8    ‘ Configure le Port 8 en sortie.
LOW 8         ‘ Met le Port 8 au niveau logique BAS (0V).
```

Lorsque qu'un Port est activé par la commande HIGH, il se retrouve connecté à VDD (lorsqu'il est configuré en LOW, ce Port est connecté à VSS).

Ncd

Variable = NCD source

Variable : Variable servant à mémoriser le résultat (ni String, ni Single).
Source : valeur source (0 à 31)

La commande NCD retourne un nombre dont le bit spécifié est forcé à 1, les autres sont à 0

```
I = NCD 0       ' Résultat 00000001     1
I = NCD 1       ' Résultat 00000010     2
I = NCD 2       ' Résultat 00000100     4
I = NCD 3       ' Résultat 00001000     8
I = NCD 4       ' Résultat 00010000     16
I = NCD 5       ' Résultat 00100000     32
I = NCD 6       ' Résultat 01000000    64
I = NCD 7       ' Résultat 10000000     128
```

Nop

NOP

Cette commande n'effectue aucune action. Elle permet simplement de monopoliser 1 cycle de commande.

```
Low 8
Nop
High 8          ' Génère une impulsion de très courte durée (Environ 50 µS)
Nop
Low 8
```

On Int

ON INTx GOSUB Etiquette

x : 0 à 3, Canal d'interruption externe

La commande ON INT doit être utilisée afin d'activer la prise en compte des interruptions externes. Les modules CUBLOC™ disposent de 4 broches d'interruptions externes.
Les broches d'interruption externes peuvent être configurées pour détecter des fronts montants, des fronts descendants ou les deux.
La commande SET ONINTx doit également être utilisée afin que les entrées d'interruption soient totalement opérationnelles.
IMPORTANT : Le module CB220 ne dispose pas d'entrée d'interruption externe.

```
Dim A As Integer
On INT0 Gosub GETINT0
Set INT0 0      ' Détection d'un front descendant sur la broche du canal 0

Do
Loop

GETINT0:
A=A+1           ' Enregistre le nombre d'interruptions
Return
```

On Ladderint Gosub

ON LADDERINT GOSUB Etiquette

Si le Registre F40 est configuré en LADDER et que la commande ON LADDERINT GOSUB est utilisée alors le CUBLOC effectuera un saut à l'étiquette spécifiée dans la commande Cette fonction est utilisée lorsque le LADDER a besoin d'exécuter une séquence de programme BASIC.
Utilisez les commandes SETOUT et DIFU pour écrire 1 dans le registre F40. Lorsque l'interruption en BASIC est terminée, le registre F40 pourra être initialisé en y écrivant un 0.
Pendant la durée de l'interruption, le fait d'écrire un 1 dans le registre F40 n'aura pas pour conséquence de générer une autre interruption.
Si le registre F40 est remis à zéro depuis le BASIC, ceci aura pour conséquence de réinitialiser l'interruption afin qu'une autre interruption puisse être prise en compte.

On Recvx

ON RECV0 GOSUB Etiquette
ON RECV1 GOSUB Etiquette
ON RECV2 GOSUB Etiquette
ON RECV3 GOSUB Etiquette

Lorsque des données arrivent sur le port RS232 du canal (0 à 3 suivant le type de module CUBLOC™ utilisé) et que la commande ON RECVx a été réalisée, le programme continuera automatiquement son exécution à l'étiquette spécifiée.

```
Dim A(5) As Byte
Opencom 1,19200, 0, 100, 50

On Recv1 gosub DATARECV_RTN    ' Saute à DATARECV_RTN lorsque des données
                               ' arrivent sur le port RS232 du canal 1
Do                             ' Boucle sans fin
Loop

DATARECV_RTN:
Set Onglobal off               ' interdire les interruptions
If Blen(1,0) > 4 Then
A(0) = Get(1,1)                ' Lecture d'un octet.
A(1) = Get(1,1)
A(2) = Get(1,1)
A(3) = Get(1,1)
A(4) = Get(1,1)
End If
Set Onglobal on                ' autoriser les interruptions
Return                         ' Fin de la routine d'interruption
```

IMPORTANT
Lorsque l'interruption générée par ON RECVx est en train d'être exécutée, il ne sera pas possible de lancer une nouvelle interruption ON RECVx. Après la fin la routine d'interruption, le CUBLOC™ pourra retourner à la routine d'interruption ON RECVx si des données sont de nouveau présentes dans le buffer de réception.

On Timer

ON TIMER(intervalle) GOSUB étiquette

Intervalle : durée entre deux interruptions 1 = 10 ms, 2 = 20 ms……

Peut prendre les valeurs de 1 à 65535
La commande ON TIMER permet de générer des interruptions à intervalles réguliers afin de réaliser cycliquement une sous-routine. Il vous faut, pour ce faire, indiquer dans la commande, le nom de la sous-routine à exécuter et l'intervalle de temps entre chaque interruptions.

