PLC / Embedded computer

CUBLOC™

Manuel utilisateur
Version 2.5

"Everything for Embedded Control"

COMFILE TECHNOLOGY
Comfile Technology Inc.
www.comfiletech.com

Copyrights et appellations commerciales

Windows XP™ est une appellation commerciale appartenant à Microsoft Corporation.
XPORT™ est une appellation commerciale appartenant à Lantronix inc.
CUBLOC™ est une appellation commerciale appartenant à Comfile Technology Inc.

Toutes les marques, les procédés, les références et les appellations commerciales des produits cités dans ce document appartiennent à leur propriétaire et Fabricant respectif. All brand names and trademarks are the property of their respective owners - Other trademarks mentioned are registered trademarks of their respective holders.

Informations techniques

Ce manuel a été conçu avec la plus grande attention. Tous les efforts ont été mis en oeuvre pour éviter les anomalies. Toutefois, nous ne pouvons garantir que ce dernier soit à 100% exempt de toute erreur. Les informations présentes dans ce manuel sont données à titre indicatif. Les caractéristiques techniques des "CUBLOC", la nature, les possibilités et le nombre de leurs instructions, ainsi que les possibilités de leurs logiciels de programmation et les caractéristiques des modules périphériques associés aux CUBLOC peuvent changer à tout moment sans aucun préavis dans le but d'améliorer la qualité et les possibilités de ces derniers. Ces produits sont protégés par des brevets à travers le monde.

Limitation de responsabilité

En aucun cas le Fabricant et LEXTRONIC ne pourront être tenus responsables de dommages quels qu'ils soient (intégrant, mais sans limitation, les dommages pour perte de bénéfice commercial, interruption d'exploitation commerciale, perte d'informations et de données à caractère commercial ou de toute autre perte financière) provenant de l'utilisation ou de l'incapacité à pouvoir utiliser les modules "CUBLOC" et leurs logiciels associés ainsi que leurs platines et modules optionnels associés, même si le Fabricant ou LEXTRONIC ont été informés de la possibilité de tels dommages.

Les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés sont destinés à être utilisés en milieu résidentiel dans les gammes de températures limites +10 à +65 °C. Les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés ne sont pas conçus, ni destinés, ni autorisés pour être utilisés au sein d'applications militaires, ni au sein d'applications médicales, ni d'alarme anti-intrusion, ni d'alerte incendie, ni au sein d'applications pour ascenseurs ou commande de feux d'artifices, ni au sein d'applications sur machine outils ou d'applications embarquées dans des véhicules (automobiles, camions, bateaux, scooters, motos, kart, scooters des mers, avions, hélicoptères, ULM...), ni au sein d'applications embarquées sur des maquettes volantes de modèles réduits (avions, hélicoptères, planeurs...).

De même, les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés ne sont pas conçus, ni destinés, ni autorisés pour expérimenter, développer ou être intégrés au sein d'applications dans lesquelles une défaillance de ces derniers pourrait créer une situation dangereuse pouvant entraîner des pertes financières, des dégâts matériels, des blessures corporelles ou la mort de personnes ou d'animaux. Si vous utilisez les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés volontairement ou involontairement pour de telles applications non autorisées, vous vous engagez à soustraire le Fabricant et LEXTRONIC de toute responsabilité et de toute demande de dédommagement.

En cas de litige, l'entière responsabilité du Fabricant et de LEXTRONIC vis-à-vis de votre recours se limitera exclusivement selon le choix du Fabricant et de LEXTRONIC au remboursement du module "CUBLOC" et/ou de ses platines et modules optionnels associés et/ou de leur réparation et/ou de leur échange. Le Fabricant et LEXTRONIC démentent toutes autres garanties, exprimées ou implicites.

L'utilisateur des modules "CUBLOC" et de ses platines et modules optionnels associés est entièrement et seul responsable des développements logiciels (de l'écriture de son programme en langage BASIC et/ou PLC) ainsi que de l'intégration matérielle, des modifications et ajouts de périphériques qu'il effectuera sur les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés. S'agissant de matériel "OEM", Il incombera à l'utilisateur de vérifier que l'application finie complète développée avec les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés soient conformes aux normes de sécurité et aux normes CEM en vigueur.

Tous les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés sont testés avant expédition. Toute inversion de polarité, dépassement des valeurs limites des tensions d'alimentation, courts-circuits, utilisation en dehors des spécifications et limites indiquées dans ce document ou utilisation pour des applications non prévues pourront affecter la fiabilité, créer des dysfonctionnements et/ou endommager les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés sans que la responsabilité du Fabricant et de LEXTRONIC ne puisse être mise en cause, ni que les produits puissent être échangés au titre de la garantie.

Rappel sur l'évacuation des équipements électroniques usagés

Ce symbole présent sur les modules "CUBLOC" ainsi que leurs platines et modules optionnels associés et/ou leurs emballages indique que vous ne pouvez pas vous débarraser de ces produits de la même façon que vos déchets courants. Au contraire, vous êtes responsable de l'évacuation de ces produits lorsqu'ils arrivent en fin de vie (ou qu'ils sont hors d'usage) et à cet effet, vous êtes tenu de les remettre à un point de collecte agréé pour le recyclage des équipements électriques et électroniques usagés. Le tri, l'évacuation et le recyclage séparés de vos équipements usagés permet-tent de préserver les ressources naturelles et de s'assurer que ces équipements sont recyclés dans le respect de la santé humaine et de l'environnement. Pour plus d'informations sur les lieux de collecte des équipements électroniques usagés, veuillez contacter votre mairie ou votre service local de traitement des déchets.

Note for all residents of the European Union

This symbol on the product or on its packaging indicates that this product must not be disposed of with other household waste. Instead, it is your responsability to dispose of your waste equipement by handing it over to designated collection point for the recycling of waste electrical and electric equipement. The separate collection and recycling of your waste equipement at the time of disposal will help to conserve natural resources and ensure that it is recycled in a manner that protects human health and environnement. For more information about where you can drop off your waste equipement for recycling, please contact your local city office or your local hold waste disposal service.

A lire avant toute utilisation…

Le CD-ROM qui vous a été livré avec les modules CUBLOC™ contient généralement la dernière version en date du logiciel de développement « CUBLOC Studio ».

Toutefois de part les mises à jours régulières opérées par Comfile Technology, il vous faudra impérativement vérifier sur le site www.comfiletech.com (rubrique « Download ») que vous disposez bien de la dernière version en date du « CUBLOC Studio » avant d'installer et d'utiliser ce dernier lors de votre première utilisation.

Contactez-nous si vous ne disposez pas de connexion Internet afin que nous puissions vous adresser si nécessaire la dernière version à jour.

Pensez également à vous connecter régulièrement sur le site www.comfiletech.com (rubrique « Download ») pour télécharger les mises à jours.

Une fois votre version du « CUBLOC Studio » mis à jour, il vous faudra également remettre à jour le Firmware de votre module CUBLOC™ afin que ce dernier dispose des dernières possibilités de programmation (consultez la page 366 pour plus d'informations à ce sujet).

A ce titre, certaines commandes et possibilités décrites dans cette documentation suppose que vous disposiez d'une version du « CUBLOC Studio » supérieur à la version 2.0.x.

Bien que nous comprenions votre impatience à vouloir utiliser immédiatement les possibilités des modules CUBLOC™ en vous inspirant des exemples de programmes et des notes d'applications présentent sur le CD-ROM, **nous vous suggérons toutefois lors de votre première utilisation de suivre pas à pas les indications décrites dans le chapitre 11 (page 341).** Vous trouverez en effet dans ces lignes certains points techniques primordiaux à respecter.

Dans tous les cas, si vous rencontrez des problèmes lors de la mise en œuvre de vos modules CUBLOC™, nous vous suggérons de lire méticuleusement les informations du chapitre 11 (page 341) ainsi que de prendre connaissance des questions/réponses de la F.A.Q des CUBLOC™ disponibles au chapitre 12 (page 366)… Les solutions à vos problèmes sont probablement présentent dans ces pages !

Table des matières

Préface

La société Comfile Technology est réputée pour développer des modules microcontrôlés PLC et BASIC depuis 1997. De part son expérience acquise dans ce domaine, Comfile Technology est en mesure de vous proposer aujourd'hui une nouvelle gamme de produits encore plus puissants et plus flexibles, lesquels bénéficient du meilleur des possibilités des contrôleurs embarqués programmables en BASIC et en PLC.

L'expérience de Comfile Technology dans le développement et la distribution des TinyPLC et des PicBASIC (qui sont respectivement des modules microcontrôlés de type PLC et programmables en BASIC), leur a permis d'améliorer leurs techniques de conception au fil des années. Contrairement à la plupart des autres produits concurrents, vous pourrez ainsi utiliser les modules CUBLOC™ uniquement en les programmant en langage BASIC ou uniquement via une programmation PLC ou encore avec les 2 technologies à la fois.

La programmation en LADDER, permet le développement d'applications de contrôles séquentiels très complexes. La programmation en langage BASIC permet l'implémentation de processus divers et variés avec une grande simplicité de prise en main et une grande souplesse de travail.

Les CUBLOC™ sont capables de gérer à la fois un programme écrit en langage BASIC et/ou en LADDER grâce à leur cœur multitâche. De plus ces 2 technologies (BASIC et LADDER) disposent d'une mémoire commune qui vous permettra ainsi de travailler avec un type de module microcontrôlé à part entière tout à fait nouveau.

Les CUBLOC™ s'adressent à la fois aux novices, comme aux professionnels et aux automaticiens.

- Les novices trouveront avec les CUBLOC™ le composant idéal pour pouvoir réaliser très rapidement leurs premières applications en toute simplicité même si ces derniers n'ont pas de connaissance approfondie sur les microcontrôleurs.

- Les professionnels apprécieront également les possibilités et la puissance des modules CUBLOC™ qui leur permettront de mettre très rapidement leurs produits sur le marché en conservant une longueur d'avance sur leurs concurrents.

- Les automaticiens pourront trouver très rapidement leur « marque » avec les CUBLOC™ (grâce à leur programmation en LADDER) tout en ayant également la possibilité de « s'essayer » progressivement aux possibilités de programmation en langage « BASIC » si nécessaire (la réciproque étant aussi vraie).

Comfile Technology, Inc.

Chapitre 1.

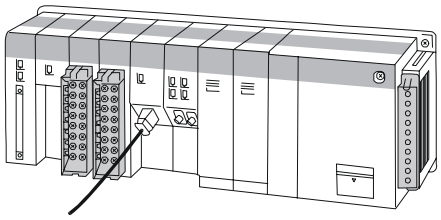
Présentation des modules CUBLOC…

Qu'est-ce qu'un module CUBLOC™ ?

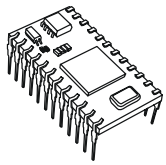
Les CUBLOC™ sont différents des automates PLC traditionnels que vous avez déjà peut être pu utiliser. Les automates traditionnels se présentent généralement sous la forme de boîtiers modulaires dotés de diverses connexions (voir représentation ci-dessous).

Les CUBLOC™ s'apparentent pour leur part à de mini automates « OEM » qui se présentent sous forme de modules hybrides, lesquels vous permettront de pouvoir les intégrer au sein de votre produit final en vous laissant ainsi une plus grande flexibilité sur la taille et les spécificités de votre application.

Les modules CUBLOC™ sont similaires aux automates traditionnels en ce sens qu'ils peuvent tout comme eux être programmés en langage LADDER. Toutefois leur petite taille vous permettra de les intégrer directement sur le circuit imprimé de votre application comme un microcontrôleur traditionnel.



traditional PLC



CUBLOC core module

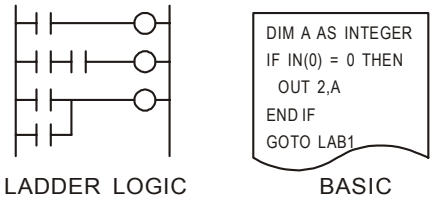
Il existe différents modules CUBLOC™, lesquels se distinguent par leur capacité mémoire, nombre d'E/S, etc… Vous trouverez ci-après un petit tableau de sélection.

Tableau comparatif

Modèle	CB220	CB280	CB290	CB405
Photo				
Mémoire programme	80 K	80 K	80 K	200 K
Mémoire de « données »	2 K (BASIC) 1 K (Ladder)	2 K (BASIC) 1 K (Ladder)	24 K (BASIC) 4 K (Ladder)	51 K (BASIC) 4 K (Ladder) 55 K (Heap)
Mémoire EEprom	4 K	4 K	4 K	4 K
Vitesse exécution	36,000/sec	36,000/sec	36,000/sec	36,000/sec
Ports d'entrées / sorties	16 entrées/sorties	49 entrées/sorties	91 répartis en 33 entrées 32 sorties 26 « E/S »	64 entrées/sorties
Ports séries Configurables de 2400 bps à 230.400 bps	2 ports séries Port 0 : RS232 Port 1 : TTL	2 ports séries Port 0 : RS232 Port 1 : TTL & RS232	2 ports séries Port 0 : RS232 Port 1 : TTL & RS232	4 ports séries Port 0 : RS232 Port 1 à 3 : TTL
Entrées de conversion « analogique / numérique » (parmis les « E/S » générales)	8 entrées Résolution 10 bits	8 entrées Résolution 10 bits	8 entrées Résolution 10 bits	16 entrées Résolution 10 bits
Sorties « PWM » Fréquence configurable 35 Hz à 1.5 MHz	3 sorties Résolution 16 bits	6 sorties Résolution 16 bits	6 sorties Résolution 16 bits	12 sorties Résolution 16 bits
Interruption externe (parmis les « E/S » générales)	-	4 canaux	4 canaux	4 canaux
Entrée de comptage rapide (parmis les « E/S » générales)	2 compteurs 32 bits	2 compteurs 32 bits	2 compteurs 32 bits	2 compteurs 32 bits
Alimentation Consommation (ports non chargés)	5 à 12 Vcc 40 mA	5 Vcc 40 mA	5 Vcc 70 mA	5 Vcc 50 mA
Horloge RTC intégrée	-	-	Oui	-
Sauvegarde RAM	-	-	En option	En option
Température fonctionnement	+10 à +65°C	+10 à +65°C	+10 à +65°C	+10 à +65°C
Présentation	DIL 24 broches 600 mil.	Module 64 broches	Module 108 broches	Module 80 broches
Dimensions (mm)	30 x 15.3 x 11	35 x 25.4 x 11	59.4 x 47.8 x 13	59.4 x 47.8 x 13

Le principal avantage des modules CUBLOC™ vis à vis des autres automates est que les CUBLOC™ peuvent compenser certaines limitations propres à la programmation en langage LADDER par une programmation additionnelle en langage BASIC (très évolué). La programmation en langage LADDER est en effet toute indiquée pour prendre en charge des actions qui s'inscrivent dans un diagramme de séquences… mais lorsqu'une application nécessite de stocker des données, d'afficher des graphiques et de réaliser d'autres tâches plus complexes les automates traditionnels ne sont alors plus adaptés. C'est la principale raison pour laquelle une programmation en langage BASIC a été ajoutée sur les modules CUBLOC™. Dès lors, vous pourrez à la fois programmer en LADDER et en BASIC !

Un des autres avantages du langage BASIC géré par les modules CUBLOC™ vis à vis d'autres modules programmables en langage BASIC vient du fait que les 2 technologies intégrées au CUBLOC™ (la programmation en BASIC et en LADDER) soient totalement indépendantes.



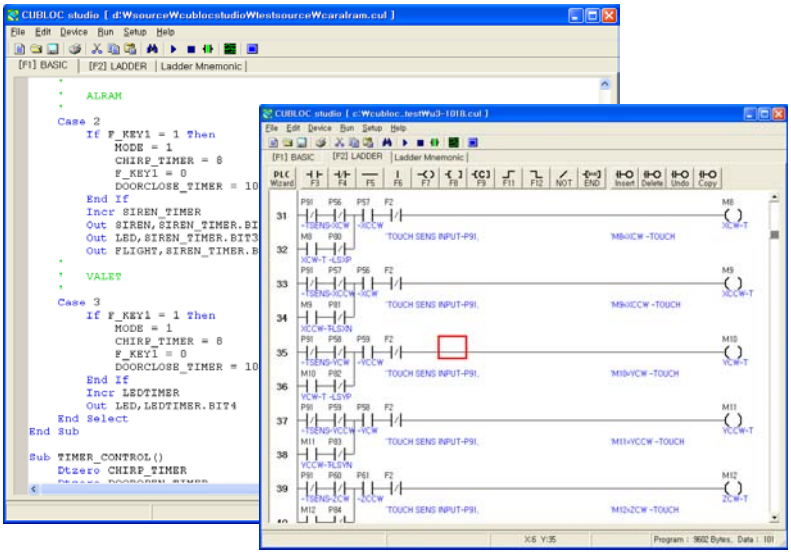
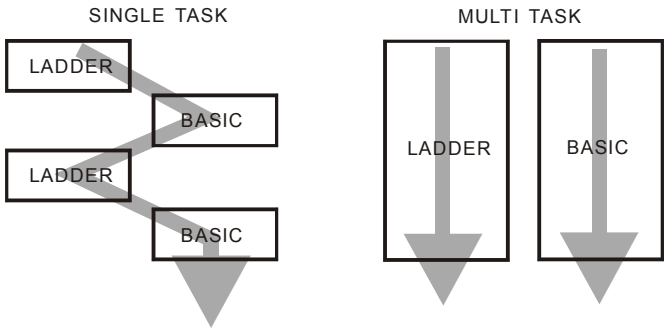


Photo de l'environnement de développement des CUBLOC™ appelé "CUBLOC Studio".

A ce jour, d'autres modules automates disponibles sur le marché sont capables de supporter une programmation en LADDER et en BASIC. Toutefois la plupart de ces derniers ne disposent pas d'une structure et d'un cœur multitâche et s'apparentent à proprement dit à des modules à structure « simple tâche ». Ce qui veut dire que les « bouts de programmes » en langage BASIC seront considérés comme des sous-parties du langage LADDER, lesquelles ne pourront pas être exécutées de façon indépendante comme vous pourrez le faire sur les modules CUBLOC™. Cette conception est à notre sens quelque part pénalisante du fait même de la « non gestion » en temps réel de la partie en langage BASIC qui pourra affecter les performances du programme en LADDER.

Les modules CUBLOC™ ont pour leur part une approche différente de part leur structure multitâche qui vous offriront une grande précision et un timing rigoureux lors de l'exécution de leur programmes. Ainsi contrairement à la plupart des autres modules programmables en langage BASIC actuellement disponibles sur le marché, les modules CUBLOC sont capables de supporté à la fois la gestion multitâche et la gestion de process en temps réel.

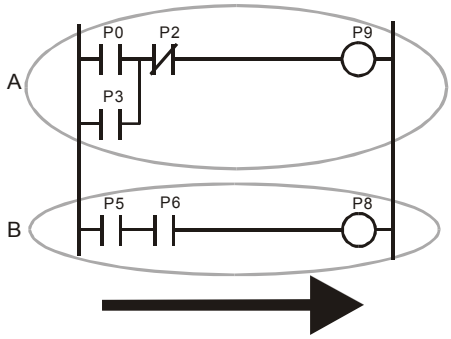
La structure multitâche des modules CUBLOC™ leur permettront ainsi de pouvoir à la fois exécuter un programme LADDER et un programme BASIC en même temps en vous permettant ainsi de pouvoir bénéficier de la précision du « timing » d'exécution du LADDER, tout en concernant les atouts d'une application programmée en langage BASIC. Dans tous les cas, vous conservez également la possibilité de pouvoir programmer toute votre application uniquement en langage LADDER ou uniquement en langage BASIC ou avec les 2 technologies à la fois



Comme vous pouvez le constater, les modules CUBLOC™ s'apparentent ainsi à de nouveaux types de contrôleurs. En vous permettant de réaliser des applications que des mini-automates traditionnels ne pourraient pas effectuer sans l'apport du langage BASIC, les CUBLOC™ ouvrent de nouvelles perspectives de développement et d'applications.

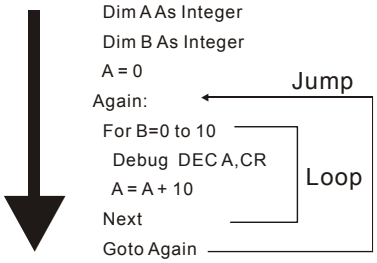
LADDER et BASIC

Le principal avantage d'une programmation en langage LADDER est que l'ensemble des circuits (de votre programme) soit scruté en "Parallèle," de telle sorte que l'exécution des processus s'effectue avec la même durée.



Comme vous le voyez ci-dessus, à la fois les circuits A et B sont en état d'attente et prêt à activer les sorties lorsqu'ils seront sollicités.

En comparaison, la gestion d'un processus en BASIC s'effectue selon un déroulement séquentiel.



Ces 2 types de langages sont utilisés depuis très longtemps dans des secteurs d'activités et pour des applications très différentes. Le LADDER est principalement exploité en automatisme par le biais d'automates programmables. Pour sa part, le langage BASIC et couramment utilisé pour la mise en œuvre d'applications diverses et variées sur compatibles PC et sur de nombreux microcontrôleurs.

Comme indiqué précédement le principal avantage de la programmation en LADDER est de permettre le traitement de l'ensemble des tâches avec la même durée. Ainsi quelque-soit la compléxité de vos « circuits », le LADDER sera toujours prêt à activer les sorties lorsque les entrées seront sollicitées. Ceci est pourquoi il est courant d'avoir recours à ce type de langage pour le pilotage de machines diverses en automatisme.

Le LADDER n'est pas un langage de programmation « traditionnel » à proprement parlé. Ce dernier est en effet d'avantage dédié à la gestion d'évènements logiques. Des lors, le recours au LADDER pour la gestion d'évènement complexes tels que la saisie d'informations issues de claviers, l'affichage sur écran LCD ou afficheurs 7 segments à Leds s'avérera très vite impossible ou très fastidieux à mettre en œuvre.

A l'inverses de telles applications sont très simples à prendre en charge avec un langage comme le BASIC. Ainsi le langage BASIC des modules est capable de gérer des nombres à virgule, des communications séries et de très nombreuses autres fonctions qui échappent aux possibilités offertes par le langage LADDER. Un des autres avantages du langage BASIC des modules CUBLOC™ est que malgré le fait que ce dernier soit très performant et doté de très nombreuses fonctions, il est également capable d'accepter certaines instructions connues de la plupart des débutants (IF, GOTO, etc…) afin que ces derniers puissent s'initier progressivement au développement sur les modules CUBLOC™ sans avoir à passer des heures, des jours ou même des mois à essayer de comprendre comment fonctionne le module.

	LADDER	*Programmation en BASIC*
Module	PLC (automate)	PC ou Microcontrôleur
Application	Automatisme	Toute application
Avantage	Gestion logique et séquentielle, gestion de Bit, Timers, Compteurs	Gestion calcule Mathématique, Communication de données (série/ I2C™…), Acquisition de données, Affichage sur écran LCD divers.
Gestion structurelle	En Parallèle	De façon Séquentielle

La gestion parralèle des processus du LADDER ainsi que la gestion séquentielle des processus du langage BASIC disposent chacuns de leurs avantages.

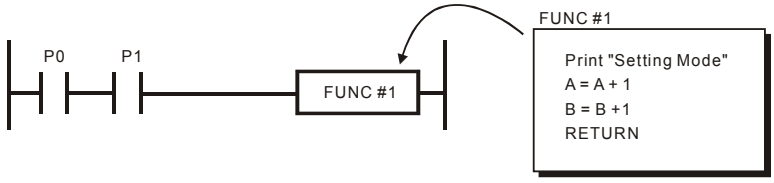
Le LADDER sera ainsi capable de gérer des applications impossibles à réaliser en BASIC.

A l'inverse, le langage BASIC sera capable de réaliser des applications impossible à réaliser en LADDER (ou très difficilement).

C'est la principale raison pour laquelle nous avons créé les modules "CUBLOC™," de telle sorte que l'utilisateur puisse librement utiliser une programmation en langage LADDER et/ou en langage BASIC. Après avoir appris à maîtriser les possibilités et avantages offerts par le langage LADDER et le langage BASIC, l'utilisateur sera ainsi en mesure de pouvoir développer des applications encore plus complexes et de façon plus rapide afin d'avoir une longueur d'avance sur ces concurrents en économisant au final du temps… et de l'argent

Gestion multitâche du LADDER et du BASIC

Il existe plusieurs façons d'implémenter une programmation BASIC et LADDER dans un même processeur. La plupart des produits similaires actuellement disponibles sur le marché utilisent le BASIC comme une « sous-partie » du langage LADDER avec à notre sens à la clef des limitations importantes.



La première limitation vient de la durée d'exécution du programme en langage BASIC qui va irrémédiablement avoir des effets sur le programme LADDER. Par exemple, si le programme BASIC réalise une boucle « sans fin », l'exécution du programme LADDER sera également stoppée. Le principal avantage du LADDER étant de disposer d'un temps d'exécution identique pour l'ensemble de ses actions, il n'est donc pas conseillé dans ce cas d'inclure une partie en langage BASIC qui aura comme effet d'annuler cet avantage !

La seconde limitation est qu'il est très problématique de ne pouvoir utiliser le langage BASIC que comme une « sous-partie » du langage LADDER en ne vous permettant pas ainsi de pouvoir utiliser toutes les possibilités et toute la puissance du langage BASIC.

La troisième limitation pourra provenir de la gestion des ports d'E/S. L'exécution du langage BASIC lors de la gestion des E/S du module pourra créer des conflits non désirés avec le LADDER.

Afin de pouvoir résoudre l'ensemble de ces limitations, nous avons décidé de concevoir un module doté d'une gestion multitâche des programmes développés en langage BASIC et en langage LADDER. Le programme développé en langage BASIC s'exécutera ainsi indépendamment et simultanément du programme développé en langage LADDER sans aucun conflit ne « collision » avec l'un l'autre.

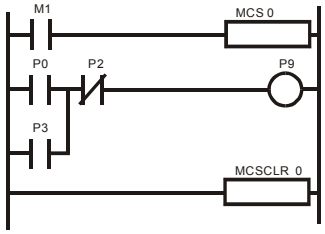
Le langage BASIC des modules CUBLOC™ vous permettra de créer de très nombreuses applications. En comparaison avec la plupart des modules concurrents programmables en langage BASIC, le langage BASIC des modules CUBLOC™ est dans la plupart des cas plus performants, plus structuré et bien plus rapide. Si vous n'avez pas besoin d'avoir recours au langage LADDER, votre application pourra très bien être entièrement programmée en BASIC.

Dans le cas des E/S, l'utilisateur pourra spécifier les E/S devant être utilisées en BASIC et en LADDER afin d'éviter les problèmes de collision des E/S.

Les modules CUBLOC™ utilisent le BASIC comme leur langage « principal ». Il est donc recommandé de controller le LADDER depuis le BASIC.

Par exemple il est possible d'avoir recours à une fonction MASTER CONTROL dans le LADDER, qui permettra à l'utilisateur d'activer/désactiver facilement des zones du LADDER.

```
If A=1 THEN _M(1) = 1
If B=1 THEN _M(1) = 0
```

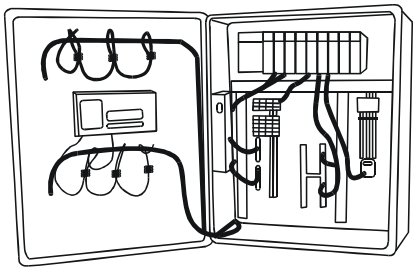


En BASIC, l'utilisateur pourra lire ou écrire dans une mémoire commune avec le langage LADDER. Dans l'exemple ci-dessus, vous pourrez avoir accès au « Registre » M1 par la variable _M(1) depuis le BASIC !

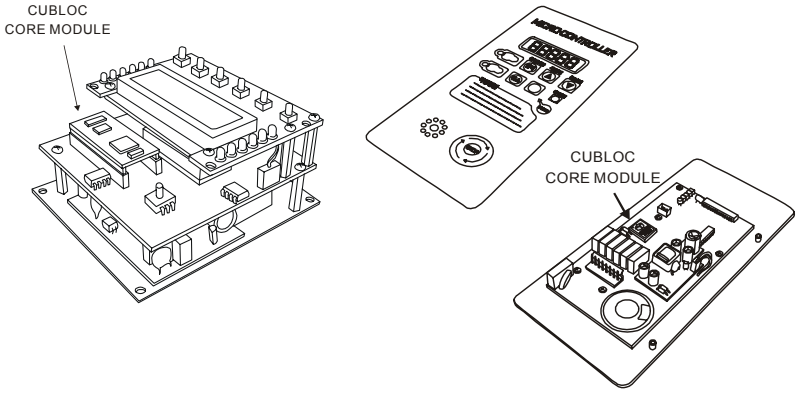
Comme vous pouvez le constater, les modules CUBLOC™ supportent les échanges mémoires multitâches entre le BASIC et le LADDER.

Les avantages des modules PLC « OEM »

Un des principaux avantages des CUBLOC™ est qu'ils se présentent sous la forme d'un module hybride « OEM ». L'utilisation d'automates (PLC) standard nécessite généralement l'utilisation d'un coffrêt destiné à recevoir le câblage de votre application.

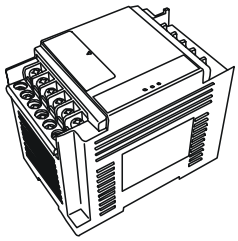
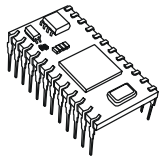


La réalisation d'une installation unique n'est généralement pas un problème. Mais lorsqu'il est nécessaire de réaliser des installations en quantité, le montage et la main d'œuvre nécessaire à leur mise en place deviennent vite rédhibitoire. Sans parler du coût additionnel des éléments à ajouter et de la taille importante de l'application finale.



De part leur conception, les modules hybrides « OEM » CUBLOC™ sont spécialement conçus pour être intégré sur votre propre platine. Vous pourrez ainsi personnaliser votre PLC ainsi que les fonctionnalités et les dimensions de votre application comme si vous utilisiez un « simple » microcontrôleur.

Le tableau ci-dessous montre les différences entre un automate traditionnel et un module CUBLOC™ « OEM ».

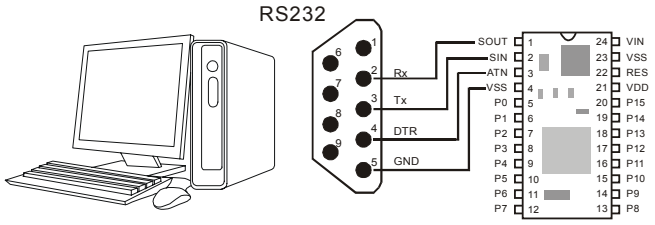
	PLC traditionnel	CUBLOC™
Photo		
Production	Support Rail Din	PCB ou Rail Din
Coût	Souvent élevé	Faible
Montage en série	Souvent difficile	Simple
Dim.	Grande	Faible
Dim. finale	Grande	Compacte

Si vous utilisez déjà des automates traditionnels, comparez les caractéristiques de nos produits et le gain de place, fonctionnalités coût que vous pourrez obtenir en intégrant les CUBLOC™ sur votre application.

Environnement de développement

Pour exploiter et programmer les modules CUBLOC™ vous aurez besoin de disposer d'un compatible PC équipé d'un système d'exploitation de type Windows XP™.

Un port série RS-232 (ou un port USB associé à un câble de conversion USB <> série) sera également nécessaire pour télécharger le programme dans le CUBLOC™ et effectuer le « monitoring » de votre application depuis le PC.



Lorsque le module CUBLOC™ sera déconnecté du PC, ce dernier fonctionnera alors de façon totalement autonome. Votre programme sera stocké dans la mémoire Flash non volatile du CUBLOC™ (la mémoire ne s'efface pas en cas de coupure d'alimentation). Cette mémoire pourra bien évidemment être facilement effacée et reprogrammée à chaque modification de votre programme.

Téléchargement et monitoring via Internet

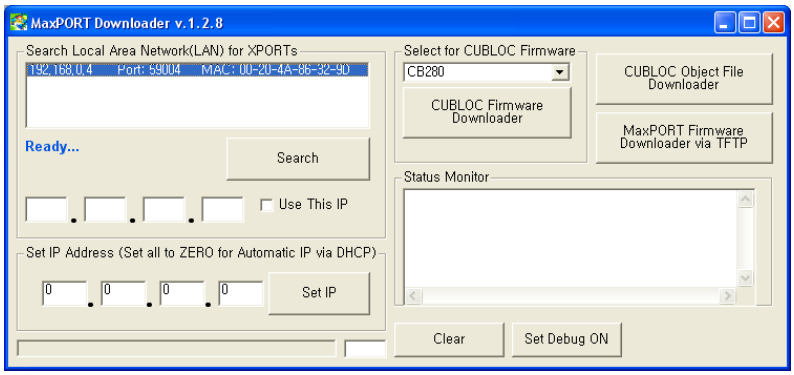
Le XPORT™ est un module additionnel optionnel cabale de convertir les signaux RS-232 en paquets Internet TCP ou UDP. Vous pourrez utilisez le module XPORT™ associé au module CUBLOC™ pour télécharger et faire le monitoring de vos programmes via Internet.

En utilisant cette fonctionnalité, vous pourrez mettre a jour votre application et offrir une prestation de service à vos Clients même si ces derniers sont à l'autre bout du monde !

Nous proposons un Firmware spécial pour le module XPORT™ permettant via un serveur et des applets de télécharger/et faire du Monitoring sur le module CUBLOC™ module. Vous pourrez utiliser ce programme pour gérer des centaines de produits via Internet !

Consultez le **Forum des CUBLOC™** sur la page de nos notes d'applications pour plus d'infos. (http://cubloc.com)

Le module Internet optionnel XPORT™

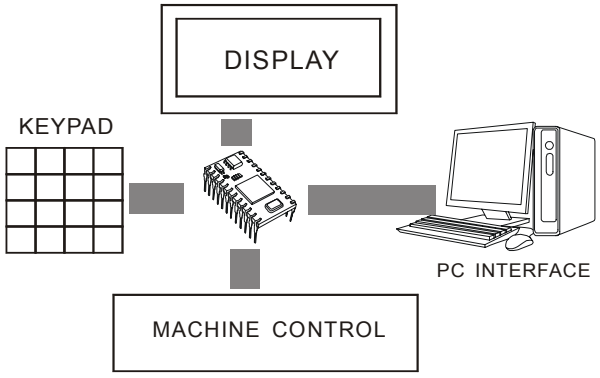


Serveur pour « Monitoring/Download » via de multiples modules XPORT™

Notes à l'attention des utilisateurs de PLC traditionnels

Les personnes habituées à la programmation sur automates traditionnels pourront programmer les modules CUBLOC™ uniquement en langage LADDER. Ces derniers pourront totalement délaisser la programmation en BASIC où s'y « essayer » progressivement à l'aide des nombreux exemples et notes d'applications disponibles en annexe. Dès lors ces personnes pourront intégrer petit à petit de nouvelles fonctions qui n'auraient pas pu être réalisées auparavant avec des automates traditionnels en leur permettant ainsi de pouvoir réaliser des applications plus performantes.

En fait, l'utilisateur n'est nullement besoin d'avoir une quelconque connaissance en langage BASIC pour pouvoir exploiter les modules CUBLOC™ en LADDER.

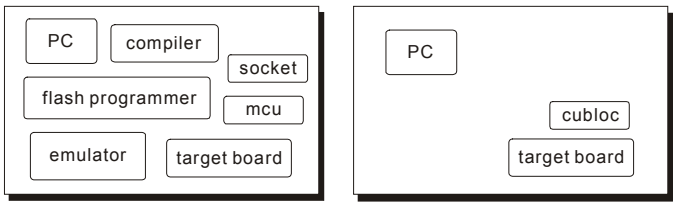


Nous fournissons toutefois de nombreuses librairies de fonctions en BASIC qui vous permettront (par simple « copier & coller ») de pouvoir gérer des claviers de saisie, des afficheurs, etc… même si vos connaissances en BASIC sont limitées.

Notes à l'attention des utilisateurs de microcontrôleurs

Afin de pouvoir réduire les coûts de production dans le cadre de réalisations en très grandes séries, les microcontrôleurs tels que les PIC™, AVR™ et autres 8051 sont tous indiqués. Toutefois le principal inconvénient de ces microcontrôleurs est qu'ils nécessitent généralement (sauf pour les applications très simples) des temps de développement très importants et des durées d'apprentissage très longues et fastidieuses (sans parler des coûts initiaux nécessaires à l'acquisition du matériel de base qui permettra de les exploiter : programmateur, compilateur, sonde de débug, émulateur, etc…).

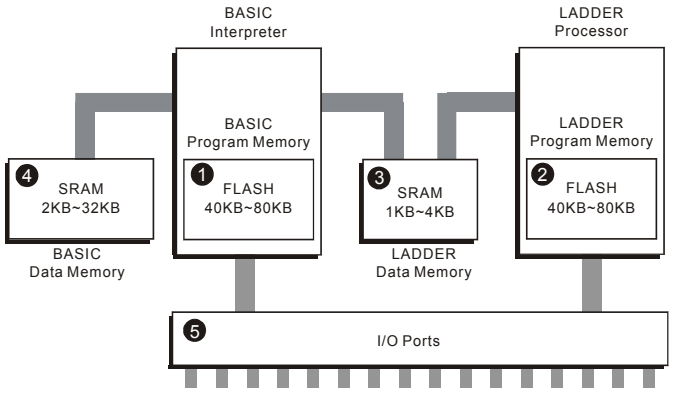
De la part même des ingénieurs expérimentés, le développement sur les microcontrôleurs « standards » est une tâche très complexe qui nécessitera de très longues heures (jour / mois… voir années suivant les cas) pour arriver à concevoir une application finale. Même après la phase de développement si un « bug » est détecté il est pratiquement impossible ou très difficile de mettre à jour le programme du micorocntrôleur.



En comparaison, les modules CUBLOC™ de Comfile vous permettront de bénéficier de temps de développement jusqu'à 20 fois moins long avec la possibilité de pouvoir effectuer des mises à jour via une liaison RS-232 ou via Internet (avec le module optionnel XPORT™) en vous permettant ainsi de disposer d'un atout indéniable sur vos concurrents.

Si en plus vous avez déjà quelques connaissances en terme de programmation sur les microcontrôleurs standards, le passage aux CUBLOC™ sera une simple « formalité ». Vous pourrez dés lors entièrement vous focaliser sur les caractéristiques de votre produit final plutôt que de devoir passer des heures et des heures devant votre ordinateur !

La structure interne des CUBLOC



L'interpréteur BASIC dispose d'une mémoire Flash dédiée au programme BASIC. Le processeur de gestion du LADDER dispose également de sa mémoire Flash pour l'exécution du programme LADDER.

Les Entrées/Sorties peuvent être partagées librement entre les programmes BASIC et LADDER.

La mémoire de données du BASIC ne pourra être accédée que par l'interpréteur BASIC tandis que la mémoire des données du LADDER pourra être accédée à la fois par l'interpréteur BASIC comme par le processeur LADDER.

La mémoire programme du BASIC **(1)** et du LADDER **(2)** se partage la même ressource de mémoire Flash. Cette ressource mémoire est de 80 K. Une application entièrement programmée en BASIC peut utiliser l'intégralité de ces 80 K. De même, une application entièrement programmée en LADDER peut également utiliser l'intégralité de cette mémoire. Une application programmée en BASIC et en LADDER pourra être développée dès lors que le total du programme BASIC et LADDER ne dépasse pas les 80 K. Les modèles CB2XX disposent actuellement de 80 K. Les futures versions de modules CUBLOC™ pourront disposer davantage de mémoire.

Les ports d'Entrées/Sorties **(5)** peuvent être partagés entre le BASIC et le LADDER. L'utilisateur doit spécifier les ports d'E/S utilisés dans le BASIC et ceux utilisés dans le LADDER. Il est possible d'utiliser tous les ports uniquement pour le BASIC ou uniquement pour le LADDER.

Les périphériques des CUBLOC

Platines « PROTO BOARD »

Les platines « Proto Board » sont spécialement conçues pour réaliser des tests et développer les prototypes de vos applications à base de modules CUBLOC™. Ces platines intègrent un étage de régulation ainsi qu'un étage de mise en forme de la sortie série des modules.

Platines « BASE BOARD » (série CUBASE)

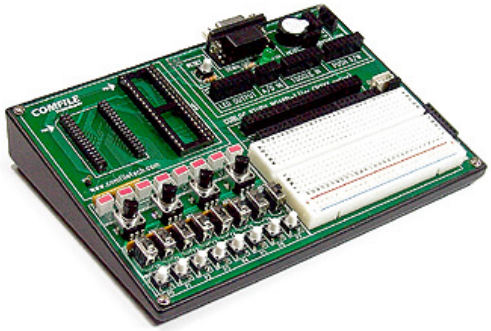
Les platines « BASE Board » sont spécialement conçues pour accélérer la mise en œuvre de vos applications. Destinées à être alimentée en 24 Vcc, ces dernières pourront directement piloter des relais ou « lire » l'état de contacts.

Platines série « CUSB »

Les platines « CUSB » sont spécialement conçues pour accélérer la mise en œuvre de vos applications. Destinées à être alimentée en 24 Vcc, ces dernières disposent de sorties relais et d'entrées tout-ou-rien opto-isolées.

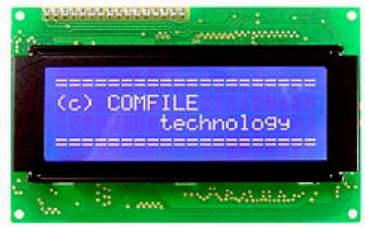
Platines « STUDY BOARD »

Ces platines sont toutes indiquées pour l'expérimentation et la prise en main rapide et simplifiée des modules CUBLOC™. Ces dernires disposent de bouton-poussoirs, de leds, de boutons-poussoirs, d'un étage de régulation, d'unn étage de mise en forme de la sortie série des modules, d'un buzzer, d'une plaque de développement sans soudure, etc…



Afficheurs LCD (Modules série CLCD, GHLCD)

De nombreux types d'afficheurs LCD pourront être utilisés par les modules CUBLOC™ via une communication de type I2C™. A l'aide d'une seule ligne de commande (PRINT, CLS, etc…), vous pourrez facilement commencer à écrire du texte sur votre afficheur sans avoir à utiliser de multiples instructions complexes.



Modules d'affichage 7 Segments à Leds (série CSG)

Divers types d'afficheurs numériques 7 segments à Leds pourront être utilisés par les modules CUBLOC™ via ™ via une communication de type I2C™.

Afficheurs graphiques à dalles tactiles (série CUTOUCH)

Le boîtier « CUTOUCH » est un afficheur graphique LCD à dalle tactile doté d'un cœur microcontrôlé « CUBLOC™ ». Des instructions BASIC vous permettront de concevoir des écrans de saisie et de gérer la dalle tactile tandis que la partie LADDER vous permettra d'avoir accès aux possibilités d'automatisme des CUBLOC.

Cet afficheur existe aussi en version « OEM » (sans boîtier et sans module microcontrôlé) afin que vous puissiez le piloter avec un module CUBLOC™ externe. De très nombreux modules optionnels supplémentaires seront prochainement disponibles, consultez les sites www.comfiletech.com et www.lextronic.fr afin de vous tenir informé de la disponibilités des nouveautés.

Chapitre 2.

Aspect matériel des CUBLOC…

Caractéristiques matérielles

Les CUBLOC™ de la série CB2XX ont les caractéristiques suivantes :

- (BASIC et LADDER) 80 K de mémoire Flash
- Vitesse d'exécution BASIC : 36,000 Instr/sec
- Vitesse d'exécution LADDER : Scan time 10 ms (Turbo Mode ~= 100 Micro seconde)
- Mémoire données pour le BASIC: 2 ~ 24 K (Sauvegarde batterie sur modèle CB290)
- Mémoire données LADDER: 1 ~ 4 K
- Mémoire EEPROM: 4 K
- 16 à 91 broches d'E/S
- 8 canaux de conversion Analogique/numérique 10 bits
- 3 ou 6 canaux PWM (8 ~ 16 bits)
- Gestion matérielle UART (RS232C)
- Interface PC via port RS232C
- Circuit d'horloge RTC intégré (sur CB290 uniquement)

Tableau de compaison

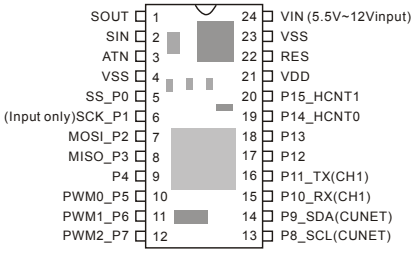
Caractéristique / Modèle	CB220	CB280	CB290	CB405
Mémoire programme	80 K	80 K	80 K	200 K
Mémoire données	BASIC 2 K LADDER 1 K	BASIC 2 K LADDER 1 K	BASIC 24 K LADDER 4 K	BASIC 51 K LADDER 4 K HEAP 55 K
Mémoire donnée sauvegardable	-	-	En option	En option
EEPROM	4 K	4 K	4 K	4 K
Ports E/S	16	49	91	64
Ports séries	2	2	2	4
Format	DIL 24broches	Module 64 broches	Module 108 broches	Module 80 broches
Conv. "A/N" 10 bits	8 Canaux	8 Canaux	8 Canaux	16 Canaux
PWM	3 Canaux	6 Canaux	6 Canaux	12 Canaux
Interruption Ext.	-	4	4	4
Entrée compt. rapide	2 Canaux	2 Canaux	2 Canaux	2 Canaux
(RTC)	-	-	Oui	-
Temp. fonctionnement	+10°C~65°C	+10°C~65°C	+10°C~65°C	+10°C~65°C

Module « CB220 »

Le CB220 se présente sous la forme d'un module hybride 24 broches au format. Doté de 16 ports d'entrées/sorties, il intègre son propre étage de régulation + 5 V. Si vous avez déjà l'habitude de travailler avec d'autres modules concurrents programmables en langage BASIC, le module « CB220 » risque de vous intéresser au plus haut point.

D'une part parce qu'il offre des possibilités et une puissance bien supérieur à la plupart des produits similaires, mais aussi parce qu'il est compatible broches à broches avec les produits que vous utilisez peut être actuellement ! (dans certains cas il vous faudra juste court-circuiter un condensateur sur votre platine de test existante pour assurer la compatibilité – voir plus d'infos page 253 de la rubrique « 9. Tutorial – Premières applications en BASIC et LADDER»).

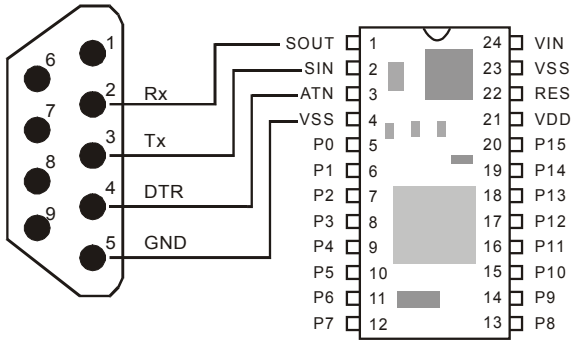
N'hésitez-pas à faire des comparaisons de performances / possibilités / nombre d'instructions / capacités mémoires / prix avec vos produits habituels... Vous vous apercevrez alors très vite qu'il est peut être temps de passer aux CUBLOC™ !



CB220

Pin	Nom	E/S	Bloc Port	Description
1	SOUT	Sortie		Téléchargement série (Sortie)
2	SIN	Entrée		Téléchargement série (Entrée)
3	ATN	Entrée		Téléchargement série (Entrée)
4	VSS	Alim.		Masse
5	P0	E/S		ADC0 / SPI SS
6	P1	Entrée		ADC1 / SPI SCK
7	P2	E/S		ADC2 / SPI MOSI
8	P3	E/S	Bloc 0	ADC3 / SPI MISO
9	P4	E/S		ADC4
10	P5	E/S		PWM0 / ADC5
11	P6	E/S		PWM1 / ADC6
12	P7	E/S		PWM2 / ADC7
13	P8	E/S		SCL (vers module CuNet)
14	P9	E/S		SDA (vers module CuNet)
15	P10	E/S		Port série 1 - Niv. TTL RX / INT2
16	P11	E/S	Bloc 1	Port série 1 - Niv. TTL TX / INT3
17	P12	E/S		
18	P13	E/S		
19	P14	E/S		Compteur rapide canal 0
20	P15	E/S		Compteur rapide canal 1
21	VDD	E/S		5 V Entrée / Sortie
22	RES	Entrée		Entrée RESET (Reset sur un Niveau Bas !)
23	VSS	Entrée		Masse
24	VIN	Entrée		5.5 V ~ 12 V Entrée alimentation

SIN, SOUT, ATN sont les broches de connexion devant être raccordées au port série du PC (ou du module XPORT™ optionnel) pour télécharger, débuguer ou avoir accès au mode moniteur du CUBLOC™ (voir schéma ci-dessous). Tous les modèles de CUBLOC™ disposent des broches SOUT, SIN, ATN.



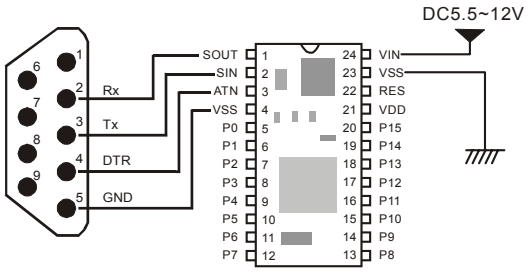
La plupart des autres broches sont des « ENTREES / SORTIES ». L'utilisateur pourra spécifier quelles seront les broches à utiliser en ENTREES et quelles seront celles à utiliser en SORTIES.

Lorsqu'elles sont utilisées en ENTREE, les broches sont dans un état HAUTE impédance. Lorsqu'elles sont utilisées en SORTIE, les broches peuvent prendre un état logique HAUT ou BAS. Le courant de sortie maximale disponible en sortie est de l'ordre de 25 mA. L'utilisateur est libre de choisir le rôle de chaque broche.

Alimentation du module « CB220 »

Le CB220 peut être alimenté de 2 façons différentes :

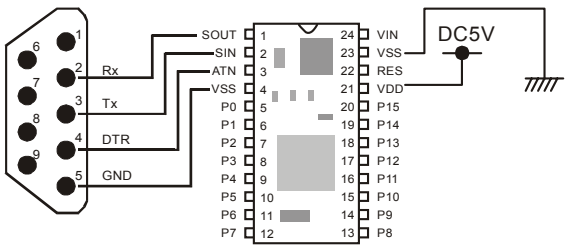
Méthode 1



Le CB220 dispose d'un régulateur interne qui vous permettra de pouvoir l'alimenter sous une tension continue comprise entre 5,5 ~12 Vcc (la tension doit être convenablement filtrée). Ce régulateur fourni également une tension de sortie stable de +5 Vcc (100 mA max.) sur une des broches du CUBLOC. Utilisez la broche 24 pour appliquer votre tension de 5,5 ~12 Vcc (dès lors vous obtiendrez du +5 Vcc sur la broche 21 - **Attention : ne connectez jamais de dispositif consommant plus de 100 mA sur cette broche sous peine de destruction de l'étage de régulation du module CB220 (non pris en compte par la garantie**).

Il ne sera par exemple pas possible d'alimenter un afficheur de type « CLCD » via la broche 21 du CUBLOC (la consommation du rétro-éclairage de l'afficheur étant trop importante).

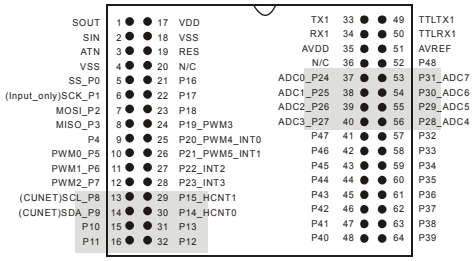
Méthode 2



Vous pouvez alimenter le module CB220 directement sous une tension continue de + 5 V régulée et filtrée en utilisant la broche 21.

Module « CB280 »

Le CB280 se présente sous la forme d'un module capôté 64 broches dont 49 broches peuvent être utilisées en E/S. Attention, le CB280 ne dispose pas de régulateur interne et vous devrez l'alimenter uniquement sous une tension régulée et filtrée de + 5 Vcc.



CB280

Les N° de broches sont rangées par fonctions et non pas par ordre chronologique.

Nom	Pin #	E/S	Bloc Port	Description
SOUT	1	Sortie		Téléchargement série (Sortie)
SIN	2	Entrée		Téléchargement série (Entrée)
ATN	3	Entrée		Téléchargement série (Entrée)
VSS	4	Alim.		Masse
P0	5	E/S	Bloc 0	SPI SS
P1	6	Entrée		SPI SCK
P2	7	E/S		SPI MOSI
P3	8	E/S		SP MISIO
P4	9	E/S		
P5	10	E/S		PWM0
P6	11	E/S		PWM1
P7	12	E/S		PWM2
P8	13	E/S	Bloc 1	SCL (vers Module CuNET)
P9	14	E/S		SDA (vers Module CuNET)
P10	15	E/S		
P11	16	E/S		
P12	32	E/S		
P13	31	E/S		
P14	30	E/S		Compteur rapide canal 0
P15	29	E/S		Compteur rapide canal 1

**** IMPORTANT :***
Les broches 20 et 36 ne sont pas utilisées – NE RIEN CONNECTER SUR CES BROCHES

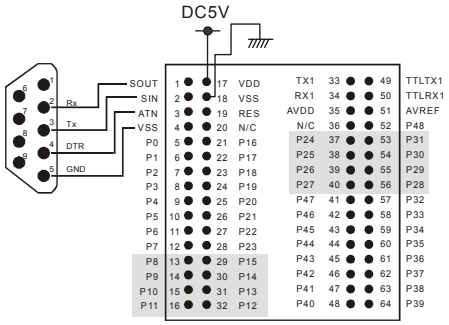
P16	21	E/S	Bloc 2	
P17	22	E/S		
P18	23	E/S		
P19	24	E/S		PWM3
P20	25	E/S		PWM4 / INT0
P21	26	E/S		PWM5 / INT1
P22	27	E/S		INT2
P23	28	E/S		INT3
P24	37	E/S	Bloc 3	ADC0 : A/N Canal 0
P25	38	E/S		ADC1 : A/N Canal 1
P26	39	E/S		ADC2 : A/N Canal 2
P27	40	E/S		ADC3 : A/N Canal 3
P28	56	E/S		ADC4 : A/N Canal 4
P29	55	E/S		ADC5 : A/N Canal 5
P30	54	E/S		ADC6 : A/N Canal 6
P31	53	E/S		ADC7 : A/N Canal 7
P32	57	E/S	Bloc 4	
P33	58	E/S		
P34	59	E/S		
P35	60	E/S		
P36	61	E/S		
P37	62	E/S		
P38	63	E/S		
P39	64	E/S		
P40	48	E/S	Bloc 5	
P41	47	E/S		
P42	46	E/S		
P43	45	E/S		
P44	44	E/S		
P45	43	E/S		
P46	42	E/S		
P47	41	E/S		
P48	52	E/S		

VDD	17	IN		Alim. 5 Vcc
VSS	18	IN		Masse
RES	19	IN		Entrée RESET Reset sur un Niveau Bas
TX1	33			Canal 1 (RS232) Niveau +/- 12V -> Sortie données
RX1	34			Canal 1 (RS232) Niveau +/- 12V -> Entrée données
AVDD	35			Alim. convertisseur « A/N »
TTLTX1	49			Canal 1 (RS232) Niveau 5 V (TTL) - Sortie données
TTLRX 1	50			Canal 1 (RS232) Niveau 5 V (TTL) - Entrée données
AVREF	51			Référence de tension ADC

**** IMPORTANT :***
Les broches 20 et 36 ne sont pas utilisées – NE RIEN CONNECTER SUR CES BROCHES

Alimentation du module « CB280 »

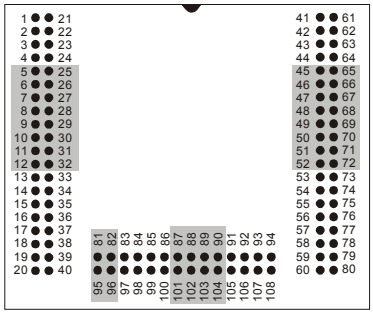
Le module CB280 ne dispose pas de régulateur interne et vous devrez l'alimenter uniquement sous une tension régulée et filtrée de + 5 Vcc comme indiqué ci-dessous.



**** IMPORTANT :***
Les broches 20 et 36 ne sont pas utilisées – NE RIEN CONNECTER SUR CES BROCHES

Module « CB290 »

Le CB290 se présente sous la forme d'un module capôté 108 broches dont 92 broches peuvent être utilisées en E/S. Attention, le CB290 ne dispose pas de régulateur interne et vous devrez l'alimenter uniquement sous une tension régulée et filtrée de + 5 Vcc. Ce dernier dispose d'une mémoire RAM de 28 K et d'une horloge RTC pouvant être sauvegardés par une pile externe (non livrée). Parmis les ports d'E/S 32 ports ne peuvent être utilisés qu'en sorties, 33 ports ne peuvent être utilisés qu'en entrées et le reste des ports peuvent être configurés à volonté.



Les N° de broches sont rangées par fonctions et non pas par ordre chronologique.

Nom	Pin #	E/S	Bloc Port	Désignation
SOUT	1	Sortie		Téléchargement série (Sortie)
SIN	2	Entrée		Téléchargement série (Entrée)
ATN	3	Entrée		Téléchargement série (Entrée)
VSS	4	Alim.		Masse
P0	5	E/S	Bloc 0	SPI SS
P1	6	Entrée		SPI SCK
P2	7	E/S		SPI MOSI
P3	8	E/S		SPI MISIO
P4	9	E/S		
P5	10	E/S		PWM0
P6	11	E/S		PWM1
P7	12	E/S		PWM2
P8	25	E/S	Bloc 1	ADC0 : A/N Canal 0
P9	26	E/S		ADC1 : A/N Canal 1
P10	27	E/S		ADC2 : A/N Canal 2
P11	28	E/S		ADC3 : A/N Canal 3
P12	29	E/S		ADC4 : A/N Canal 4
P13	30	E/S		ADC5 : A/N Canal 5
P14	31	E/S		ADC6 : A/N Canal 6
P15	32	E/S		ADC7 : A/N Canal 7
P16	83	E/S	Bloc 2	SCL (vers module CuNET)
P17	84	E/S		SDA (vers module CuNET)
P18	85	E/S		INT 2
P19	86	E/S		INT 3
P20	97	E/S		
P21	98	E/S		
P22	99	E/S		Compteur rapide canal 0
P23	100	E/S		Compteur rapide canal 1

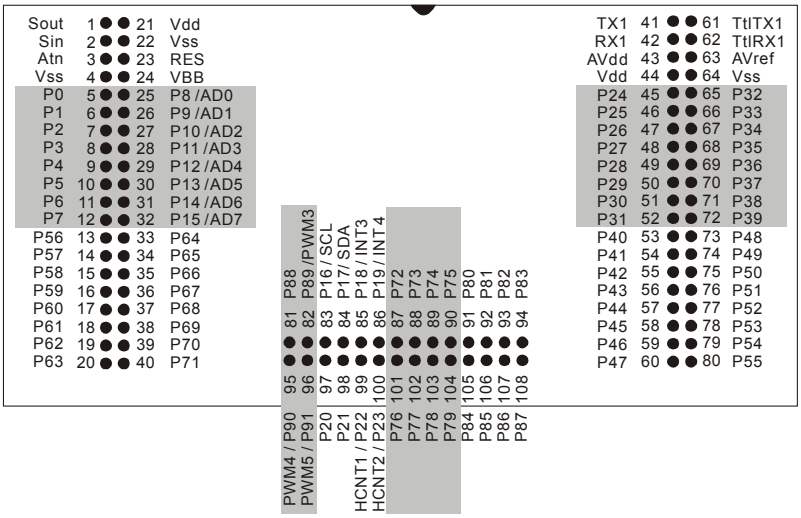
*** Ne pas utiliser le port P88 et ne rien connecter dessus.**

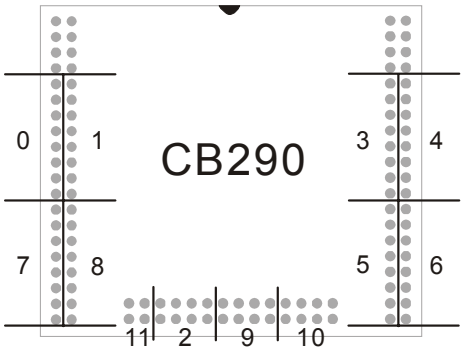
P24	45	Sortie	Bloc 3	
P25	46	Sortie		
P26	47	Sortie		
P27	48	Sortie		
P28	49	Sortie		
P29	50	Sortie		
P30	51	Sortie		
P31	52	Sortie		
P32	65	Sortie	Bloc 4	
P33	66	Sortie		
P34	67	Sortie		
P35	68	Sortie		
P36	69	Sortie		
P37	70	Sortie		
P38	71	Sortie		
P39	72	Sortie		
P40	53	Sortie	Bloc 5	
P41	54	Sortie		
P42	55	Sortie		
P43	56	Sortie		
P44	57	Sortie		
P45	58	Sortie		
P46	59	Sortie		
P47	60	Sortie		
P48	73	Sortie	Bloc 6	
P49	74	Sortie		
P50	75	Sortie		
P51	76	Sortie		
P52	77	Sortie		
P53	78	Sortie		
P54	79	Sortie		
P55	80	Sortie		
P56	13	Entrée	Bloc 7	
P57	14	Entrée		
P58	15	Entrée		
P59	16	Entrée		
P60	17	Entrée		
P61	18	Entrée		
P62	19	Entrée		
P63	20	Entrée		
P64	33	Entrée	Bloc 8	
P65	34	Entrée		
P66	35	Entrée		
P67	36	Entrée		
P68	37	Entrée		
P69	38	Entrée		
P70	39	Entrée		
P71	40	Entrée		
P72	87	Entrée	Bloc 9	
P73	88	Entrée		
P74	89	Entrée		
P75	90	Entrée		
P76	101	Entrée		
P77	102	Entrée		
P78	103	Entrée		
P79	104	Entrée		

*** Ne pas utiliser le port P88 et ne rien connecter dessus.**

P80	91	Entrée	Bloc 10	
P81	92	Entrée		
P82	93	Entrée		
P83	94	Entrée		
P84	105	Entrée		
P85	106	Entrée		
P86	107	Entrée		
P87	108	Entrée		
P88	81	N/C	Bloc 11	N/C (Ne rien connecter)
P89	82	E/S		PWM3
P90	95	E/S		PWM4 / INT0
P91	96	E/S		PWM5 / INT1
VDD	21,44	Entrée		Alim. 5 Vcc
VSS	22,64	Entrée		Masse
RES	23	Entrée		Entrée RESET Reset sur un Niveau Bas
VBB	24	Entrée		Pile Backup
TX1	41			Canal 1 (RS232) +/- 12V -> Sortie données
RX1	42			Canal 1 (RS232) +/- 12V -> Entrée données
AVDD	43			Alim. convertisseur « A/N »
TTLTX1	61			Canal 1 (RS232) Niveau 5 V (TTL) Sortie données
TTLRX 1	62			Canal 1 (RS232) Niveau 5 V (TTL) Entrée données
AVREF	63			Référence de tension ADC

*** Ne pas utiliser le port P88 et ne rien connecter dessus.**





Port Blocks

Note concernant les ports P24 à P55.

Il existe à ce jour 2 versions de modules « CB290 ». La version « CB290 » **rev A.** et la version « CB290 » **rev B.** (vous pouvez contrôler votre version grâce à une inscription présente au dos du module).

Pour les modules « CB290 » (ou CT1720) en version rev A.
Lorsque vous utilisez ces modules avec une pile de sauvegarde (y compris sur les platines de test « CB290 Proto » ou « Baseboard 64 » ou « CT1720 ») la mémoire RAM des données (relative aux ports P24 à P55) sera sauvegardée en cas de coupure d'alimentation (y compris l'état de sortie des « E/S »).A la « remise » sous tension, ces ports retrouveront l'état qu'ils avaient au moment de la coupure d'alimentation. Ceci permettra au module de reprendre son fonctionnement « normal » en cas de perte d'alimentation passagère.

Toutefois cette fonctionnalité nécessite d'être vigilent si vous ne connaissez pas l'état des données de la RAM alors que vous utilisez une pile de sauvegarde afin d'éviter que des valeurs aléatoires n'activent ces sorties lors de la mise sous tension. Si vous devez disposer de sorties dont le niveau doit être impérativement « BAS » à la mise sous tension, utilisez alors des ports d'entrées/sorties standards (autres que P24 à P55).

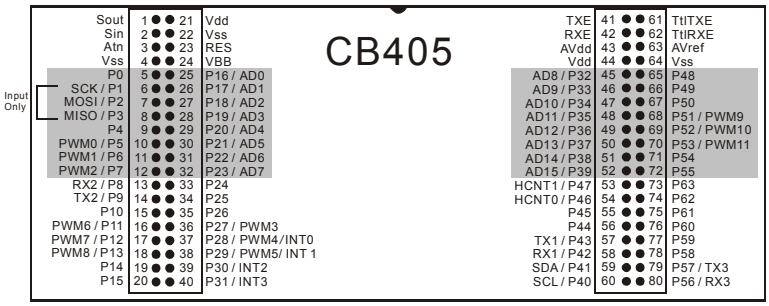
Pour les modules « CB290 » (ou CT1720) en version rev B.
Les ports P24 à P55 sont en haut impédance (High-Z) à la mise sous tension du module « CB290 » (ou CT1720) afin d'éviter que des valeurs « aléatoires » ne soient appliquées à ce moment sur ces derniers. Aussi afin de pouvoir exploiter tous ces Ports en sortie, il vous faudra utiliser au préalable l'instruction « Set Outonly On ».

Set Outonly On

Le port P88 est en fait utilisé pour réagir à l'instruction « Set Outonly » et piloter les étages des ports P24 à P55. Il est donc impératif de ne JAMAIS utiliser le port P88 au sein de votre application (ni en BASIC, ni en Ladder).

Module « CB405 »

Le CB405 se présente sous la forme d'un module capôté 80 broches dont 64 broches peuvent être utilisées en E/S. Attention, le CB405 ne dispose pas de régulateur interne et vous devrez l'alimenter uniquement sous une tension régulée et filtrée de + 5 Vcc. Ce dernier dispose d'une mémoire RAM de 55 K pouvant être sauvegardée par une pile externe (non livrée).



Les N° de broches sont rangées par fonctions et non pas par ordre chronologique.

Nom	Pin #	E/S	Bloc Port	Désignation
SOUT	1	Sortie		Téléchargement série (Sortie)
SIN	2	Entrée		Téléchargement série (Entrée)
ATN	3	Entrée		Téléchargement série (Entrée)
VSS	4, 22, 64	Alim.		Masse
VDD	21, 44	Alim.		5 Vcc
AVDD	43	Alim.		Alim. convertisseur « A/N »
AVREF	63	Alim.		Référence de tension ADC
VBB	24	Entrée		Pile Backup
RES	23	Entrée		Entrée RESET Reset sur un Niveau Bas
TTL TXE	61	Sortie		Broche libre du "MAX232" interne Niveau 5 V (TTL) Sortie données
TTL RXE	62	Entrée		Broche libre du "MAX232" interne Niveau 5 V (TTL) Entrée données
TXE	41	Sortie		Broche exploitable du MAX232 Niveau +/- 12V Sortie données
RXE	42	Entrée		Broche exploitable du MAX232 Niveau +/- 12V Entrée données

Les N° de broches sont rangées par fonctions et non pas par ordre chronologique

Nom	Pin #	E/S	Bloc Port	Désignation
P0	5	E/S	Bloc 0	SPI SS
P1	6	Entrée		SPI SCK
P2	7	Entrée		SPI MOSI
P3	8	Entrée		SPI MISIO
P4	9	E/S		
P5	10	E/S		PWM0
P6	11	E/S		PWM1
P7	12	E/S		PWM2

P8	13	E/S	Bloc 1	Canal 2 - TTLRX2 Niveau 5 V (TTL) Entrée données
P9	14	E/S		Canal 2 - TTLTX2 Niveau 5 V (TTL) Sortie données
P10	15	E/S		
P11	16	E/S		PWM6
P12	17	E/S		PWM7
P13	18	E/S		PWM8
P14	19	E/S		
P15	20	E/S		

P16	25	E/S	Bloc 2	ADC0 : A/N Canal 0
P17	26	E/S		ADC1 : A/N Canal 1
P18	27	E/S		ADC2 : A/N Canal 2
P19	28	E/S		ADC3 : A/N Canal 3
P20	29	E/S		ADC4 : A/N Canal 4
P21	30	E/S		ADC5 : A/N Canal 5
P22	31	E/S		ADC6 : A/N Canal 6
P23	32	E/S		ADC7 : A/N Canal 7

P24	33	E/S	Bloc 3	Co-processeur (SCL) *
P25	34	E/S		Co-processeur (SDA) *
P26	35	E/S		Co-processeur (INT) *
P27	36	E/S		PWM3
P28	37	E/S		PWM4 / INT0
P29	38	E/S		PWM5 / INT1
P30	39	E/S		INT2
P31	40	E/S		INT3

* Pour pilotage ultérieur d'un coprocesseur externe (essayez dans la mesure du possible de ne pas exploiter ces broches afin que vous puissiez y avoir recours lorsque le module coprocesseur externe sera disponible).

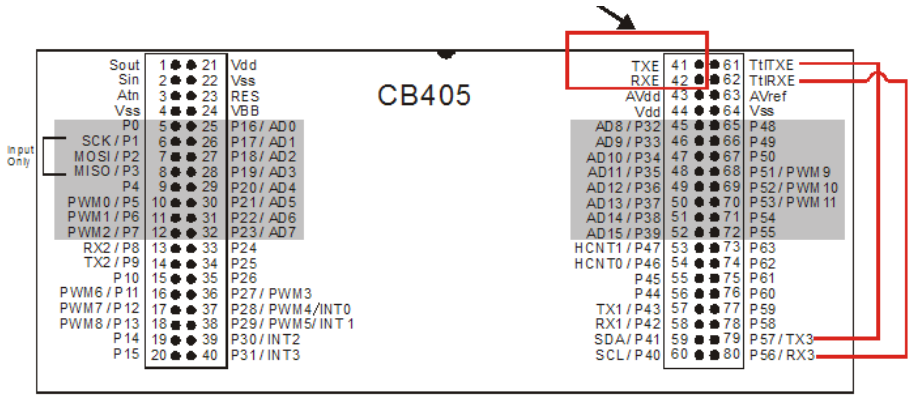
P32	45	E/S	Bloc 4	ADC8 : A/N Canal 8
P33	46	E/S		ADC9 : A/N Canal 9
P34	47	E/S		ADC10 : A/N Canal 10
P35	48	E/S		ADC11 : A/N Canal 11
P36	49	E/S		ADC12 : A/N Canal 12
P37	50	E/S		ADC13 : A/N Canal 13
P38	51	E/S		ADC14 : A/N Canal 14
P39	52	E/S		ADC15 : A/N Canal 15

P40	60	E/S	Bloc 5	SCL (vers module CuNET)
P41	59	E/S		SDA (vers module CuNET)
P42	58	E/S		Canal 1 – TTLRX1 Niveau 5 V (TTL) Entrée données
P43	57	E/S		Canal 1 – TTLTX1 Niveau 5 V (TTL) Sortie données
P44	56	E/S		
P45	55	E/S		
P46	54	E/S		Compteur rapide canal 0
P47	53	E/S		Compteur rapide canal 1

P48	65	E/S	Bloc 6	
P49	66	E/S		
P50	67	E/S		
P51	68	E/S		PWM9
P52	69	E/S		PWM10
P53	70	E/S		PWM11
P54	71	E/S		
P55	72	E/S		

P56	80	E/S	Bloc 7	Canal 3 – TTLRX3 Niveau 5 V (TTL) Entrée données
P57	79	E/S		Canal 3 – TTLTX3 Niveau 5 V (TTL) Sortie données
P58	78	E/S		
P59	77	E/S		
P60	76	E/S		
P61	75	E/S		
P62	74	E/S		
P63	73	E/S		

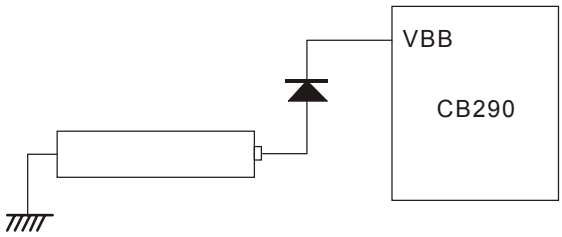
Le module « CB405 » dispose d'un composant interne de mise à niveau « MAX232 » dont les broches (TTLTXE et TTLRXE) pourront être connectées à l'un de ses Ports séries (niveau logique 0 / 5 V) afin que le port en question puisse être directement utilisé sur une liaison série au niveau logique +/- 12 V via les broches « TXE et RXE ». Dans l'exemple ci-dessous, le Port série TX3 initialement doté d'un niveau logique 0 / 5 V pourra être exploité via les broches « TXE et RXE » sur une liaison série RS232 de niveau logique + /- 12 V.



Comment connecter une pile de sauvegarde sur les modules « CB290 » et « CB405 » ?

Si vous utilisez une « super capacité » sur le port VBB du module CB290, il vous sera possible de sauvegarder les données RAM et RTC pendant plusieurs jours en cas de coupure d'alimentation.

Pour des durées de sauvegardes plus importantes une pile devra être utilisée. Le recours à une pile et à une large capacité pourra suivant les cas d'utilisation vous permettre d'entre la durée de sauvegarde jusqu'à 1 an. Utilisez impérativement le schéma ci-dessous AVEC la diode pour réaliser le dispositif de sauvegarde.



Caractéristiques électriques communes

- Tension de fonctionnement : 4.5 à 5.5 Vcc
- Fréquence de cadencement : 18.432 MHz
- Source Current port E/S : 20 mA
- Sink Current port E/S: 25 mA
- Températures limites de fonctionnement : +10 à +65 Degrés (Celsius)
- Fonctionnement taux d'humidité : 10 ~ 80 % RH

Précautions d'usages

Il conviendra d'être extrêmement vigilant avec le type de signaux appliqués sur les broches des modules CUBLOC™ et le type de dispositifs pilotés par ces broches. Ceci est d'autant plus vrai lors des premières phases d'utilisation ou pour le besoin de vos tests, pendant lesquels vous serez amené à changer souvent le rôle (entrée ou sortie) de vos "broches".

Correctement utilisés, vos modules CUBLOC™ vous permettront de réaliser d'innombrables quantités de montages et d'applications dont vous ne pouvez même pas imaginer la puissance. Toutefois il vous faut impérativement garder à l'esprit que les modules CUBLOC™ ne sont rien d'autre que des microcontrôleurs et au même titre qu'avec tout autre microcontrôleur il vous faut respecter certaines règles de bases afin d'éviter qu'il ne rende l'âme !

1) Ne jamais dépasser les limites des tensions d'alimentation des modules CUBLOC™.

2) Si vous appliquez des tensions issues de capteurs ou de dispositifs extérieurs sur les CUBLOC™:

   - Vérifiez toujours que ces tensions soient égales ou inférieures à + 5 Vcc.

- Coupez **en PRIORITÉ** l'alimentation des capteurs externes **AVANT** de couper celle du module CUBLOC™ afin d'éviter qu'une tension soit toujours présente sur l'entrée du module CUBLOC™ alors que ce dernier n'est plus alimenté (sans quoi le port du CUBLOC™ serait endommagé).

- Selon la même recommandation que ci-dessus, vérifiez que vous ne disposez pas de condensateurs de forte valeur reliés sur les entrées des CUBLOC™, lesquels pourront stocker une tension qui viendra alors se décharger dans le CUBLOC lorsque vous couperez les alimentations.

3) Ne jamais inverser la polarité d'alimentation du module CUBLOC™.

4) Lorsque vous utilisez les ports du CUBLOC™ en entrées, n'utilisez jamais de grand fils pour y raccorder des boutons-poussoirs et autres capteurs sans avoir recours à un circuit de mise en forme et de protection (voir exemples de montage dans la documentation). Si pour vos tests vous n'utilisez pas de protection de ce type, limitez la longueur de vos fils à 3 - 4 cm afin d'éviter les phénomènes de "latch-up" ou de destruction par électricité statique.

5) Utilisez impérativement des diodes et autres dispositfs de protection lorsque vous pilotez (via une électronique de puissance) des charges inductives (moteurs par exemple) et éviter de placer le câble de téléchargement à côté de cette source.

6) Découplez rigoureusement l'alimentation du CUBLOC™ (au plus près de celui-ci).

7) Avant d'appliquer une quelconque tension (+ 5V ou masse) sur une des broches du CUBLOC™, vérifiez IMPERATIVEMENT que cette broche ai bien été configurée en ENTREE. Dès lors, ne reliez aucune tension (+ 5V ou masse) sur les ports du CUBLOC™ configurés en sorties (sous peine de court-circuit et de destruction de ces derniers).

8) Passez toujours par un montage d'interface (voir exemple dans la notice) pour alimenter et piloter un dispositif consommant plus d'une vingtaine de milli-ampère.

9) Si certaines broches du module CUBLOC™ ne sont pas utilisées pour les besoins de votre application, configurez tout de même impérativement ces dernières en ENTREES. Dans le même esprit, mettez toutes les broches de SORTIES non utilisées au niveau logique BAS. Remettez également à jour l'état de toutes les broches des CUBLOC™ régulièrement (même celles non utilisées) au sein de la « boucle » principale de votre programme (ne vous contentez pas d'une simple configuration au début du programme).

10) Comme TOUT microcontrôleur, les CUBLOC sont sensibles à l'électricité statique. Ces derniers devront donc être manipulés (et soudés) avec les précautions qui s'imposent afin d'éviter leur destruction ou leur fragilisation.

11) Le port P1 est UNIQUEMENT un port utilisable en entrée.

12) Lorsqu'elles ne sont pas utilisées, les broches SIN, SOUT et ATN ne doivent jamais être connectées à quoi que ce soit. Le connecteur de programmation doit être câblé directement à la sortie et au plus près des broches SIN, SOUT et ATN du module CUBLOC

13) N'utilisez pas de câble de raccordement série supérieur à 2 mètres pour le téléchargement de votre module CUBLOC™.

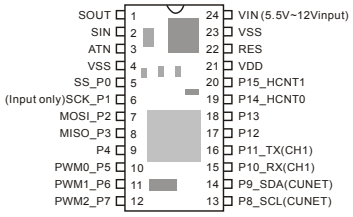
14) Ne déconnectez JAMAIS le câble de programmation et/ou ne coupez JAMAIS l'alimentation du module CUBLOC™ lorsque ce dernier est en cours de programmation ou que vous remettez à jour son « Firmware ».

15) Si vous déconnectez le câble de programmation du module CUBLOC™ ou que vous éteignez l'alimentation du PC alors que le CUBLOC™ est encore relié au PC via le câble de programmation, le module CUBLOC™ peut être ammené à effectuer un Reset (prenez en compte lors du développement de votre application).

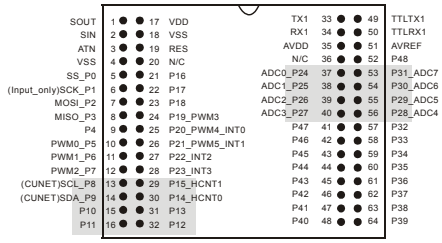
16) Il est impératif de faire fonctionner les modules CUBLOC™ en dessous des valeurs limites de températures, lequelles correspondent aux conditions extrêmes d'utilisation.

En cas de non respect des limites et des conditions d'utilisations indiquées dans ce manuel, la fiabilité et la durée de vie des modules CUBLOC™ sera remise en cause (sans que la responsabilité de Lextronic puisse être mise en cause, ni que l'échange du module CUBLOC™ ne puisse être pris en charge au titre de la garantie).

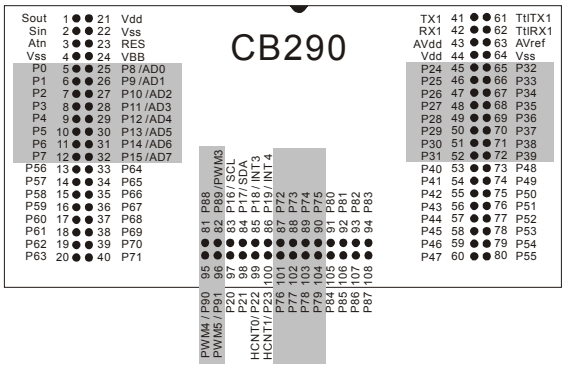
Rappel des brochages des modules « CUBLOC »

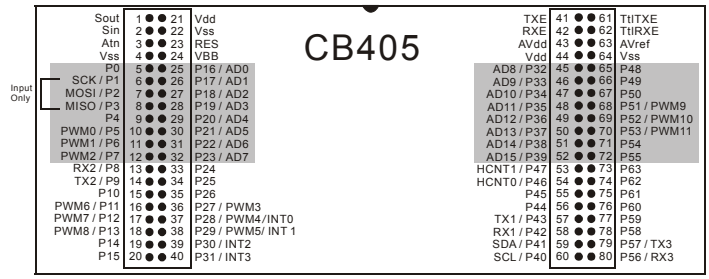


CB220

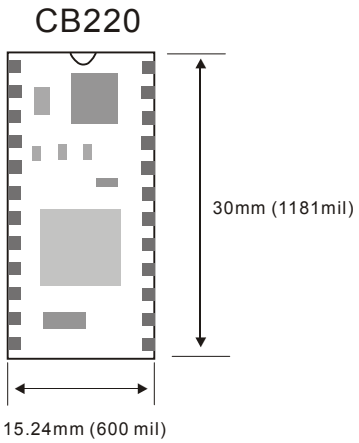


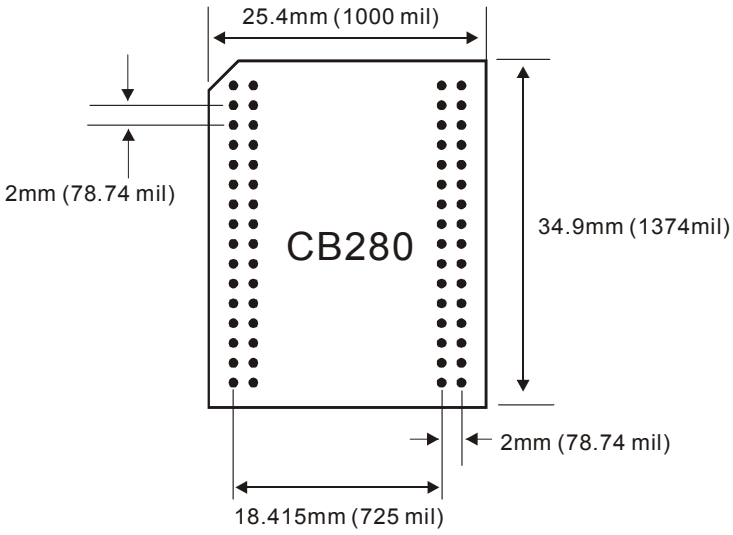
CB280

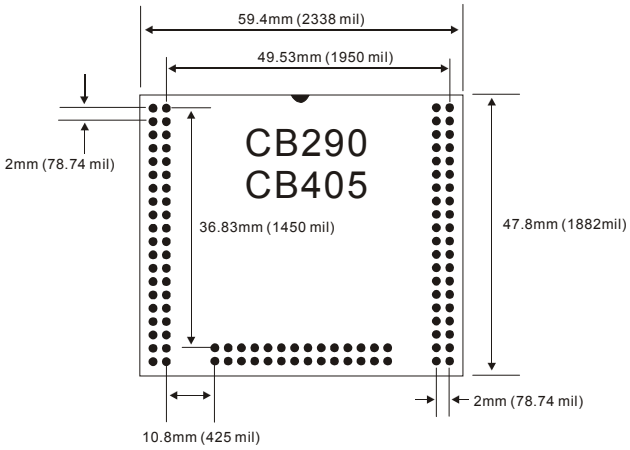


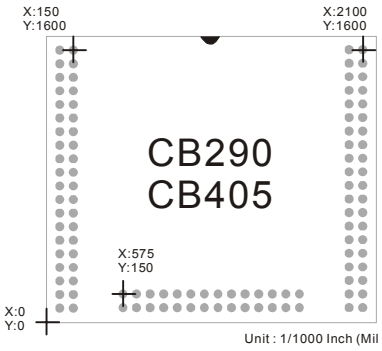


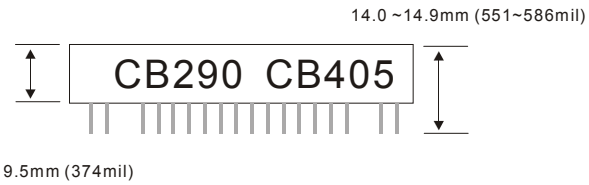
Dimensions des modules « CUBLOC »







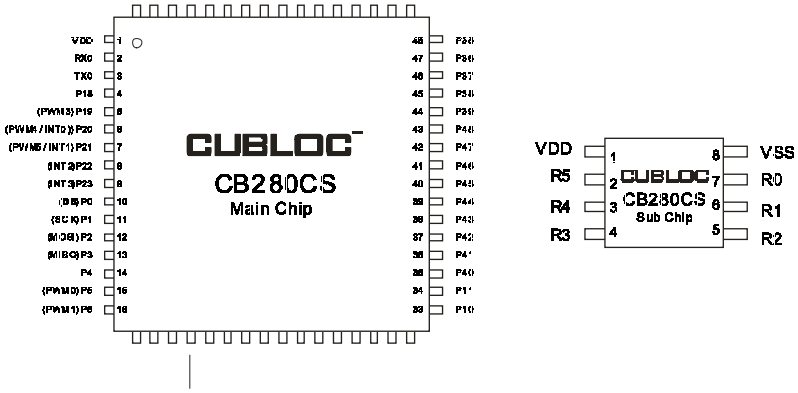




Reportez-vous aux schémas ci-dessus pour l'implantation sur votre circuit imprimé. Les valeurs sont indexées sur les coordonnées 0, 0.