```
On TIMER(100) Gosub TIMERTN
Dim I As Integer
I = 0

Do
Loop

TIMERTN:
Incr I                  ' La variable I est incrémentée toutes les 1 secondes.
Return
```

IMPORTANT
Prenez impérativement garde à ce que la durée d'exécution de la routine d'interruption soit plus courte que l'intervalle entre 2 interruptions sinon un dysfonctionnement surviendra dans votre programme.

Opencom

OPENCOM canal, Debit, protocole, recvsize, sendsize

canal : canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)
Debit : Debit
protocole : Protocole
recvsize : Taille du buffer de réception (Max. 1024)
sendsize : Taille du buffer d'émission (Max. 1024)

Avant toute communication via la liaison série RS232 du module CUBLOC™, il sera nécessaire d'exécuter la commande OPENCOM.
Les modules CUBLOC™ disposent de 2 ou 4 canaux de communication RS232 (suivant les modèles). Le canal 0 est réservé pour le téléchargement/monioring/débug (toutefois l'utilisateur pourra également utiliser ce dernier si le débug et le monitoring ne sont pas nécessaires).
Vous trouverez ci-dessous les différents débits acceptés par les modules CUBLOC™: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200, 230400 Pour le paramètre « protocole », vous pouvez vous référer au tableau ci-dessous:

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
			Parité		Bit de stop	Nombre de bits transmis	
			0	0 = aucune	0 = 1 bit	0	0 = 5 bits
			0	1 = réservé	1 = 2 bits	0	1 = 6 bits
			1	0 = paire		1	0 = 7 bits
			1	1 = impaire		1	1 = 8 bits

Le tableau ci-dessous montre les configurations les plus utilisées et les valeurs associées du paramètre « protocole » calculé par rapport au tableau ci-dessus.

Bits	Parité	Bit de Stop	Valeur du paramètre « protocole »
8	AUCUNE	1	3
8	PAIR	1	19 (Hex = 13)
8	IMPAIR	1	27 (Hex = 1B)
7	AUCUNE	1	2
7	PAIR	1	18 (Hex = 12)
7	IMPAIR	1	26 (Hex = 1A)

OPENCOM 1, 19200, 3, 30, 20 ‘ Configure communication en 8-N-1
Vous pouvez configurer la taille du buffer de réception et d’émission. Les buffers de réception et d’émission sont en mémoire RAM et réduisent la place disponible pour les variables.
Bien que la taille max. de chaque buffer puisse aller jusqu’à 1024 octets, nous vous suggérons, pour les applications courantes, de donner une taille de 30 à 100 octets au buffer de réception et de 30 à 50 octets au buffer d’émission.

Pour le CB220, les ports 1 et 2 peuvent être utilisés pour le canal RS232 N° 0 (niveaux +/-12V). Les ports 10 et 11 peuvent être utilisés pour le canal RS232 N° 1 (niveaux TTL +5 / 0V).
Note : Vous pouvez utiliser la commande Set RS232 pour modifier le débit et les paramètres d’un port de communication série durant l’exécution de votre programme.

Out

OUT Port, Valeur

Port : N° du Port du module CUBLOC (1 à 255)
Valeur : Valeur à appliquer sur la broche de sortie (1 ou 0)

La commande OUT applique un niveau logique sur un Port du module CUBLOC™.
Lorsque vous exécutez cette instruction, le module CUBLOC™ configure automatiquement le Port en sortie avant de lui attribuer le niveau désiré. Il ne sera donc pas nécessaire de définir au préalable le rôle du Port avec OUTPUT.

```
OUT 8,1      ‘ Applique un niveau logique HAUT sur le Port 8.
OUT 8,0      ‘ Applique un niveau logique BAS sur le Port 8.
```

Output

OUTPUT Port

Port : N° du Port du module CUBLOC (1 à 255)

Configure le Port du module CUBLOC™ en sortie. Rappelons que tous les Ports du module CUBLOC™ sont configurés en mode haute impédance à la mise sous tension.

```
OUTPUT 8    ‘ Configure le Port 8 en sortie.
```

Vous pouvez également (et de préférence) utiliser les commandes HIGH et LOW pour définir un Port en sortie.

```
LOW 8       ‘ Place un niveau logique BAS sur le Port 8.
```

En effet, lorsque vous utilisez la commande OUTPUT, le Port est configuré pour fonctionner comme une sortie, sans pour autant avoir sélectionné de façon clairement définie le niveau HAUT ou BAS à appliquer sur cette sortie. Il est donc impératif de définir ce niveau via les commandes HIGH ou LOW si vous utilisez la commande OUTPUT.

Outstat( )

Variable = OUTSTAT(Port)

Variable : Variable servant à mémoriser le résultat (non String, ni Single).
Port : N° du Port du module CUBLOC (1 à 255)

Permet de connaître le niveau logique présent sur un Port du module CUBLOC™. Cette commande est différente de la commande IN(), elle retourne l'état d'un port de sortie, et non pas la valeur présente en entrée.