Chipset CUBLOC « CB280CS »

Le Chipset CUBLOC « CB280CS » correspond aux 2 principaux circuits intégrés utilisés dans le module "CB280". Ils sont proposés au détail afin que vous puissiez directement implanter les possibilités du module "CB280" à moindre coût au sein de votre propre carte électronique. Le premier composant est au format QFP 64 broches, le second au format SOIC 8 broches. Si vous n'avez jamais utilisé les modules CUBLOC, il est préférable dans un premier temps d'utiliser le module « CB280 » avant d'envisager ensuite dans un second temps d'intégrer le Chipset Cubloc « CB280CS ». Le Chipset Cubloc « CB280CS » comprend les 2 circuits intégrés.



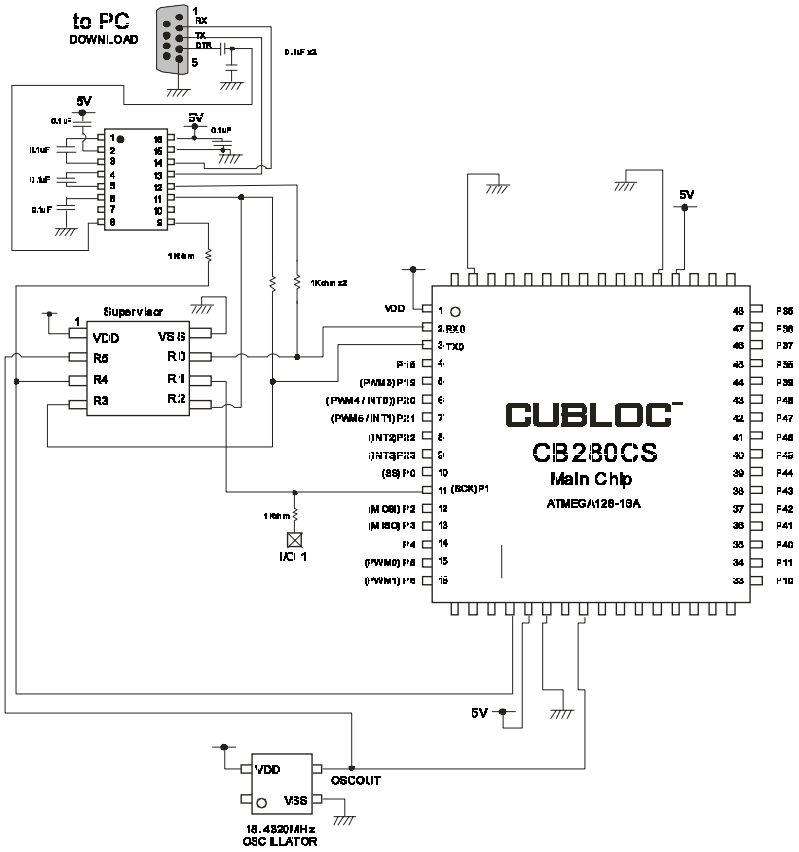
Brochage du circuit intégré pricncipal du Chipset CUBLOC « CB280CS »

Pin #	Port	Fonction	Description.
1	VDD		Alimentation
2	RX0	DOWNLOAD RX	RS232-RX
3	TX0	DOWNLOAD TX	RS232-TX
4	P18		Port E/S
5	P19	PWM3	Port E/S
6	P20	PWM4 / INT0	Port E/S
7	P21	PWM5 / INT1	Port E/S
8	P22	INT2	Port E/S
9	P23	INT3	Port E/S
10	P0	SS	Port E/S
11	P1	SCK	Port E/S
12	P2	MOSI	Port E/S
13	P3	MISO	Port E/S
14	P4		Port E/S
15	P5	PWM0	Port E/S
16	P6	PWM1	Port E/S
17	P7	PWM2	Port E/S
18	P16		Port E/S
19	P17		Port E/S
20	/RESET		Reset (Actif niveau bas)
21	VDD		Alimentation
22	VSS		Masse
23	XTALOUT		Entrée Quartz
24	XTALIN		Sortie Quartz
25	P8	SCL	Port E/S (vers module CuNET)
26	P9	SDA	Port E/S (vers module CuNET)
27	RX1	RS232 CH1 RX	RS232 Channel 1 Rx
28	TX1	RS232 CH1 TX	RS232 Channel 1 Tx
29	P12		Port E/S
30	P13		Port E/S
31	P14	HCOUNT0	Port E/S

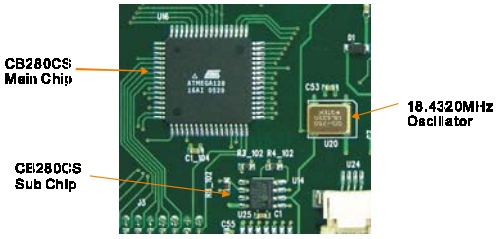
32	P15	HCOUNT1	Port E/S
33	P10		Port E/S
34	P11		Port E/S
35	P40		Port E/S
36	P41		Port E/S
37	P42		Port E/S
38	P43		Port E/S
39	P44		Port E/S
40	P45		Port E/S
41	P46		Port E/S
42	P47		Port E/S
43	P48		Port E/S
44	P39		Port E/S
45	P38		Port E/S
46	P37		Port E/S
47	P36		Port E/S
48	P35		Port E/S
49	P34		Port E/S
50	P33		Port E/S
51	P32		Port E/S
52	VDD		Alimentation
53	VSS		Masse
54	P31	ADC7	Port E/S
55	P30	ADC6	Port E/S
56	P29	ADC5	Port E/S
57	P28	ADC4	Port E/S
58	P27	ADC3	Port E/S
59	P26	ADC2	Port E/S
60	P25	ADC1	Port E/S
61	P24	ADC0	Port E/S
62	AREF		Référence pour ADC
63	VSS		Masse
64	AVDD		Alimentation pour ADC

Consultez l'appendice F pour plus d'information sur les spécifications du « CB280CS ».

Schéma de mise en œuvre du chipset CUBLOC « CB280CS »



Exemple

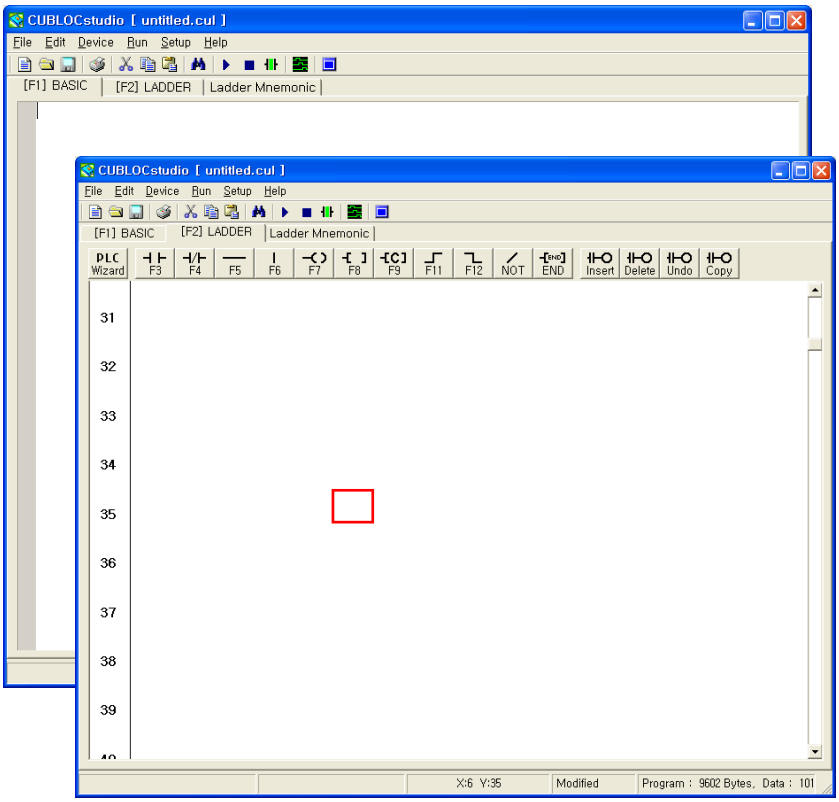


Chapitre 3.

L'éditeur / compilateur CUBLOC STUDIO

Les bases de CUBLOC STUDIO

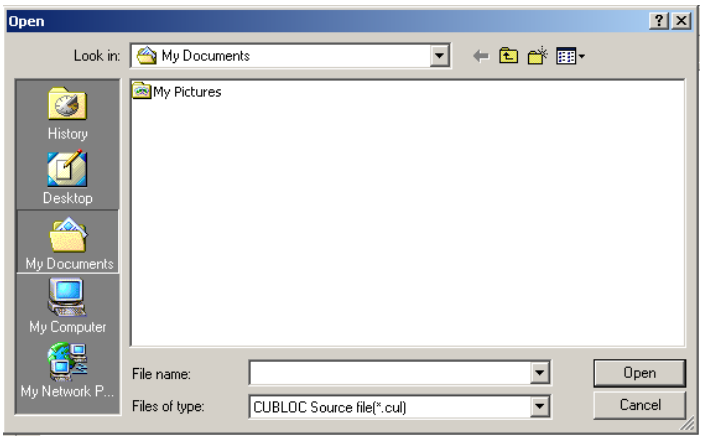
Après avoir installé (voir procédure sur la feuille jointe avec le module CUBLOC™) et l'exécution de CUBLOC STUDIO, vous verrez cet écran.



En premier lieu, CUBLOC STUDIO vous place sous la fenêtre de l'EDITEUR DE TEXTE.

Si vous pressez la touche F2, la fenêtre de l'EDITEUR du LADDER s'affichera alors et si vous pressez la touche F1, la fenêtre de l'EDITEUR DE TEXTE s'affichera à nouveau.

Le fichier source de votre programme sera sauvegardé dans 2 fichiers (portant les extensions « .CUL » et « .CUB ». Si vous devez réaliser des sauvegardes de vos programmes pensez impérativement à copier les 2 fichiers à la fois.



Lorsque vous chargez un de vos programmes, seule l fichier avec l'extension « .CUL » sera visible (les fichiers avec les extensions « .CUB » ne sont pas visibles bien qu'ils soient dans le même répertoire). Lorsque vous chargerez un fichier avec l'extension « .CUL », le CUBLOC STUDIO chargera automatiquement le fichier « .CUB » associé.

Votre code source ne peut être chargé que depuis le PC. Il n'est pas possible de récupérer un programme source depuis un module CUBLOC™ déjà programmé.

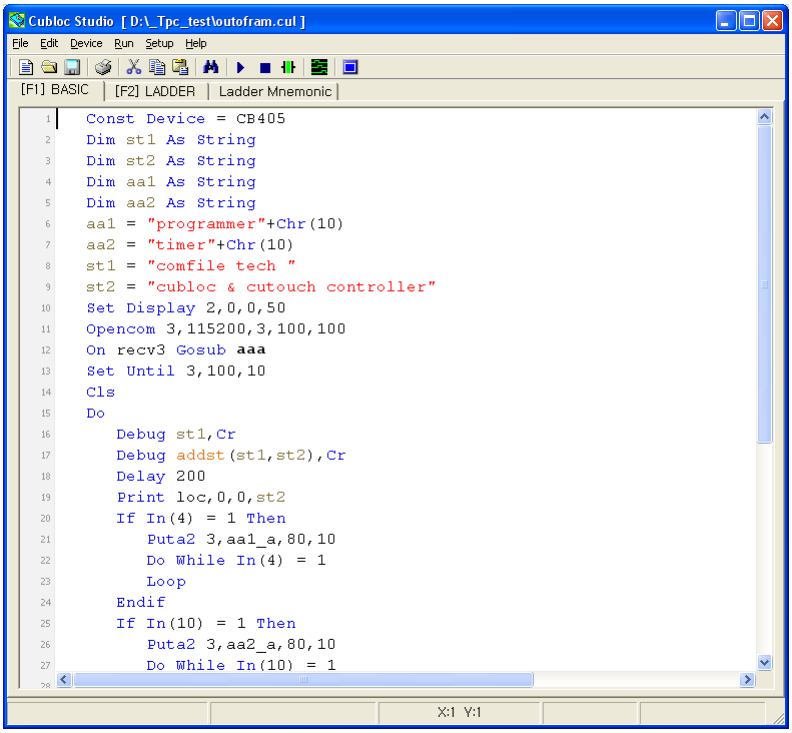
IMPORTANT

Les modules CUBLOC™ disposent d'un système de protection de code avec cryptage qui interdira à un autre utilisateur (comme à vous même) de relire tout ou partie du programme que vous aurez téléchargé dans le module.

Lorsque vous sollicitez le bouton « RUN » (ou la combinaison de touches « CTRL-R »), le processus « Save **->Compile -> Download -> Execute** » est automatiquement réalisé. Les programmes LADDER et BASIC sont compilé dans une seule opération. Si des erreurs sont trouvées, l'écran affichera la partie du programme où l'erreur a été détectée.

Développement en « BASIC »

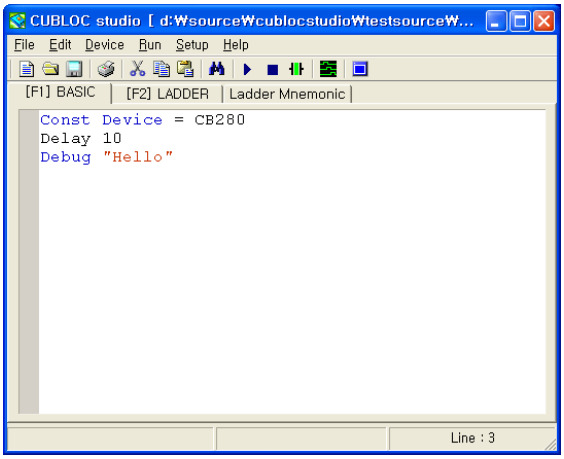
Vous pouvez créer et écrire votre programme BASIC comme l'exemple de l'écran ci-dessous. L'éditeur de TEXT du CUBLOC STUDIO est similaire à la plupart des autres éditeurs de textes.



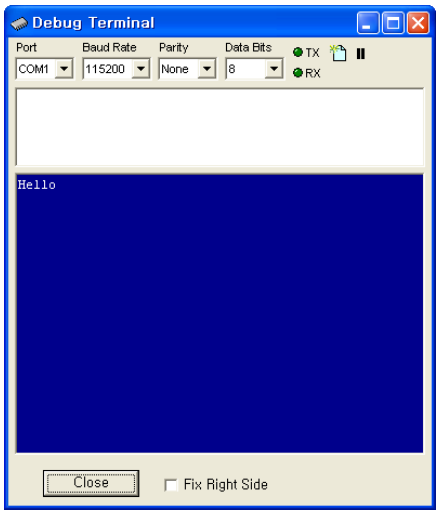
Ce dernier supporte différents raccourcis clavier (voir ci-dessous)à ainsi que la « coloration syntaxique » permettant d'obtenir une plus grande visibilité du code.

Raccorcis	Explications
CTRL-Z	UNDO
CTRL-O	OPEN (Ouvrir fichier)
CTRL-S	SAVE (Sauvegarder fichier)
CTRL-C	COPY (Copier)
CTRL-X	CUT (Couper)
CTRL-V	PASTE (Coller)
CTRL-F	FIND (Trouver)
CTRL-HOME	Go to the very beginning (Aller au tout début)
CTRL-END	Go to the very end (Aller à la fin)
CTRL-Y	REDO

Débugage



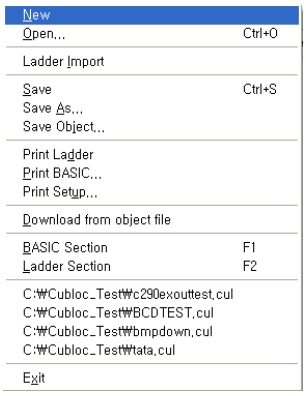
Afin de faciliter la mise au point de vos applications, vous avez la possibilité d'activer des fonctions de DEBUG (via des commandes spéciales à placer au sein de votre programme BASIC) afin de pouvoir afficher l'état de variables ou d'informations diverses sur une fenêtre spéciale du PC à un ou plusieurs moments particuliers du déroulement de votre programme.



Attention, il n'est cependant pas possible d'utiliser à la fois le mode DEBUG du BASIC et la fonction de monitoring du langage LADDER. Si vous désirez utiliser la fonction de monitoring du LADDER, vous devrez enlever ou désactiver les commande de DEBUG que vous aurez placez dans votre programme BASIC (il vous suffit de faire passer ces commandes comme des commentaires en plaçant une apostrophe devant ou d'utiliser la commande "Set Debug Off", qui désactivera totalement les possibilités de DEBUG du BASIC). De plus il est déconseillé de « remonter » des informations via le mode débug en boucle de façon répétée, sans quoi vous pourrez avoir des difficultés à transférer votre programme au sein du module CUBLOC™ (la ligne série étant alors occupée) – En ce cas, le module CUBLOC™ pourra au moment de la programmation vous signaler un problème de communication série (il vous suffira alors de recommencer l'opération de téléchargement plusieurs fois pour arriver à « tomber » dans un intervalle de temps ou la ligne série est inoccupée).

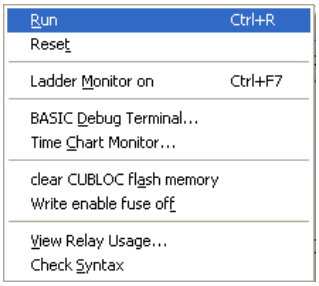
Détail des menus de la fenêtre de développement en « BASIC »

Menu « **File** »



	Explications
New	Permet de créer un nouveau fichier
Open	Ouvre un fichier.
Ladder Import	Importe une partie de LADDER dans le programme du CUBLOC™.
Save	Sauvegarde le programme courrant.
Save As	Sauvegarde le programme courrant sous une nom différent.
Save Object	Sauvegarde le programme courrant sous la forme d'un fichier binaire « objet ». Ceci permet de protéger votre code source afin qu'il ne puisse pas être analysé par une autre personne. Vous pourrez ensuite utiliser le menu "Download from Object File" pour télécharger un programme initialement sauvegardé en fichier « object » dans un module CUBLOC™. Vous pouvez créer des fichiers « object » capables d'être téléchargés via Internet avec les utilitaires CuMAX ou CuMAX Server.
Print Ladder	Imprimer le programme LADDER uniquement.
Print Basic	Imprimer le programme BASIC uniquement.
Print Setup	Réglage de l'impression pour la partie LADDER uniquement.
Download from Object file	Télécharger un fichier « Object » dans une module CUBLOC™.
Basic Section	Basculer vers la fenêtre de l'éditeur BASIC (Utilisez aussi F1).
Ladder Section	Basculer vers la fenêtre de l'éditeur LADDER (Utilisez aussi F2).
Last 4 Files Edited	Voir les 4 derniers fichiers utilisés.
Exit	Sortir du programme CUBLOC Studio

Menu « **Run** »



Menu	*Explications*
Run	Compile le programme Basic et Ladder, puis les télécharge dans le module CUBLOC™ (à condition qu'aucune erreur n'est été détectée). Une fois téléchargé, le programme s'éxécute immédiatement. Vous pouvez désactiver le démarrage automatique du programme depuis le menu **Setup -> Studio Option**.
Reset	Reset le module CUBLOC™.
Ladder Monitor on	Active le Monitoring du LADDER.
BASIC Debug Terminal	Ouvre la fenêtre du terminal de Debug du BASIC. Cette fenêtre s'ouvre également automatiquement lorsqu'une commande DEBUG est détectée lors de l'exécution du programme.
Time Chart Monitor	Voir la fenêtre moniteur « Time Chart ».
Clear CUBLOC's Flash Memory	Efface la mémoire Flash du CUBLOC™.
Write enable fuse off	Cette option permet de désactiver la fonction de téléchargement du module CUBLOC afin de protéger son programme en vérouillant l'accès de sa mémoire Flash (conseillé si ce dernier est exploité en environnement perturbé). Une fois sélectionnée, plus aucun programme ne pourra être téléchargé dans le CUBLOC (La seule façon pour pouvoir télécharger à nouveau un programme est de reprogrammer le Firmware du CUBLOC)
View Register Usage	(Après compilation) Visulalise les Registres utilisés dans le LADDER.
Check Syntax	Permet de vérifier la syntaxe.

Menu « **Edit** »

Menu	*Explications*
Find / replace / Go Line / Undo / Redo / Copy / Cut / Paste / Sellect All	Vous retrouvez ici toutes les fonctions usuelles (copier/coler/ etc…) propres à tout éditeur de texte.

Menu « **Device** »

Menu	*Explications*
CB220 / CB280 / CB290 / CB405 / CT1720	Permet de modifier automatiquement l'entête de déclaration du programme BASIC en fonction du type de CUBLOC™ utilisé.

Menu « **Setup** »

Menu	*Explications*
PLC Setup Wizard	Interface d'aide à la génération automatique des codes d'initialisation du programme BASIC en cas d'utilisation conjoint avec le LADDER.
PC Interface Setup	Configuration du N° de pour COM RS232 (COM1 à COM4) dédié au téléchargement de votre programme.
Editor Environment Setup	Configuration de l'environnement de l'Editeur de texte du BASIC.
Environnement Options	Options de paramétrage d'option diverses du CUBLOC STUDIO.
Use Korean Menu	Permet l'utilisation de menu en Coréen.
Firmware Download	Permet de remettre à jour le Firmware du module CUBLOC™.

Le menu « **Help** »

Menu	*Explications*
Upgrade History / About CUBLOC STUDIO	Permet d'obtenir des informations sur la version du CUBLOC STUDIO et sur l'historique de ses mises à jour.

*** Lisez impérativement cette page, même si vous ne comptez utiliser que le langage LADDER.**

Chapitre 4.
Le langage BASIC des Modules CUBLOC™

Const Device

IMPORTANT

Vous devez déclarer au sein du programme BASIC le modèle de CUBLOC™ que vous allez utiliser (que votre application soit développée en BASIC ou en LADDER).

L'exemple ci-dessous montre comment déclarer que vous travaillez avec un module CUBLOC™ « CB220 ».

```
CONST DEVICE = CB220          ' Use CB220.
```

Cette déclaration doit être faite à la première ligne de votre programme BASIC. Si cette déclaration est absente, le CUBLOC™ choisira le modèle « CB220 » par défaut.

Autres exemples de déclarations :

```
CONST DEVICE = CT1720         ' Use CT1720.
CONST DEVICE = CB280          ' Use CB280.
```

Caractéristiques principales du BASIC des CUBLOC™

Interfaçage avec un PC depuis le port RS232C

Le BASIC du CUBLOC™ utilise le port RS232 pour s'interfacer avec un PC. Il vous est également possible (en option) d'utiliser un module additionnel XPORT™ afin de pouvoir effectuer des téléchargements/monitoring depuis Internet.

Le BASIC du CUBLOC™ supporte les fonctions et sous-routines.

Au même titre qu'avec le langage « C », l'utilisateur du langage BASIC du CUBLOC™ sera en mesure de créer des sous-routines et fonctions afin de limiter la complexité de son programme. En ayant la possibilité de pouvoir réutiliser des sous-routines et des fonctions que vous avez déjà écrites (par simple copier/coller) dans vos nouveaux programmes, vous pourrez accélérer vos temps de développements. Vous trouverez ci-dessous un petit exemple de fonction que vous pourrez réaliser.

```
Function  SUM( A As Integer,  B As Integer) As Integer
        Dim RES As Integer
        RES = A + B
        SUM = RES
End Function
```

La gestion de fonctions mathématiques complexes est pris en compte.

```
A = SIN(A) * LOG(3.0) + 100 + ( B * 3.14 + 100) / 210
```

Les CUBLOC™ pourront dont assurer des calculs complexes.

Des traitements conditionnés pourront être effectués au moyen de fonctions telles que If, While, etc…

```
IF ((A + 1) = 100) THEN GOTO ABC

IF ((A + 1) = 100) AND (B / 100 = 20) OR C = 3 THEN GOTO ABC
```

Les tableaux Multi-dimensions sont supportés.

Les CUBLOC™ supportent les tableaux multi-dimensions (jusqu'à 8 dimensions et une dimension avec des champs de type « caractère »).

```
DIM A(100,10,20) AS BYTE
```

Les communication RS232 « Hardware » sont supportées

Les CUBLOC™ disposent d'un buffer de communication afin de pouvoir gérer les liaisons RS232 sans interférence avec le déroulement de votre programme.

Les déclarations conditionnées sont supportées

Le BASIC des CUBLOC™ supporte les instructions de type : SELECT CASE et DO…LOOP

Librairie graphique pour gestion de LCD spécifiques

Les CUBLOC™ disposent d'une libraire graphique permettant la gestion d'afficheur LCD graphiques spéciaux de type GHLCD. Tracer des rectangles, des lignes, des cercles et utiliser des commandes graphiques s'effectuera en quelques lignes de codes.

De nombreux protocoles de communication sont supportés

- Pour la gestion de périphériques « CuNET » spéciaux et dédiés aux modules CUBLOC™ tels que des afficheurs LCD, des afficheurs 7 segments à Leds…
- RS232
- MODBUS : HMI et protocole Touch screen
- I2C™ : A l'aide des commandes du type I2CREAD et I2CWRITE
- SPI™ : A l'aide des commandes du type SHIFTIN, SHIFTOUT
- PAD: pour la gestion de clavier

Le BASIC du CUBLOC™ peut être comparé sur certains points au « C »

Supporte les déclarations #define, #include
Supporte #if..#ifdef..#endif
Commandes de type Incr, Decr
Supporte les pointeurs (PEEK, POKE et MEMADR)
Tableau de type « String » (1 dimension)

Exemple simple de programme BASIC

Vous trouverez ci-dessous un petit exemple de programme BASIC avec l'instruction Do…Loop.

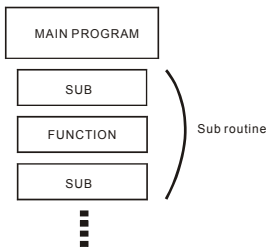
```
Dim A As Byte
Do
        Byteout 0, A
        A=A+1
Loop
```

Ce programme « sort » la valeur de la variable A (qui s'incrémente automatiquement) sur le port P0-P7. Le programme suivant utilise le recours à une « sous-routine » pour réaliser la même chose:

```
Dim A As Byte
Do
        Byteout 0, A
        A=ADD_VALUE(A)
Loop
End

Function ADD_VALUE(B As Byte) As Byte
        ADD_VALUE = B + 1
End Function
```

Même si ce programme peut vous sembler plus compliqué que le précédent, on aura pu mettre en valeur que l'instruction A=A+1 a été transformée en une « fonction » qui pourra dès lors être appelée depuis n'importe quelle autre partie du programme. Dans le cas d'un traitement plus complexe qu'un simple A=A+1, vous pourrez réaliser des actions très complexes au sein d'une fonction, laquelle pourra alors s'apparenter à une nouvelle instruction. Une fonction pourra par exemple récupérer les valeurs d'un mini-module sonar externe au CUBLOC™ afin de déterminer la distance qui le sépare d'un obstacle. Cette fonction pourra alors s'apparenter à elle seule à une nouvelle instruction permettant de réaliser cette action, instruction que vous pourrez appeler à tout moment dans votre programme en simplifiant ainsi sa composition.

Ces fonctions devront être écrites à la fin du programme principal.



Sous-routines « Sub » et « Fonction »

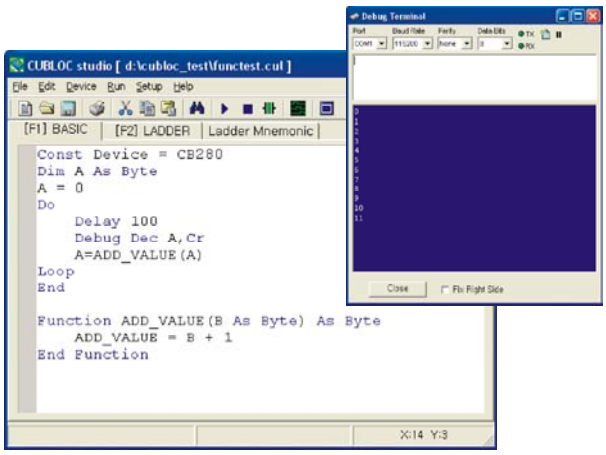
A noter que pour la génération de sous-routines, il vous est possible d'utiliser les instructions « Sub » ou « Fonction ». « Sub » ne vous permettra pas de retourner de valeur alors que « Fonction » vous permettra de retourner une valeur en sortie de la sous-routine.

```
 Sub SubName (Param1 As DataType [,ParamX As DataType][,…])
          Statements
          [Exit sub]       ' Sort de la sous-routine
End Sub

Function FunctionName (Param1 As DataType [,…])[As ReturnDataType]
          Statements
         [Exit Function]  ' Sort de la sous-routine
End Function
```

Pour retourner une valeur au programme principal en sortie d'une « Fonction », il vous suffit de stocker cette valeur dans une variable protant le même nom que la « Fonction » (voir exemple ci-dessous).

```
Function ADD_VALUE(B As Byte) As Byte
        ADD_VALUE = B + 1       ' Retourne B+1.
End Function
```

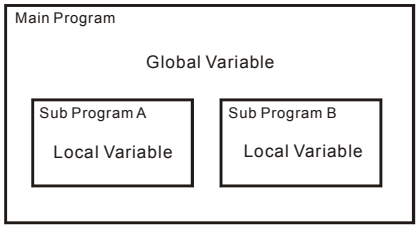
Vous pouvez tester l'utilisation des « Fonctions » en saisissant le programme ci-dessous.



Variables « globales » et « locales »

Lorsque vous déclarez des variables à l'intérieur de sous-programmes de type « Sub » ou « Fonction », ces variables sont dites de type « locale ». Ces à dire que ces dernières sont crées temporairement en mémoire vive, puis effacées lors de la sortie du sous-programme. L'utilisation de ces variables ne peut alors de faire que dans le sous-programme.

A l'opposé, les variables utilisées dans le programme principal sont de type « global » et peuvent être utilisées dans n'importe quelle partie de votre code.




```
Dim  A  As  Integer              ' Déclare A comme une variable de type « global »
LOOP1:
        A = A + 1
        Debug  Dp(A),CR      ' Affiche A dans le fenêtre de débug
        DELAYTIME            ' Appel la sous-routine DELAYTIME
        Goto LOOP1
        End                  ' Fin du programme principal

Sub  DELAYTIME()
        Dim  K  As  Integer  ' Déclare K comme une variable de type « Local »
        For K=0 To 10
        Next
End Sub
```

Dans le programme ci-dessus, la variable A (de type global) pourra être utilisée dans TOUT le programme alors que la variable K (de type local) ne pourra être utilisée que dans la sous-routine « DELAYTIME() ».

A noter que les variable de type « tableau » ne pourront pas être utilisé comme des variables de type « local ». **Les variables de type « tableau » doient être uniquement déclarée comme des variables de type « global ».**

Appels aux sous-routines

Lorsque vous avez créé des sous-routines, il vous sera possible de les utiliser comme s'il s'agissait d'une nouvelle instruction. Pour les sous-routines de type « Sub », vous n'avez pas besoin d'utiliser de parenthèses entre les paramètres. Pour utiliser des paramètres multiples, utilisez des virgules pour les séparer (voir exemple ci-dessous).

```
DELAYTIME 100                ' Appel de la sous-routine
End

Sub  DELAYTIME(DL As Integer)
        Dim  K  As  Integer     ' Déclare K comme une variable locale
        For K=0 To DL
        Next
End Sub
```

Lorsque vous appelez une « Fonction », vous devez utiliser des parenthèses entre les paramètres (les parenthèses doivent être utilisée même s'il n'y a pas de paramètres).

```
Dim K As Integer
K = SUMAB(100,200)          '  Appel de la sous-routine et stock valeur retournée dans K
Debug Dec K,cr
End

Function  SUMAB(A AS INTEGER, B AS INTEGER) As Integer
        SUMAB = A + B
End Function
```

Emplacement des sous-routines

Les « sous-routines » de type « Sub » et « Fonction » doivent être placées APRES le programme principal. Pour ce faire, il vous faut placer l'instruction « End » à la fin de votre programme principal, puis les sous-routine après. L'instruction « End » à la fin de votre programme principale ne devra être utilisée que si vous disposez de sous-routine de type « Sub » ou « Fonction ».

```
Dim  A  As  Integer
LOOP1:
        A = A + 1
        Debug  DP(A),CR
        DELAYTIME
        Goto Loop1

        End             '  Fin du programme principal

Sub  DELAYTIME()
        Dim  K  As  Integer
        For K=0 To 10
        Next
End Sub
```

Si par contre vous utilisez des sous-routines « conventionnelles » appelée à l'aide de l'instruction « Gosub », il vous faudra placer ces dernières à l'intérieur de votre programme principal comme indiqué ci-dessous :

```
Dim A As Integer
  :
  :
Gosub ABC
  :
ABC:
  :
End
```

```
Sub  DEF(B as Byte)
  :
  :
End Sub
```

```
Function  GHI(C as Byte)
  :
  :
End Function
```

* Attention l'utilisation de l'instruction « End » est différente suivant que vous êtes en BASIC ou en ladder. L'instruction « End » en Ladder signifie la fin « définitive » de votre programme Ladder.

Renvoi de paramètres et valeurs depuis les « sous-routines »

Les « fonctions » peuvent utiliser presque tous les types de données pour le renvoi de paramètres et de valeurs.

```
Dim A(10) As Integer
Function ABC(A AS Single) as Single                    ' Retourne une valeur de type « Single »
End Function

Function ABC(A AS String * 12) as String *12           ' Retourne une « Chaîne de caractères »
End Function

Function ABC(A AS long)                                ' Valeur de type « Long » comme paramètre
End Function                                  ' Si la valeur de retour n'est pas déclarée,
                                              ' une valeur de type Long sera utilisée.
```

La variables de type « tableau » ne peuvent pas par contre être utilisés pour le retour de « Fonction ».

```
Function ARRAYUSING(A(10) AS Integer)  ' Les tableaux ne peuvent pas être utilisés comme
                                       ' paramètres.
End Function
```

Mais vous pouvez utiliser un seul élément d'un tableau comme paramètre.

```
Dim b(10) as integer
K = ARRAYUSING(b(10))       ' Utilise le 10 ème élément du tableau comme paramètre.

Function ARRAYUSING(A AS Integer) as integer
End Function
```

Tous les paramètres des « fonctions » sont gérés commes des « valeurs de sortie ». C'est à dire que ces valeurs sont uniquement gérées comme variable « de référence ». Même si la valeur du paramètre change, ceci n'affectera pas l'état actuel de la variable utilisée comme paramètre (voir l'exemple ci-dessous).

```
Dim  A  As  Integer
Dim  K  As  Integer
A = 100
K = ADDATEN(A)
Debug Dec? A, Dec? K,CR              ' A est égale à 100 et K est égal à 110
End

Sub ADDATEN(V As Integer)
        V = V + 10                   ' A ne sera pas modifié lorsque V est modifié.
        ADDATEN = V
End Sub
```

Trop de caractères sur une seule ligne ?

Si la ligne de votre instruction est trop grande pour apparaître en totalité à l'écran, vous pouvez utiliser le caractère « underscore » (_) pour indiquer de continuer à la ligne suivante.

```
ST = "COMFILE TECHNOLOGY"
```

Revient à écrire

```
ST = "COMFILE _
        TECHNOLOGY"
```

Affichage de commentaires dans votre programme

Utilisez le caractère (') pour indiquer le début d'un commentaire. Ces commentaires seront ignorés lors de la compilation du programme et ne monopoliserons aucune ressource mémoire.

```
ADD_VALUE = B + 1       ' Ajouter 1 à B (ceci est un commentaire)
```

Sous-routines imbriquées

Les CUBLOC™ supportent le recours à des sous-routines imbriquées.

```
A=FLOOR(SQR(F))          ' Réalise Floor() de SQR(F).
```

Gestion des séparations

La séparation par un caractère (:) n'est pas reconnu par les CUBLOC™.

```
A=1: B=1 : C=1   ' Non correct.

A=1              ' Correct.
B=1
C=1
```

Variables

Il y a 5 types de variables pouvant être gérées par le BASIC des CUBLOC™.

BYTE	8 bits nombre positif 0~255
INTEGER	16 bits nombre positif 0~65535
LONG	32 bits nombre positif/négatif -2147483648 ~ +2147483647
SINGLE	32 bits Nombre à virgule flottante, -3.402823E+38 ~ 3.402823E+38
STRING	String (chaîne), 0 à 127 octets

Une variable « Byte » est un nombre positif sur 8 bits pouvant être compris entre 0 et 255.

Une variable « Integer » est un nombre positif sur 16 bits pouvant être compris entre 0 et 65535.

Une variable « Long » est un nombre positif/négatif sur 32 bits pouvant être compris entre -2,147,483,648 to 2,147,483,647.

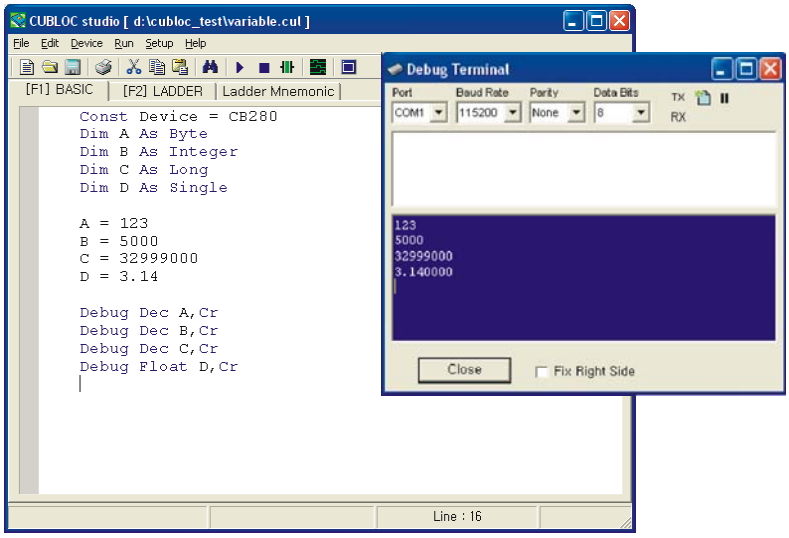
Une variable « Single » est un nombre positif/négatif sur 32 bits à virgule flottante pouvant être compris entre $-3.402823x10^{38}$ to 3.402823×10^{38}.

*Pour mémoriser des nombres négatifs, utilisez des variables de type LONG ou SINGLE.

Utilisez la commande « DIM » pour déclarer les variables en début de programme comme indiqué ci-dessous.

```
DIM  A  AS  BYTE                    ‘ Declare A comme en BYTE.
DIM  B  AS  INTEGER, C AS BYTE      ‘ La virgule ne doit pas être utilisée.
DIM  ST1  AS STRING * 12            ‘ Déclaration de la taille de la chaîne.
DIM  ST2  AS STRING                 ‘ Déclaration d’une chaîne avec taille par défaut (64 octets).
DIM  AR(10) AS BYTE                 ‘ Déclaration d’un tableau d’octets.
DIM  AK(10,20) AS INTEGER           ‘ Déclaration d’un tableau 2 D.
DIM  ST(10) AS STRING*10            ‘ Déclaration d’un tableau de chaînes
```

Exemple de programme de test mettant en œuvre la déclaration des variables.



Instruction « VAR »

L'instruction « VAR » pourra être utilisée pour la déclaration des variables à l'image de l'instruction « DIM » en utilisant la syntaxe ci-dessous.

```
A     Var   Byte          ' Déclare A comme BYTE.
ST1   Var   String * 12   ' Déclare ST1 comme une chaîne de of 12 octets.
AR    Var   Byte(10)      ' Déclare AR comme un tableau de 10 octets.
AK    Var   Integer(10,20)' Déclare AK comme un tableau à 2 dimensions (Interger)
ST    Var   String *12 (10)' Déclare un tableau de type chaîne
```

Chaînes

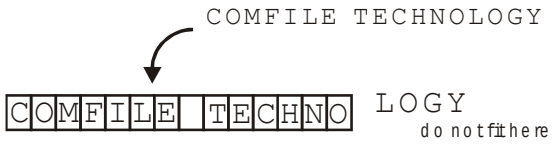
Une chaîne peut être dotée d'une taille maximale de 127 octets. Si la taille de la chaîne n'est pas déclarée, celle-ci sera d'office configurée avec 64 octets.

```
DIM  ST  AS  STRING * 14      ' Cette chaîne aurra une taille max. de 14 octets
DIM  ST2  AS  STRING          ' Cette chaîne aurra une taille max. de 64 octets
```

Il est important de noter que lorsque vous sélectionnerez une taille maximale de 14 octets pour une chaîne, un autre octet est alloué par le processeur du module CUBLOC™ pour stocker une chaîne NULL.

Si vous essayez de stocker une variable de taille supérieur à ce que la chaîne peu contenir, vous ne pourrez utiliser que la place déclarée. Ainsi dans l'exemple ci-dessous, les 4 derniers caractères ne seront pas mémoriés.

```
DIM  ST  AS  STRING * 14
ST = "COMFILE TECHNOLOGY"     '  "LOGY" is not stored
```



Dans le BASIC du CUBLOC™, le caractère (") doit être utilisé pour définir une chaîne par contre, certains caractères ne pourront pas être utilisé à l'intérieur d'une chaîne.

```
ST = "COMFILE " TECHNOLOGY"     ' (") Ce caractère ne peut pas être utilisé à dans une chaîne.
ST = "COMFILE ' TECHNOLOGY"     ' (') Ce caractère ne peut pas être utilisé à dans une chaîne.
ST = "COMFILE , TECHNOLOGY"     ' (,) Ce caractère ne peut pas être utilisé à dans une chaîne.
```

Vous pouvez utiliser la commande CHR(&H22) pour définir le caractère (") et CHR(&H27) pour définir le caractère (') ou encore la commande CHR(&H2C) pour définir le caractère (,).

```
PRINT CHR(&H22),"COMFILE TECHNOLOGY",CHR(&H22)     ' (")
PRINT CHR(&H27),"COMFILE TECHNOLOGY",CHR(&H27)     ' (') Apostrophe
```

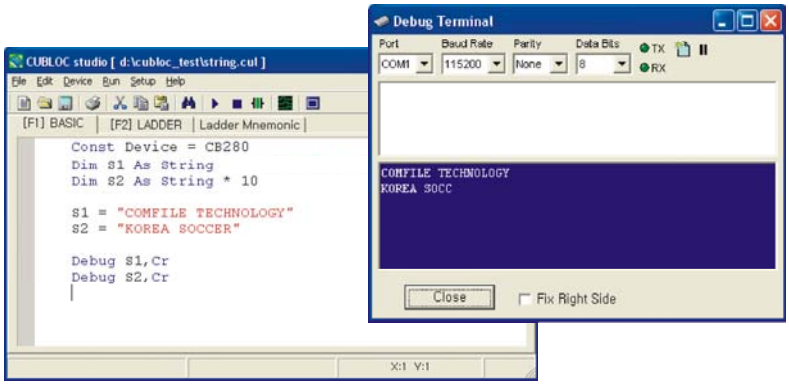
Pour associer de multiples chaînes, vous pouvez utiliser des virgules.

```
PRINT “ABC”,”DEF”,”GHI”  ‘ Same as PRINT “ABCDEFGHI”.
```

Utilisez CR pour le retour à la ligne.

```
PRINT “California”,CR        ‘ Affiche California et va à la ligne suivante.
```

Voir exemples sur le programme de test ci-dessous :

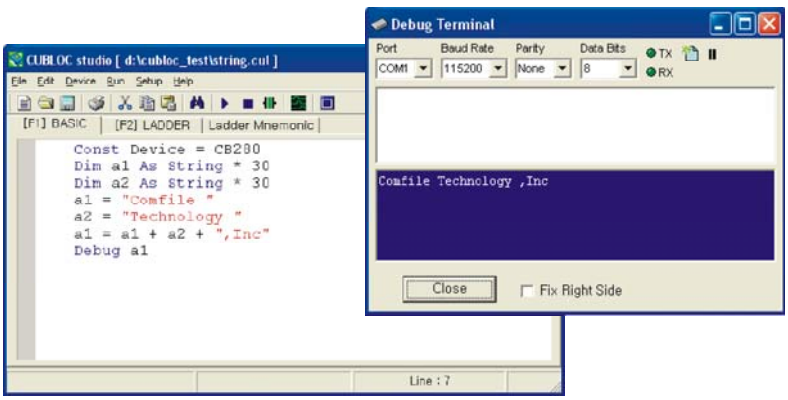


Fusionner de multiples chaînes

Pour fusionner de multiples chaînes, utilisez le caractère & comme indiqué ci-dessous :

```
Dim a1 As String * 30
Dim a2 As String * 30
a1 = "Comfile "
a2 = "Technology "
a1 = a1 & a2 & ",Inc"
Debug a1,cr
```

Le programme ci-dessous affichera “Comfile Technology, Inc” sur la fenêtre de débug.



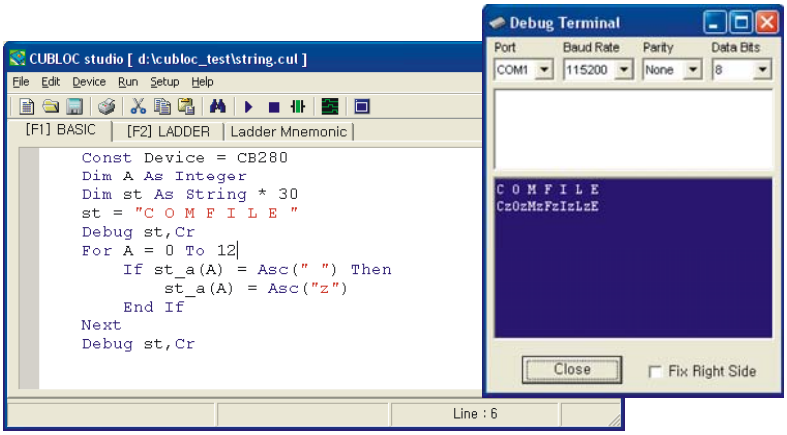
Comment accéder individuellement aux caractères d'une chaîne

Pour ce faire, vous pouvez utiliser les chaînes comme un « tableau » en ajoutant simplement "_A" à la variable de la chaîne comme indiqué dans l'exemple ci-dessous:

```
DIM ST1 AS STRING * 12   ' Le champ ST1_A est créé en même temps.
ST1 = "123"
ST1_A(0) = ASC("A")          ' Mémorise A dans le premier caractère de ST1.
```

Lorsque vous déclarez « Dim ST1 AS STRING * 12 », un tableau de type « ST1_A(12) » est alors automatiquement créé par le « système d'exploitation » du CUBLOC™. Les chaînes et les tableaux utilisent en fait la même ressource mémoire et vous y accédez de la même façon.

L'exemple ci-dessous montre comment remplacer les caractères « vides » par des caractères « z ».



Attention, toutefois les tableau de type « chaîne » ne peuvent pas utiliser cette possibilité.

```
Dim st(10) As String * 3
```

A propos de l'espace mémoire dédié aux variables

En ce qui concerne les modules CB220 et CB280, vous disposez de 2 K de mémoire de donnée. Vous ne pourrez pas utiliser toute la mémoire pour vos variables (la mémoire est en effet également utilisée pour d'autres périphériques tels que les AFFICHEURS optionnels à commandes séries ou I2C™ ainsi que le buffer RS232). Vous disposez en fait d'environ 1800 octets pour vos variables et les périphériques et de 80 octets seront dédiés aux commandes de DEBUG. Ainsi lorsque vous utilisez les commandes SET DISPLAY ou OPENCOM, ou que vous exploitez des sous-routines de type « Sub » ou « Fonction » la mémoire restante pour vos variables sera donc d'autant plus réduite.

Initialisation de la mémoire

La mémoire de données des CUBLOC™ n'est pas initialisée à la mise sous tension. Vous devrez donc penser à mettre vos variables à zéro ou utiliser la commande RAMCLEAR pour effacer toute la mémoire.

```
Ramclear
```

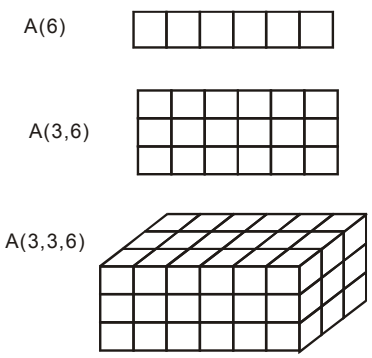
En absence d'exécution de cette instruction, la mémoire RAM contiendra des valeurs indéfinies après la mise sous-tension.

Si vous utilisez une sauvegarde par pile (pour les CUBLOC disposant de cette possibilité), les variables conserveront leur valeur après une remise sous tension du module principal.

Tableaux (matrices)

Le BASIC des CUBLOC™ peut gérer des tableaux (matrices) jusqu'à 8 dimensions (de 65535 éléments max.)

```
DIM    A(20)  AS  BYTE              ‘ Déclare tableau à 20 éléments
DIM    B(200)  AS   INTEGER        ‘ Déclare tableau avec des variables de type “Integer”
DIM    C(200)  AS   LONG           ‘ Déclare tableau avec des variables de type “Long”
DIM    D(20,10)  AS   SINGLE       ‘ Déclare tableau 2 D avec variables de type “Single”
DIM    ST1(10) AS STRING * 12      ‘ Déclare tableau avec variables de “String” (chaîne)
```



Prenez garde à calculer (suivant la nature des variables) l'occupation mémoire en RAM lorsque vous utilisez des tableaux multi-dimensions.

```
‘ 13 * 10 = 130 octets de mémoire de données !
            (les variables Chaînes nécessitent 1 octets en plus)

DIM    ST1(10) AS STRING * 12

 ‘ 4*10 * 20 = 800 octets de mémoire de données !

DIM    D(20,10)  AS   SINGLE
```

Gestion des bits et des Octets

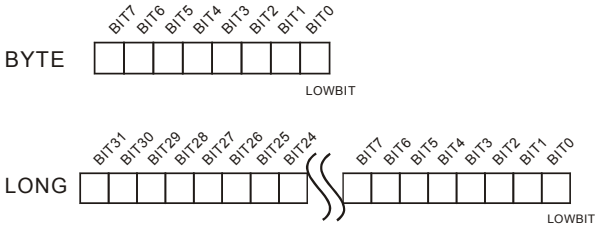
Les variables de type « bit » et « octet » peuvent être individuellement adressé à l'aide des commandes ci-dessous.

```
DIM  A  AS INTEGER
DIM  B  AS BYTE
A.LOWBYTE = &H12        ' Mémorise &H12 dans les octets de poid faible de A
```

Bit

LOWBIT	Variable -> bit 0
BIT0~31	Variable -> bit 0 à 31

```
A.BIT2 = 1     ' Place le bit 2 de la variable A à 1.
```

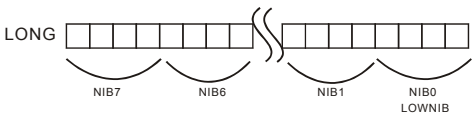


Nibble

Cette commande permet de traiter 4 bits à la fois afin de disposer d'une plus grande fléxibilité pour manipuler les données.

LOWNIB	Variable -> NIBBLE 0
NIB0~7	Variable -> NIBBLE 0~7

```
A.NIB3 = 7     ' Mémorise 7 en Nibble 3 de A
```

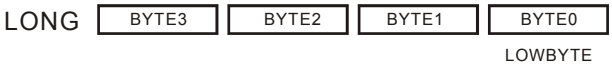


Byte

Pour déterminer un octet particulier d'une variable, les noms suivant peuvent être utilisés. (L'octet A est sur 8 bits)

LOWBYTE, BYTE0	Octet 0 de la Variable
BYTE1	Octet 1 de la Variable
BYTE2	Octet 2 de la Variable
BYTE3	Octet 3 de la Variable

A.BYTE1 = &HAB ‘Mémorise &hAB dans l'octet 1 de la variable A

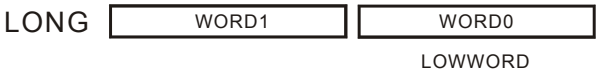


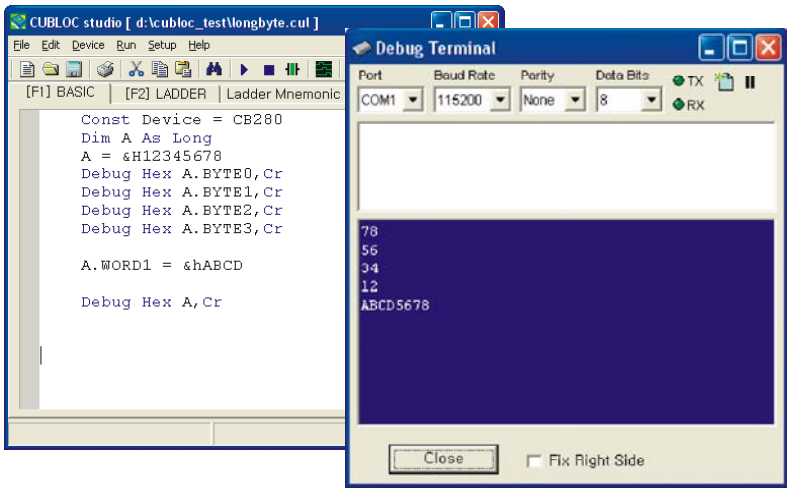
Word

Pour déterminer une donnée de type « word » particulière d'une variable, les noms suivant peuvent être utilisés: (La donnée de type Word est sur 16 bits)

LOWWORD, WORD0	Donnée Word 0 de la variable
WORD1	Donnée Word 1 de la variable

A.WORD1 = &HABCD ‘Mémorise &hABCD dans la donnée “word” 1 de la variable A





Astuce : Vous désirez accedez à 5 bits d'une variable ?
Essayez **NewVariable = Variable and 0x1F**
Ceci aura pour effet de masquer les 5 derniers bits de la variable

Constantes

Les constantes peuvent être utilisées pour déclarer une valeur fixe au début de votre programme. En utilisant celles-ci, votre programme sera plus « lisible » et plaus simple à comprendre. Utilisez la commande CONST pour déclarer vos constantes dans le CUBLOC™.

```
CONST  PI  AS  SINGLE = 3.14159
CONST  WRTTIME  AS  BYTE  = 10
CONST  MSG1 AS STRING = “ACCESS PORT”
```

Lorsque le type de contante n’est pas indiqué, le compilateur BASIC du CBLOC™ essayera de déterminer le type de donnée le plus approprié:

```
CONST  PI = 3.14159              ‘ Déclare PI comme un “SINGLE”
CONST  WRTTIME  = 10             ‘ Déclare WRTTIME come un “Byte”
CONST  MYROOM  = 310             ‘ Déclare MYROOM comme “Integer” (car il dépasse 255).
CONST  MSG1 = “ACCESS PORT”      ‘ Déclare comme une chaîne (String)
```

CON (autre mémthode similaire à CONST)

La commande « CON » peut également être utilisée pour déclarer les constantes comme suit :

```
PI          CON 3.14159          ‘ Déclare PI comme un “SINGLE”.
WRTTIME CON 10                   ‘ Déclare WRTTIME come un “Byte”
MYROOM      CON 310              ‘ Déclare MYROOM comme “Integer” (car il dépasse 255).
MSG1        CON “ACCESS PORT”    ‘ Déclare comme une chaîne (String)
```

Constantes de type tableau...

En utilisant des constantes de type tableau, vous pourrez spécifier une liste de nombres au début de votre programme afin de simplifier vos déclarations:

```
Const Byte DATA1 = (31, 25, 102, 34, 1, 0, 0, 0, 0, 0, 65, 64, 34)
I = 0
A = DATA1(I)          ' Mémorise 31 dans A.
I = I + 1
A = DATA1(I)          ' Mémorise 25 dans A.
```

```
Const Byte DATA1 = ("CUBLOC SYSTEMS")
```

Les données de type chaîne peuvent être stockées dans des tableaux d'octets. Le code ASCII du caractère sera retourné. Si DATA1(0) est lu, le code ASCII 'C' sera retourné. De la même façon si la donnée DATA1(1) est lu, le code ASCII 'U' sera retourné.

Des tableaux de variables diverses peuvent être utilisés:

```
CONST INTEGER DATA1 = (6000, 3000, 65500, 0, 3200)
CONST LONG DATA2 = (12345678, 356789, 165500, 0, 0)
CONST SINGLE DATA3 = (3.14, 0.12345, 1.5443, 0.0, 32.0)
```

Si vous devez définir de très longues suite de nombres, procédez comme dans les exemples ci-dessous:

1)
```
CONST BYTE DATA1 = (31, 25, 102, 34, 1, 0, 0, 0, 0, 0, 65, 64, 34,
                    12, 123, 94, 200, 0, 123, 44, 39, 120, 239,
                    132, 13, 34, 20, 101, 123, 44, 39, 12, 39)
```
2)
```
CONST BYTE DATA2 = (31, 25, 102, 34, 1, 0, 65, 64, 34,_
                        101, 123, 44, 39, 12, 39)
```

Les chaînes peuvent être utilisées comme ci-dessous:

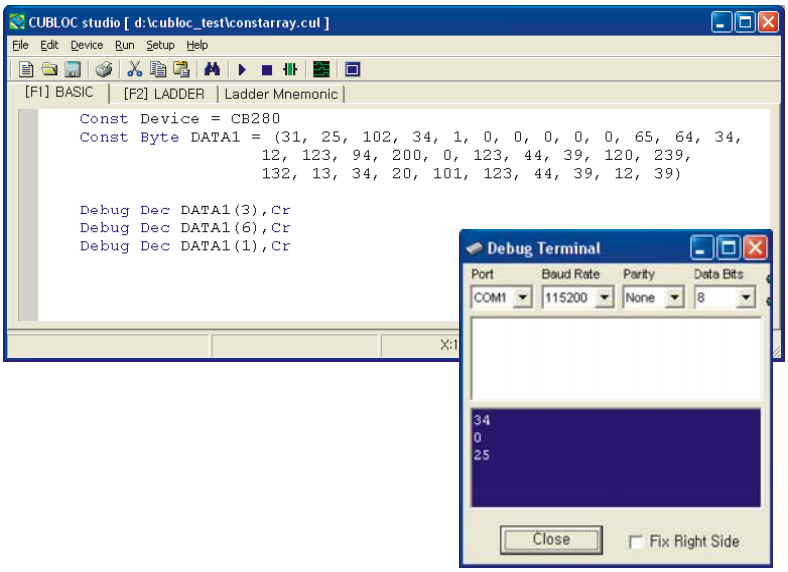
CONST STRING * 6 STRTBL = (“COMFILE”, “BASIC”, “ERROR”, “PICTURE”)

Configurez la taille de la chaîne afin qu’elle soit supérieure à la taille de tous les éléments de la liste.

Il n’est possible de déclarer que des tableaux à une seule dimension.

Comparaison	Tableau	Constante (tableau)
Stockage	Mémoire données (SRAM)	Mémoire programme (FLASH)
Durée de stockage	Pendant l’exécution du programme	Pendant le téléchargement
Peut être changé ?	Oui	Non
Objectif	Valeurs modifiables	Valeurs non modifiables
Coupure alimentation	Disparition données	Sauvegardée

Exemple de programme :



Opérateurs

Lorsque vous utilisez des opérateurs logiques, la table de priorité ci-dessous indique l'ordre de traitement effectué par le BASIC du CUBLOC™.

Opérateur	Explication	Type	Priorité
^	A la puissance	Math	La plus haute
*,/,MOD	Multiplier, Diviser, MOD	Math	
+,-	Ajouter, Soustraire	Math	
<<, >>	Décalage à gauche, Décalage à drote	Comparaison	
<, >, <=, >=	Plus petit que, Plus grand que, Moins ou égal à , plus grand ou égal à.	Comparaison	
=, <>	Egale, Différent	Comparaison	
AND, XOR, OR	AND, XOR, OR	Logique	La plus basse

Consulter le tableau ci-dessus pour vérifier l'ordre de priorité de vos calculs. Dans une ligne de calcul, l'ordre de priorité part de la gauche vers la droite.

Vous pouvez utiliser des opérateurs en tant que condition comme ci-dessous:

```
IF A+1 = 10 THEN GOTO ABC
```

Tous les nombres peuvent être mixés avec les nombres à virgule flottante. Le résultat final sera associé (ou converti) par rapport au type de la variable dans lequel il sera sauvegardé.

```
DIM F1 AS SINGLE
DIM A AS LONG
F1=1.1234
A = F1 * 3.14     ' A = 3 même si le résultat est 3.525456 (car A est de type LONG).
```

Veillez à bien inclure des (.) lorsque vous utilisez des nombres à virgule flottante.

```
F1 = 3.0/4.0                    ' 3/4 doit être écrit 3.0/4.0 pour obtenir des virgules flottantes
F1 = 200.0 + FLOOR(A) * 12.0 + SQR(B)   '200 doit être écrit 200.0 et 12 doit être écrit 12.0
```

AND, XOR, OR sont utilisés pour des opérations logiques et comme opérateur sur le bits.

```
IF A=1 AND B=1 THEN C=1     ' if A=1 et B=1 …(Opération logique)
IF A=1 OR B=1 THEN C=1      '  if A=1 ou B=1…( Opération logique)

A = B AND &HF         ' Met les 4 bits de poids fort à 0 (Opération sur les bits)
A = B XOR &HF         ' Inverse les 4 bits de poids faible. (Opération sur les bits)
A = B OR &HF          ' Met les 4 bits de poids faible à 1. (Opération sur les bits).
```

Les chaînes peuvent être comparées avec le signe “=”.
Les valeurs ASCII sont comparées pour les chaînes.

```
DIM ST1 AS STRING * 12
DIM ST2 AS STRING * 12
ST1 = “COMFILE”
ST2 = “CUBLOC”
IF ST1=ST2 THEN ST2 = “OK”    ‘ Vérifie si ST1 est égal à ST2.
```

Les opérateurs utilisés dans le langage BASIC des CUBLOC peuvent avoir une signification légèrement différente des autres langage BASIC (voir le tableau ci-dessous).

Opération	Math	Basic	Exemple
Addition	+	+	3+4+5, 6+A
Soustraction	-	-	10-3, 63-B
Multiplication	X	*	2 * 4, A * 5
Division	÷	/	1234/3, 3843/A
A la puissance de	5^3	^	5^3, A^2
MOD	Reste de	mod	102 mod 3

Avec le BASIC des CUBLOC™, le caractère (/) est utilisé pour désigner une division.
Vérifiez que vous utilisez les parenthèses de façon appropriée pour formuler correctement les opérations de calcul.

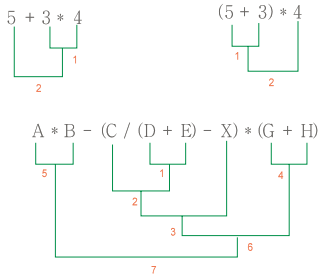
$\frac{1}{2}$ ▶ $1/2$ $\frac{5}{3+4}$ ▶ $5/(3+4)$

$\frac{2+6}{3+4}$ ▶ $(2+6)/(3+4)$

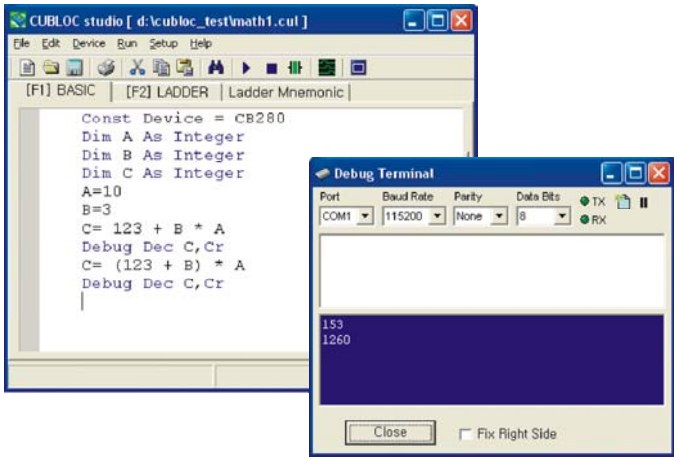
Priorité des opérations

Lorsque vous utilisez des opérations multiples, l'ordre de priorité ci-dessous est utilisé :

1) Opération entre parenthèses
2) Signe négatif (–)
3) (^)
4) Multiplication, Division, Reste (*, /, MOD)
5) Addition / Soustraction (+,-)
6) décalage à gauche, Décalage à droite (<<, >>)



Exemple de programme :



Exemples d'expression des nombres (en bits)

Il existe 3 façons pour exprimer les nombres avec les CUBLOC™.
Binaire (2 bits), Décimal (10 bits) et Héxadecimal (16 bits).

Exemples:

Binaire : &B10001010, &B10101,
0b1001001, 0b1100

Décimal : 10, 20, 32, 1234

Hexadecimal : &HA, &H1234, &HABCD
0xABCD, 0x1234 ← Similaire au langage C
$1234, $ABCD ← Similaire au langage Assembleur

Le « pré-processeur » BASIC

Le « pré-processeur » BASIC est un « macro processeur » automatiquement utilisé par le compilateur pour transformer votre programme avant la compilation. Il est appelé « macro processeur » car il vous permet de définir des « macros » qui vous permettent d'alléger la structure de programmes de grande envergure.

Le BASIC des CUBLOC™ est à ce titre similaire au langage « C » avec le recours possible à des directives telles que #include et #define afin d'ajouter des fichiers et des processus avant la compilation.

#include « nom de fichier »

Permet d'ajouter un fichier au code source. Pour les fichiers présents dans le même répertoire que le fichier source, vous pouvez utilisez la syntaxe ci-dessous :

#INCLUDE "MYLIB.cub"

Pour les fichiers stockés dans un autre répertoire, vous devrez indiquer le chemin d'accès complet comme dans l'exemple ci-dessous :

#INCLUDE "c:\mysource\CUBLOC\lib\mylib.cub"

En utilisant des fichiers externes via la commande #include, vous pourrez par exemple stocker toutes vos sous-routines dans des fichiers séparés.

Utilisez impérativement les commandes #include à la fin de votre programme (après « End » pour les « sous-routines »).

#define nom de constante
En utilisant la commande #define, vous pourrez définir des constantes avant la compilation.

```
#define motorport 4
low motorport
```

Dans l'exemple ci-dessus, l'expression motorport sera compilé comme l'expression 4.
Vous pouvez aussi utiliser plus simplement CONST comme dans l'exemple ci-dessous:

```
CONST motorport = 4
low motorport
```

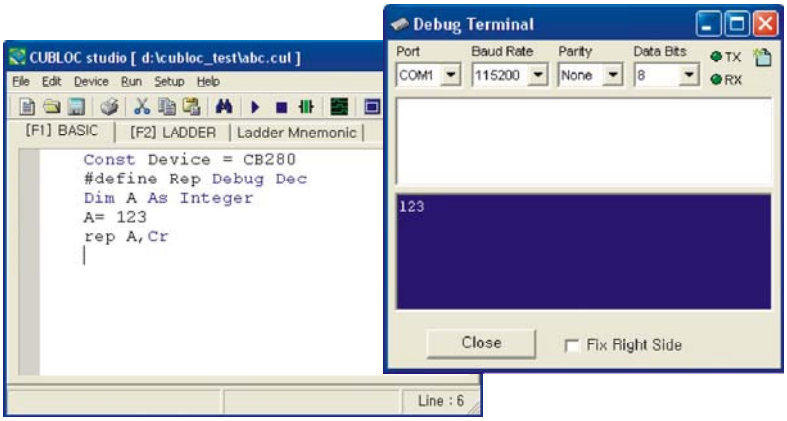
L'exemple suivant utilise #define pour replacer une ligne de commande :

```
#define FLAGREG1 2
#define f_led FLAGREG1.BIT0
#define calc (4+i)*256
f_led = 1                          ' positionne 1er bit FLAGREG1 à 1.
IF f_led = 1 then f_led = 0        ' Permet une lecture du programme plus simple
j = calc                           ' L'expression du calcul est simplifiée
```

Note :
#define ne fait pas de différence entre les caractères minuscules et majuscule. L'ensemble de ces derniers seront interprétés comme des caractères majuscules.

Par exemple #define ALPHA 0 et #define alpha 0 seront considérés comme identique.

Exemple de programme :



Directives conditionnelles

Une directive conditionnelle permet d'indiquer au « pré-processeur » de sélectionner ou non l'intégration d'une partie du code avant la compilation. Les directives conditionnelles peuvent tester des expressions arithmétiques ou un nom défini comme une macro ou les 2 en même temps en utilisant un opérateur spécial.

Voici quelques raisons pour lesquelles vous pouvez avoir recours à une directive conditionnelle :

- Un programme peu nécessiter l'utilisation de différents codes en fonction du modèle de CUBLOC™ utilisé (certains CUBLOC™ disposent de spécificités propres). Dès lors en utilisant une directive conditionnelle, vous pourrez adapter automatiquement le code de votre programme en fonction des spécificités de chaque module et disposer d'un programme « universel » pouvant s'adapter à tous les modules CUBLOC™.

- Il peut être nécessaire de compiler le même fichier source pour obtenir 2 programmes différents. Une première version pourra par exemple afficher des variable de Débug sur l'écran du PC alors que la seconde version ne comportera aucun retour d'infos de débug vers le PC.

#if constante
#endif

La directive #if du « pré-processeur » comparera une constante déclarée avec l'instruction CONST avec une autre constante. Si la condition testée par #if est vrai, le contenu compris entre les commandes #if … #endif seront compilés (sinon le code sera ignoré).

```
Const Device = CB280

Delay 500
#If Device = 220
  Debug "Module CB220 utilisé !"
#endif
```

Dans l'exemple ci-dessus, le code compilé sera différent suivant que vous utilisé un module CUBLOC « CB280 » ou un module « CB220 ».

#elseif
#else
En utilisant les directives #elseif ou #else, vous pourrez créer des tests plus complexes.

```
Const Device = CB220

Delay 500

#If Device = 220
  Debug "Module CB220 utilisé !"
  #elseif device = 280
  Debug " Module CB280 utilisé!"
  #elseif device = 290
  Debug " Module CB290 utilisé !"
  #elseif device = 1720
  Debug " Module CT1720 utilisé !"
#endif
```

#else ne pourra être utilisé qu'après une expression #if. Vous ne pouvez comparer que des constantes déclarées avec la commande CONST dans les tests #if.

#ifdef nom
#endif
Lorsque vous utilisez #if pour comparer des constantes, vous pouvez utiliser #ifdef pour voir si la constante a été préalablement déclarée via une commande #define ou CONST.

Si la constante a été préalablement déclarée, le code présent entre #if …#endif sera compilé (sinon il sera ignoré).

```
#define  LOWMODEL 0
#ifdef LOWMODEL
        LOW 0
#endif
```

Dans l'exemple ci-dessus, du fait que LOWMODEL ai été déclaré, la code LOW 0 sera ajouté au programme lors de la compilation.

#else #elseifdef peuvent être utilisés pour réaliser des tests plus complexes :

```
#ifdef LOWMODEL
      LOW 0
#elseifdef HIGHMODEL
      HIGH 0
#else
      LOW 1
#endif
```

#ifndef nom
#endif
#ifndef est exactement l'opposé de la directive #ifdef. Si une constante n'a pas été définie, le code entre #if…#endif sera compilé (sinon il sera ignoré).

```
#define  LOWMODEL 0
#ifndef LOWMODEL
        LOW 0
#endif
```

#elseifndef et #else peuvent être utilisés pour réaliser des tests plus complexes :

```
#ifndef LOWMODEL
        LOW 0
#elseifndef HIGHMODEL
        HIGH 0
#else
        LOW 1
#endif
```

Il est également possible de « mixer » les directive comme dans l'exemple ci-dessous:

```
#if MODELNO = 0
        LOW 0
#elseifdef HIGHMODEL
        HIGH 0
#else
        LOW 1
#endif
```

Il n'est toutefois pas possible d'utiliser #if à l'intérieur d'un autre test #if.

Pour utiliser uniquement le LADDER

Si vous ne désirez pas programmer en BASIC, votre application pourra être entièrement développée en langage LADDER. Toutefois, le programme BASIC devra toutefois comporter quelques instructions d'initialisation comme indiqué ci-dessous:

```
Const Device = CB280              ' Sélectionnez le type de module CUBLOC™ utilisé

Usepin 0,In,START             ' Déclarez les broches utilisées en LADDER
Usepin 1,Out,RELAY

Alias M0 = MOTORSTATE  ' Déclarez éventuellement des alias pour les Registres
Alias M1 = RELAY1STATE

Set Ladder On                      ' Démarrer le LADDER.
```

Le type de CUBLOC™ et les broches utilisés ainsi que la définition des « alias » devront se faire en BASIC. L'exécution du programme LADDER devra être activée depuis le programme BASIC par la commande SET LADDER ON.

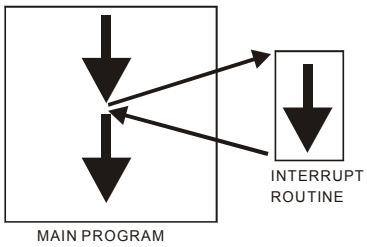
Pour utiliser uniquement le BASIC

Programmez simplement en BASIC ! (Le LADDER est désactivé apr défaut).

```
SET LADDER ON      ‘ N'utilisez pas cette commande.
LADDERSCAN         ‘ Ni celle-ci.
```

Interruptions

Une interruption peut intervenir durant l'exécution du programme principal afin de réaliser une routine particulière. La commande ON…GOSUB doit être utilisée pour configurer une nouvelle interruption. Lorsque cette interruption intervient, le programme principal est stoppé et le programme poursuit son exécution à l'adresse indiquée préalablement par la commande ON…GOSUB. Lorsque la sous routine d'interruption est terminée, la commande RETURN permet au programme principal de reprendre le cours de son exécution à l'endroit où il l'avait stoppé.



Des interruptions pour quoi faire ?
La saisie d'une touche ou encore l'arrivée de données sur le port RS-232 pouvant intervenir à tout moment, il est dans ce cas intéressant d'avoir recours à l'utilisation des interruptions afin de ne pas avoir à faire surveiller en permanence ces entrées dans le programme principal.

A ce titre, les modules CUBLOC™ disposent d'une gestion des interruptions extrêmement flexible et puissante. Si une routine d'interruption est en cours d'exécution**, toute autre requête d'interruption de même nature sera ignorée.** Si par exemple une interruption de type RS232 RECV survient durant l'exécution d'une sous-routine d'interruption RS232 RECV, cette dernière sera ignorée. Si par contre une interruption de type INT Edge survient durant l'exécution d'une sous-routine d'interruption RS232 RECV, cette dernière sera alors immédiatement exécutée.

Type d'Interruption	Explanation
On Timer	Génère une interruption à intervalle de temps donné
On Int	Génère une interruption lorsqu'une entrée externe est sollicitée
On Recv	Génère une interruption lorsque des données arrivent sur la liaison RS-232
On LadderInt	Génère une interruption lorsque celle-ci est demandée par le LADDER
On Pad	Génère une interruption lorsqu'un clavier de saisie est sollicité

En savoir plus sur les interruptions des modules CUBLOC™…
Les modules CUBLOC™ (ainsi que le module CUTOUCH™) disposent d'un mini-système d'exploitation qui prend en charge les interruptions. La gestion des interruptions est légèrement différente de celle que l'on peu retrouver sur les microcontrôleurs standards.

1. Lorsqu'une interruption « A » survient, durant l'exécution de la sous-routine associée à cette interruption, une nouvelle interruption « A » ne pourra pas être à nouveau générée. Mais une interruption différente « B » par exemple pourra survenir. Afin d'être plus explicite, on pourra par exemple indiquer que les interruptions de type « A » et « B » correspondent à des interruption générées par « On Timer » et « On Recv ».

2. Lorsque l'interruption « B » survient (pendant l'exécution de la sous-routine d'interruption « A »), la sous-routine de l'interruption « B » sera immédiatement exécutée et le programme principal reviendra à l'interruption A afin de terminer celle-ci.

3. A la fin de votre routine d'interruption il vous faudra impérativement ajouter une instruction **return** (sous peine de dysfonction de votre programme).

4. Il n'y a pas de limite sur le nombre d'interruptions, ni même sur la taille de la routine d'interruption (sans la limite de l'espace mémoire de votre programme bien sûr).

5. Les commandes **Delay**, **Pulsout** peuvent être utilisées pendant une interruption. Toutefois il est important de savoir que les commandes **Delay**, **Pulsout** peuvent être affectées par d'autres interruptions intervenant lors de leur exécution. Pour éviter ce phénomène, utilisez alors la commande Set Onglobal Off **avant** d'exécuter les commandes **Delay** et **Pulsout** comme dans l'exemple ci-dessous :

```
Set Onglobal Off
Delay 100                    ' La commande Delay ne sera pas affectée
Set Onglobal On
```

6. Si aucune interruption n'est requise durant l'exécution de votre programme, vous pouvez accélérer sensiblement l'exécution de votre programme ou de votre CUTOUCH en désactivant toutes les interruptions à l'aide de la commande **Set Onglobal Off**.

   * Par défaut, la commande **Set Onglobal** est active (« ON »).

7. Lors de la génération d'une interruption par On Recv, les données reçues durant l'exécution de la sous-routine On Recv seront simplement stockées dans le buffer de réception (afin qu'aucune donnée ne soit perdue). Dès que la sous-routine générée par le On Recv est terminée, s'il n'y a une nouvelle donnée dans le buffer de réception, une nouvelle interruption est alors immédiatement générée. La commande Bclr pourra être utilisée si vous ne désirez pas régénérer une nouvelle interruption.

8. Si vous déclarez une interruption 2 fois, le dernier appel sera effectif.

Pointeurs utilisant Peek, Poke et Memadr

L'exemple suivant montre comment utiliser les commandes EEWRITE et EEREAD pour lire des données comportant des nombres à virgule flottante:

```
Const Device = CB280
Dim f1 As Single, f2 As Single
f1 = 3.14
Eewrite 0,f1,4
f2 = Eeread(0,4)
Debug Float f2,cr
```

Lorsque vous exécutez ce code, la fenêtre de DUBUG affichera 3.00000 au lieu de 3.14.
La raison est que la commande EEWRITE convertie automatiquement les nombres à virgule flottante en nombre entier.

Afin de pouvoir stocker et relire des nombres à virgule flottante, vous devrez utiliser les commandes Peek et Poke comme indiqué dans l'exemple ci-dessous.

```
Const Device = CB280
Dim F1 As Single, F2 As Single
F1 = 3.14
Eewrite 10,Peek(Memadr(F1),4),4
Poke Memadr(F2),Eeread(10,4),4

Debug Float F2,CR
```

La fenêtre Debug affichera maintenant 3.14.

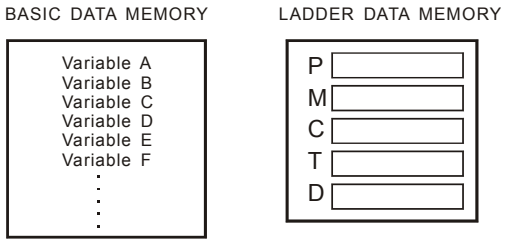
La commande Memadr(F1) est utilisée pour trouver l'adresse mémoire de F1, puis la commande Peek est utilisée pour accéder directement la mémoire en y insrivant 4 octets dans la mémoire EEPROM.

A l'inverse, les commandes Memadr(F2) et Poke sont utilisées pour lire à nouveau directement les 4 octets.

Attention : Prenez garde en utilisant ces commandes, sans quoi les pointeurs risquent d'affecter le programme en entier et/ou le fonctionnement du CUBLOC™. Les commandes Peek et Poke peuvent également être utilisées pour accéder la mémoire de donnée de type SRAM.

Partage de données

Les modules CUBLOC™ disposent d'une mémoire de donnée individuelle pour le BASIC et le LADDER.



La mémoire de donnée du LADDER peut être facilement accédée par le BASIC en utilisant des variables systèmes. En utilisant ces variables systèmes, cette mémoire données pourra également être lue et modifiée par le LADDER.

Variables systèmes	Type de donnée	« Registres » du LADDER
_P	Bits _P(0) ~ P(127)	Registre P
_M	Bits _P(0) ~ P(511)	Registre M
_WP	Words _WP(0) ~ _WP(7)	Registre P (Word Access)
_WM	Words _WM(0) ~ _WM(31)	Registre M (Word Access)
_T	Words _T(0) ~ _T(99)	Registre T (Timer)
_C	Words _C(0) ~ _C(49)	Registre C (Counter)
_D	Words _D(0) ~ _D(99)	Registre D (Data)

Les « Registres » P et M du LADDER pourront être adressés bit à bit. Les autres « Registres » C, T, et D pourront être adressés en tant que données de type « Word ». Pour accéder aux « Registres » P et M en tant que données de type « Words », utilisez « _WP » et « _WD ».

Par exemple _WP(0) reprsente P0 à P15.

Voir également d'autres exemples ci-dessous:

```
_D(0) = 1234
_D(1) = 3456
_D(2) = 100
FOR I = 0 TO 99
        _M(I) = 0
NEXT
IF _P(3) = 1 THEN _M(127) = 1
```

A l'inverse, il n'est pas possible d'avoir accès aux variables du BASIC depuis le LADDER. Vous pourrez toutefois déclencher des interruptions dans le BASIC, lesquelles pourront être générées par le LADDER.

Utilisation des broches du Ladder en BASIC via commandes ALIAS

Les commandes ALIAS pourront être utilisées depuis le BASIC pour donner des « alias » aux « Registres » du LADDER (**tous sauf D**). Il vous sera dès lors possible d'utiliser ces « alias » librement en BASIC ou en LADDER. Ceci est idéal pour améliorer la lisibilité de votre programme.

```
Usepin 0,In,START
Usepin 1,Out,RELAY
Alias M0 = MOTORSTATE
Alias M1 = RELAY1STATE
Alias T1 = SUBTIMER

RELAY = 0                ' Mettre le port 1 au niveau BAS
MOTORSTATE = 1           ' Mettre le Registre M0 au niveau HAUT.  Idem _M(0) = 1.

A = RELAY1STATE          ' Mémorise état de M1 dans la variable A.
B = SUBTIMER             ' Mémorise état de T1 dans la variable B
```

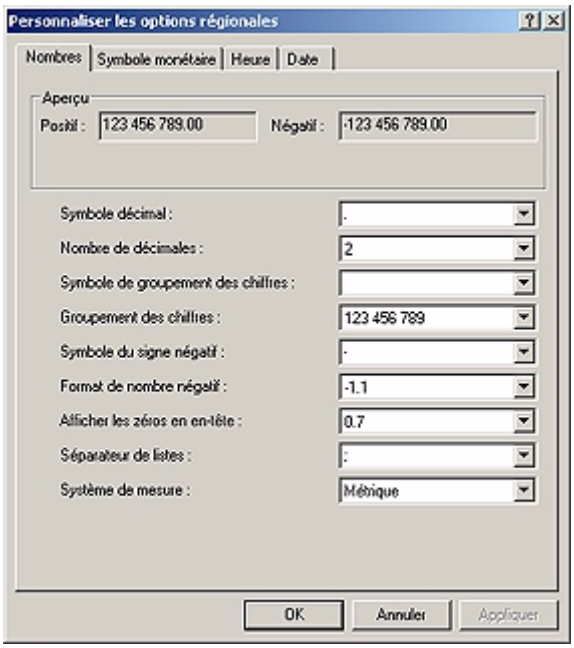
Chapitre 5.

Les principales fonctions BASIC des CUBLOC™

Fonctions Mathématiques

Avant de pouvoir utiliser correctement les fonctions mathématiques des CUBLOC, il vous faut impérativement (sous votre système d'exploitation Windows XP™) sélectionner le menu « Démarrer » -> « Paramètres » - > « panneau de configuration ».

Réalisez alors un « double click » sur l'icône « Options régionales… ».



Cliquez ensuite sur le bouton « Personnaliser… » et sélectionnez le « . » (point) comme symbole décimal. Puis validez les modifications en cliquant à chaque fois sur le bouton « OK » pour referner les fenêtres.

Si vous ne procédez pas à cette modification, vous ne pourrez pas utiliser correctement les fonctions et calculs mathématiques des CUBLOC (les résultats retournés seront erronés).