```
DIM A AS BYTE
A = OUTSTAT(0)      ' Lit l'état du Port 0 et sauvegarde la valeur dans A.
```

Pause

PAUSE valeur

Commande équivalente à la commande DELAY

Pulsout

PULSOUT Port, Periode

Port : Port de sortie (0 à 255)
Periode : Durée de l'impulsion (1 à 65535)

Génère une impulsion.
Pour produire une impulsion HAUTE, la sortie doit au préalable être placée au niveau logique BAS.
Pour produire une impulsion BASSE, la sortie doit au préalable être placée au niveau logique HAUT.
Si le paramètre « Periode » est configuré avec la valeur 10, vous génèrerez une impulsion de l'ordre de 2.5 ms. De la même façon, si le paramètre « Periode » est configuré avec la valeur 100, vous génèrerez une impulsion de 23 ms
.

```
sub pulsout(pt as byte, ln as word)
dim dl1 as integer
reverse pt for dl1 = 0 to ln
next
reverse pt
end sub
```

Put

PUT canal, data, NbOctet

canal : canal RS232 (0 à 3 selon module CUBLOC™ utilisé)
data : Données à envoyer (type Long ou inférieur)
NbOctet : Nombre de données à envoyer (1 à 4)

Cette commande envoie des données sur le port RS232 spécifié. Les données peuvent être de type « variable » ou « constante ». Pour envoyer une chaîne de caractères, il faut utiliser la commande PUTSTR.

```
OPENCOM 1,19200,0,50,10
DIM A AS BYTE
A = &HA0
PUT 1,A,1      ' Envoie &HA0 (0xA0) sur le canal RS232 N° 1.
```

IMPORTANT
La commande OPENCOM doit être utilisée au préalable
Le module CUBLOC™ va en premier lieu stocker les données dans son buffer pour ensuite les envoyer jusqu'à ce que le buffer soit vide.
Il faut s'assurer que le buffer d'émission soit vide avant d'utiliser la commande PUT.
Il est donc important d'utiliser conjointement la commande BFREE afin de connaître l'état du buffer afin d'éviter une perte de données.

```
IF BFREE(1,1) > 2 THEN ' Si le buffer dispose d'au moins 2 octets de libre
PUT 1,A,2
END IF
```

La commande BFREE() retourne le nombre d'octets de libre dans le buffer d'émission.

Astuce :
Après avoir utilisé les commandes PUT ou PUTSTR, on peut utiliser la fonction SYS(0) pour vérifier que les données ont été enregistrées dans le buffer d'émission.

```
OPENCOM 1,19200,0,50,10
PUTSTR 1,"COMFILE"
DEBUG DEC SYS(0)  ' Si l'écran de DEBUG affiche 7 c'est que toutes les données
                  ' ont été envoyées dans le buffer d'émission.
```

Puta

PUTA canal, ArrayName, NbOctet

canal : canal RS232 (0 à 3 selon module CUBLOC™ utilisé)
ArrayName : Nom Tableau
NbOctet : Nombre d'octets à envoyer (1 à 65535)

La commande PUTA peut être utilisée pour envoyer les données présentes dans un tableau Pour ce faire, il vous suffit d'indiquer le nom du tableau et le nombre d'octets à envoyer. Les données du tableau seront envoyées à partir du premier élément.

```
Dim A(10) As Byte
Opencom 1,19200,0,50,10
Puta 1,A,10             ' Envoi de 10 éléments du tableau A
```

IMPORTANT:
Si vous essayez d'envoyer plus de données que le tableau n'en contient, le module CUBLOC™ enverra des données aléatoires

Puta2

PUTA2 canal, ArrayName, NbOctet, CarStop

canal : canal RS232 (0 à 3 selon module CUBLOC™ utilisé)
ArrayName : Nom Tableau
NbOctet : Nombre de données à envoyer (1 à 65535)
CarStop : caractère ASCII d'arrêt CarStop = 0x1A pour (ctrl Z)

La commande PUTA2 s'utilise de la même façon que la commande PUTA mais la transmission des données sera stoppée après la survenue du caractère ASCII CarStop (ce caractère sera le dernier envoyé).

Putstr

PUTSTR canal, data…

canal : canal RS232 (0 à 3 selon module CUBLOC™ utilisé)
Data : Chaîne de données (variable ou constante)

Envoie une chaîne de données sur le canal RS232.

```
OPENCOM 1,19200,0,50,10
PUTSTR 1,"COMFILE TECHNOLOGY", DEC I, CR
```

Fonctionne de façon similaire à la commande PUT, en envoyant une chaîne de caractères dans le buffer d'émission après quoi l'interpréteur du module CUBLOC™ prendra soin d'envoyer ces données sur la liaison série. Comme précédément, prenez soin de ne pas dépasser la capacité du buffer d'émission sans quoi vous perdrez des données…

Pwm

PWM Canal, Duty, Periode

Canal : N° du canal PWM (0 à 15)
Duty : Durée niveau haut, doit être inférieur au paramètre
Periode Période -> valeur maximale 65535

Cette commande génére des signaux PWM. Attention, le paramètre Canal
correspond au N° du canal PWM et non pas aux N° des ports d'« E/S ». Pour les modules CUBLOC™ CB280, les broches 5, 6 et 7 sont utilisées respectivement pour les signaux PWM 0, 1 et 2. Avant d'utiliser les signaux PWM assurez-vous que les ports aient bien été configurés en SORTIE.
En fonction de la valeur du paramètre Periode, un signal PWM avec une résolution max. de 16 bits pourra être créé.
Lorsque Periode est configurée à 1024, la résolution du signal PWM sera de 10 bits. Lorsque Periode est configurée à 65535, la résolution du signal PWM sera de 16 bits.
La valeur du paramètre Duty devra être être inférieure à la valeur du paramètre Periode. Si la valeur du paramètre Duty est à 50 % de la valeur de du paramètre Periode, vous obtiendrez un signal carré.
Les signaux PWM sont gérés automatiquement par le module CUBLOC™ (il vous suffit d'exécuter la commande PWM pour que le signal soit généré en permanence par le CUBLOC™, jusqu'à ce que la commande PWMOFF soit utilisée).