SIN, COS, TAN

Retourne les valeurs Sinus, Cosinus et Tangente. Les CUBLOC™ utilisent les radians comme unité. Utilisez des variables de type « SINGLE » pour une plus grande précision.

```
A=SIN B      ‘ Retourne la valeur du Sinus de B.
A=COS B      ‘ Retourne la valeur du Cosinus de B.
A=TAN B      ‘ Retourne la valeur de la Tangente de B.
```

ASIN, ACOS, ATAN

Retourne les valeurs Arc Sine, Arc Cosine et Arc Tangente. Les CUBLOC™ utilisent les radians comme unité. Utilisez des variables de type « SINGLE » pour une plus grande précision.

```
A=ASIN B     ‘ Retourne la valeur de l’Arc Sinus de B.
A=ACOS B     ‘ Retourne la valeur de l’Arc Cosinus de B.
A=ATAN B     ‘ Retourne la valeur de l’Arc Tangente de B.
```

SINH, COSH, TANH

Retourne les valeurs Sinus Hyperbolic, Cosinus Hyperbolic et Tangente Hyperbolic.

```
A=SINH B     ‘ Retourne la valeur du Sinus Hyperbolic de B.
A=COSH B     ‘ Retourne la valeur du Cosinus Hyperbolic de B.
A=TANH B     ‘ Retourne la valeur de la Tangeante Hyperbolic de B.
```

SQR

```
A=SQR B      ‘ Retourne la valeur de la racine carée de B.
```

EXP

```
A=EXP X      ‘ Retourne E^X.
```

LOG, LOG10

```
A=LOG B      ‘ Retourne la valeur de LOG de B.
Ou
A=LOG10 B
```

Astuce : Pour le logarithme naturel (Ln), utilisez simplement : A = Log(B) / Log(Exp(1))

ABS (ne fonctionne que pour les variables de type LONG)

```
Dim A As Long, B As Long
B = -1234
A=ABS B          ' Retourne |B|.
Debug Dec A      ' Affiche 1234
```

FABS (ne fonctionne que pour les variables de type SINGLE)

```
Dim A As Single, B As Single
B = -1234.0
A=FABS B         ' Retourne |B|.
Debug Float A    ' Affiche 1234.00
```

FLOOR

```
Dim A As Single, B As Single
B = 3.14
A=FLOOR B      ' FLOOR de 3.14 donne 3.
Debug Float A  ' Affiche 3.0
```

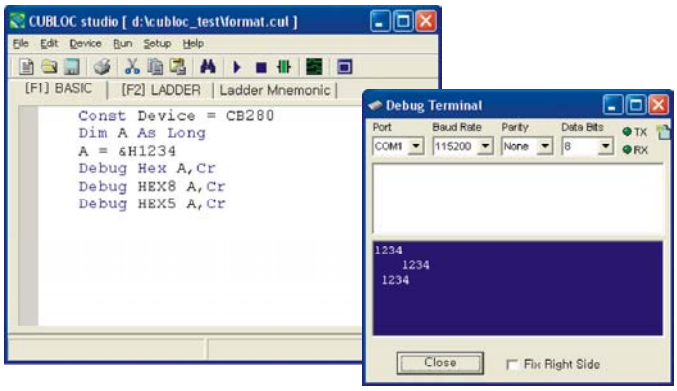
Conversions de « formats »

Des commandes vous permettront de modifier une variable vers une représentation différente.

HEX

Convertie la variable en hex (16 bits). HEX8 signifie une conversion en décimal sur 8 caractères (1 à 8 peut être utilisé pour les places décimales)

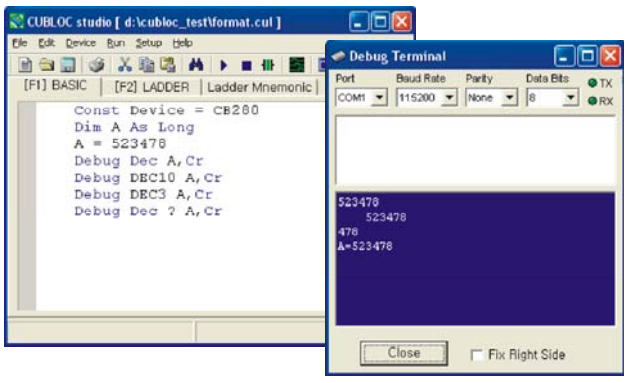
```
DEBUG HEX A       ‘ Si A est 123ABC, 123ABC sera affiché dans la fenêtre de DEBUG
DEBUG HEX8 A      ‘ Si A est 123ABC, bb123ABC sera affiché dans la fenêtre de DEBUG
                  ‘  b correspond à un caractère blanc.
DEBUG HEX5 A      ‘ Si A est 123ABC, 23ABC sera affiché dans la fenêtre de DEBUG
                  ’ Le premier caractère est coupé.
```



DEC

Converti une variable en un décimal (10 bits). DEC8 signifie une conversion en décimal sur 8 caractères (1 à 11 peut être utilisé pour les places décimales)

```
DEBUG DEC A       ‘ Si A est 1234, 1234 sera affiché dans la fenêtre de DEBUG.
DEBUG DEC10 A     ‘ Si A est 1234, bbbbbb1234 sera affiché dans la fenêtre de DEBUG
                  ‘ b correspond à un caractère blanc.
DEBUG DEC3 A      ‘ Si A est 1234, 234 sera affiché dans la fenêtre de DEBUG
                   ‘Le premier caractère est coupé
```



?

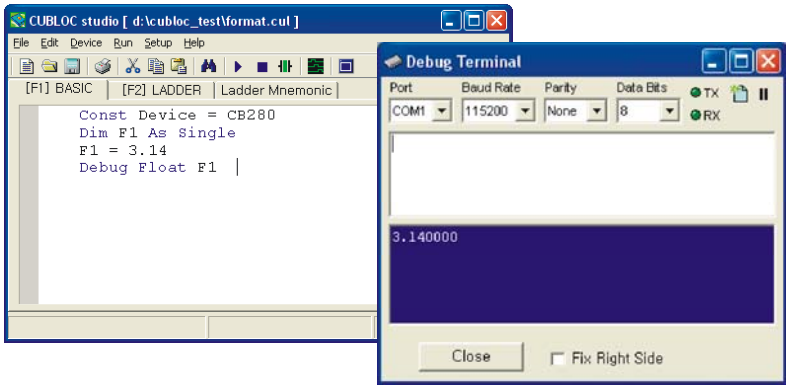
Permet d'ajouter le nom de la variable (ne peut être utilisé qu'avec des variables de type HEX ou DEC).

```
DEBUG DEC ? A      ' Si A est 1234, "A=1234" sera affiché dans la fenêtre de DEBUG.
DEBUG HEX ? A      ' Si A est ABCD, "A=ABCD" sera affiché dans la fenêtre de DEBUG.
DEBUG HEX ? B      ' Si B est une sous-routine (CONV par exemple)
                   ' "B_@_CONV=ABCD" sera affiché.
```

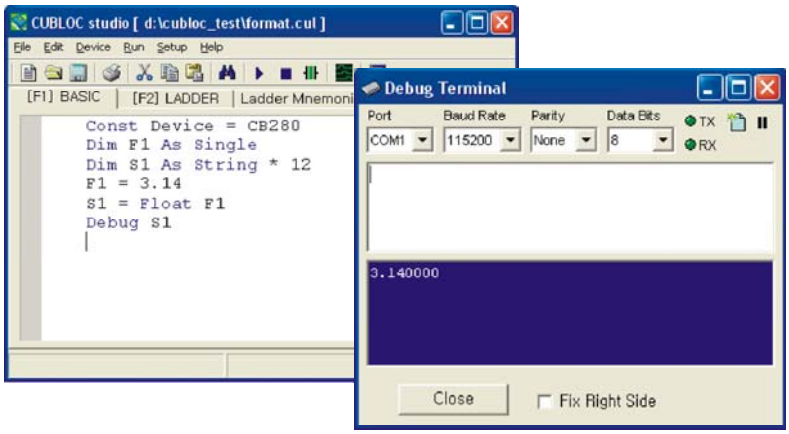
FLOAT

Utilisez FLOAT pour convertir les nombres à virgule flottante en chaîne.

```
Const Device = cb280
Dim F1 As Single
F1 = 3.14
Debug Float F1,cr          ' Affiche "3.14000" et va à la ligne
```




```
Dim ST As String * 12
ST = Float F1              ' Stock F1 dans une chaîne.
ST = Left(ST,3)            ' Ne récupère que les 3 caractères de gauche.
 Debug ST                  ' Affiche "3.14" dans la fenêtre DEBUG.
```



Fonctions de gestion de chaînes

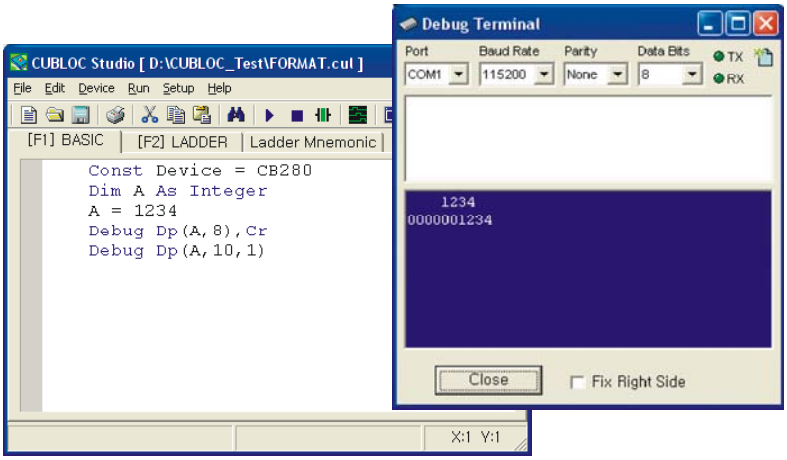
Ces fonctions permettent de réaliser des traitements simplifiés sur les chaînes.

DP(Variable, Place Decimale, Affichage zéro)

Permet de convertir une variable en une chaîne de représentation décimale.

Si le paramètre Affichage zéro est mis à 1, des zéros remplacent les espaces libres.

```
Dim A as Integer
DEBUG DP(A,10,0)    ' Convertion de A dans une chaîne de représentation décimale.
                    ' Réserve 10 emplacements pour les chiffres.
                    ' Si A = 1234, bbbbbb1234 sera affiché.
                    ' (b sera remplacé par des « espaces ».)
DEBUG DP(A,10,1)    ' Si A = 1234 alors 0000001234 sera affiché.
```

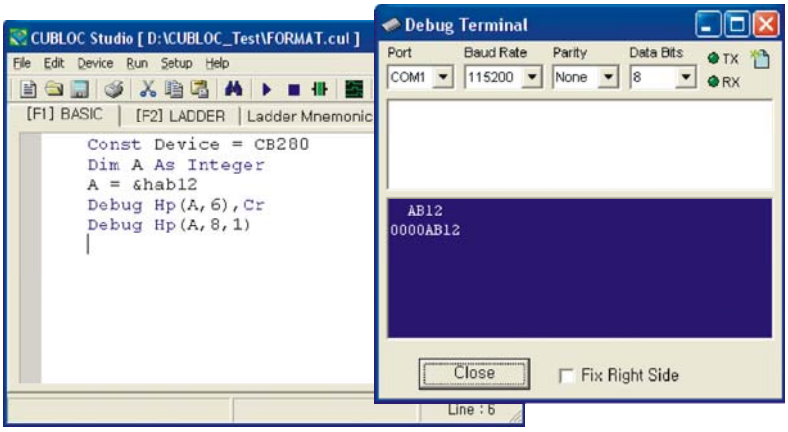


HP(Variable, Place Decimale, Affichage zéro)

Permet de convertir une variable en une chaîne de représentation héxadécimale.

Si le paramètre Affichage zéro est mis à 1, des zéros remplacent les espaces libres.

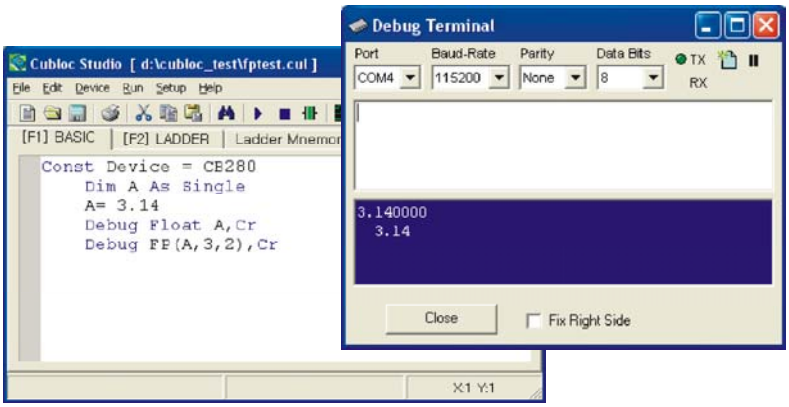
```
DEBUG HP(A,4,0)  ' Convertion de A dans une chaîne de représentation héxadécimale
                 ' Réserve 4 emplacements pour l'affichage.
                 ' Si A = ABC, alors  bABC  sera affiché.
                 ' (b sera remplacé par des « espaces ».)
DEBUG HP(A,4,1)  ' Si A = ABC, alors   0ABC   sera affiché.
```



FP(Variable, Nombre de Digits, Nombre après virgule)

Permet de convertir un nombre à virgule flottante dans une chaîne formatée avec la possibilité de sélectionner le format d'affichage.

```
Dim A as Single
A = 3.14
DEBUG Float A              ' Affiche 3.1400000 (avec tous les digits)
DEBUG FP(A,3,2)            ' Affiche valeur selon le format défini : 3.14
```



En utilisant la fonction « FP » il vous sera possible de définir le format d'affichage de vos expressions sur la fenêtre de Débug ou sur les afficheurs LCD par exemple.

La gestion des nombres à virgule flottante des CUBLOC™ est au format IEEE724. Les valeurs retournées par la fonction « FP » et « Float » peuvent être différentes (mais les valeurs stockées dans les variables seront bien les mêmes).

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

LEFT(Variable, Nb caractères)

Sélectionne un nombre défini de caractères d'une chaîne en partant de la gauche.

```
DIM ST1 AS STRING * 12
ST1 = "CUBLOC"
DEBUG LEFT(ST1,4)       ' "CUBL" sera affiché dans la fenêtre DEBUG.
```

RIGHT(Variable, Nb caractères)

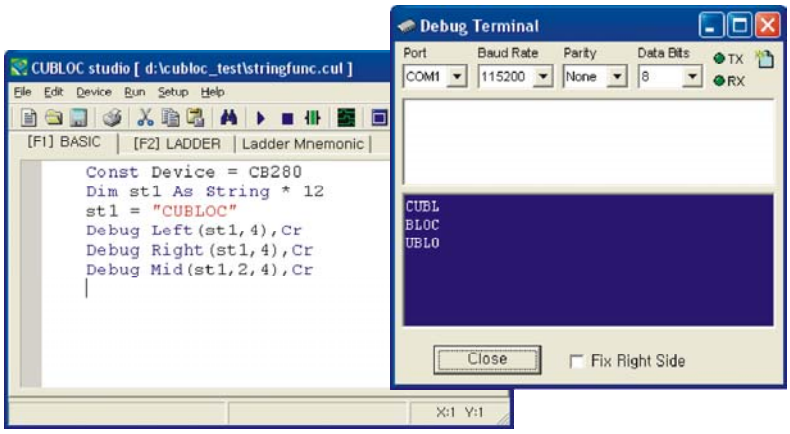
Sélectionne un nombre défini de caractères d'une chaîne en partant de la droite.

```
DIM ST1 AS STRING * 12
ST1 = "CUBLOC"
DEBUG RIGHT(ST1,4)  ' "BLOC" sera affiché dans la fenêtre DEBUG.
```

MID(Variable, Position, Nb caractères)

Sélectionne un nombre défini de caractères d'une chaîne en partant d'une position donnée.

```
DIM ST1 AS STRING * 12
ST1 = "CUBLOC"
DEBUG MID(ST1,2,4)  ' "UBLO" sera affiché dans la fenêtre DEBUG.
```



LEN(Variable)

Retourne la longeur d'une chaîne.

```
DIM ST1 AS STRING * 12
ST1 = "CUBLOC"
DEBUG DEC LEN(ST1)  ' 6 sera affiché car la chaîne comprend 6 caractères.
```

STRING(ASCII code, longueur)

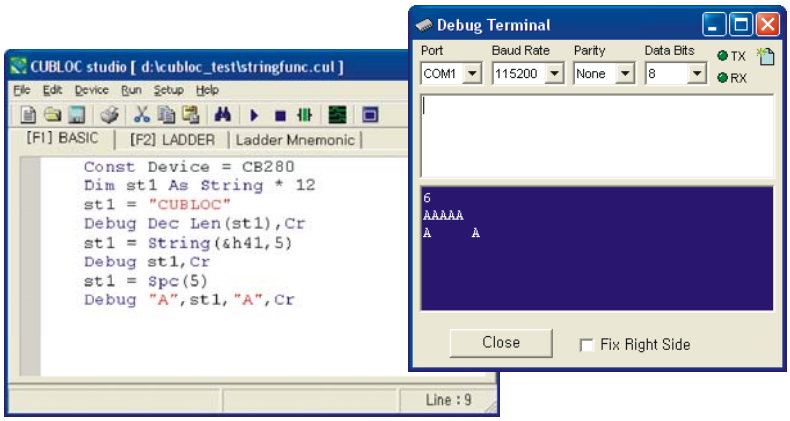
Permet de créer une chaîne de longeur spécifiée remplie avec un caractère ASCII donné.

```
DIM ST1 AS STRING * 12
ST1 = STRING(&H41,5)
DEBUG ST1  'AAAAA sera affiché dans la fenêtre DEBUG.
           &H41est la valeur ASCII de la lettre A.
```

SPC(Nb Caractères)

Permet de créer une chaîne de caractères « avec des espaces »

```
DIM ST1 AS STRING * 12
ST1 = SPC(5)
DEBUG "A",ST1,"A"      'AbbbbbA sera affiché dans la fenêtre DEBUG. b sera des espaces.
```



LTRIM(variable Chaîne)

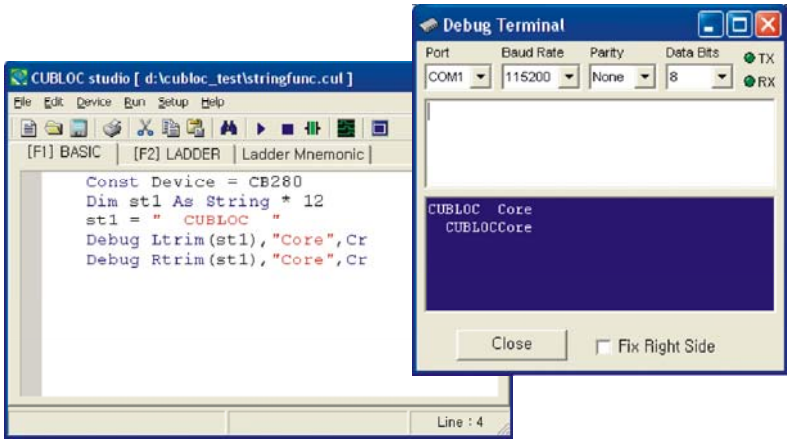
Enlève tous les espaces à gauche d'une chaîne de caractères.

```
DIM ST1 AS STRING * 12
ST1 = "  COMFILE"
ST1 = LTRIM(ST1)
DEBUG "AAA",ST1  ' AAACOMFILE sera affiché dans la fenêtre DEBUG.
                 ' Tous les espaces à gauche de COMFILE on été retirés.
```

RTRIM(variable Chaîne)

Enlève tous les espaces à droite d'une chaîne de caractères.

```
DIM ST1 AS STRING * 12
ST1 = "COMFILE  "
ST1 = RTRIM(ST1)
DEBUG ST1,"TECH"  ' COMFILETECH sera affiché dans la fenêtre DEBUG.
                  'Tous les espaces à gauche de COMFILE on été retirés.
```



VAL(variable Chaîne)

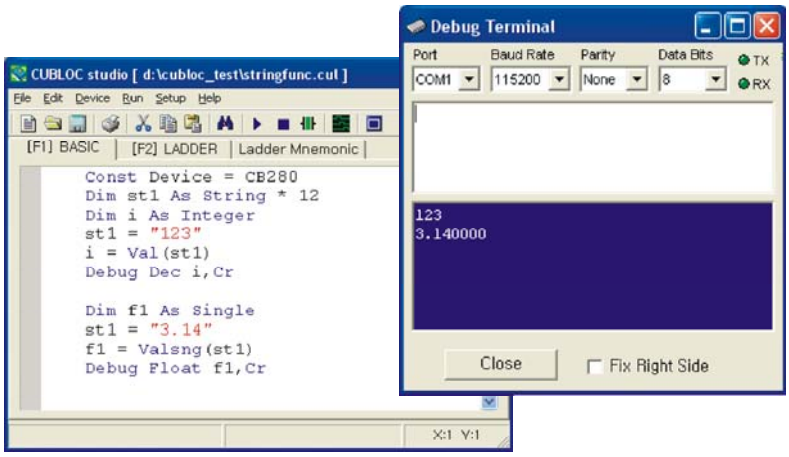
Conversion de la valeur numérique d'une chaîne.

```
DIM ST1 AS STRING * 12
DIM I AS INTEGER
ST1 = "123"
I = VAL(ST1)      ' 123 est mémorisé dans la variable numérique I.
```

VALSNG(variable Chaîne)

Conversion de la valeur numérique (avec virgule flottante) d'une chaîne.

```
DIM ST1 AS STRING * 12
DIM F AS SINGLE
ST1 = "3.14"
F = VALSNG(ST1)      ' 3.14 est mémorisé dans la variable numérique F.
```



VALHEX(variable Chaîne)

Conversion de la valeur héxadécimal d'une chaîne.

```
DIM ST1 AS STRING * 12
DIM I AS LONG
ST1 = "ABCD123"
I = VALHEX(ST1)      ' &HABCD123 est mémorisé dans la variable I.
```

CHR(code ASCII)

Retourne le code ASCII d'un caractère.

```
DIM ST1 AS STRING * 12
ST1 = CHR(&H41)
DEBUG ST1              ‘ Affiche A, &H41 est le code ASCII de la lettre A.
```

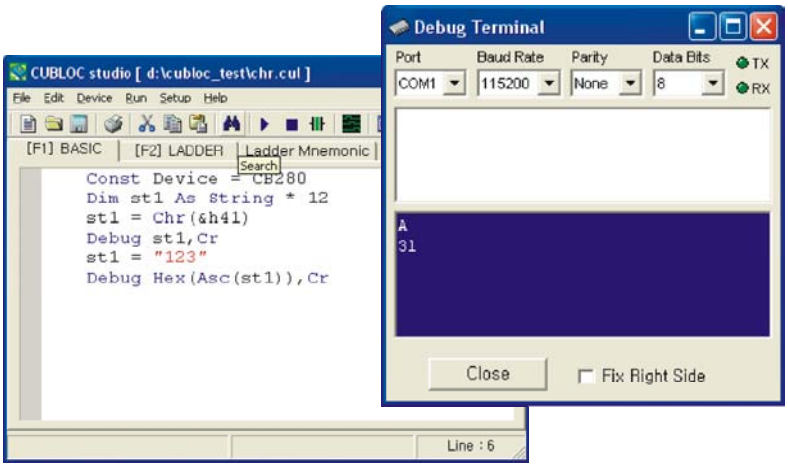
ASC(variable String ou Constante)

Retourne le code ASCII converti du premier caractère d'une chaîne.

```
DIM ST1 AS STRING * 12
DIM I AS INTEGER
ST1 = “123”
I = ASC(ST1)   ‘ &H31 est stocké dans la variable I.  Le code ASCII du chiffre 1
               ‘  est &H31 ou 0x31.
```

NOTE
Seule une variable peut être utilisée lorsque vous exploitez des fonctions de chaînes.

```
DEBUG LEFT(“INTEGER”,4)      ‘ Impossible d'utiliser une chaine de caractère directement.

ST1 = “INTEGER”
DEBUG LEFT(ST1,4)            ‘ Vous devez utiliser une variable pour réaliser cette instruction.
```



Chapitre 6.

Les librairies BASIC des CUBLOC™

Adin( )

Variable = ADIN (Canal)

Variable : Variable servant à mémoriser le résultat

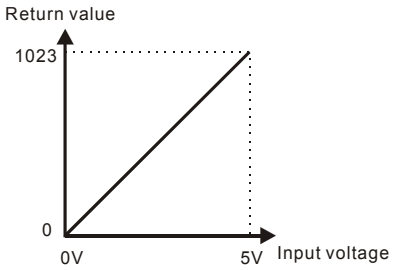
Canal : Port de conversion Analogique/Numérique (pas le N° de la broche du CUBLOC)

Les CUBLOC™ disposent d'entrées de convertisseurs Analogiques/Numériques avec une résolution de 10 bits et de sorties PWM avec une résolution de 16 bits. L'utilisateur pourra utiliser les convertisseurs « A/N » pour mesurer des tensions analogiques ou utiliser les sorties PWM pour piloter des moteurs « cc » ou générer des signaux analogiques.

La commande **ADIN** effectue une lecture de la valeur analogique d'un signal 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 au préalable être déclarés en tant que port d'entrée avant toute utilisation.**

Lorsqu'une tension comprise entre 0 et AVREF est mesurée, cette tension est convertie en une valeur comprise entre 0 et 1023. * AVREF peut être compris entre +2 et +5 V (la valeur usuelle est généralement de +5 V). Si vous utilisez une tension de +3 V pour AVREF, alors les tensions comprises entre 0 et +3 V seront converties 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° du port de conversion « Analogique/Numérique » du module CUBLOC™. Ainsi dans l'exemple ci-dessus, le port conversion « Analogique/Numérique » N° 0 correspond en fait au Port 24 du CB280.

Les schémas et la table ci-dessous donnent la correspondance des ports/N° broches de conversion « Analogique / Numérique » en fonction des différents modèles de CUBLOC™.

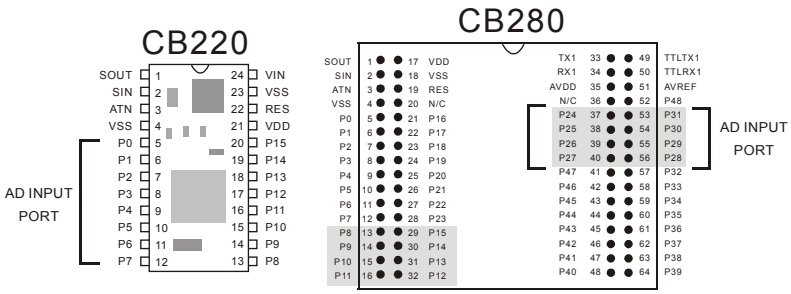


Tableau de corespondance.

	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 l’exécution de la commande). En comparaison, la commande **TADIN** retourne la valeur moyenne de 10 conversions afin de donner une plus grande précision de la mesure. Il conviendra donc d’utiliser la commande **TADIN** dans le cadre d’applications nécessitant une grande précision.

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 vous permettront de nommer en « clair » les Regsitres utilisés dans le LADDER (tels que P0, M0, C0) afin que votre programme soit plus simple à « lire » et à interpréter.

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

Bcd2bin

Variable = BCD2BIN(valeurbcd)
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 (vous pouvez sélectionner la nature du buffer).

```
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 : Période de sortie des impulsions (1 à 65535)

La commande **BEEP** est utilisée pour créer des « bips » sonores. Un buzzer (sans oscillateur) doit pour ce faire être relié au préalable sur un des ports du module CUBLOC™. Un bip sonore court sera alors généré. Cette commande est utile par exemple pour confirmer la saisie d'une touche ou réaliser des signaux sonores divers. Lorsque vous utilisez cette commande, la broche utilisée et automatiquement configurée en sortie.

```
BEEP 2, 100   ' Génère un Bip sonore sur P2 avec une période de 100
```

Bfree( )

Variable = BFREE(canal, typebuffer)
Variable : Variable servant à mémoriser le résultat (Non 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 la valeur du nombre d'octets de libres disponibles dans le buffer de réception ou d'émission. Cette commande pourra par exemple 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, 49 octets max. de libre pourront être retournés. La fonction retournera un nombre d'octets inférieur d'une unité par rapport à la taille du buffer lorque le buffer est totalement vide.

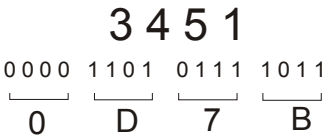
Bin2bcd

Variable = BIN2BCD(valeurbinaire)

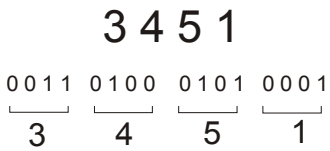
Variable : Variable servant à mémoriser le résultat (Retourne un LONG)
valeurbinaire : Valeur binaire à convertir

La commande **BIN2BCD** convertie un nombre binaire en valeur BCD (afin de pouvoir en avoir une expression décimale).

Par exemple: 3451 en binaire sera représenté comme suit:



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 digit du nombre binaire.



Cette commande est utile lorsque vous désirerez convertir une variable binaire afin qu'elle puisse par exemple être représentée sur un afficheur 7 segments à Leds.

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

Blen( )

Variable = RBLEN(canal, typebuffer)

Variable : Variable servant à mémoriser le résultat (Non 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 courrant 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 pouvoir 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
Loop                           ' Boucle sans fin

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

Bytein( )

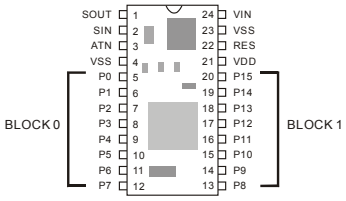
Variable = BYTEIN(Bloc)

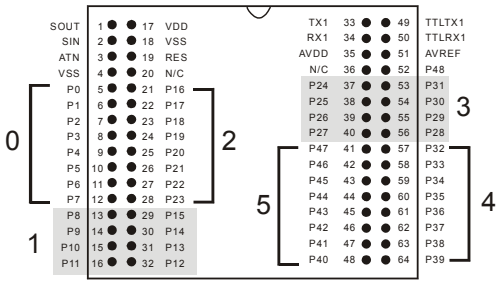
Variable : Variable servant à mémoriser le résultat (Non String, ni Single)
Bloc : N° du Bloc d'« E/S » (0 à 15 suivant le type de CUBLOC™ utilisé)

Effectue la lecture de l'état en cours d'un Bloc d'E/S. 8 broches d'E/S (ports) sont associées à ce qu'on appel 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 représente 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 a appliquer sur les sorties (comprise entre 0 et 255).

Permet d'appliquer une valeur sur un Bloc. 8 broches d'E/S (ports) sont associées à ce qu'on appel 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 (Non String, ni Single)
canal : Canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)

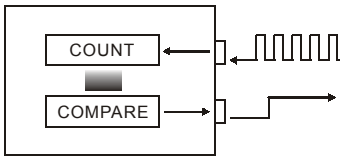
Sans affecter le contenu du buffer de réception RS232, la commande **CheckBf()** peut être utilisée pour vérifier le contenu du buffer. Cette commande vous permettra ainsi de lire ce qui est présent dans le buffer (1 seul octet à la fois), sans l'effacer (contrairement à la commande **GET**).

```
A = Checkbf(1)         ' Vérifie la donnée dans le buffer de réception
```

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



Lorsque un compteur rapide atteind la valeur prédéfinie (Cible#), le CUBLOC™ applique un état logique (Etatcible) sur un Port donné.

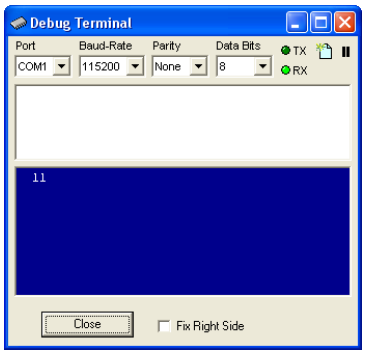
Si Etatcible = 1 et que le compteur atteind la valeur Cible# alors le Port prendra le niveau logique haut « 1 ». A l'inverse, si Etatcible = 0 et que le compteur atteind la valeur Cible# alors le Port prendra le niveau logique bas « 0 ».

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 de pouvoir 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 atteind 11 impulsions, le port P61 « passe » au niveau logique « haut ».

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

Count( )

Variable = COUNT(canal)

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

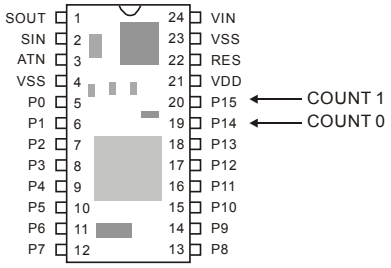
Retourne la valeur du canal compteur spécifié. Il vous faut au préalable configurer le port en entrée avant d'utiliser cette commande.

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™).

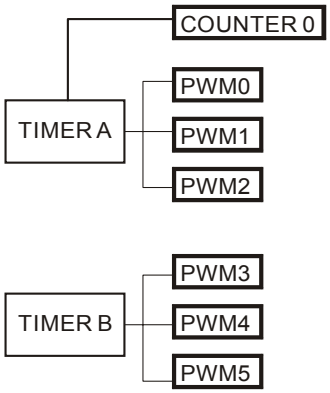
Les modules CUBLOC™ disposent de 2 Compteurs. Le compteur du Canal 0 utilise les mêmes ressources que les fonctions PWM0, 1, 2 et ne pourra donc pas être utilisé en même temps que ceux-ci. Toutefois le compteur du Canal 1 pourra être utilisé librement.

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 vous sera pas possible 'utiliser toutes ces fonctions en même temps.




```
' Exemple de mesure de fréquence des impuslions 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)

Reset (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

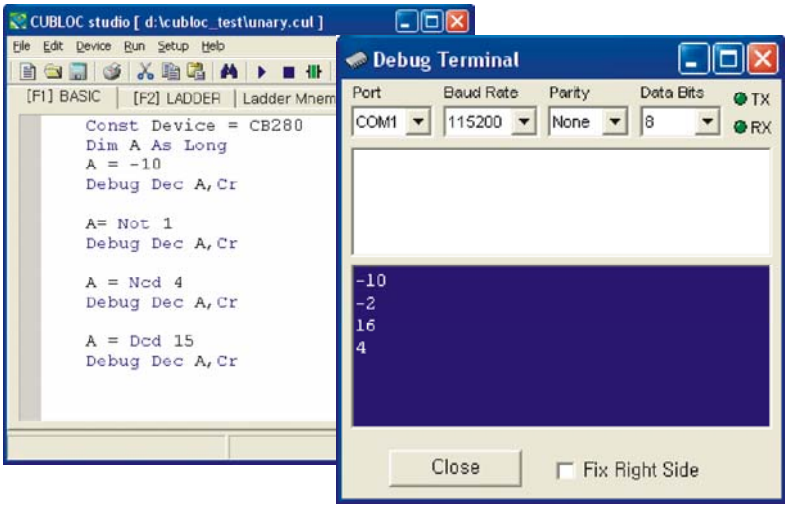
Variable = Source DCD

Variable : Variable servant à mémoriser le résultat (Non String, ni Single).
Source : Valeur source

La commande **DCD** réalise l'inverse de la commande **NCD**.

Cette commande retourne la position du bit (en partant du bit de poid faible LSB 0) de la position la plus haute du bit qui est à 1.

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



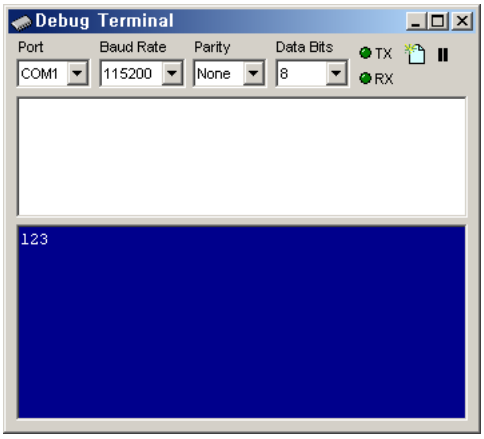
Debug

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.

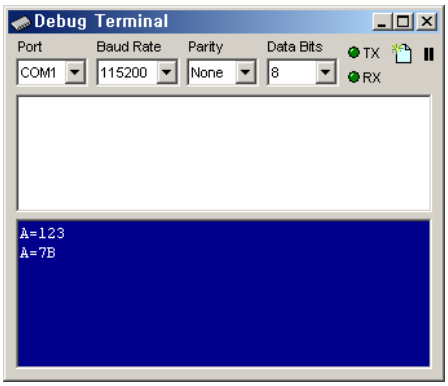
```
DIM A AS INTEGER
A = 123
DEBUG  DEC  A
```



Utilisez **DEC** ou **HEX** pour afficher des nombres. Si vous n'utilisez pas DEC, ni HEX, les nombres seront représentés par leur code ASCII. Utilisez également **DEC** ou **HEX** pour afficher l'état des variables.

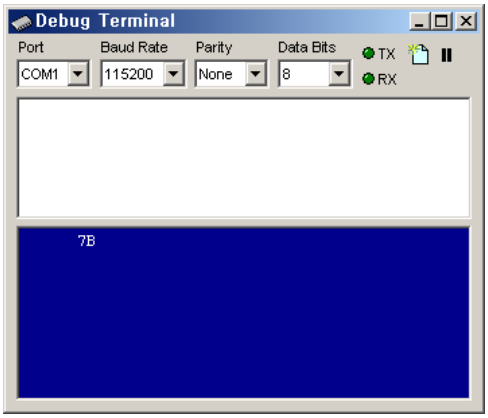
Si vous insérez un point d'interrogation (**?**) avant DEC ou HEX, le nom de la variable sera affiché en même temps que sa valeur.

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



Vous pouvez également utiliser des nombres pour limiter le nombre de décimal à afficher.

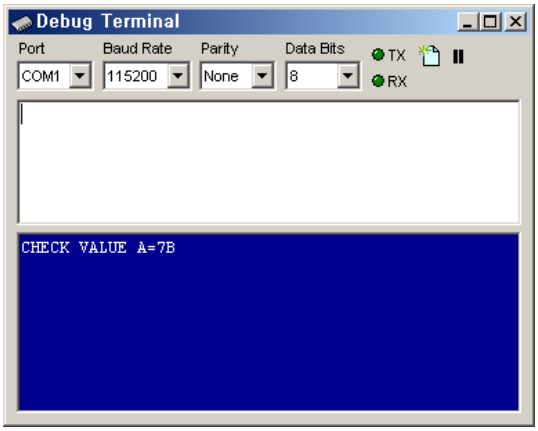
```
DEBUG  HEX8  A
```



1 à 8 peuvent être utilisés avec HEX. HEX8 affichera un nombre héxadécimal à 8 digits max.
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 vous aider à trouver rapidement et facilement vos « bug » de programmation. Elles vous permettront d’une part de vous assurer que le programme est en train de s’exécuter à un endroit particulier (celui où vous avez placé la commande DEBUG) et d’autre part, ces dernières vous permettront 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 sera réservée pour un futur usage.

ATTENTION !
Les commandes de DEBUG du BASIC ne doivent JAMAIS être utilisées en même temps que le mode monitoring du LADDER. De la même façon, le mode monitoring du LADDER ne doit JAMAIS être utilisé en même temps que les commandes de DEBUG du BASIC.

Le tableau ci-dessous montre les différentes possibilités offertes par les commandes DEBUG (comme vous pourrez le constater vous retrouverez de grande similitude avec les commandes d'afficheurs LCD).

Commande	Code	Explications	Exemple d'utilisation
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	Equivalent "Line Feed"	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 commandes DEBUG.

```
DEBUG GOXY,5,5,DEC I
DEBUG CLR,"TEST PROGRAM"
```

Decr

DECR variable
Variable : Variable à décrémenter (Non String, ni Single).

Décrémente la variable d'une unité (similaire à "A - -" en langage « C »)

```
DECR A          ' Décrémente A d'une unitée.
```

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 l'utilisation de cette commande pour réaliser des mesures de durées ou des applications de gestion temporelle pour lesquelles la précision doit être prépondérante.

```
DELAY 10        ' Delai d'environ 10 ms.
DELAY 200       ' Delai d'environ 200 ms.
```

A titre « didactque », l'instruction Delay reviendrait à réaliser cette sous-routine :

```
sub delay(dl as long)
  dl1 var long
  dl2 var integer
  for dl1=0 to dl
  for dl2=0 to 1
      nop
      nop
      nop
  next
  next
end sub
```

Do...Loop

La commande **DO...LOOP** réalise une boucle sur elle-même. Les commandes **DO WHILE** ou **DO UNTIL** réalisent également une boucle sur elles-mêmes associée à une condition permettant de sortir de cette boucle. La commande **EXIT DO** permet également de forcer le programme à sortir d'une boucle de type DO...LOOP.

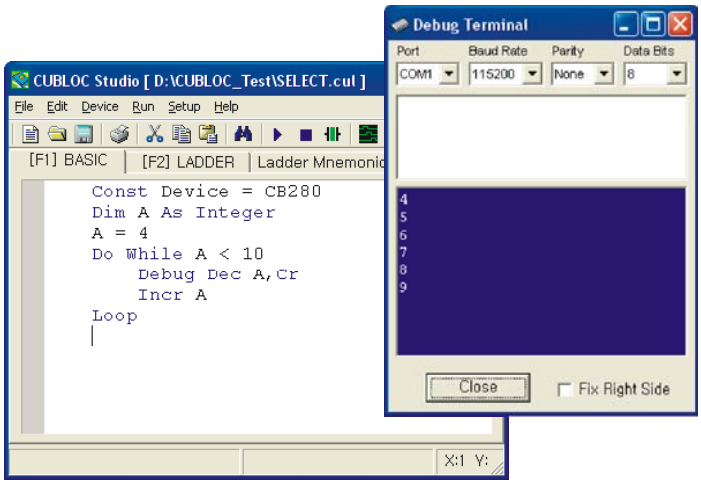
```
Do
      Commands
Loop

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 sans fin entre **DO** et **LOOP**. EXIT DO or GOTO command must be used to get out of the infinite loop.

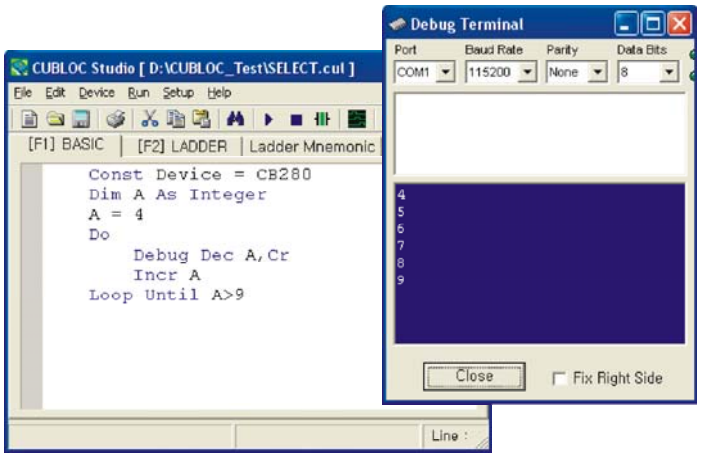
```
Do While [Condition]
      Commands
      [Exit Do]
Loop

Do
      Commands
      [Exit Do]
Loop While [Condition]
```

DO..WHILE réalisera une boucle sans fin jusqu'à ce que la condition en WHILE soit remplie.

```
Do Until [Condition]
        Commands
        [Exit Do]
Loop

Do
        Commands
        [Exit Do]
Loop Until [Condition]
```



DO..UNTIL réalisera une boucle sans fin jusqu'à ce que la condition en UNTIL soit remplie.



Dtzero

DTZERO variable

Variable : Variable à décrémenter (Non String, ni Single).

Décrément la variable d'une unité. Lorsque la variable arrive à 0, celle-ci n'est plus décrémentée.

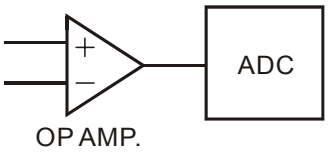
```
DTZERO A        ' Décrément A d'une unité.
```

Eadin( )

Variable = EADIN (mux)

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

Cette commande est utile si vous désirez obtenir une conversion « A/N » plus précise. Les modules CUBLOC™ disposent d'un OPAMP intégré. Lorsque vous utilisez la commande ADIN, cet OPAMP n'est pas exploité. En utilisant la commande EADIN, l'utilisateur pourra mettre en œuvre l'OPAMP pour un résultat plus précis.



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 (Utilisé 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 temprisation
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 EADIN. Lorsque vous utilisez les multiplicateurs X1 et X10 vous travaillez avec une résolution sur 8 bits. Lorsque vous utilisez des multiplicateurs X8 et 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 des canaux de conversion « A/N » et des 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 (Non String, ni Single).
Addresse : 0 à 4095
NbOctet : Nombre d'octets à lire (1 à 4)

Lecture de données depuis une adresse en EEPROM.

```
DIM A AS INTEGER
DIM B AS INTEGER
A = 100
EEWRITE 0,A,2          ' Mémorise la variable A à l'adresse 0.
B = EEREAD(0,2)        ' Lecture depuis l'adresse 0 et mémorisation dans B.
```

Eewrite

EEWRITE Addresse, Data, NbOctet
Addresse : 0 à 4095
Data : Variable à écrire en EEprom (Type Long ou inférieur)
NbOctet : Nombre d'octets à écrire (1 à 4)

Mémorise des données à une adresse spécifique de la mémoire EEPROM.

```
Dim A As Integer
Dim B As Integer
A = 100
EEWRITE 0,A,2          ' Mémorise la variable A à l'adresse 0.
B = EEREAD(0,2)        ' Lecture depuis l'adresse 0 et mémorisation dans B.
```

Attention pensez que la durée d'écriture en EEPROM nécessite généralement 3 à 5 millisecondes.

La lecture de données depuis l'EEPROM ne dispose pas de contrainte temporelle.

Pour rappel, sachez également que la limite d'écriture en mémoire EEprom est de l'ordre de 100.000 cycles (évitez impérativement les boucles sans fin dans lesquelles une écriture en EEprom aurait lieu).

Si vous envisagez de développer une application nécessitant des stockages de données fréquents et nombreux, il conviendra de ne pas utiliser la mémoire EEProm, mais plutôt la mémoire SRAM sauvegardée par pile (comme cela est possible sur le module CB290).

Le tableau ci-dessous présente un petit comparatif entre les mémoire de type SRAM et EEPROM.

Type	Battery Backup SRAM	EEPROM
Rétention des données	Quelques heures à 1 an (en fonction de la capacité de la pile)	40 ans
Nombre de cycles d'écritures max.	Infini	Environ 100,000
Durée d'écriture	Quasiment immédiat	3 to 5 ms
Utilisation	Système d'acquisition (type data-looging)	Mémoirisation de petite quantité de données (paramétres d'un système par exemple)

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)

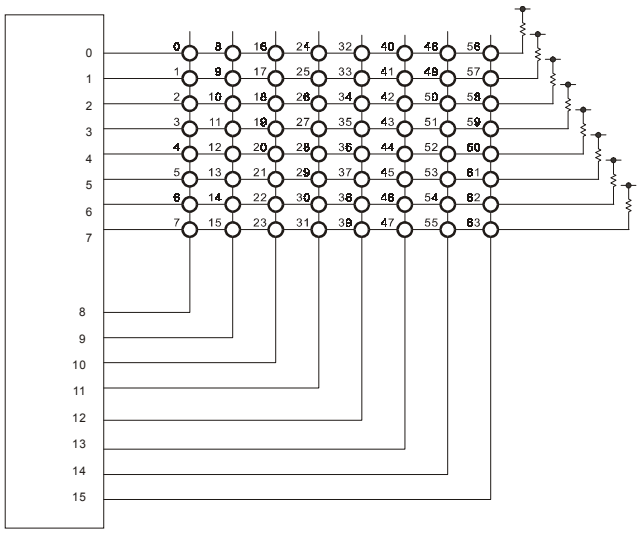
La commande **EKEYPAD** permet d'étendre les possibilités de la commande KEYPAD afin de pouvoir lire jusqu'à 64 touches. 2 blocs de ports devront à ce titre être mis à profit. Le bloc de port d'entrées (avec résistances de tirage à ramener au +5 V) et le bloc de port de sorties devront être sélectionnés 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 resté « 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 sollicité sera retourné.

For...Next

FOR…NEXT réalisera une boucle sur elle-même pendant un nombre de fois donné.

```
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 utilisé. 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 que cette dernière 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.

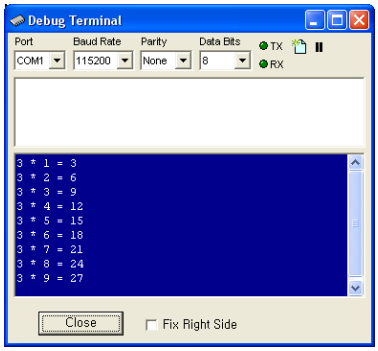
```
Dim K As Byte
For K=0 To 255
        Debug Dp(K),CR
Next
```

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

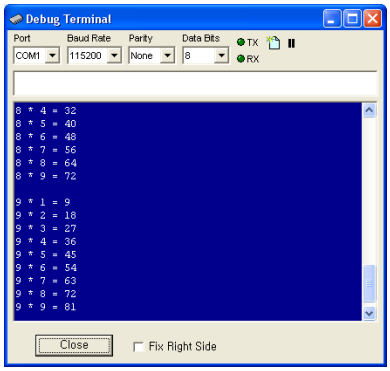
```
Dim LK As Long
For LK=255 To 0 Step –1     ' Le nombre -1 sera atteind en fin de boucle
        Debug Dp(LK),CR
Next
```

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.

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

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 corrélation entre le paramètre ValFreq et la valeur de 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.

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

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

ValFreq = 2304000 / fréquence désirée

Avant de pouvoir générer une fréquence, il vous faut le port PWM en sortie. Pour stopper le signal, vous pouvez 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 d'utilisation 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.

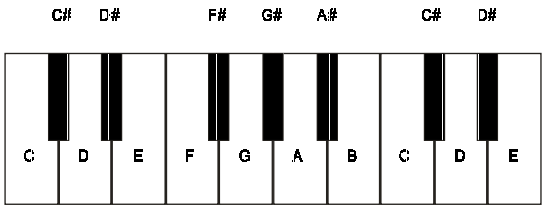
A noter qu'il est possible de générer différentes fréquence sur les canaux PWM 0 et 3.

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 (entre autre) être utile pour générer des notes de musiques. Le tableau ci-dessous donne une correspondance entre la valeur du paramètre ValFreq et les notes de musique.

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 donc être à 1. Pour recevoir une donnée de type Long, le paramètre NbData devra donc être à 4. Pour pouvoir recevoir une plus grande quantité de donnée, utilisez la commande **GETSTR()**.

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

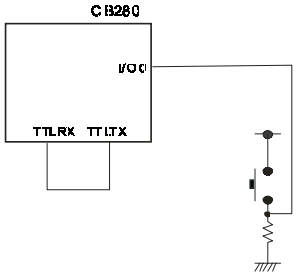
Geta

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 qui seront stockés 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 « stoppant »

La commande **GETA2** s'utilise de la même façon que la commande GETA à l'exception que la lecture des données sera stoppée à la détection du caractère ASCII CarStop (même si d'autres données reste 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 lire 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 ASCII 10 soit trouvé ou que
                             ' 20 octets aient été reçus.
```

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

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. La commande retourne une valeur de CRC sur 16 bits de type Integer via le tableau. Vous pouvez déterminer le nombre d'octets à utiliser pour le calcul du CRC à partir du tableau en partant de 0.

```
Const Device = CB280
Opencom 1,115200,3,80,20
Set Modbus 1,9
Dim A(20) As Byte
Dim B As Integer
Ramclear
Usepin 0,Out
Usepin 9,Out

Set Ladder On

A(0) = 9
A(1) = 2
A(2) = 3
A(3) = 0
A(4) = 10
A(5) = 23

Getcrc B,A,6            'Nom du tableau
Debug Hex B,Cr
```

*** Utilisez uniquement un tableau de type Byte avec cette commande**

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

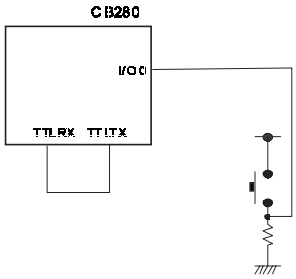
Getstr( )

Variable = GETSTR(canal, NbData)

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

Idem Get() mise 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
        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:
        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 « stoppant »

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

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

Gosub..Return

La commande GOSUB est destinée à appeler une sous-routine. La commande **RETURN** doit être utilisée pour clôturer 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 (appelé « étiquette »). Cette commande est présente sur la plupart des langages BASIC usuelle. Toutefois il n'est pas conseillé d'avoir recours à cette dernière si vous voulez conserver une programmation structurée

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

A propos des « étiquettes »…
Une « étiquette » sera définie en ajoutant à la fin le caractère ':'. 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:
```

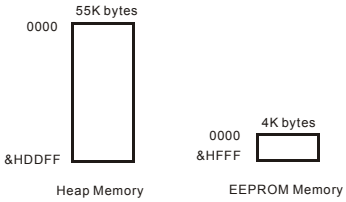
Une « étiquette » ne doit par contre pas utiliser de « mots » ou constantes réservés, ni de chiffre et encore moins contenir d'espace.

Ci-dessous quelques enxemple «d'étiquettes » **qu'il ne faut pas utiliser**:

```
Ladder:         ' Constante réservée (Problème)
123:            ' Etiquette avec des nombre (Problème).
Aboot 10:       ' Etiquette avec un espace (Problème).
```

Accès à la mémoire HEAP du module « CB405 »

La mémoire HEAP (uniquement disponible sur le module CUBLOC™ « CB405 ») peut être accédée depuis l'adresse 0 jusqu'à 56831 (octet par octet) de 0 jusqu'à &HDDFF en hex. Vous disposez ainsi de près de 55 K de mémoire. Cette mémoire est idéale pour stocker de grande quantité de données (graphiques, table de température, etc…). Avec une sauvegarde par pile externe (non livrée) la mémoire pour disposer d'une sauvegarde en cas de coupure d'alimentation.



Il existe 5 commandes donnant accès à la mémoire HEAP. Ces commandes ne fonctionnent que sur le « CB405 » et qu'avec le logiciel « Cubloc Studio » version 2.0.x et supérieur.

Commande	Syntaxe	Fonctionnalité
HEAPCLEAR	Heapclear	Efface la totalité de la mémoire HEAP
HREAD	Variable = HREAD(Adresse, Long)	Lecture d'un nombre d'octets (Long) depuis la mémoire HEAP à l'adresse (Adresse) et stockage dans une Variable
HWRITE	HWRITE Adresse, Variable, Long	Ecriture d'un nombre d'octets (Long) d'une Variable dans la mémoire HEAP à l'adresse (Adresse)
HEAPW	HEAPW Adresse, Variable	Mémorise un octet dans la mémoire HEAP à l'adresse (Adresse)
HEAP	Variable = HEAP(Adresse)	Lecture d'un octets depuis la mémoire HEAP à l'adresse (Adresse) et stockage dans une Variable

Hread( )

Variable = HREAD (Adresse , Long)

Variable : Variable pour le stockage du résultat
Adresse : adresse dans la mémoire HEAP
Long : Nombre d'octets à lire (contante ou variable de 1 à 4)

Cette commande permet de lire de 1 à 4 octets à la fois depuis la mémoire HEAP.

Hwrite( )

HWRITE Adresse, Donnée, Long

Adresse : adresse dans la mémoire HEAP
Donnée : Constante ou variable contenant la donnée
Long : Nombre d'octets à écrire

Cette commande permet d'écrire des données dans la mémoire HEAP.

```
DIM A AS INTEGER
DIM B AS INTEGER
A = 100
HWRITE 0,A,2            ' Mémorise la variable de type Integer A à l'adresse 0
B = HREAD(0,2)          ' Lecture données sur 2 octets depuis l'adresse 0 et stock dans B
```

NOTE :
Bien que la syntaxe des commandes EEREAD et EEWRITE soit identique à la syntaxe des commandes HREAD et HWRITE, il est important de rappeler les différences qu'il existe entre les 2 types de commandes.

Commande	Type de mémoire	Caractéristiques
EEWRITE, EEREAD	EEPROM	La sauvegarde (en cas de coupure d'alimentation) des données mémorisées s'effectue sans pile externe La commande EEWRITE nécessite un miniumum de 5 ms pour être réalisée Vous ne disposez que de 4 K de mémoire à votre disposition
HREAD, HWRITE	SRAM	La sauvegarde (en cas de coupure d'alimentation) des données mémorisées nécessite une pile externe La commande HWRITE nécessite un miniumum de 20 µS pour être réalisée Vous disposez 55 K de mémoire à votre disposition

Heapclear

HEAPCLEAR

Met toute la mémoire HEAP (les 55 K) à la valeur 0

Heap( )

Variable = HEAP (Adresse)

Variable : Variable pour le stockage du résultat
Adresse : adresse dans la mémoire HEAP

Cette commande permet de lire de 1 octet depuis la mémoire HEAP.

Heapw

HEAPW Adresse, Donnée

Adresse : adresse dans la mémoire HEAP
Donnée : constante ou variable avec la donnée à sauvegarder (Byte uniquement)

Cette commande permet d'écrire 1 octet dans la mémoire HEAP.

A propos de l'adressage de la mémoire « HEAP »
L'adressage de la mémoire « Heap » est accessible octet par octet. Si vous désirez stocker une variable de type « Long », vous mémoriserez alors 4 octets à la fois et 4 emplacements mémoires seront monopolisés.

HWRITE 0, &H1234ABCD, 4

0	CD
1	AB
2	34
3	12

La table ci-dessus représente les 4 emplacements mémoire utilisés pour le stockage de la variable de type « LONG ». Dès lors, il vous faudra être vigilent dans la gestion des adresses de votre mémoire « Heap » afin d'éviter que des données ne se chevauchent (comme c'est le cas dans le petit exemple de code ci-dessous).

```
HWRITE 0, &HABCD, 2
HWRITE 1, &H6532, 2
```

Dans cet exemple de code, il y a un chevauchement entre les 2 données stockées au niveau de l'adresse mémoire 1 de la mémoire « Heap ».

Programme de démo montrant l'utilisation de la mémoire « Heap »

```
Const Device = CB405
Dim A As Byte
Dim I As Long,J As Long

I = &HABCD1234
Heapclear
Hwrite 0,I,4
Do
  Heapw 56830,100
  Heapw 56831,123

  Debug Dec Heap(56830),Cr
  Debug Dec Heap(56831),Cr
  J = Hread(0,4)
  Debug Hex J,Cr
  Delay 100
Loop
```

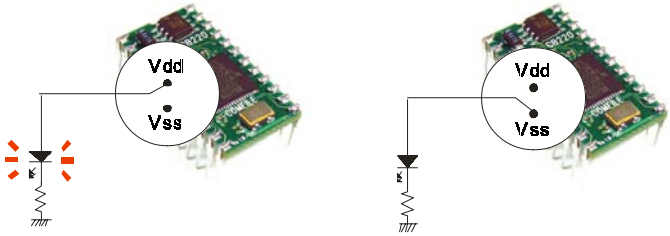
High

HIGH Port
Port : N° de Port du module CUBLOC™

Cette commande applique un niveau logique HAUT (+5 V) sur le Port du module CUBLOC™.
Cette commande 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.
```

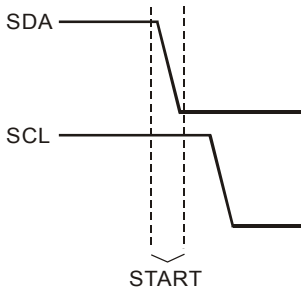
Lorsque qu’un Port est activé par la commande HIGH, ce dernier se retrouve connecté à VDD (lorsqu’il est configuré en LOW, ce Port est connecté à VSS).



I2Cstart

I2CSTART

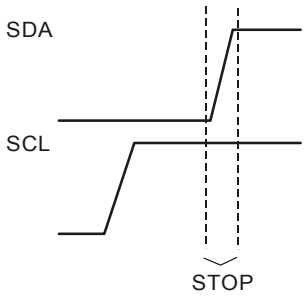
Génère (via les signaux SDA et SCL) une condition « Start » sur le bus I2C™ . Après cette commande les signaux SDA et SCL sont au niveau logique BAS.



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.



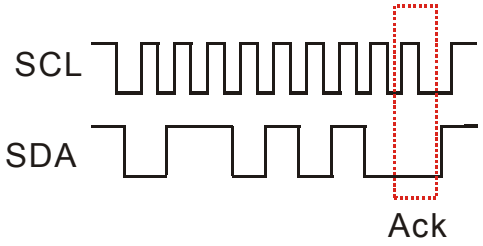
I2Cread( )

Variable = I2CREAD(dummy)

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

Lecture d'un octet depuis le bus I2C™ (pré-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™.

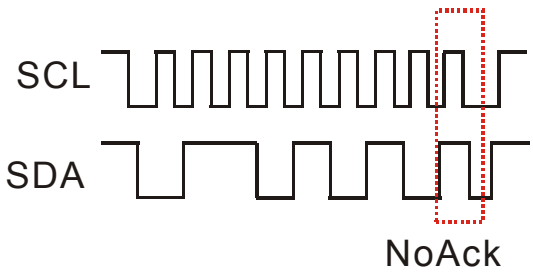
I2Creadna( )

Variable = I2CREADNA(dummy)

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

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

A = I2CREADNA(0)



Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

I2Cwrite( )

Variable = I2CWRITE donnée

Variable : Acknowledge(1 = Acquittement, 0= Sans Acquittement)
donnée : Donée à envoyer

Envoi un octet sur le bus I2C™. Cette commande génère une impulsion d'acquittement sur le bus I2C™ et retourne la valeur 0 si l'acquittement du composant adressé survient et 1 si le composant n'a pas envoyé de signal d'acquittement. Ce cas de figure peut intervenir à cause de plusieurs cas de figure: Adresse du composant I2C™ mal configurée, mauvais raccordement des signaux SDA et SCL, problème d'alimentation, problème sur le composant I2C™, etc… Il est intéressant dans ce cas de prévoir une vérification de la bonne communication I2C™ (voir exemple ci-dessous):

```
IF I2CWRITE(DATA)=1 THEN GOTO ERR_PROC
```

Lorsque vous n'avez pas besoin de traiter l'information d'acquittement, vous pouvez utiliser n'importe quelle variable pour recevoir cette information (voir exemple ci-dessous):

```
A = I2CWRITE(DATA)
```

La transmission d'un octet nécessite environ 60 µs
Consultez le chapitre « A propos du Bus I2C™… » pour plus d'infos.

If..Then..Elseif…Endif

Vous pouvez utiliser la commande **If…Then…Elseif...Else…EndIf** afin de réaliser des actions « conditionnées » dans votre programme.

```
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 après 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
```

Exemple N° 6 de « rédaction » possible

```
If  A<10  Then
      B=1
Elseif A<20 Then
      C=1
Elseif A<40 Then
      C=2
Else
      D=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 une 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 (BAS ou HAUT) 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 configurées en haute impédance (HIGH-Z)à la mise sous tension.

Lorsqu'un Port est configuré en sortie, la sortie pourra alors être amener à prendre un niveau logique « BAS » (0 V) ou « HAUT » (+ 5 V).

Incr

INCR variable
Variable : Variable à incrémenter.

Incrémente la variable d'une unité.

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

Input

INPUT Port
Port : Port du module CUBLOC (0 à 255)

Configure le Port spécifiée en enrée haute impédance (High-Z).
Toutes les entrées des CUBLOC™, sont configurées en entrées haute impédance à la mise sous tension du module. Ceci signifie que la résistance d'entrée est si élevée, que le niveau n'est jamais HAUT, ni BAS.

INPUT 8 Configure le Port 8 en entrée haute impédance.

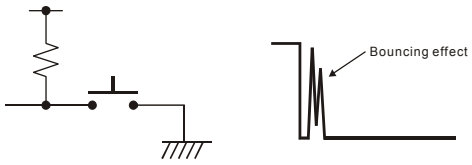
Keyin

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 ce peu 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 auant que faire ce peu 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.
```

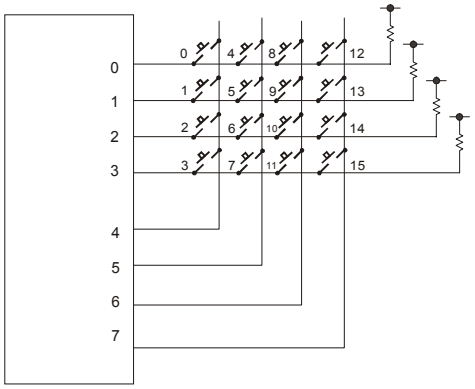
Keypad

Variable = KEYPAD(BlocPort)

Variable : Variable servant à mémoriser le résultat (Retourne un octet « type Byte »)
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 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 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 sollicité sera retourné.

Ladderscan

LADDERSCAN

La commande **LADDERSCAN** configure le module CUBLOC™ pour qu'il ne soit utilisé qu'en mode LADDER (sans possibilité d'utliser de programme BASIC). Lorsque la commande est placée dans une boucle sans fin telle que DO…Loop, il en résulte que la vitesse de traitement du LADDER pourra atteindre jusqu'à 10 ms par « time scan ».

```
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
```

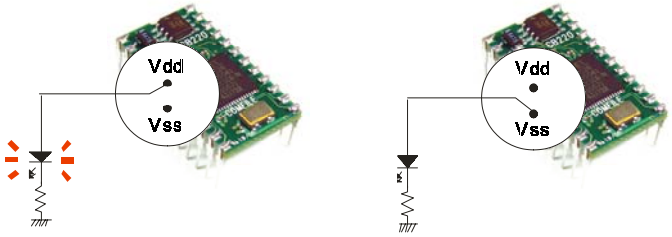
Low

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 applique un 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, ce dernier se retrouve connecté à VDD (lorsqu'il est configuré en LOW, ce Port est connecté à VSS).



Memadr( )

Variable = MEMADR (VariableCible)
Variable : Variable servant à mémoriser le résultat (non String, ni Single)
VariableCible : Variable servant à touver l'adresse mémoire physique

Comme pour le langage « C », vous pourrez utiliser des pointeurs dans le BASIC des modules CUBLOC™. En utilisant ces derniers, vous pourrez trouver l'adresse mémoire physique de variable en RAM afin de pouvoir y stocker ou y lire des données.

```
Dim A as Single
Dim Adr as Integer
Adr = Memadr(A)          ‘ Retourne l’adresse physique de la variable A.
```

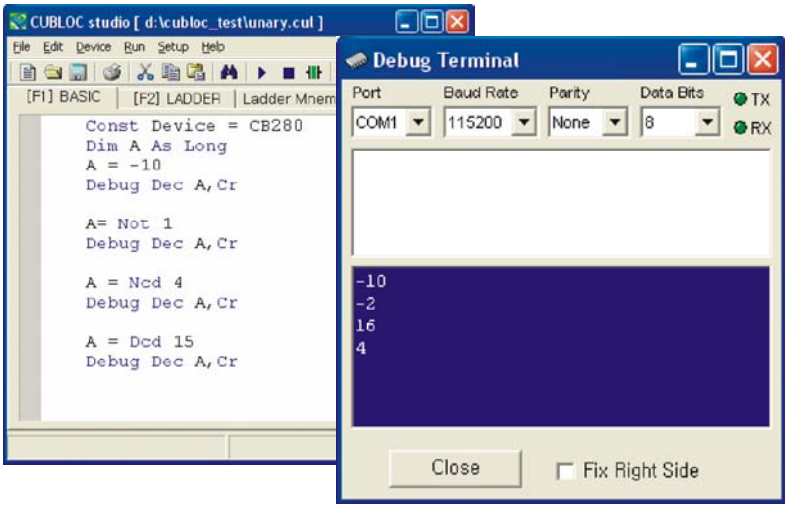
Ncd

Variable = NCD source

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

La commande **NCD** peur être utilisé pour mettre à 1 le bit de la valeur 0x00000000 et de retourner le résultat sous la forme d'une valeur 32 bits.

```
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

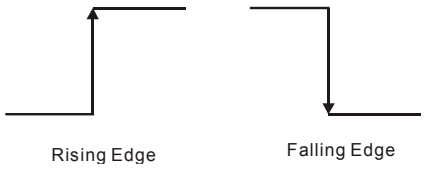
ON INTx GOSUB Etiquette
x : 0 à 3, Canal d'interruption externe

La commande **ON INT** doit être utilisée afin de pouvoir 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
Do
Loop

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

On Ladderint Gosub

ON LADDERINT GOSUB Etiquette

Si le Registre **F40** est utilisé en LADDER et que la commande **ON LADDERINT GOSUB** est également utilisée alors le CUBLOC effectuera un saut à l'étiquette spécifiée dans la commande On Ladderint. Cette fonctionnalité permettra au LADDER d'exécuter un « bout » de programme BASIC à un moment précis.

Utilisez les commandes **SETOUT** et **DIFU** pour écrire 1 dans le registre F40. Lorsque l'interruption en BASIC est finie, le registre F40 pourra être initialisé en y écrivant 0.

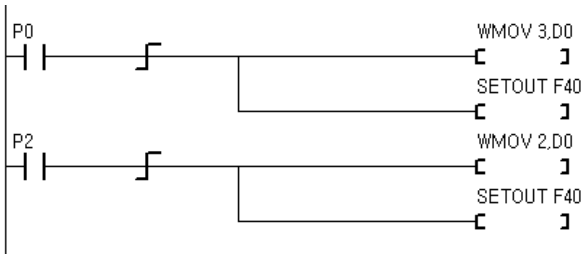
Pendant la durée de l'interruption, le fait d'écrire un 1 dans le registre F40 n'aura pas pour conséquence de générer une autre interruption.

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.

```
        Usepin 0,In
        Set Ladder On
        Set Display 0,0,16,77,50
        On Ladderint Gosub msg1_rtn
        Dim i As Integer
        Low 1

        Do
              i=i+1
              Byteout 1,i
              Delay 200
        Loop
msg1_rtn:
        Locate 0,0
        Print "ON Ladderint",Dec i
        Reverse 1
        Return
```

Lorsque P0 est mis à ON, le registre F40 sera mis à 1 et la sous-routine d'interruption BASIC msg1_rtn sera exécutée. Cette sous routine affiche un message sur un écran LCD.

Bien qu'il n'y ait qu'un seul registre F40 pour générer une interruption en BASIC depuis le LADDER, il est possible d'utiliser le registre D pour pouvoir générer plusieurs programmes d'interruptions BASIC.



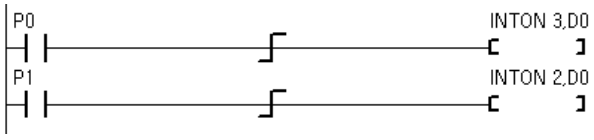
Lorsque P0 est activé, D0 prend la valeur 3 et une interruption sera générée par le BASIC.

Lorsque P0 est activé, D0 prend la valeur 2 et une interruption sera générée par le BASIC.

Il vous suffira alors dans le programme BASIC d'interruption d'exécuter telle ou telle routine en fonction de la valeur présente dans D0.

```
msg1_rtn:
        If _D(0)=3 Then
                Locate 0,0
                Print "ON Ladderint",Dec i
        End If
        If _D(0)=2 Then
                Locate 0,0
                Print "TEST PROGRAM",Dec i
        End If
        Return
```

Le programme ci-dessous fait la même chose en utilisant la commande INTON du LADDER qui accomplie à la fois une commande WMOV et SETOUT en une seule opération.



On Pad Gosub

ON PAD GOSUB Etiquette

Les modules CUBLOC™ disposent de la possibilité de pouvoir piloter un module optionnel de gestion de clavier. Ce module se présente sous la forme d'une petite platine électronique conçue pour recevoir différents types de claviers matricés et pour communiquer l'état de leurs touches au module CUBLOC™ via une liaison 4 fils (de type esclave SPI™). Le recours à cette platine optionnelle permet d'une part d'économiser le nombre « d' E/S » nécessaire à la gestion d'un du clavier et d'autre part de pouvoir générer des interruptions sur le CUBLOC™ suite à la sollicitation des touches du clavier (voir commande SET PAD pour plus d'info).

La commande ON PAD générera un « saut » à une routine d'interruption « étiquette » lorsque la quantité de données présente dans le buffer du module optionnel de gestion de clavier aura été atteinte (selon la valeur que vous aurez fixée dans la commande SET PAD).

Utilisez impérativement une commande Return à la fin de la sous-routine d'interruption.

```
        Const Device = Ct1720
        Dim TX1 As Integer, TY1 As Integer
        Contrast 450
        Set Pad 0,4,5
        On Pad Gosub GETTOUCH
        Do
        Loop

GETTOUCH:
        TX1 = Getpad(2)
        TY1 = Getpad(2)
        Circlefill TX1,TY1,10
        Pulsout 18,300
        Return
```

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
Do                               ' arrivent sur le port RS232 du canal 1
Loop                             ' Boucle sans fin

DATARECV_RTN:
    If Blen(1,0) > 4 Then
        A(0) = Get(1,1)  ' Lecture d'un octet.
        A(1) = Get(1,1)  ' Lecture d'un octet.
        A(2) = Get(1,1)  ' Lecture d'un octet.
        A(3) = Get(1,1)  ' Lecture d'un octet.
        A(4) = Get(1,1)  ' Lecture d'un octet.
    End If
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 d'exécuter une nouvelle interruption ON RECVx. Après que la routine d'interruption ait terminé son exécution, le CUBLOC™ pourra revenir à la routine d'interruption ON RECVx si des données (dans le buffer de réception) ont encore été reçues.

On Timer

ON TIMER(intervalle) GOSUB étiquette

Intervalle : Intervalle d'intérruption 1 = 10 ms, 2 = 20 ms……65535 = 655350 ms
Peu prendre les valeurs 1 à 65535

La commande ON TIMER permet de générer des interruptions à intervalles réguliers afin de pouvoir réaliser cycliquement une sous-routine. Il vous faut pour ce faire indiquer dans la commande le nom de la sous-routine à réaliser 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 la durée de l'intervalle entre 2 interruptions. Si par exemple vous avez choisi un intervalle de 10 ms entre la génération de chaque intervalle, il faudra que la durée d'exécution du sous-programme compris entre le nom de l'étiquette et l'instruction RETURN de fin d'interruption soit plus court que 10 ms sans quoi un problème de collision et un dysfonctionnement interviendra 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 vous sera nécessaire d'utiliser au préalable 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 nécessaire).

Vous trouverez ci-dessous les différents débits supportés 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		Stop Bit	Bit	# of Bits
			0	0 = Aucune	0=1 bit de Stop	0	0 = 5 bits
			0	1 = Reservé*	1=2 bit de Stop	0	1 = 6 bits
			1	0 = Pair		1	0 = 7 bits
			1	1 = Impair		1	1 = 8 bits

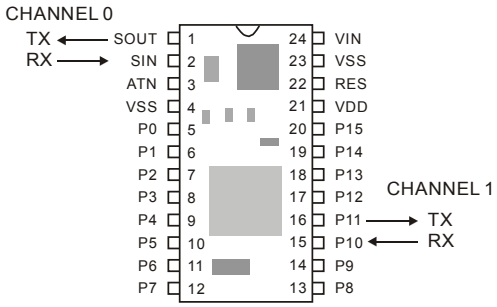
Le tableau ci-dessous montre les configurations les plus utilisées et les valeurs associées du paramètre « protocole » calculé par rapport au tableau ci-dessus.

Bits	Parité	Bit de Stop	Valeur du paramètre « protocole » à utilser
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
```

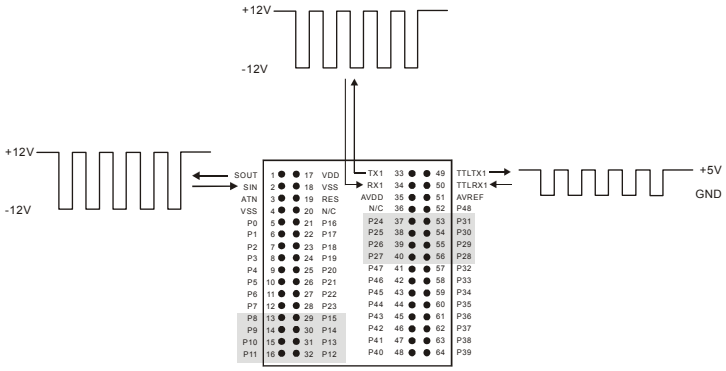
Vous pouvez configurer la taille du buffer de réception et d'émission. La taille des buffers de réception et d'émission influencera directement la taille de la mémoire RAM restante pour l'utilisation de vos variables. Si vous choisissez des buffers de taille très importante, vous ne pourrez pas utiliser beaucoup de variables. Bien que la taille max. de chaque buffer puisse aller jusqu'à 1024 octets, nous vous suggérons pour les applications standards de configurer la taille du buffer de réception de 30 à 100 octets et celle du buffer d'émission de 30 à 50 octets.

Pour le CB220, les ports 1 et 2 peuvent être utilisés pour le canal RS232 N° 0 (niveaux +/- 12V). Les ports 10 et 11 peut être utilisés pour le canal RS232 N° 1 (niveaux TTL +5 / 0V).



Pour le module CB280, vous disposez de ports RS232 dédiés. Ainsi pour le canal 1, vous pouvez utiliser 2 types de signaux +/- 12V et niveaux TTL (+5 / 0V).

Attention, il vous faut toutefois utiliser un seul type de digal à la fois.



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.

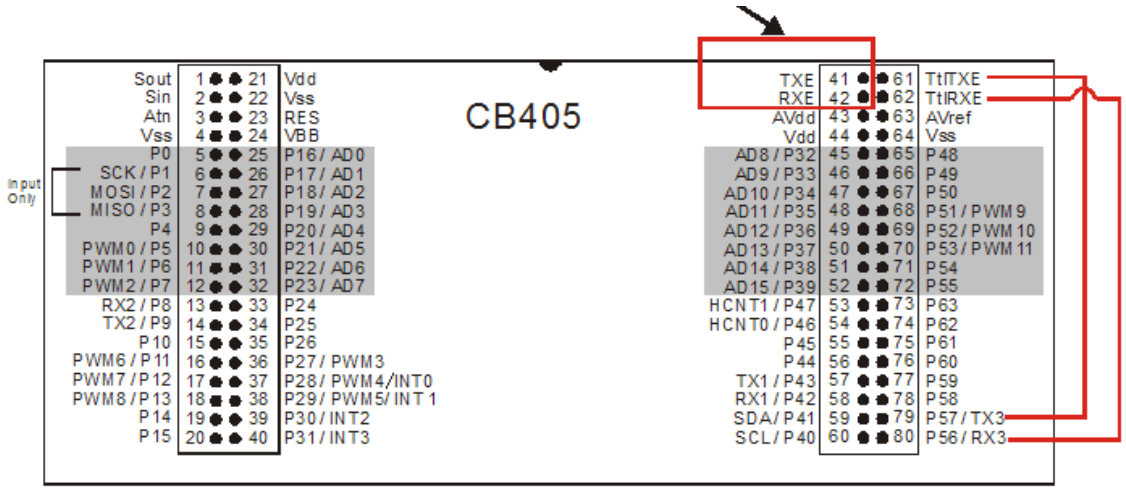
Comment utiliser les ports RS232 sur le module « CB405 » ?

Le tableau ci-dessous indique la correspondance des Ports du module « CB405 » et de ses ports de communication RS232 (**au niveau logique TTL 0 / 5 V**).

Canal	Port
1	P42 (RX Niveau logique TTL – 0 / +5 V) P43 (TX Niveau logique TTL – 0 / +5 V)
2	P8 (RX Niveau logique TTL – 0 / +5 V) P9 (TX Niveau logique TTL – 0 / +5 V)
3	P56 (RX Niveau logique TTL – 0 / +5 V) P57 (TX Niveau logique TTL – 0 / +5 V)

Le module « CB405 » dispose d'un composant interne de mise à niveau « MAX232 » dont les broches (TTLTXE et TTLRXE) pourront être connectées à l'un de ses Ports séries (niveau logique 0 / 5 V) afin que le port en question puisse être directement utilisé sur une liaison série au niveau logique +/- 12 V via les broches « TXE et RXE ».

Dans l'exemple ci-dessous, le Port série TX3 initialement doté d'un niveau logique 0 / 5 V pourra être exploité via les broches « TXE et RXE » sur une liaison série RS232 de niveau logique + /- 12 V.



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 permet d'appliquer un niveau logique sur un Port du module CUBLOC™. Lorsque vous exécutez cette instruction, le module CUBLOC™ configurera 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 lorsque vous utilisez cette commande.

```
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. Pour rappel, toutes les Ports du module CUBLOC™ sont configurés en mode haute impédance à la mise sous tension.

```
OUTPUT 8        ‘ Configure le Port 8 pour fonctionner 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() en ce sens qu'elle donne l'état du port, et non pas la valeur présente à son entrée.

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

Pause

PAUSE valeur

Commande équivalente à la commande DELAY

Peek( )

Variable = PEEK (Adresse, Long)

Variable : Variable servant à mémoriser le résultat
Adresse : Adresse RAM
Long : Nb d'octets à lire (1 à 4)

Lecture d'un nombre d'octets de données à partir d'une adresse en RAM

Poke

POKE Adresse, Valeur, Long

Addrese : Adresse RAM
Valeur : Variable servant à mémoriser le résultat
Long : Nb d'octets à écrire (1 à 4)

Ecriture d'un nombre d'octets de données à partir d'une adresse en RAM.

Attention cette instruction doit être utilisée avec précaution afin de ne pas dépasser les limites de la RAM sous peine de dysfonctionnement du module CUBLOC™.

```
Const Device = CB280
Dim F1 As Single, F2 As Single
F1 = 3.14
Eewrite 10,Peek(Memadr(F1),4),4
Poke Memadr(F2),Eeread(10,4),4

Debug Float F2,CR
```

Pulsout

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

Cette sous librairie vous pemettra de générer une impulsion.

Pour générer une impulsion HAUTE, la sortie doit au préalable être placée au niveau logique BAS.

Pour générer 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.

```
LOW 2                          HIGH 2
PULSOUT 2, 100 ' 23 mS         PULSOUT 2, 100 ' 23 mS
HIGH Pulse                     LOW Pulse
```

```
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 envoi des données au travers du port RS232 spécifié. Les données peuvent être de type « variable » ou « constante ». Pour envoyer une chaîne de caractères, il vous faudra utiliser la commande **PUTSTR**.

```
OPENCOM 1,19200,0,50,10
DIM A AS BYTE
A = &HA0
PUT 1,A,1              ‘ Envoi &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.

Si le buffer est plein lorsque la commande **PUT** est exécutée, la commande PUT ne va pas attendre que le buffer se vide (en d'autre terme vous allez perdre les données à envoyer…). Il est donc important d'utiliser conjointement la commande **BFREE** afin de s'assurer de l'état du buffer afin d'éviter ce cas de figure.

```
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() vérifie combien d'octets de libre le buffer dispose.

Astuce :
Après avoir utilisé les commandes **PUT** ou **PUTSTR**, la fonction **SYS(0)** peut être utilisée 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 de données à envoyer (1 à 65535)

La commande **PUTA** peut être utilisée pour envoyer les données (présentent dans un tableau) sur la liaison RS232.

Pour ce faire, il vous suffit d'indiquer le nom du tableau et le nombre de données 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 contient le tableau, le module CUBLOC™ enverra des données plus ou moins 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 « stoppant »

La commande **PUTA2** s'utilise de la même façon que la commande PUTA à l'exception que la transmission des données sera stoppée à la détection du caractère ASCII CarStop (ce caractère sera le dernier envoyé).

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

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 au travers de la liaison série. Comme précédément, prenez soin à ne pas dépasser la capacité du buffer d'émission sans quoi vous risquez de perdre 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 permet de générer 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 à utiliser aient bien été configurés en SORTIE.

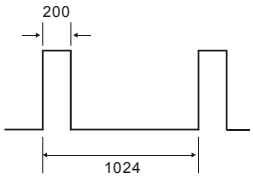
En fonction de la valeur du paramètre Periode, un signal PWM avec une résolution max. de 16 bits pourra être créé.

Lorsque Periode est configuré à 1024, la résolution du signal PWM sera de 10 bits.

Lorsque Periode est configuré à 65535, la résolution du signal PWM sera de 16 bits.

La valeur du paramètre Duty devra être être inférieur 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. Leur paramètre duty peut en revanche être différent.

Les signaux PWM 3, 4 et 5 doivent avoir la même valeur pour le paramètre Periode car ils utilisent les mêmes ressources. Leur paramètre duty peut en revanche être différent.

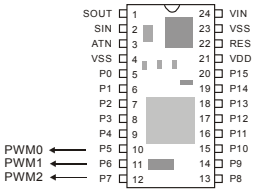
Pwmoff

PWMOFF Canal

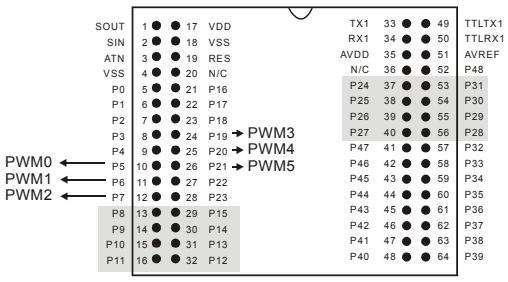
Canal : Canal PWM (0 à 15)

Stop 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 « incertaines ».