```
LOW 5      ' Configure le port 5 en sortie avec un niveau logique BAS.
PWM 0,200,1024      ' Génère un signal PWM de résolution 10 bits avec un niveau haut
                    ' de 200 et une durée totale de 1024
```

IMPORTANT:
Les signaux PWM 0, 1 et 2 doivent avoir la même valeur pour le paramètre Periode car ils utilisent les mêmes ressources. Les paramètres duty peuvent être différents.
Les signaux PWM 3, 4 et 5 doivent avoir la même valeur pour le paramètre Periode car ils utilisent les mêmes ressources. Les paramètres duty peuvent être différents.

Pwmoff

PWMOFF Canal

Canal : Canal PWM (0 à 15)
Stoppe le signal de sortie PWM du canal spécifié.
Vous trouverez ci-dessous le repérage des canaux PWM en fonction des différents modules CUBLOC™:

Pour le module CB220, 3 canaux PWM sont disponibles via les broches P5, P6 et P7.

Le tableau ci-dessous donne la correspondance des Ports avec les signaux PWM.

Canal PWM	CB220	CB280	CB290	CT17X0	CB405
PWM0	PORT 5	PORT 5	PORT 5	PORT 8	PORT 5
PWM1	PORT 6	PORT 6	PORT 6	PORT 9	PORT 6
PWM2	PORT 7	PORT 7	PORT 7	PORT 10	PORT 7
PWM3		PORT 19	PORT 89	PORT 11	PORT 27
PWM4		PORT 20	PORT 90	PORT 12	PORT 28
PWM5		PORT 21	PORT 91	PORT 13	PORT 29
PWM6					PORT 11
PWM7					PORT 12
PWM8					PORT 13
PWM9					PORT 51
PWM10					PORT 52
PWM11					PORT 53

Ramclear

RAMCLEAR

Efface la mémoire RAM reservée au BASIC des modules CUBLOC™. Cette opération est nécessaire car à la mise sous tension du module CUBLOC™, la RAM peut contenir des valeurs quelconques.
* Certains modules CUBLOC peuvent avoir une sauvegarde de leur RAM par une pile externe. Si vous ne réalisez pas la commande RAMCLEAR, la pile pourra sauvegarder des valeurs non souhaitées présentes dans la RAM lors de la précédente mise sous tension.
Reverse
REVERSE Port
Port : N° du Port (0 à 255)
Inverse le niveau logique présent sur un Port (lequel devra avoir été au préalablement défini comme une sortie). Un Port à l'état logique HAUT prendra un niveau logique BAS et un Port au niveau logique BAS, prendra un niveau logique HAUT.

```
OUTPUT 8    ' Configure le Port 8 en sortie.
LOW 8       ' Place le Port 8 au niveau logique BAS.
REVERSE 8   ' Inverse le niveau logique du Port 8 -> Passe à HAUT.
```

Rnd( )

Variable = RND(0)

La commande RND() retourne un nombre pseudo-aléatoire. Ce nombre (compris entre 0 et 65535) sera stocké dans la variable. Le nombre à l'intérieur des parenthèses de la commande RND () n'a pas d'importance.

```
DIM A AS INTEGER A = RND(0)
```

La génération des nombres est pseudo-aléatoire parce que la même séquence de nombres se répétera à chaque mise sous tension (d'où le nom de pseudo-aléatoire).

Select...Case

Select Case Variable

Structure de choix multiple.
Le traitement exécuté dépend de la valeur de Variable.

```
Dim A as integer
Select Case A
    case 1              'si la variable A vaut 1 effectuer le traitement 1
        traitement 1
    case 2, 3, 4        'si la variable A vaut 2 ou 3 ou 4 effectuer le traitement 2
        traitement 2
    case else           'dans tous les autres cas, exécuter le traitement 3
        traitement 3
end select
```

```
Dim A as integer
Select Case A
    case is < 3         'si la variable A < 3 effectuer le traitement 1
        traitement 1
    case is < 10        'si la variable 3 ≤  A < 10  effectuer le traitement 2
        traitement 2
    case else           'dans tous les autres cas, exécuter le traitement 3
        traitement 3
end select
```

Set Debug

SET DEBUG on [/Off]

Vous pouvez utiliser cette commande pour « activer / désactiver » la fenêtre DEBUG du PC du BASIC. Cette commande pourra ainsi être utilisée pour désactiver le fonctionnement des commandes DEBUG de tout votre programme, sans avoir besoin de les retirer de votre listing (les commandes sont simplement ignorées lors de la compilation).

Set i2c

SET I2C DataPort, ClockPort

DataPort : SDA, Data Send/Receive Port (0 à 255)
ClockPort : SCL, Clock Send/Receive Port (0 à 255)

Cette commande permet d'attribuer un Port du module CUBLOC™ aux signaux I2C™ SDA et SCL. Une fois exécutée, les 2 Ports désignés seront automatiquement configurés en Sortie (haute impédance).
Seuls les Ports d' « E/S » peuvent être utilisés pour une communication I2C™. De plus, il vous faudra utiliser des résistances de tirage externes de 4.7 K (à relier au + 5 Vcc comme indiqué ci-dessous).

Attention certains Ports des modules CUBLOC™ ne peuvent être utilisés soit qu'en entrée, soit qu'en sortie (seuls les Ports utilisables en entrées/sorties doivent être choisis pour les communications I2C™).

Set Int

SET INTx mode

x : 0 à 3, Canal Interruption Externe
mode : 0 = Front descendant, 1 = Front montant, 2 = Changement d'état

Cette commande doit être utilisée avec la commande On Int afin que le module CUBLOC™ puisse générer des interruptions via des signaux externes. Il est possible de configurer le module afin qu'il réagisse à des fronts montants, des fronts descendants ou des changements d'états.

```
SET INT0 0    ‘ Configure pour génération d'interruption sur front descendant.
```

Set Ladder

SET LADDER On[/Off]

Active le fonctionnement du programme LADDER. Cette commande doit être exécutée afin de rendre le programme LADDER opérationnel (par défaut Set Ladder est « OFF »). L'exemple de programme ci-dessous donne le code BASIC minimal à saisir pour utiliser une programmation en Ladder.

```
Const Device = CB280          ' Déclaration du type de CUBLOC utilisé

Usepin 0,In,START    ' Déclaration des Ports
Usepin 1,In,RESETKEY
Usepin 2,In,BKEY
Usepin 3,Out,MOTOR

Alias M0=RELAYSTATE       ' Alias
Alias M1=MAINSTATE
Set Ladder On          ' Lance le Ladder
Do
Loop  ' Boucle sans fin en BASIC
```

Set modbus

Voir la documentation officielle. §

Set Onglobal

SET ONGLOBAL on[/off]

A la mise sous tension du CUBLOC™, le paramètre Set Onglobal est ON par défaut.
Cette commande active (ON) ou désactive (OFF) la prise en compte de toutes les interruptions.
Lorsque Onglobal est configuré en Off, puis en On, toutes configurations d'interruptions modifiées avant la mise sur Off seront affectées