* 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 les valeurs «incertaines » présentent dans la RAM lors de la première 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()** génère 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'à pas d'importance.

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

La génération des nombres est « pseudo » aléatoire en ce sens que la même séquence de suite de nombres se répétera à chaque mise sous tension (d'où le nom de « pseudo » aléatoire).

Select...Case

Si le « cas » est vrai (c'est à dire qu'il correspond à la même valeur que la variable) ou qu'il rempli la condition IS, la commande sous le « cas » sera exécutée.

```
Select Case Variable
      [Case Value [,Value],…
            [Statement 1]]
      [Case Value [,Value],…
            [Statement 2]]
      [Case Else
            [Statement 3]]
End Select
```

Exemples :

```
Select Case A
      Case 1
            B = 0
      Case 2
            B = 2
      Case 3,4,5,6      ‘ Utilisez la virgule (,) pour plus d’une valeur.
            B = 3
      Case Is < 1       ‘ Utilisez < pour les opérations logiques.
            B = 3
      Case Else         ‘ Utilisez ELSE pour tous les autres cas.
            B = 4
End Select
```

```
Select Case K
      Case  Is < 10           ‘ Si inférieur à 10
            R = 0
      Case Is < 40            ‘ Si inférieur à 40
            R = 1
      Case Is < 80
            R = 2
      Case Is < 100
            R = 3
      Case Else
            R = 4
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).

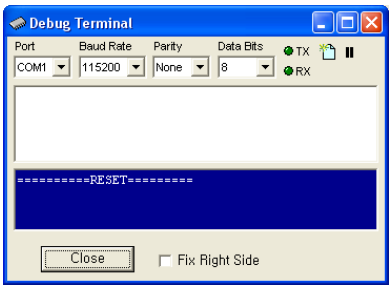
En savoir plus sur le mode Debug…

Correctement utilisé, le mode Débug vous permettra de développer vos applications encore plus facilement et rapidement en vous permettant d'identifier et de corriger efficacement vos « bugs » ou encore de simuler la présence d'un afficheur LCD. Vous trouverez ci-dessous quelques unes des possibilités du mode « Débug » :

1) Vérifier si le programme a effectué un « Reset »

Il peut être quelquefois intéressant de vérifier si le programme n'a pas réalisé un RESET (cas pouvant arriver si vous utilisez une mauvaise structure dans votre code). Mettez alors simplement une commande de DEBUG signifiant que le CUBLOC a effectué un RESET en début de programme. Si le message apparaît plus d'une fois après l'exécution du programme, c'est que le CUBLOC a effectué à nouveau un RESET (à vous de vérifier si cela est voulu ou pas dans votre application !).

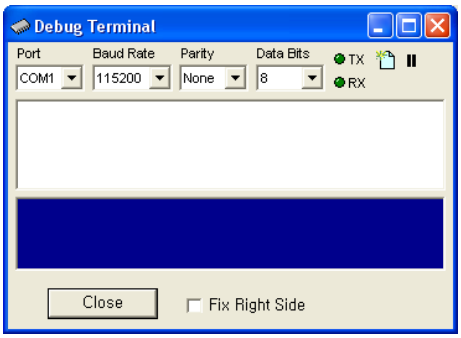
```
Const Device = CB280
Debug "==========Reset========="
Do
        High 0
        Delay 200
        Low 0
        Delay 200
Loop
```



2) Vérifier si le programme « passe bien à un endroit particulier »
Mettez simplement une commande de DEBUG signifiant que le CUBLOC passe à « cet endroit » et contrôler la progression du programme sur l'écran de DEBUG du PC.

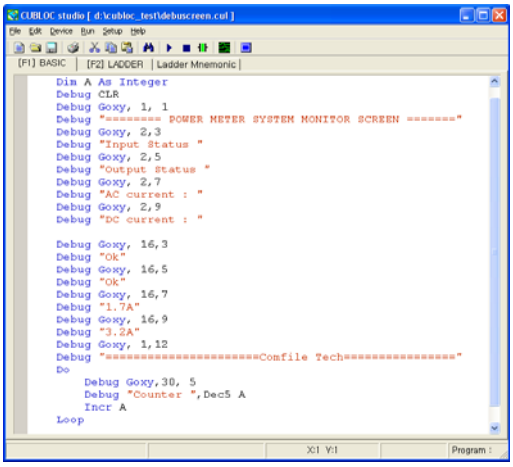
```
Const Device = CB280
Do
      High 0
      Delay 200
      Low 0
      Delay 200
Loop
Debug "Passe à cet endroit !"
```

Dans cet exemple le message ne s'affichera jamais car le CUBLOC est dans une boucle « do … Loop » sans fin.



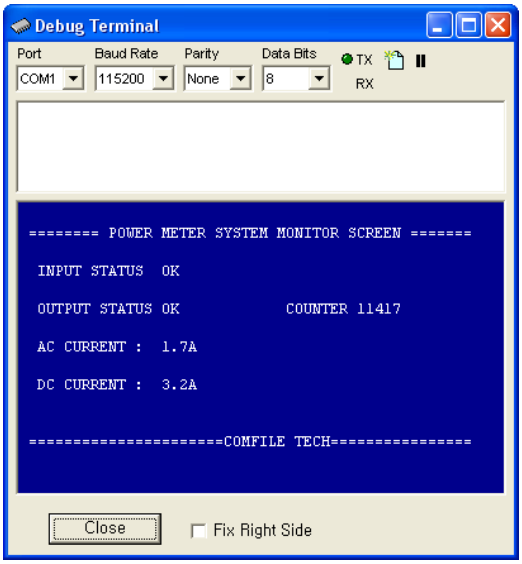
3) Simulez un écran LCD

Vous pouvez simuler un écran LCD en utilisant le terminal de DEBUG. Utilisez simplement les commandes goxy, XX, YY pour positionner le pointeur du LCD sur l'écran de Débug.



```
Dim A As Integer
Debug CLR
Debug Goxy, 1, 1
Debug "======== POWER METER SYSTEM MONITOR SCREEN ======="
Debug Goxy, 2,3
Debug "Input Status "
Debug Goxy, 2,5
Debug "Output Status "
Debug Goxy, 2,7
Debug "AC current : "
Debug Goxy, 2,9
Debug "DC current : "

Debug Goxy, 16,3
Debug "Ok"
Debug Goxy, 16,5
Debug "Ok"
Debug Goxy, 16,7
Debug "1.7A"
Debug Goxy, 16,9
Debug "3.2A"
Debug Goxy, 1,12
Debug "======================Comfile Tech================="
Do
    Debug Goxy,30, 5
    Debug "Counter ",Dec5 A
    Incr A
Loop
```

Utilisez la commande **Debug CLR** pour effacer l'écran de Débug du PC.



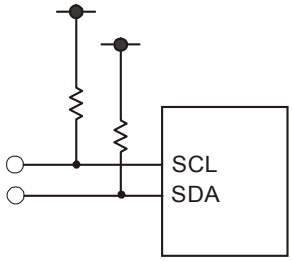
Comme indiqué précédement, il vous est possible de désactiver (sans avoir besoin de les retirer de votre code) toutes les commandes de Débug à l'aide de la commande **Set Debug Off**.

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™).

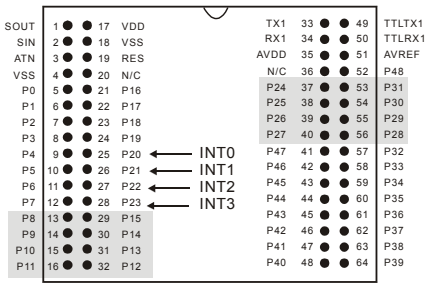
Set Int

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 on/off

SET LADDER On[/Off]

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

Do
Loop                       ' Boucle sans fin en BASIC
```

Set modbus

SET MODBUS mode, slaveaddress, returninterval
mode : 0=ASCII, 1=RTU
slaveaddress : Adresse esclave (1 à 254)
returninterval : Interval Retour (1 à 255)

Les modules CUBLOC™ supportent les communications avec le protocole MODBUS. Le bus MODBUS pourra être connecté sur le canal RS232 N° 1.

Pour activer le mode esclave MODBUS, utilisez la commande **SET MODBUS**. Cette commande doit intervenir après la commande OPENCOM et ne pourra être opérationnelle que sur le canal RS232 N° 1. Les paramètres Baurate, bit et parité devront être configurés avec la commande OPENCOM.

```
Opencom 1,115200,3,80,80      ‘ Configurez une taille de buffer de réception d’au
                              ‘ moins 50 octets.
Set Modbus 0,1                ' Mode ASCII, Adresse esclave = 1
```

Après cette commande, le module CUBLOC™ répondra automatiquement. Le module CUBLOC™ supporte les commandes MODBUS 1,2,3,4,5,6,15 et 16.

Commande	Nom de la commande
01, 02	Bit Read
03, 04	Word Write
05	1 Bit Write
06	1 Word Write
15	Multiple Bit Write
16	Multiple Word Write

Consultez le Chapitre 9 pour une description détaillée sur la gestion des communications de type MODBUS et des exemples associés.

Le terme Returninterval est la durée que le CUBLOC ou que le CUTOUCH mettra à répondre à un dispositif MODBUS maître. Si le paramètre Returninterval est trop rapide, le dispositif Maître ne pourra pas recevoir toutes les données. La valeur par défaut est 1 (ce qui correspond à environ 200 µs). L’utilisateur pourra également configurer cette valeur à 100 (ce qui correspond à une durée de 4 ms) ou à 255 (ce qui correspond à une durée de 11 ms).

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 toutes les désactiver 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 pouvoir 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 en « 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.

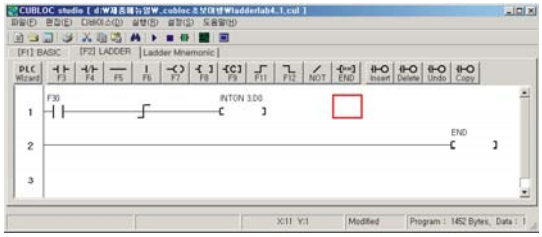
```
SET ONINT0 ON
SET ONINT1 ON
SET ONINT1 OFF
SET ONINT2 OFF
SET ONINT3 ON
SET ONINT3 OFF
```

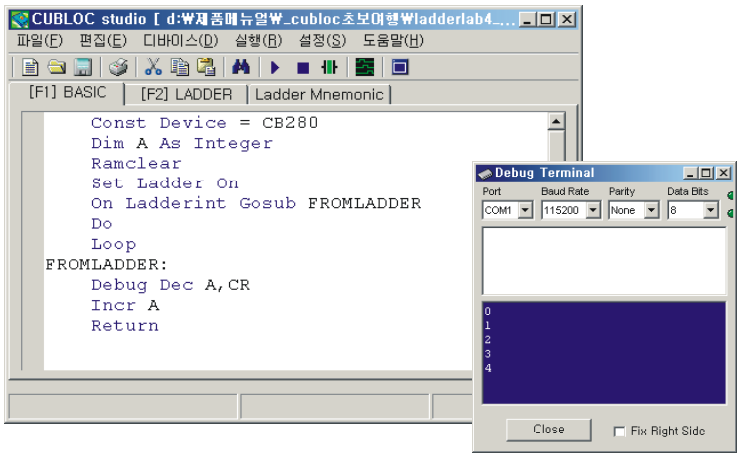
Set OnLadderint

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 » alors 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 ONRECV1 ON
SET ONRECV1 OFF

Set Ontimer

SET ONTIMER on[/off]

A la mise sous tension du CUBLOC™, le paramètre 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
```

Modèle de CUBLOC / CUTOUCH	Port utilisables en sortie uniquement
CB290	P24 à P55
CT1720 / CT1721	P24 à P55

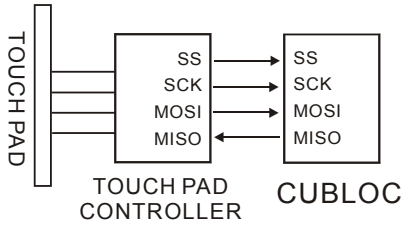
Set Pad

SET PAD mode, packet, NbDonnee

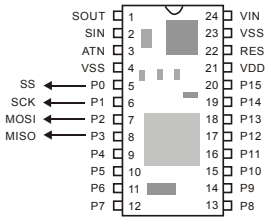
mode : Bit mode (0 à 255)
packet : Longueur du packet (1 à 255)
NbDonnee : Taille du buffer de réception (1 à 255)

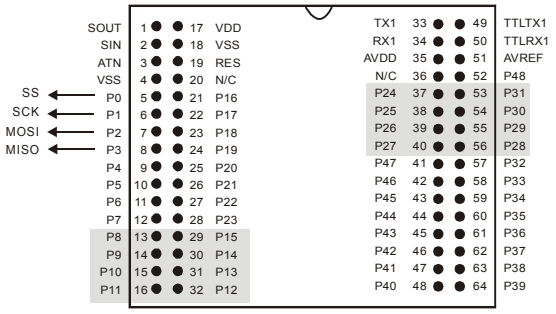
Les CUBLOC™ disposent de la possibilité de pouvoir piloter un module optionnel de gestion de clavier. Ce module se présente sous la forme d'une petite platine électronique conçue pour recevoir différents types de claviers matricés et pour communiquer l'état de leurs touches au module CUBLOC™ via une liaison 4 fils (de type esclave SPI™). Le recours à cette platine optionnelle permet d'une part d'économiser le nombre « d' E/S » nécessaire à la gestion d'un du clavier et d'autre part de pouvoir générer des interruptions sur le CUBLOC™ suite à la sollicitation des touches du clavier.

Afin de pouvoir exploiter le module optionnel de gestion de clavier en association avec les CUBLOC™, il vous faudra utiliser la commande Set Pad au début de votre programme. La communication entre le module CUBLOC™ et le module optionnel de gestion de clavier utilise 4 fils (SCK correspondant au signal d'horloge, SS comme signal de sélection « esclave », MOSI et MISO comme signaux de communication).



Les ports P0 à P3 seront monopolisés pour assurer la communication entre le CUBLOC™ et le module optionnel de gestion de clavier.

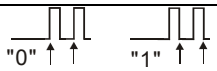




Le paramètre « Packet » correspond au nombre de données après lesquelles une interruption sera déclenchée sur le CUBLOC™. Par exemple si le module de gestion de clavier nécessite la réception de 4 octets avant de générer une interruption, alors le paramètre « Packet » devra être configuré avec la valeur 4.

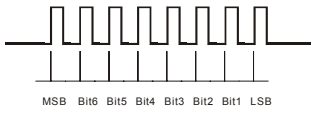
La paramètre « NbDonnee » correspond à la taille maxi du buffer de réception. La taille de ce buffer doit être supérieur d'au moins un octet par rapport à la taille du paramètre « Packet » (NbDonnée = Packet + 1). Une capacité importante du paramètre « NbDonnee » vous donnera essentiellement plus de temps pour exploiter la routine d'interruption. La taille généralement utilisée pour le buffer doit être de l'ordre de 5 à 10 fois supérieur à celle du paramètre « Packet ».

Le paramètre « Mode » permet de configurer le mode de réception des données (consultez le tableau ci-dessous pour plus d'infos).

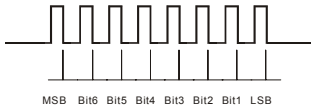
Mode	Valeur	Bit Pattern	Diagrame
LSB en premier	&H20	0010 xxxx	
MSB en premier	&H00	0000 xxxx	
SCK Low-Edge Triggered	&H08	xxxx 1xxx	
SCK High-Edge Triggered	&H00	xxxx 0xxx	
Sampling après SCK	&H04	xxxx x1xx	
Sampling avant SCK	&H00	xxxx x0xx	

Vous pouvez ajouter les valeurs des modes de réception. Par exemple pour sélectionner une réception avec les paramètres « MSB en premier, High-Edg Triggered SCK et Sampling après SCK : vous utiliserez le calcul 0x00 + 0x00 + 0x04 = 0x04). Vous trouverez ci-dessous d'autres exemples « types » pour la sélection du paramètre « Mode ».

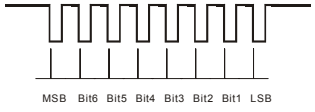
&H00



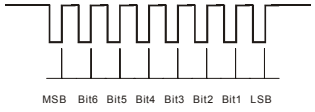
&H04



&H08



&H0C



La commande Set Pad configure automatiquement le fonctionnement des ports P0 à P3 (vous n'aurez donc pas à configurer ces derniers).

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 pouvoir 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 supportés 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		Stop Bit	Bit	# of Bits
			0	0 = Aucune	0=1 bit de Stop	0	0 = 5 bits
			0	1 = Reservé*	1=2 bit de Stop	0	1 = 6 bits
			1	0 = Pair		1	0 = 7 bits
			1	1 = Impair		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 » à utilser
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

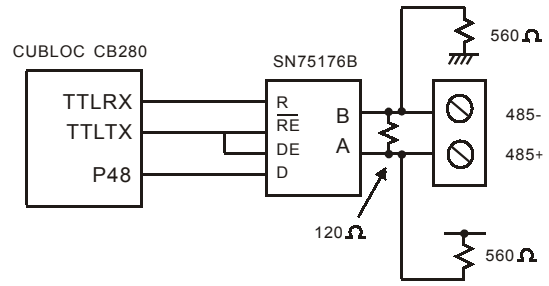
canal : canal RS232 (0 à 3 suivant type de CUBLOC™ utilisé)
NoPort : No d'activation du Port de transmission

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 ait un seul maître et que le reste des périphériques soient des esclaves.

Il vous faudra bien sûr adjoindre au module CUBLOC™ une interface électronique de mise à niveau RS485 (vous pourrez par exemple utiliser des circuits intégrés du type SN75176B ou un convertisseur de type RS232 <> RS485).

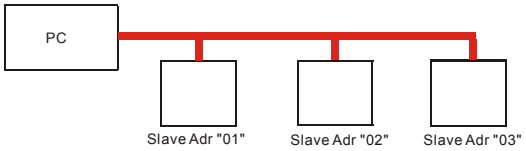
Avec les communications RS485, la transmission et la réception des données devront intervenir au même moment. Ce type de communication est réputé pour sa grande fiabilité (y compris en milieu perturbé et parasité).

Vous pourrez par exemple vous inspirer du schéma ci-dessous pour connecter les signaux TTL du module CB280 au circuit d'interfaçage RS485 (SN75176B).



La communication RS485 nécessitera l'utilisation d'un signal « Transmit Enable » pour indiquer si le dispositif est en train de transmettre ou de recevoir. Il ne peut y avoir qu'un seul dispositif qui transmet pendant que les autres sont en mode de réception.

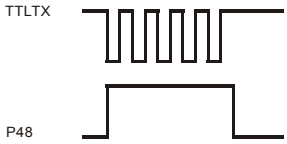
Exemple : Lorsqu'un PC transmet, tous les modules esclave seront en réception des données.



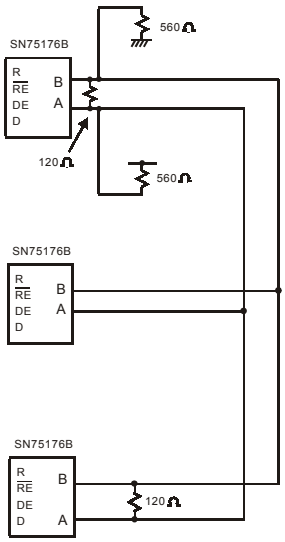
La commande SET RS485 permet aux modules CUBLOC ou CUTOUCH de contrôler la ligne de données à chaque fois qu'ils voudront transmettrent ou recevoir. Pendant que les données seront transmises, la broche Transmit Enable sera placée au niveau Haut. Ceci sera automatiquement réalisé par le « système » d'exploitation du CUBLOC™.

NOTE : Si vous utilisez un convertisseur RS232 <> RS485 et que ce dernier supporte le mode automatique, vous n'aurez pas besoin d'utiliser cette commande.

SET RS485 1,48



Lorsque vous utilisez la commande SET RS485, le Port utilisé peut ne pas être exploité.



Référez-vous à la figure ci-dessus si vous désirez relier plusieurs modules « CUBLOC » ou « CUTOUCH » via une liaison RS485.

Utilisez une résistance de 120 ohms de terminaison sur le dispositif en fin de ligne.

Les 2 résistances de tirage de 560 ohms sont nécessaires pour assurer une communication fonctionnelle.

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

Set Until

SET UNTIL canal, packetlength, Stopchar

canal : Canal RS232 (0 à 3 suivant modèle de CUBLOC™ utilisé)
packetlength : Longueur du packet (0 à 255)
Stopchar : Caractères à vérifier

Cette commande conditionnelle 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** pourra être utilisée pour déterminer quand l'interruption sera générée.

Lorsque les caractères reçus correpondront à ceux du paramètre « Stopchar » ou que le nombre d'octets reçus sera supérieur à la valeur du paramètre « packetlength », alors la commande ON RECV continuera son programme à l'étiquette d'interruption spécifiée. Avec cette méthode, vous aurez une plus grande souplesse de traitement.

Le paramètre « packetlength » est nécessaire (dans le cas où le caractère attendu n'arrive jamais).

Cette commande DOIT ETRE utilisée avec la commande 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'autre terme, 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).

Le caractère à détecter peut également être formulé sous forme décimal (comme indiqué ci-desous) :

```
SET UNTIL 1,100,4
```

Shiftin( )

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

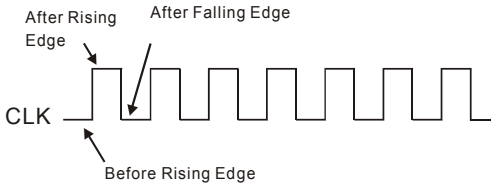
Variable : Variable servant à mémoriser le résultat (non String, ni Single)
Clock : Port du Signal d'horloge (0 à 255)
Data : Port du Signal de données (0 à 255)

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

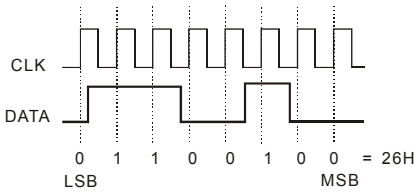
bitlength : Nombre de bits (1 à 16)

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 travail 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)

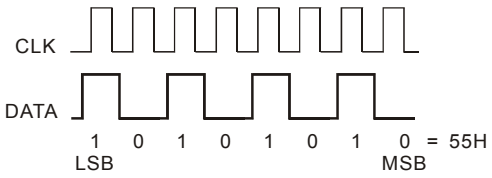
Mode : 0 = LSB en premier (Least Significant Bit First)
1 = MSB en premier (Most Significant Bit First)
2 = MSB en premier (Most Significant Bit First) , Création ACK (Pour I2C™)

variable : Variable servant à mémoriser la 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 envoi 0x55
                         ' bitlength = envoi de 8 bits,
```



Steppulse

STEPULSE canal, Port, Freq, Qt

Canal : Canal StepPulse (0 ou 1)
Port : Port du Cubloc
Freq : Fréquence de sortie (jusqu'à 15 KHz)
Qté : Nomnbre 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 peu varier.

Module	Canaux	Canal	Canaux PWM ne pouvant pas être utilisés pendant cette commande
CB220, 280, 290, CT17XX	1	0	Canal 0 (soit PWM 3, 4, 5)
CB405	2	0 ou 1	Canal 0 (soit PWM 3, 4, 5) Canal 1 (Soit PWM 6, 7, 8)

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. Ce port restera attribué à une sortie (même lorsque les impulsions auront fini d'être générées).

La fréquence de sortie peut être configurée de 1 Hz à 15 KHz. Le nombre d'impulsions peut être configuré de 1 à 2147483647. Toutes les impulsions sont générées en tâche de fond de telle sorte que le CUBLOC™ puisse être utilisé pour d'autres actions en même temps.

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

Stepstop

STEPSTOP canal

Canal : Canal StepPulse (0 ou 1)

Cette commande stoppe immédiatement la génération des impulsions sur le canal sélectionné.

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

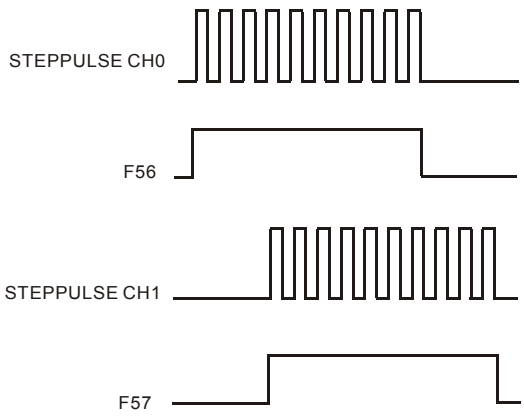
Stepstat()

Variable = STEPSTAT (canal)

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

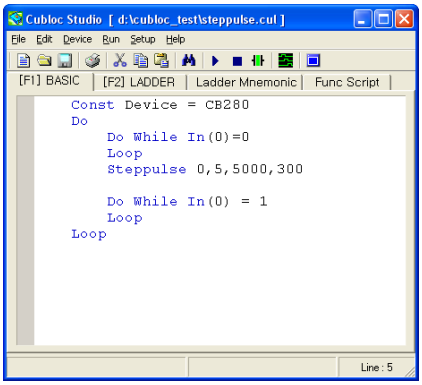
Cette commande vous permet de contrôler combien d'impulsions ont été générées depuis l'exécution de la dernière commande SETPPULSE. STEPSTAT vous retournera le double du nombre d'impulsions restant à être générées. Si par exemple il reste 500 impulsions à générer, STEPSTAT vous retournera la valeur 1000.

Vous pouvez également contrôler l'état de la sortie des impulsions en utilisant les registres –F(56) et _F(57) en ladder. Lorsque le canal 0 génère des impulsions le registre _F(56) sera au niveau HAUT (1). Lorsque le canal 1 génère des impulsions le registre _F(57) sera au niveau HAUT (1). Si aucune impulsion n'est générée, les registres seront au niveau BAS (0).

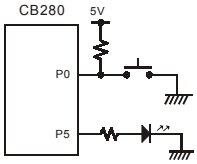


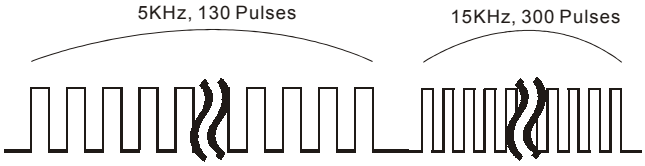
Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

Programme de test :

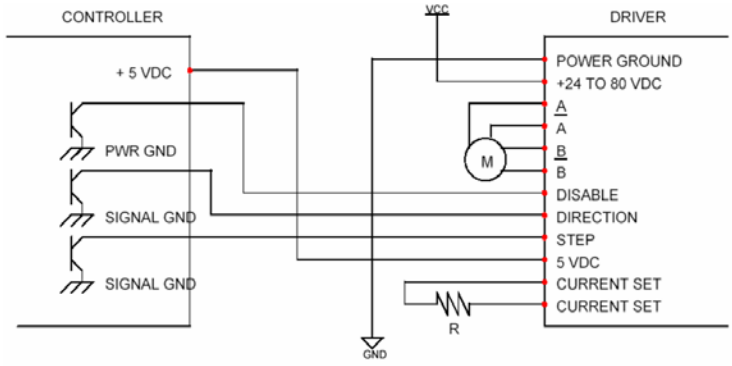


Ce programme générera 300 impulsions à une fréquence de 5 KHz, lorsque le port P0 sera sollicité via un bouton-poussoir.





Les commandes STEPPULSE, STEPSTOP et STEPSTAT sont idéalement conçues pour piloter des moteurs pas-à-pas (à l'aide d'interfaces de puissance additonnelles).



3 Ports seront généralement nécessaires pour piloter un moteur pas à pas via un module d'interface de puissance. Un premier port sera alloué à la sélection du sens de rotation, un second port à la désactivation de la commande du moteur et le troisième port à la génération des impulsions de commande (consultez la documentation de votre module de puissance – non livré avec les CUBLOC™ pour plus d'infos – et plus particulièrement sur le nombre d'impulsions nécessaires pour faire avancer le moteur d'un pas).

Sys( )

Variable = SYS(adresse)

Variable : Variable servant à mémoriser le résultat (non 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).
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 sera à 0. Après l'appel à une sous-routine ou une Fonction, l'adresse mémoie sera incrémentée. Si l'adresse mémoire de la RAM excède la capacité du module CUBLOC™, vous risquez de réaliser une action de dépassement pouvant générer 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)    ' Stock l'adresse de la mémoire RAM dans la variable A
```

Tadin()

Variable = TADIN(canal)

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

La commande Tadin() est similaire à Adin() mise à part qu'elle retourne une valeur moyenne de 10 mesures. Si vous travaillez en environnement perturbé, cette commande vous apportera une plus grande précision de mesure.

Le programme ci-dessous décompose le fonctionement 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
```

Time( )

Variable = TIME (adresse)

Variable : Variable servant à mémoriser le résultat (non String, ni Single)
adresse : Adresse des données présentent dans l'horloges (0 à 6)

Lorsque cette commande est utilisée avec le module CUBLOC™ CB290, vous bénéficierez de son horloge interne RTC. Vous pouvez utiliser les commandes Time() et Timeset pour mettre à jour et lire les données de cette horloge. L'horloge vous permettra ainsi de savoir l'heure, le jour de la semaine ou encore l'année.

Il est possible de sauvegarder le fonctionnement de l'horloge même si l'alimentation du module est coupée (à condition de prévoir une pile de sauvegarde).

Le tableau ci-dessous montre les différentes adresses relatives aux différentes informations présentes dans l'horloge RTC du module CB290.

*** Vous ne pouvez pas utiliser cette commande sur les modules CUBLOC™ CB220, CB280 et CB405 du fait qu'ils ne disposent pas d'horloge RTC.**

Adresse	Valeur	Gamme	Structure
0	Seconde	0~59	2nd digit place | 1st digit place
1	Minute	0~59	2nd digit place | 1st digit place
2	Heure	0~23	2nd digit place | 1st digit place
3	Date	01~31	2nd digit place | 1st digit place
4	Jour	0~6	1st digit place
5	Mois	1~12	2nd digit | 1st digit place
6	Année	00~99	2nd digit place | 1st digit place

Le tableau ci-dessous donne la correspondance des jours de la semaine vis à vis des valeurs numériques:

Dimanche	0
Lundi	1
Mardi	2
Mercredi	3
Jeudi	4
Vendredi	5
Samedi	6

Horloge système (RTC)

Il est toutefois possible sur les modules CUBLOC™ CB220, CB280 et CB405 d'exploiter l'horloge système interne du microcontrôleur pour bénéficier d'une horloge exploitable à l'aide des commandes TIME() et TIMESET.

Le tableau ci-dessous vous indique les correspondances d'adressage :

Adresse	Valeur retournée	Gamme
10	Secondes	0~59
11	Minutes	0~59
12	Heures	0~65535
13	Seconde continuelle	0~65535

La valeur à l'adresse 10 sera incrémentée toutes les secondes. Lorsque celle-ci atteindra 60, la valeur de l'adresse 11 sera incrémentée à son tours. Lorsque la valeur de l'adresse 11 attendra 60, la valeur de l'adresse 12 sera alors également incrémentée à son tours. Lorsque cette valeur atteindra 65535, elle sera réinitialisée à la valeur 0. A la mise sous tension toutes les valeurs de ces adresses sont à 0.

La commande TIMESET pourra être utilisée pour « programmer » l'heure.

Les valeurs des emplacements mémoire 10 à 13 sont stockées en mémoire RAM. Il ne sera pas nécessaire comme sur le CB290 de procéder à leur conversion (via les commandes BCD2BIN et BIN2BCD) pour les rendre exploitables.

La valeur présente à l'adresse 13 s'incrémente également toutes les secondes (comme la valeur de l'adresse 10), mis à part qu'elle ira jusqu'à 65535 avant de « retomber » à 0.

L'utilisation des adresses mémoire 10 à 13 pour la génération d'une horloge RTC système interne ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

Il va de soit que la génération de cette horloge RTC interne dispose d'une précision inférieure à celle d'une « vrai » horloge RTC dédiée – comme sur le CB290 par exemple (il faut compter environ un décalage potentiel < 1 % par 24 heures).

```
Const Device = CB405
Dim i As Integer
Cls
Timeset 10,58
Timeset 13,254
Do
        i = Time(10)
        Debug Goxy,0,0,dec4 i,Cr
        Debug Goxy,0,1,dec4 Time(13)
        Delay 100
Loop
```

Timeset

TIMESET adresse, valeur

adresse : Adresse des données présentent dans l'horloges (0 à 6)
valeur : valeur (0 à 255)

Utilisez la commande **TIMESET** pour modifier les paramètres de l'horloge RTC du module « CB290 ».

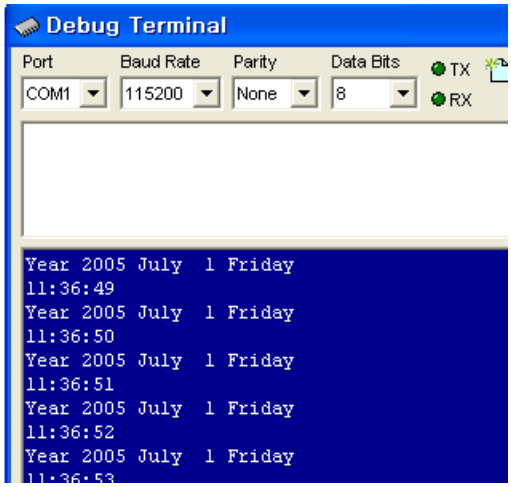
Adresse	Valeur	Gamme	Structure bit
0	Seconde	0~59	2nd digit place / 1st digit place
1	Minute	0~59	2nd digit place / 1st digit place
2	Heure	0~23	2nd digit place / 1st digit place
3	Date	01~31	2nd digit place / 1st digit place
4	Jour	0~6	1st digit place
5	Mois	1~12	10 / 1st digit place
6	Année	00~99	2nd digit place / 1st digit place

L'exemple qui suit montre comment mettre à l'heure l'hrologe RTC du module « CB290 » et comment l'afficher dans le fenêtre DEBUG du PC:

```
Const Device =CT1700
    Dim I As Byte
    Timeset 0,0         ' Seconde
    Timeset 1,&H32      ' Minute
    Timeset 2,&H11      ' Heure
    Timeset 3,&H1       ' Date
    Timeset 4,&H5       ' Jour de la semaine
    Timeset 5,&H6       ' Mois
    Timeset 6,&H5       ' Année

    Do
        I = Time(6)
        Debug "Année ","200",Hex I, " "
        I = Time(5)
        Select Case I
        Case 0
            Debug "Janvier"
        Case 1
            Debug "Fevrier"
        Case 2
            Debug "Mars"
        Case 3
            Debug "Avril"
        Case 4
            Debug "Mai"
        Case 5
            Debug "Juin"
        Case 6
            Debug "Juillet"
        Case 7
```

```
        Debug "Aout"
Case 8
        Debug "Septembre"
Case 9
        Debug "Novembre"
Case 10
        Debug "Decembre"
End Select

I = Time(3)                    ' Affiche la date
Debug " ", Hex2 I
Debug " "

I = Time(4)
Select Case I
Case 0
        Debug "Dimanche "
Case 1
        Debug "Lundi "
Case 2
        Debug "Mardi "
Case 3
        Debug "Mercredi "
Case 4
        Debug "Jeudi "
Case 5
        Debug "Vendredi "
Case 6
        Debug "Samedi "
End Select
Debug cr

I = Time(2)
Debug Hex2 I,":"
I = Time(1)
Debug Hex2 I,":"
I = Time(0)
Debug Hex I,cr
Delay 1000
Loop
```

La fenêtre de DEBUG du PC:



Udelay

UDELAY time

Time : Interval (1 à 65535)

Cette commande permet de générer 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 10          ' Delay d'environ 1630 microsecondes.
```

Usepin

Usepin I/O, In/Out, NomAlias

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

La commande **Usepin** est utilisée pour configurer le fonctionnement (Entrée ou sortie) ainsi que le nom des broches dans le programme LADDER. Cette commande doit être utilisée au préalable avant d'utiliser les ports d' « E/S » dans le LADDER.

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

Utmax

UTMAX variable

Variable : Variable à incrémenter (non String, ni Single).

Incrémente une variable d'une unité. Lorsque la variable atteint sa valeur maximale, cette dernière 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). Cette temporisation dispose d'une précision de l'ordre 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
```

Cette commande ne fonctionne qu'avec le « Cubloc Studio » version 2.0.x et supérieur.

WaitTx

WAITTX canal

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

La commande **WaitTx** attendra jusqu'à ce que le buffer d'émission RS232 soit purgé.
Le programme ci-dessus réalise la même fonction que la commande WaitTX :

```
OPENCOM 1,19200,0, 100, 50
PUTSTR 1,“ILOVEYOU”,CR

DO WHILE BFREE(1,1)<49  ‘ Attend que toutes les données soient envoyées
LOOP
```

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 jusqu'à ce que toutes les données soient expédiées
```

Lorsque cette commande est en phase d'attente, les autres interruptions peuvent être appelées.
En d'autres termes, cette commande n'affecte pas le fonctionnement du module CUBLOC™.

Chapitre 7.

Les librairies d'affichages des CUBLOC™…

Les modules CUBLOC™ peuvent piloter très facilement des modules LCD **spéciaux** de type **alphanumérique** ou **graphique** au moyen de commandes très simples qui vous permettront suivant les modèles d'afficher des textes, de tracer des lignes, des cercles, des carrés, etc…

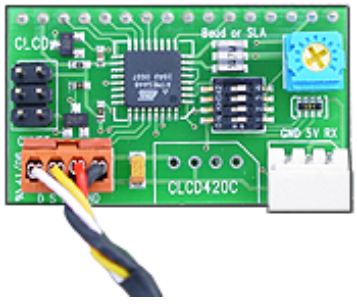
Afficheurs LCD alphanumériques de la gamme « CLCD »

Les afficheurs LCD de la gamme « CLCD » vous permettrons d'afficher des caractères alphanumériques ainsi que des nombres.

Ces derniers sont composés d'un module afficheur standard (2 x 16 ou 4 x 20 caractères suivant les modèles) associé au dos à une petite platine de contrôle qui recevra **au choix** des données de type **série** ou **I2C™** en provenance du module CUBLOC™ afin de les interpréter en « commandes » comprises par l'afficheur.

A noter que nous commercialisons également les platines de commandes seules afin que vous puissiez utiliser des afficheurs standards « 2 x 16 » ou « 4 x 20 » déjà en votre pocession et les transformer ainsi en afficheurs de type « CLCD » à commandes I2C™ ou séries. Consultez-notre site Internet www.lextronic.fr pour plus d'infos.

La platine de commande dispose de 3 connecteurs, d'un potentiomètre de réglage de luminosité et d'un commutateur DIL de paramétrage. Le premier connecteur mâle 6 points à l'extrême gauche de la platine **ne doit JAMAIS être utilisé** – il sert à la programmation de la platine en sortie d'usine. Ne court-circuitez jamais ces broches et n'y connectez aucun signal.



Le second connecteur mâle 4 points marron (CUNET) en bas à gauche est destiné à piloter l'afficheur via une liaison 4 fils de type I2C™. Ce connecteur est pré-équipé d'un câble de liaison. On retrouve sur ce connecteur une broche +5 V (servant à alimenter l'afficheur), la masse et les signaux SDA et SCL. Des résistances de tirage de 4,7 Komhs (à relier au + 5 Vcc) devront être ajoutées sur ces 2 signaux au niveau de votre platine d'application.

Le troisième connecteur en bas à droite est un modèle 3 points. Il est destiné à piloter l'afficheur via un signal série (avec niveau logique 0 / +5 V). On retrouve sur ce connecteur une broche +5 V (servant à alimenter l'afficheur), la masse et l'entrée de commande série de l'afficheur à relier sur la sortie TX de votre microcontrôleur. Le signal série de commande relatif à cette entrée ne pourra pas avoir un débit supérieur à 115200 bps. Le câble nécessaire au raccordement de ce connecteur est livré en option (consultez-nous).

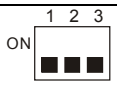
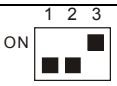
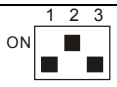
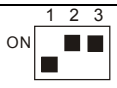
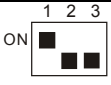
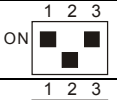
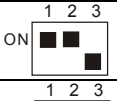
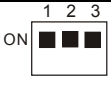
A noter la présence de 4 trous de connexion (au dessus du marquage CLCD420C). Ces trous ne sont pas utilisés et rien ne doit être raccordé à cet endroit.

Le commutateur DIL présent sur la platine a une double fonction.

- Lorsque l'afficheur est utilisé en mode série, il permet de paramétrer la vitesse de communication.

- Lorsque l'afficheur est utilisé en mode I2C™, il permet de paramétrer l'adresse de commande de l'afficheur.

Vous trouverez ci-après le rôle de chaque DIL (le 4ème DIL n'est pas utilisé).

DIP Switch	Débit communication RS232	Adresse esclave I2C™
	2400	0
	4800	1
	9600	2
	19200	3
	28800	4
	38400	5
	57600	6
	115200	7

L'afficheur détectera automatiquement si vous utilisez l'entrée I2C™ ou l'entrée série pour le piloter via le CUBLOC™ (une seule entrée devra toutefois être utilisée à la fois).

Vous trouverez ci-après la liste des commandes du CUBLOC™ dédiées au pilotage de ces afficheurs.

Set Display

SET DISPLAY type, Mode, baud, Taillebuffer
type : 0=Série, 1=Graphique, 2=CLCD
Mode : Mode de communication 0=CuNET, 1=COM1
baud : Adresse I2C™ esclave (si Mode = 0) ou Baud rate (si Mode = 1)
Taille Buffer : Taille du Buffer d'émission (jusqu'à 128)

La commande **SET DISPLAY** doit être utilisée (une seule fois) en début de programme pour indiquer au CUBLOC™ avec quel type d'afficheur vous allez communiquer et comment vous l'avez interfacé.

La commande permet aussi d'initialiser cet afficheur avant de pouvoir communiquer avec ce dernier. 4 paramètres doivent être définis afin de pouvoir initialiser correctement l'afficheur (type, Mode, baud, Taillebuffer).

La paramètre « **type** »
Ce paramètre permet de sélectionner le type d'afficheur LCD que vous allez utiliser.

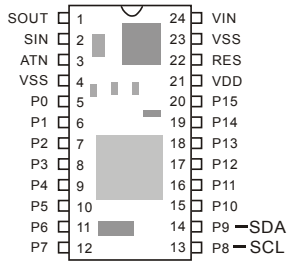
> La valeur **0** vous permettra d'utiliser les afficheurs modèles « **CLCD** » via leur liaison série. (Attention les afficheur de type « **ELCD** » (utilisés avec les PICBASIC - qui sont d'autres types de contrôleur programmable en BASIC développés par Comfile Technology) ne sont pas compatibles avec les modules CUBLOC™ et ne pourront pas être pilotés via ce mode).

> La valeur **1** vous permettra d'utiliser des afficheurs graphiques (voir description ci-après).

> La valeur **2** vous permettra d'utiliser les afficheurs modèles « **CLCD** » via leur liaison I2C™.

Le paramètre « **Mode** »
Ce paramètre sert à indiquer au CUBLOC™ comment vous aller piloter l'afficheur (en mode série ou via une communication I2C™).

> Lorsque Mode = **0**, le module CUBLOC™ sera configuré pour dialoguer avec un afficheur de type « CLCD » via une liaison I2C™. Par exemple, pour un module CB220 les ports pour gestion de l'afficheur seront les Port 8 (Clock) et Port 9 (Data). Il vous faudra ajouter des résistances de tirage de 4,7 Kohms au + 5 V sur ces 2 ports. Attention : Bien qu'il s'agisse d'une communication I2C™, un seul afficheur doit être connecté sur le module CUBLOC™.



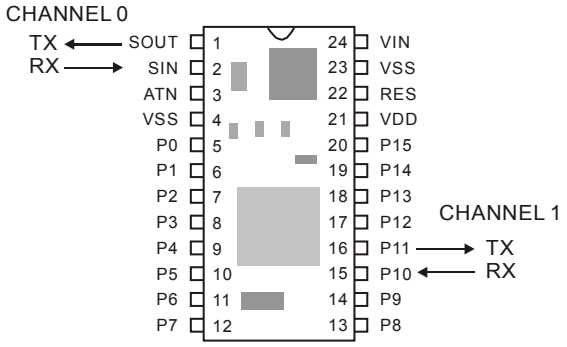
Dans le cadre d'une communication avec un afficheur CLCD piloté en mode I2C™, le paramètre « baud » devra correspondre à l'adresse esclave de l'afficheur (voir le tableau donné en page 218 pour la correspondance des adresses esclaves de l'afficheur et de ses mini-interrupteur –dils).

Exemple :

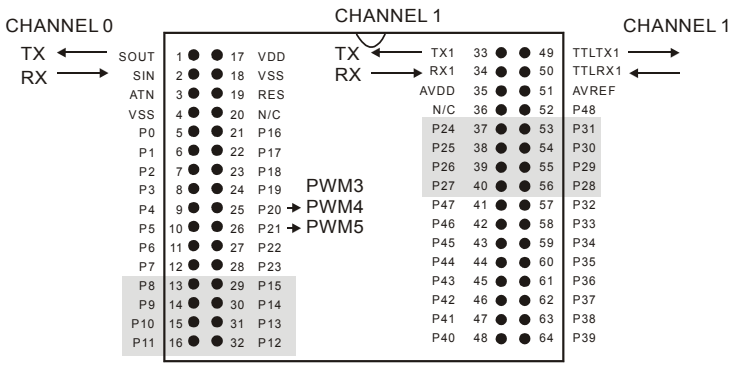
SET DISPLAY 2,0,1,50

Ici on initialisation la communication pour un afficheur de type « CLCD » en liaison I2C™ dont l'adresse esclave est 1 et avec un buffer d'émission de 50 octets.

> Lorsque Mode = **1**, le module CUBLOC™ sera configuré pour dialoguer avec un afficheur de type « CLCD » en liaison série (avec niveau logique 0 / 5 V) via le canal de communication série N° 1 (pour le CB220, utilisez alors le port 11 - TX).



Pour le CB280 la broche 49 peut être utilisée.



Le paramètre « **baud** »
Ce dernier permet de sélectionner le débit de communication série (lorsque vous exploitez le mode = 1) parmi les valeurs suivantes : 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200, 230400). Le paramètre « baud » est remplacé par l'adresse esclave de l'afficheur lorsque vous utilisez le mode = 0.

Le paramètre « **Taillebuffer** »
Ce dernier permet de sélectionner la taille du buffer de communication alloué au pilotage de l'afficheur. Une valeur comprise entre 50 et 128 est conseillée. Si la taille du buffer est trop petite, l'affichage ne se fera pas correctement. Si la taille est trop grande, vous gaspillerez de la capacité mémoire RAM du CUBLOC™.

SET DISPLAY 0,1,19200,50

Ici on initialisation la communication pour un afficheur de type « CLCD » en liaison série dont le débit est de 19200 bds avec un buffer d'émission de 50 octets.

Rappels :

Attention la commande **SET DISPLAY** ne doit être utilisée qu'au début de votre programme.

N'utilisez pas la sortie +5 V régulée des modules CUBLOC™ « CB220 » pour alimenter les afficheurs « CLCD » sans quoi le régulateur interne du « CB220 » risque une destruction en raison de son incapacité à pouvoir alimenter le rétro-éclairage de l'afficheur.

Cls

CLS
Initialise l'afficheur LCD.

(Pensez à ajouter un délai après cette commande avant de poursuivre le reste de votre programme afin de laisser le temps à l'afficheur de s'initialiser correctement.)