```
SET ONGLOBAL OFF          ‘ Désactive TOUTES les interruptions.
```

Si vous n'utilisez aucune interruption sur votre module CUBLOC™, pensez à les désactiver toutes afin de disposer d'une vitesse d'exécution accrue de votre module CUBLOC™.

Set Onint

SET ONINTx on[/off]

Cette commande active (ON) ou désactive (OFF) individuellement la possibilité de prendre en compte les interruptions externes. Le n° de l'interruption doit être indiqué dans la commande. Par exemple ONINT1 est utilisé pour l'interruption 1.
Lorsque la commande Set Onintx est positionnée sur « ON » pour une interruption spécifique, alors une interruption pourra être générée en utilisant la commande associée ON INTx. Si la commande Set Onintx est positionnée en « OFF » alors la commande ON INTx ne générera pas d'interruption. Voir aussi la commande SET INTx qui vous permettra de configurer le mode de déclenchement des interruptions.

SET ONINT0 ON
SET ONINT1 ON
SET ONINT1 OFF
SET ONINT2 OFF
SET ONINT3 ON
SET ONINT3 OFF

Set OnLadderint

SET ONLADDERINT on[/off]

A la mise sous tension du CUBLOC™, le paramètre Set OnLadderint est ON par défaut.
Cette commande active (ON) ou désactive (OFF) la prise en compte des interruptions générées via le Ladder. Lorsque la commande Set OnLadderint est positionnée en « ON » une interruption pourra être générée par la commande On ladderint. Si la commande Set OnLadderint est positionnée en « OFF » alors la commande On ladderint ne générera pas d'interruption.

Voir aussi la commande On ladderint qui vous permettra de configurer le mode de déclenchement des interruptions.

Set Onpad

SET ONPAD on[/off]

A la mise sous tension du CUBLOC™, le paramètre Set Onpad est ON par défaut.
Cette commande active (ON) ou désactive (OFF) les interruptions générées via la commande On Pad. Lorsque la commande Set Onpad est positionnée en « ON » alors une interruption pourra être générée par la commande On Pad. Si la commande Set Onpad est positionnée en « OFF » alors la commande On Pad ne générera pas d'interruption.
Voir aussi la commande Set Pad qui vous permettra de configurer le mode de déclenchement des interruptions.

Set Onrecv

SET ONRECV0 on[/off]
SET ONRECV1 on[/off]
SET ONRECV2 on[/off]
SET ONRECV3 on[/off]

A la mise sous tension du CUBLOC™, le paramètre Set Onrecv est ON par défaut.
Cette commande active (ON) ou désactive (OFF) les interruptions générées si des données sont réceptionnées dans le buffer de réception des ports séries. Lorsque la commande Set On Recv est positionnée en « ON » alors une interruption pourra être générée par la commande On Recv. Si la commande Set OnRecv est positionnée en « OFF » alors la commande On Recv ne générera pas d'interruption. Voir aussi la commande On Recv pour le paramétrage de la génération des interruptions.

Set Ontimer

SET ONTIMER on[/off]

A la mise sous tension du CUBLOC™, le paramètre de Set Ontimer est ON par défaut.
Cette commande active (ON) ou désactive (OFF) les interruptions temporelles générées via la commande On Timer(). Lorsque la commande Set Ontimer est positionnée en « ON » alors une interruption pourra être générée par la commande On Timer(). Si la commande Set Ontimer est positionnée en « OFF » alors la commande On Timer() ne générera pas d'interruption. Voir aussi la commande On Timer pour le paramétrage de la génération des interruptions.

Set Outonly

SET Outonly on[/off]

Les ports de sorties des modules CUBLOC « CB290 » (Rev. B) et CUTOUCH « CT1720 » sont à l'état haute impédance à la mise sous tension afin d'éviter que ces derniers ne présentent des niveaux de sortie incertains. Vous devrez utiliser la commande « SET OUTONLY ON » pour que ces ports soient utilisables en sorties.
Const device = cb290 Set outonly on
Low 24

Set Pad

SET PAD mode, packet, NbDonnee

mode : Bit mode (0 à 255)
packet : Longueur du packet (1 à 255)
NbDonnee : Taille du buffer de réception (1 à 255)

Les CUBLOC™ peuvent piloter un module optionnel de gestion de clavier.
Voir la documentation officielle. §

Set RS232

Set RS232 canal, Debit, protocole

canal : canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé) Debit : Debit
protocole : Protocole
La commande Opencom est utilisée pour ouvrir et initialiser une communication série sur le module CUBLOC™. Afin de modifier la vitesse et le protocole de communication au cours de l'exécution de votre programme, vous devrez utiliser la commande SET RS232.
Vous trouverez ci-dessous les différents débits acceptés par les modules CUBLOC™: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200, 230400 Pour le paramètre « protocole », vous pouvez vous référer au tableau ci-dessous:

Bit 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
			Parité		Bit de stop	Nombre de bits transmis	
			0	0 = aucune	0 = 1 bit	0	0 = 5 bits
			0	1 = réservé	1 = 2 bits	0	1 = 6 bits
			1	0 = paire		1	0 = 7 bits
			1	1 = impaire		1	1 = 8 bits

Le tableau ci-dessous montre les configurations les plus utilisées et les valeurs associées du paramètre « protocole » calculé par rapport au tableau ci-dessus.

Bits	Parité	Bit de Stop	Valeur du paramètre « protocole »
8	AUCUNE	1	3
8	PAIR	1	19 (Hex = 13)
8	IMPAIR	1	27 (Hex = 1B)
7	AUCUNE	1	2
7	PAIR	1	18 (Hex = 12)
7	IMPAIR	1	26 (Hex = 1A)

```
Opencom 1, 19200, 3 , 30, 20     ' Ouvre communication RS232 sur le canal 1
Set Rs232 1, 115200, 19          ' Modification du débit et de la parité
```

Set RS485

Set RS232 canal, NoPort

canal : canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)
NoPort : n° d'activation du Port de transmission

La commande RS485 vous permettra d'établir un réseau de communication afin de relier plusieurs modules CUBLOC™ entre eux sur de longues distances (jusqu'à 1 Km !). Cette communication impose qu'il y n'y ait qu'un seul maître et que le reste des périphériques soient des esclaves.
Voir la documentation officielle. §

Set Until

SET UNTIL canal, packetlength, Stopchar

canal : Canal RS232 (0 à 3 suivant modèle de CUBLOC™ utilisé)
packetlength : Longueur de la trame (0 à 255)
Stopchar : Caractère de fin

Cette commande peut être utilisée après la commande ON RECV. Alors que la commande ON RECV générera une interruption même si un seul octet est reçu par le port série, la commande Set Until permet de préciser les conditions de déclenchement de l'interruption.
Le sous-programme d'interruption sera lancé lorsque le caractère reçu correspondra à celui du paramètre « Stopchar » ou que le nombre d'octets reçus sera supérieur à la valeur du paramètre « packetlength »,

Le paramètre « packetlength » est nécessaire (dans le cas où le caractère attendu n'arrive jamais).
Cette commande doit être utilisée avec ON RECV.

Voir l'exemple ci-dessous:

```
Dim A(5) As Byte
Opencom 1,19200,0, 100, 50
On Recv1 DATARECV_RTN
Set Until 1,99,"S"
```

Comme vous pouvez le voir, on prévoit une longeur max. de données pouvant arriver de 99 octets. En d'autres termes, si le caractère "S" n'est pas reçus dans les 99 octets qui arrivent, l'interruption interviendra.

```
SET UNTIL 1,5
```

Vous pouvez également uniquement indiquer une taille de données à recevoir (sans vérification sur le caractère reçu).

Shiftin( )

Variable = SHIFTIN(clock, data, mode, bitlength)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single)
Clock : Port du Signal d'horloge (0 à 255) Data : Port du Signal de données (0 à 255)
Mode : 0 = LSB en premier (Least Significant Bit First), après front montant
1 = MSB en premier (Most Significant Bit First), après front montant
2 = LSB en premier (Least Significant Bit First), après front descendant
3 = MSB en premier (Most Significant Bit First), après front descendant
4 = LSB en premier (Least Significant Bit First), avant front montant
5 = MSB en premier (Most Significant Bit First), avant front montant
bitlength : Nombre de bits (1 à 16)

La commande SHIFTIN() permet la réception de données via une communication 2 fils (CLOCK et DATA).
Les commandes SHIFTIN et SHIFTOUT peuvent être utilisées pour communiquer avec des composants via un bus SPI™, Microwire™ ou des protocoles similaires. Il vous sera ainsi possible de piloter des composants SPI™ externes tels que des EEPROM, des convertisseurs « A/N » ou « N/A ».

```
DIM A AS BYTE
A = SHIFTIN(3,4,0,8)          ' Le port 3 est l'horloge, le port 4 correspond aux données,
                              ' Mode 0 (on travaille avec une réception sur 8 bits).
```

Shiftout

SHIFTOUT clock, data, mode, variable, bitlength

Clock : Port du Signal d'horloge (0 à 255) Data : Port du Signal de données (0 à 255)
Mode : 0 = LSB en premier (Least Significant Bit First)
1 = MSB en premier (Most Significant Bit First)
2 = MSB en premier (Most Significant Bit First) , Création ACK (Pour I2C™)
variable : Variable servant à mémoriser les données à envoyer (jusqu'à 65535)
bitlength : Nombre de bits (1 à 16)
La commande SHIFTOUT) permet l'envoi de données via une communication 2 fils (CLOCK et DATA). Il est possible d'utiliser 3 modes différents. Le mode 2 est utilisable pour les communication de type I2C™ (pour ce mode, un signal acquitement sera nécessaire (ACK) après chaque envoi de 8 bits).

```
SHIFTOUT 3,4,0,&H55,8       ' Port 3 = Horloge, Port 4 = Données
                            ' Mode = 0, on envoie 0x55
                            ' bitlength = envoi de 8 bits,
```

Voir la documentation sur la liaison I²C $

Steppulse

STEPPULSE canal, Port, Freq, Qt

Canal : Canal StepPulse (0 ou 1)
Port : Port du Cubloc
Freq : Fréquence de sortie (jusqu'à 15 KHz)
Qt : Nombre d'impulsions à générer (jusqu'à 2147483647)
Cette commande génère un nombre d'impulsions à une fréquence donnée (jusqu'à 15 KHz). Bien que les commandes FREQOUT et PWM soient également destinées à générer des impulsions, il n'est pas possible avec ces dernières de sélectionner un nombre d'impulsions défini. De plus ces commandes monopolisent les ports PWM. Pour sa part, la commande

Steppulse vous permettra non seulement de travailler avec la plupart des Ports du CUBLOC™, mais aussi de sélectionner la fréquence de sortie et le nombre d'impulsions à générer.
En fonction du modèle de CUBLOC™ utilisé, le nombre de canaux peut varier.

La commande STEPPULSE utilise les compteurs PWM du CUBLOC™ pour fonctionner, de telle sorte qu'il ne sera plus possible d'exploiter les signaux PWM3, PWM4 et PWM5 lorsque vous utilisez cette commande.
Pour le CB405, lorsque vous exploitez le canal 1, se sont les signaux PWM6, PWM7 et PWM8 qui ne pourront plus être utilisés. Avec la série des CUBLOC CB2xx, seul le canal 0 pourra être utilisé. Avec le CB405, vous avez 2 canaux simultanés à votre disposition.
Vous pouvez utiliser n'importe quel Port d'E/S du CUBLOC™. Lorsque vous exécutez la commande STEPPULSE, ce Port sera automatiquement configuré en sortie et le restera même après la fin des impulsions impulsions.
La fréquence de sortie est réglable de 1 Hz à 15 KHz et le nombre d'impulsions de 1 à 2147483647. Les impulsions sont générées en tâche de fond et n'empêchent pas le CUBLOC™ de traiter d'autres actions en même temps.

Stepstop

STEPSTOP canal

Canal : Canal StepPulse (0 ou 1)
Stoppe immédiatement la génération des impulsions sur le canal sélectionné.

Stepstat()

Variable = STEPSTAT (canal)

Variable : Variable servant à mémoriser le résultat
Canal : Canal StepPulse (0 ou 1)

Vous permet de connaître le nombre d'impulsions qui ont été générées depuis l'exécution de la dernière commande SETPPULSE. STEPSTAT retourne le double du nombre d'impulsions restant à produire. Si par exemple il reste 500 impulsions à générer, STEPSTAT vous retournera la valeur 1000.

Voir la documentation officielle. §

Sys( )

Variable = SYS(adresse)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single) adresse : Adresse (0 à 255)
Utilisez la commande Sys() pour lire l'état des buffers RS232 (canal 0 et 1).
Adresse = 0 : Nombre d'octets envoyés dans le buffer d'émission après l'exécution des commandes PUT ou PUTSTR.
Addresse = 1 : Nombre d'octets envoyés dans le buffer de réception après l'exécution des commandes GET ou GETSTR.
Adresse = 5 : Valeur d'un Timer dont la valeur s'incrémente toutes les 10 ms.
SYS(5) retournera la valeur du timer système qui s'incrémentera toutes le 10 ms jusqu'à la valeur 65535 puis reviendra à 0 (idéal pour des applications nécessitants l'utilisation d'unTimer supplémentaire.
Adresse = 6 : Adresse mémoire de donnée (RAM) = pointeur de pile.

SYS(6) retournera la valeur courante de l'adresse de la mémoire de donnée. A la mise sous tension l'adresse de la mémoire de donnée est à 0. Après l'appel d' une sous-routine ou d'une fonction, l'adresse mémoire sera incrémentée. Si l'adresse mémoire de la RAM excède la capacité du module CUBLOC™, vous risquez de réaliser une action de dépassement pouvant créer un dysfonctionnement du module CUBLOC™. La commande SYS(6) vous aidera à éviter cette situation. Par exemple le module CB280 dispose de 1948 octets de mémoire RAM. Prenez garde de conserver une marge d'au moins 100 octets de libre pour une plus grande sécurité.

```
A = SYS(6)    ' Stocke l'adresse courante de la mémoire RAM dans la variable A
```

Tadin()

Variable = TADIN(canal)

Variable : Variable servant à mémoriser le résultat (ni String, ni Single)
canal : n° canal de conversion « A/N » (Pas le N° de Port)

La commande Tadin() est similaire à Adin() mais elle retourne la valeur moyenne de 10 mesures. Si vous travaillez en environnement perturbé, cette commande vous apportera une plus grande précision si le signal d'entrée évolue lentement.

Le programme ci-dessous reproduit le fonctionnement de la commande TADIN à l'aide de la commande ADIN.

```
function tadin(num as byte) as integer dim ii as integer, ta as long ta = 0
For ii = 0 To 9
ta = ta + Adin(num)
Next
TADIN = TA / 10
End Function
```

Udelay

UDELAY time

Time : Interval (1 à 65535)
Cette commande permet de produire des délais spéciaux. La commande débute avec une durée de base de 70 microsecondes. Chaque unité ajoutée au paramètre « time » provoquera l'ajout de 14 à 18 microsecondes à la valeur de base.
Par exemple, la commande Udelay 0 générera une temporisation de 70 microsecondes. Udelay 1 générera une temporisation de l'ordre de 82 à 84 microsecondes. Lorsque des interruption ou le LADDER sont activés en même temps, la durée de « Udelay » peut être affectée. De même, durant cette durée, si une interruption BASIC intervient, la durée de « Udelay » sera également affectée.
Si vous ne désirez pas que cette commande soit affectée par le LADDER ou par les interruptions, nous vous suggérons de stopper au préalable l'action du LADDER et de toutes les interruptions.

```
UDELAY 100 ' Delay d'environ 1630 microsecondes.
```

Usepin

Usepin E/S, In/Out, NomAlias

E/S: N° du port d' « E/S » (0 à 255) In/Out : Entrée ou Sortie
NomAlias : Alias pour le port (optionnel)

La commande Usepin déclare les ports utilisés par un programme LADDER on trouve : - le numéro du port
- sa fonction, entrée ou sortie.
- optionnellement son Alias.
Comme toute déclaration, elle doit se faire préalablement à l'utilisation des broches

```
USEPIN 0,IN,START
USEPIN 1,OUT,RELAY
USEPIN 2,IN,BKEY
USEPIN 3,OUT,MOTOR
```

Utmax

UTMAX variable

Variable : Variable à incrémenter (ni String, ni Single).
Incrémente la variable d'une unité. Lorsque la variable atteint sa valeur maximale, elle n'est plus incrémentée. Cette valeur maximale dépend du type de la variable. Dans le cas d'octets, il s'agit de la valeur 255, pour un Integer, on pourra aller jusqu'à 65535.

```
UTMAX A      ' Incrémente A d'une unité
```

Wait

WAIT time

time : Interval (variable ou constante) en ms de 10 à 2147483640
La commande Wait effectuera une temporisation (en utilisant l'horloge système). La résolution est de 10 ms.

```
Wait 10      ' tempo de 10 ms
Wait 15      ' tempo de 10 ms
Wait 110     ' tempo de 110 ms
Wait 115     ' tempo de 110 ms
```

WaitTx

WAITTX canal

canal : Canal RS232 (0 à 3 suivant le modèle de CUBLOC™ utilisé)

La commande WaitTx attend que le buffer d'émission RS232 soit vide.

En utilisant la commande WaitTx, la gestion des communications RS232 est simplifiée:

```
OPENCOM 1,19200,0, 100, 50
PUTSTR 1,"ILOVEYOU",CR
WAITTX 1                    ' Attend que toutes les données soient expédiées
```

Lorsque cette commande est active, les autres interruptions peuvent être appelées.
En d'autres termes, cette commande n'affecte pas d'autres parties du système d'exploitation du CUBLOC™.