```
CLS
DELAY 200
```

Cson

CSRON
Active l'affichage du Curseur (ON). (Par défaut le curseur est désactivé).

Csoff

CSROFF
Désactive l'affichage du Curseur (OFF).

Locate

LOCATE x,y

X : position sur l'axe des « X » de l'afficheur LCD
Y : position sur l'axe des « Y » de l'afficheur LCD

Défini la position du curseur (qu'il soit visible ou non). Après une commande CLS, le curseur est en position 0,0.

```
LOCATE 1,1              ' Déplace le curseur en 1,1
PRINT "COMFILE"         ' et affiche « COMFILE »
```

Print

PRINT String/Variable
String : String
Variable : variable.

Affiche des caractères à l'écran

```
LOCATE 1,1                ' Il est possible de mixer les données à afficher
PRINT "COMFILE",DEC I     ' constantes et variables (à séparer par des virgules).
```

Note importante : Si vous devez envoyer des commandes consécutives à l'afficheur LCD, prenez garde à ce que la commande en cours d'éxécution soit arivée à terme avant d'envoyer une nouvelle commande, sans quoi l'afficheur risque de ne pas prendre en compte votre demande (ce dernier étant alors occupé). Ainsi par exemple après une commande « cls », réalisez une pause de l'ordre de 200 ms.

Dans le même ordre d'esprit, si vous devez envoyer des commandes à l'afficheur dès la mise sous tension du CUBLOC™, effectuez impérativement une pause minimale de 100 ms avant d'envoyer votre premier ordre (afin de s'assurer que l'afficheur soit lui même initialisé).

Commandes spéciales pour les afficheurs de type « CLCD »

Les afficheurs « CLCD » peuvent également réaliser certaines actions lorsqu'on leur envoi une suite de caractères spéciaux dont le tableau ci-dessous donne la liste.

Commande	Exemple (hex)	Octets	Durée exécution	Explication
ESC ' C'	1B 43	2	15 mS	Efface l'écran. Une temporisation de 15 ms mini. doit être réalisée après cette fonction avant de pouvoir adresser correctement l'afficheur
ESC 'S'	1B 53	2		Curseur ON
ESC 's'	1B 73	2		Curseur OFF (par défault)
ESC 'B'	1B 42	2		Backlight ON (par défault)
ESC 'b'	1B 62	2		Backlight OFF
ESC 'H'	1B 48	2		Equivalent LOCATE 0,0
ESC 'L' X Y	1B 4C xx yy	4	100 uS	Change la position du cursor.
ESC 'D' 8byte	1B 44 adresse (8 à 15) 8 octets	11		Les caractères de l'afficheur présents aux adresses 8 jusqu'à 15 peuvent être modifiés par vos soins grâce à l'envoi de cette suite de codes (consultez les notes d'applications présentent sur le CD-ROM ou sur le www.lextronic.fr pour plus d'infos).
1	01	1		Se place au début de la rangée 1
2	02	1		Se place au début de la rangée 2
3	03	1		Se place au début de la rangée 3
4	04	1		Se place au début de la rangée 4

Si les afficheurs reçoivent une suite de caractères qui ne correspondent pas à la liste de commandes reconnues dans le tableau ci-dessus, ces caractères seront affichés à l'écran.

Les afficheurs de type « ALCD » disposent également d'un jeu de commandes spéciales. Le nombre de ces commandes est plus important que celui des afficheurs « CLCD ». Consultez la notice des afficheurs « ALCD » pour plus d'informations à ce sujet.

Exemple d'utilisation d'un afficheur de type « CLCD »

Le programme ci-dessous donne un exemple de programme permettant d'afficher un message et un compteur sur un écran « CLCD204-BLB ». L'afficheur sera connecté aux Ports P8 et P9 d'un module « CUBLOC™ » CB220. Les commutateurs DIL de l'afficheur devront être positionnés comme ci-dessous.

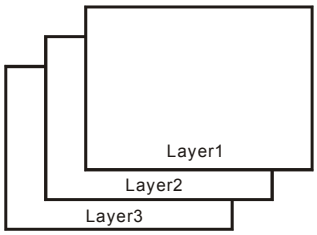
DIL1 = OFF
DIL2 = OFF
DIL3 = ON
DIL4 = OFF

```
Const Device = Cb220
  Set Display 2,0,1,50              ‘ Pilotage en mode I2C™ de l’afficheur de type CLCD.
  Dim i As Integer
  Delay 100                         ' Délai pour initialisation de l’afficheur
  Cls
  Delay 200                         ' Délai après une commande CLS
  Csroff
  Locate 5,2
  Print "Start !!!"
  Delay 500
  Cls
  Delay 100
  Do
        Incr i
        Locate 0,0
        Print "COMFILE"
        Locate 1,3
        Print "CUBLOC ",Dec i
        delay 100
  Loop
```

Afficheurs LCD graphiques série « GHB3224 »

Les afficheurs de la série « GHLCD » sont capables d'afficher des caractères et des graphiques sur 3 couches différentes. Comme tous les afficheurs de la gamme Comfile Technology, les « GHLCD » supportent différents types de commandes de la part des modules CUBLOC™ qui leur permettra de tracer facilement des lignes, des cercles et des rectangles. D'autres commandes permettent également des opérations du type « Copier / couper / coller » ou encore l'affichage d'images BMP préalablement chargées dans le « GHLCD » au moyen d'un utilitaire spécialisé (CuCanvas). Ces commandes sont également compatibles avec les modules « CUTOUCH ».

Les modèles « GHB3224 » disposent d'un affichage LCD de type STN (noir et blanc) ou (blanc et bleu) avec une résolution de 320 x 240 pixels. Sur leurs 3 couches, 1 est dédié à l'affichage de textes et les 2 autres à l'affichage des graphiques.



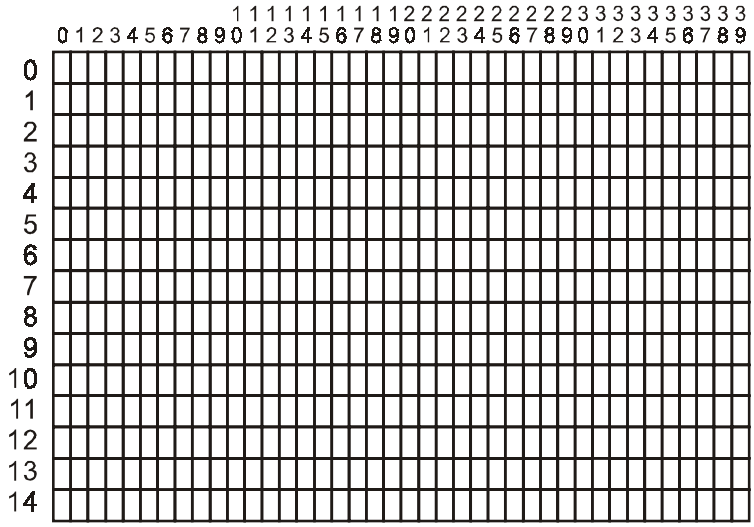
Les versions « GHB3224A » supportent les communications séries.

Les versions « GHB3224C » supportent les communications au protocole I2C™ (ce qui permet de libérer un port série sur le CUBLOC).

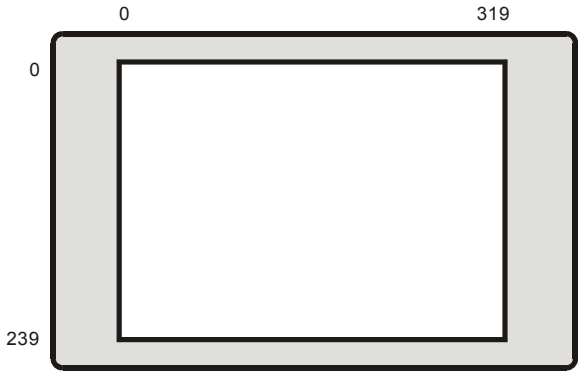
Vous trouverez ci-dessous un exemple de déclaration pour utiliser un « GHB3224C »:

```
SET DISPLAY 1,0,1,50        'GHLCD, mode I2C™, Adresse esclave 1, Taille buffer 50
```

La taille de la couche dédiée au texte est de 40 x 15 (comme on le voit sur la grille ci-dessous). Chaque caractère est doté d'une taille de 8 par 16.



Pour vos graphiques, vous disposez d'une résolution de 320 x 240 pixels.



A noter que les graphiques et les textes apparaîtront de façon aléatoire sur l'écran si vous essayer de les afficher en dehors des postions autorisées (c'est à dire en dehors de la résolution 320 x 240 pixels pour les graphiques et en dehors de la grille 40 x 15 pour les textes). Nous vous recommandons de dessiner le fond de votre affichage sur une des couches graphiques et les caractères sur la couche texte.

Vous trouverez ci-après la liste des commandes du CUBLOC™ dédiées au pilotage de ces afficheurs.

Cls

CLS
Initialise le LCD et efface toutes les couches.
(Prévoir un délai afin que l'afficheur ait le temps de réaliser cette opération)

```
CLS
DELAY 200
```

Clear

CLEAR layer
Efface une couche spécifique.

```
CLEAR 1          ' Efface la couche 1 (Texte).
CLEAR 2          ' Efface la couche 2 (Graphique).
CLEAR 0          ' Efface toutes les couches (même résultat que la commande « CLS »).
```

Csron

CSRON
Active l'affichage du Curseur (ON). (par défaut le curseur est désactivé).

Csroff

CSROFF
Désactive l'affichage du Curseur (OFF).

Locate

LOCATE x,y

X : position axe « X » sur le LCD
Y : position axe « Y » sur le LCD

Définie une position sur la couche texte. Après une commande CLS, le LCD se met d'office en position 0,0.

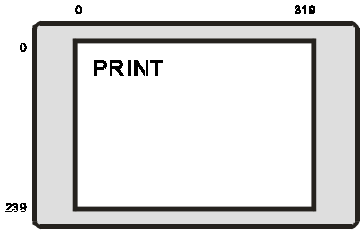
```
LOCATE 1,1                    ' Positionne le curseur en 1,1
PRINT "COMFILE"
```

Print

PRINT String/Variable
String : String
Variable : lorsque vous utilisez des variables/constantes, la représentation des chaînes/ variables seront affichés.

Cette commande affiche des caractères sur la couche texte.
Pour afficher des caractères sur la couche graphique, utilisez la commande GPRINT.

```
LOCATE 1,1                  ‘ Positionne le curseur en 1,1
PRINT “COMFILE”,DEC I       ‘ Affiche le mot COMFILE et la valeur décimale de la variable I
```



Layer

LAYER layer1mode, layer2 mode, layer3 mode
Layer1mode : Set Layer 1 mode (0=off, 1=on, 2=flash)
Layer2mode : Set Layer 2 mode (0=off, 1=on, 2=flash)
Layer3mode : Set Layer 3 mode (0=off, 1=on, 2=flash)

Configure le mode sur une couche spécifique. Le mode « flash » fera clignoter la couche à une fréquence d’environ 16 Hz. Les couches 1 et 2 sont « ON » et la couche 3 est « OFF » à la mise sous tension de l’afficheur.

Utilisez cette fonction pour dissimuler les phases pendant lesquelles vous tracerez des lignes, des cercles, des rectangles, etc…(en mettant la couche en « OFF » avant les tracés et en faisant apparaître l’ensemble des tracés d’un seul coup en mettant la couche en « ON » une fois que toutes les opérations de dessin seront terminées).

GLayer

GLAYER layernumber
Layernumber : Attribution de la couche graphique (0,1,2)

Les afficheurs de la série « GHB3224 » disposent de 3 couches. Une de ces couches peut être utilisée pour les graphiques. Les commandes du type LINE, CIRLCLE et BOX pourront être utilisées sur la couche qui aura été attribuée pour les graphiques. En temps normal, la couche 1 est utilisée pour les textes et la couche 2 pour les graphiques. Les couches 2 et 3 disposent de caractéristiques quelque peu différentes. Nous recommandons l'exploitation de la couche 2 pour la génération de graphiques qui devront être effacés souvent.

La couche 1 peut également si nécessaire être utilisée en tant que couche graphique. Dans ce cas, vous pourrez effacer du texte à l'aide de commandes prévues pour les graphiques. Pour attribuer la couche N° 3 aux graphiques, utilisez la commande LAYER.

Overlay

OVERLAY overmode
overmode : Mode logique (0 = or, 1 = and, 2 = xor)

La commande **Overlay** détermine l'interaction logique entre les couches 1 et 2.
La couche 1 étant dédiée aux textes et la couche 2 aux graphiques.
En utilisant cette commande, vous pourrez choisir comment réagira le LCD lorsque les couches 1 et 2 affichent quelque chose aux mêmes endroits. Le mode par défaut (XOR) provoquera une inversion lorsque quelque chose sera affiché aux mêmes endroits sur la couche 1 et 2. Pour ne pas avoir cette inversion, choisissez le mode OR.

Contrast

CONTRAST valeur
value : Valeur du contrast

Cette commande permet de contrôler le contrast du LCD.

Contrast 450

Light

LIGHT valeur
valeur : Back light 0 = OFF, 1 = ON

Active le back light « ON » ou « OFF ». Par défaut, le back light est « ON ».

Font

FONT fontsize, efontwidth

fontsize : 0 ~ 8 Sélection de la taille de la fonte
efontwidth : 0 = Largeur fixe, 1 = Largeur variable

Les afficheurs « GHB3224" disposent de 4 tailles de fontes associées à 2 largeurs différentes.

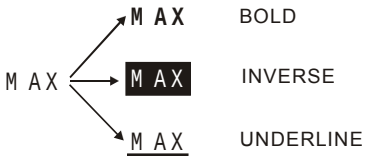
Type de fonte	Fonte
0,1	10 x 16
2,3,4,5	16 x 16
6,7	24 x 24
8	48 x 48

```
Const Device = cb220
SET DISPLAY 1,1,19200,50          ' GHLCD, série 19200 bds, Taille buffer 50
Cls
Delay 100
Font 0,0
Glocate 10,10
GPrint "FONT 0,0 :ABCDEFGHIJKLMN"
Font 2,0
Glocate 10,30
GPrint "FONT 2,0 :ABCDEFGHIJKLMN"
Font 6,0
Glocate 10,50
GPrint "FONT 6,0 :ABCDEFGHIJKLMN"
Font 8,0
Glocate 10,72
GPrint "FONT 8,0 "
Font 0,1
Glocate 10,120
GPrint "FONT 0,1 :ABCDEFGHIJKLMN"
Font 2,1
Glocate 10,140
GPrint "FONT 2,1 :ABCDEFGHIJKLMN"
Font 6,1
Glocate 10,160
GPrint "FONT 6,1 :ABCDEFGHIJ"
Font 8,1
Glocate 10,185
GPrint "FONT 8,1 "
```

Style

STYLE bold, inverse, underline
bold : 0 = Normal, 2 ou 3 = Gras
inverse : 0 = Normal, 1 = Inversé
underline : 0 = Normal, 1 = Souligné

Vous pouvez utiliser la commande STYLE pour modifier les paramètres d'affichage des textes Bold (gras), Inverse (inversé) ou UnderLine (souligné).



Cmode

CMODE valeur
valeur : 0 = Type BOX, 1 = type Under Line (souligné)

Choisissez le type de curseur. Par défaut le curseur est de type Under Line (souligné).

■ 0 : BOX Type

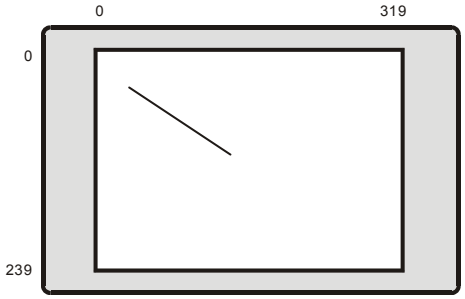
— 1 : Under Line Type

Line

LINE x1, y1, x2, y2

Dessigne une ligne depuis x1, y1 vers x2, y2.

```
LINE  10,20,100,120                    ' Dessigne une ligne
```

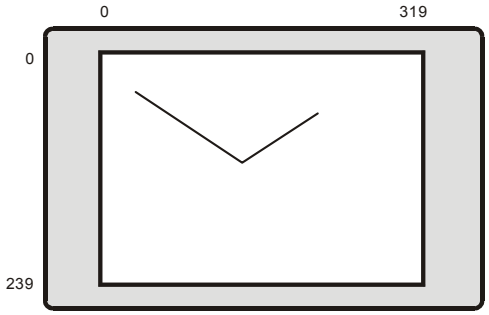


Lineto

LINETO x, y

Dessine une ligne depuis le dernier point vers x,y.

LINETO 200,50 ‘ Continue le tracé de la ligne depuis le dernier point.

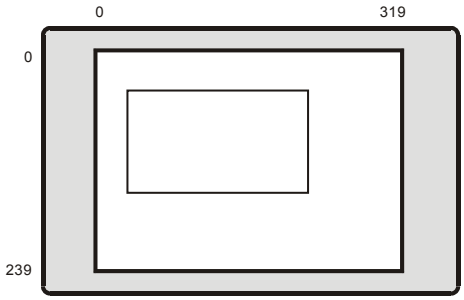


Box

BOX x1, y1, x2, y2

Dessine un rectangle d’après les positions X1,Y1 et X2,Y2.

BOX 10,20,200,100 ‘ Dessine un rectangle

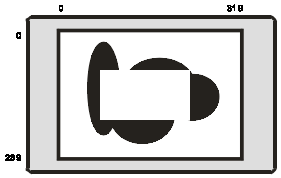


Boxclear

BOXCLEAR x1, y1, x2, y2

Efface un rectangle d'après les positions X1,Y1 et X2,Y2.

BOXCLEAR 10,20,200,100 ' Efface le rectangle



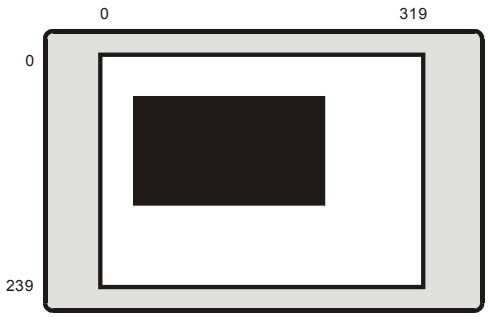
Boxfill

BOXFILL x1, y1, x2, y2,logique
logique : 0 =OR, 1 = AND, 2 = XOR

Dessine un rectangle d'après les positions X1,Y1 et X2,Y2 et le remplie suivant la logique définie.

0 = OR -> affichera toutes les zones qui se chevauchent.
1 = AND -> affichera seulement les zones qui se chevauchent.
2 = XOR -> affichera les zones qui se chevauchent en inversé.

BOXFILL 10,20,200,100,0 ' Dessine un rectangle rempli.

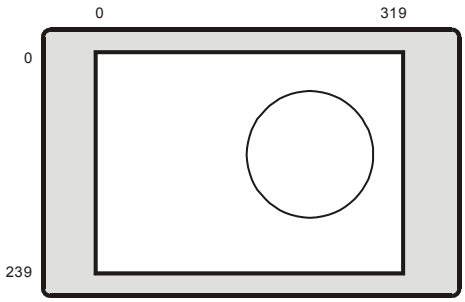


Circle

CIRCLE x, y, r

Dessine un cercle avec pour centre x,y et un rayon de r.

```
CIRCLE  200,100,50                    ‘ Dessine un cercle
```

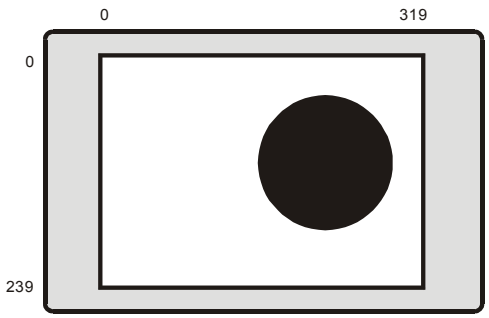


Circlefill

CIRCLEFILL x, y, r

Dessine un cercle plein avec pour centre x,y et un rayon de r.

```
CIRCLEFILL  200,100,50                ‘ Dessine un cercle plein
```

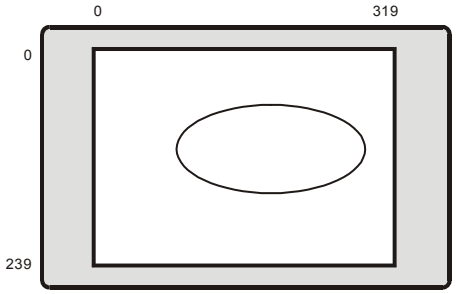


Ellipse

ELLIPSE x, y, r1, r2

Dessine une ellipse avec pour centre x,y et pour rayon horizontal r1 et pour rayon vertical r2.

ELLIPSE 200,100,100,50 ‘ Dessine une ellipse

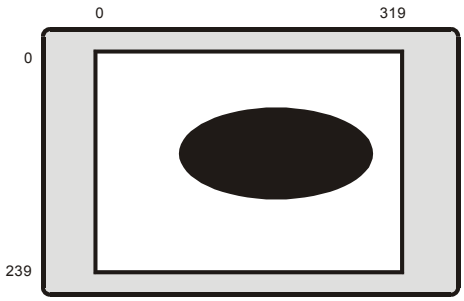


Elfill

ELFILL x, y, r1, r2

Dessine une ellipse pleine avec centre x,y et pour rayon horizontal r1 et pour rayon vertical r2.

ELFILL 200,100,100,50 ‘ Dessine une ellipse pleine

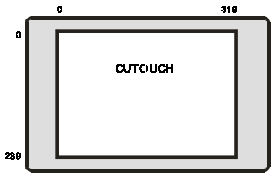


Glocate

GLOCATE x, y

Détermine une nouvelle position pour la couche graphique.

GLOCATE 128,32 ‘ Détermine la nouvelle position

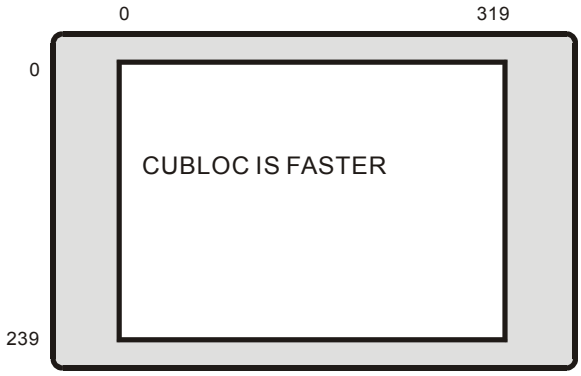


Gprint

GPRINT string

Affiche des chaînes sur la couche graphique. Ceci vous permet de positionner votre texte de façon plus précise du fait que vous travaillez sur la couche graphique en utilisant la commande GLOCATE, puis en affichant votre texte avec la commande GPRINT.

GPRINT “CUBLOC IS FASTER”,CR ‘ Affiche la phrase et passe à la ligne (CR)

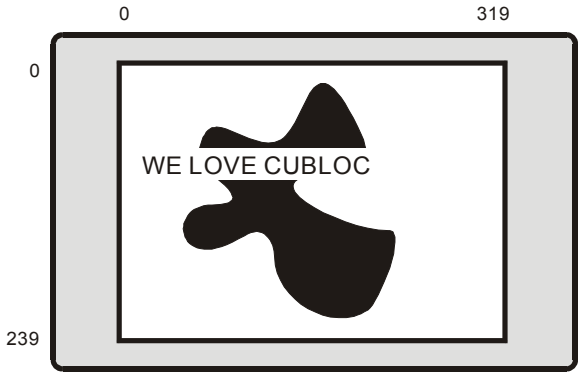


Dprint

DPRINT string

DPRINT est similaire à GPRINT mise à part qu'il efface les graphiques du dessous.

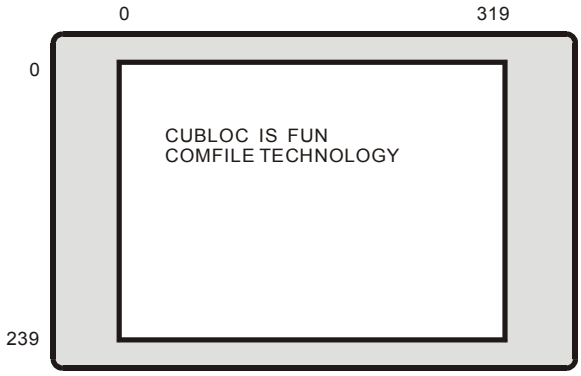
GPRINT "WE LOVE CUBLOC",CR 'Affiche la phrase et passe à la ligne (CR)



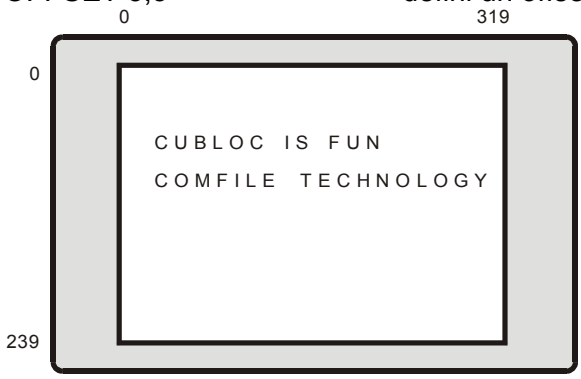
Offset

OFFSET x, y

Vous pouvez définir l'offset pour la chaîne affichée sur la couche graphique. La valeur par défaut est 0. Vous pouvez contrôler l'offset sur l'axe x ou y.



OFFSET 3,3 ' défini un offset de 3 sur l'axe x et y.



Après la commande, la chaîne s'affichera automatiquement avec le nouvel offset.

Pset

PSET x, y

Affiche un point en x,y

PSET 200,100 ' Affiche un point

Color

COLOR valeur

Définie la couleur du LCD. 1 = noir et 0 = blanc. La valeur par défaut est 0.

```
COLOR 0              'Définie la couleur à 0.
```

Linestyle

LINESTYLE valeur

Cette commande permet de définir le style de vos lignes. Vous pourrez ainsi tracer des lignes en pointillé en augmentant cette valeur. La valeur par défaut est 0 (pour une ligne pleine).

```
LINESTYLE 1                    ' Utilise des lignes pointillées
```

Dotsize

DOTSIZE valeur, style

Définie la taille des points. Valeur représente la taille du point et style le type de point (0 pour un point de type rectangulaire et 1 pour un point circulaire).

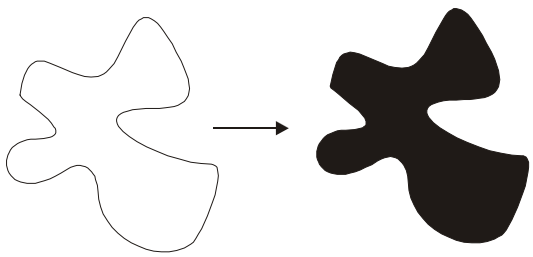
```
DOTSIZE 1,1                    ' Définie une taille de 1 avec un point de type rectangulaire
```

Paint

PAINT x, y

Rempli une forme « fermée » à partir de la position x,y.

```
PAINT 100,100                  ' Rempli la forme à partir de la position 100,100
```

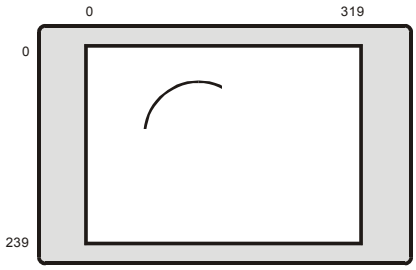


Arc

ARC x, y, r, start, end

Dessine un arc avec x, y pour centre. Start et end peuvent prendre des valeurs (en degré) compris entre 0 et 360.

ARC 200,60, 100, 10, 20 ‘ Dessine un arc avec des angles de 10 et 20.



Defchr

DEFCHR Code, Data

Code : Custom character code (&hdb30 ~ &hdbff)

Data : 32byte bitmap data

Cette commande permet la création de caractères personnalisés. Des caractères de 16 x 16 pixels peuvent être mémorisés dans la mémoire du LCD. Ces caractères pourront ensuite être utilisés comme n’importe quel autre caractère avec les commandes usuelles du type : PRINT ou GPRINT ou DPRINT. Un total de 207 caractères personnalisés peuvent ainsi être mémorisés dans l’écran LCD.

Les caractères ne sont pas sauvegardés lors des coupures d’alimentation du LCD.

```
DEFCHR &HDB30,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA,_
      &HAA,&HAA,&HAA,&H55,&HAA,&HAA,&HAA,&HAA,_
      &HAA,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA,_
      &HAA,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA,&HAA

print CHR(&HDB30)
```

Bmp

BMP x, y, filenumber, layer

X, y : position x,y pour afficher l'image BMP
Filenumber : N° du fichier BMP
Layer : Couche où afficher l'image BMP

Le « GHB3224 » dispose d'une mémoire FLASH permettant la mémorisation d'images BMP sous la forme de fichiers. Un utilitaire dédié vous permettra à ce titre de télécharger des images depuis votre PC vers l'afficheur. Il vous suffira alors d'utiliser la commande BMP pour afficher les images à l'écran.

*** L'afficheur « GHB3224 » dispose d'une mémoire Flash de 102,400 octets permettant la mémorisation de vos images au format. Vous pourrez ainsi mémoriser jusqu'à 10 images d'une résolution de 320 x 240 !**

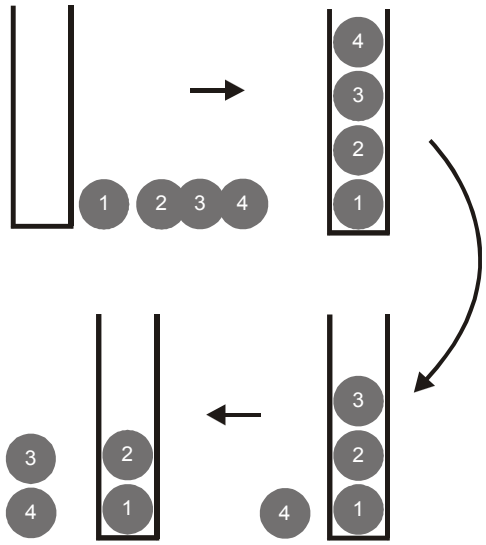
Commandes graphique Data PUSH, POP

Les afficheurs de la série « GHB3224 » disposent d'un emplacement mémoire spécialement dédié au stockage de données graphiques. Vous pourrez stocker ou récupérer un écran (ou une partie de ce dernier) au sein de cet emplacement mémoire. En utilisant cette possibilité de stockage, vous pourrez aisément utiliser des fonctions similaires aux traditionnels « Copier / couper / coller » propres aux traitements de textes. Les commandes GPUSH et GPOP sont à utiliser pour extraire des parties précises de vos écrans alors que les commandes HPUSH et HPOP sont utilisables pour des traitements très rapides.

L'emplacement mémoire est un modèle de type « LIFO » (Last in First out – Dernier entré… premier sorti) qui permet ainsi de récupérer les dernières données sauvegardées.

Vous disposez pour ce faire d'une mémoire de 32 KB qui vous permettra ainsi que sauvegarder 3 ou 4 écrans pleins.

Consultez les images ci dessous pour assimiler le mode de fonctionnement de cet emplacement mémoire:

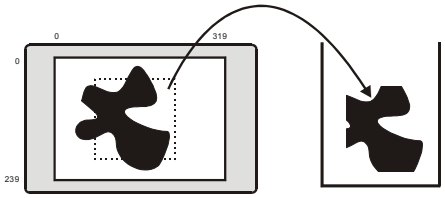


Gpush

GPUSH x1, y1, x2, y2, layer

Envoi le rectangle d'image compris entre x1,y1 et x2, y2 dans la mémoire.

GPUSH 10,20,200,100,2

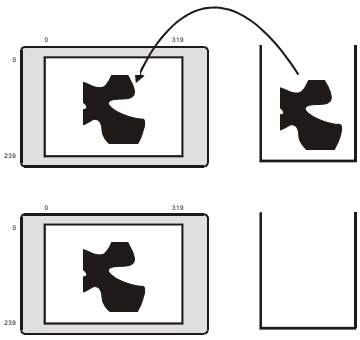


Gpop

GPOP x, y, layer, logique
logique =0 : OR
logique =1 : AND
logique =2 : XOR
logique =3 : Efface l'écran puis récupère l'image

Récupère l'image depuis la mémoire, puis l'affiche sur la couche spécifiée à la position x,y en y associant une logique d'affichage (l'image est effacée de la mémoire tampon).

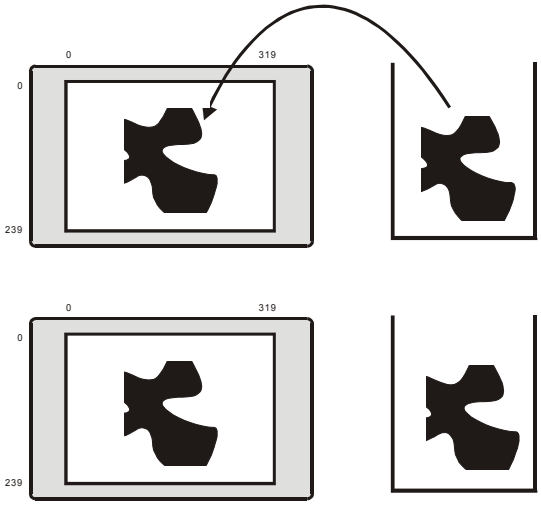
GPOP 120,20,2,0



Gpaste

GPASTE x, y, layer, logique
Logique =0 : OR
logique =1 : AND
logique =2 : XOR
logique =3 : Efface l'écran puis récupère l'image

Réalise la même opération que la commande GPOP à l'exception du fait que l'image n'est pas effacée de la mémoire tampon.

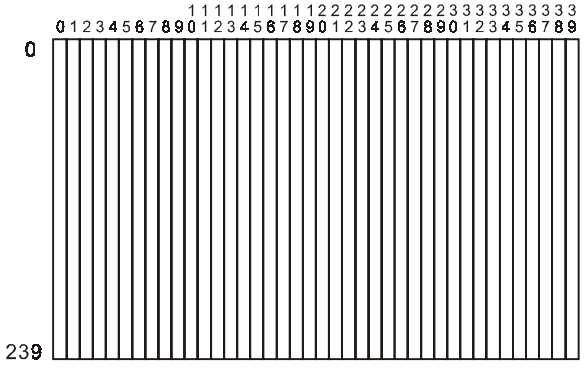


Hpush

HPUSH x1, y1, x2, y2, layer

Les commandes HPUSH, HPOP et HPASTE sont similaires aux commandes GPUSH, GPOP et GPASTE à l'exception du fait que les colonnes ne peuvent être que des multiples de 8 comme montré ci-dessous:

*Les 320 pixels ont étés divisés par 8 (il y a 40 colonnes de 8 pixels de large).



HPUSH 6,20,12,100,2

Hpop

HPOP x, y, layer

Similaire à la commande GPOP (mais avec des valeurs comprises entre 0 et 39).

HPOP 10,20,2,0

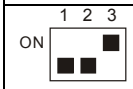
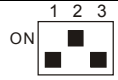
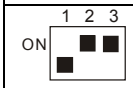
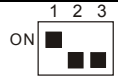
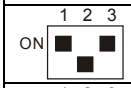
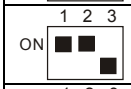
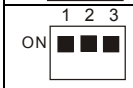
Hpaste

Hpaste x, y, layer,

Similaire à la commande GPASTE (mais avec des valeurs comprises entre 0 et 39).

Configuration des DIP switch des modèles « GHB3224C »

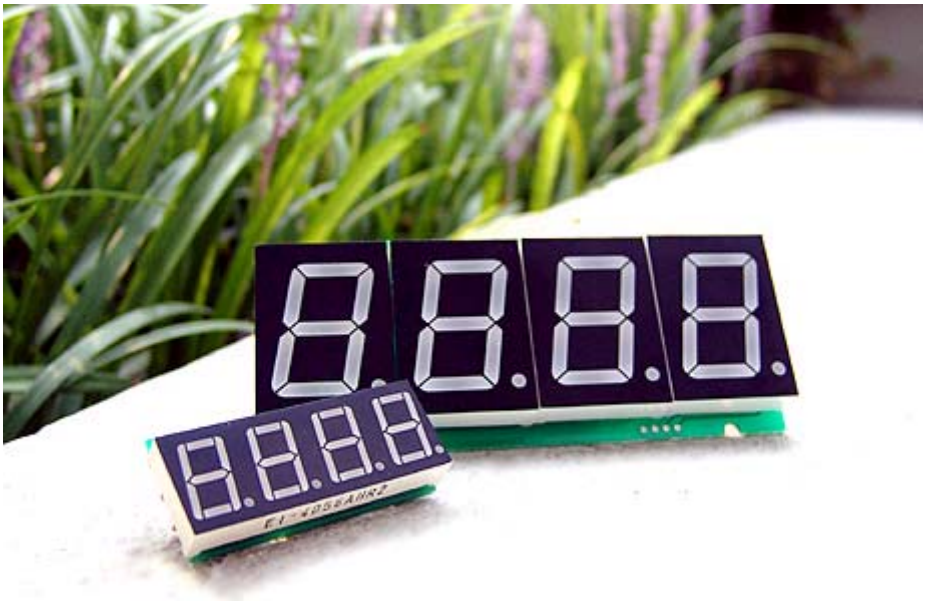
Au dos de l'afficheur « GHB3224C » vous disposez de DIP switch permettant la configuration de la vitesse de communication RS232 (si vous l'utilisez dans ce mode) ou de l'adresse esclave I2C™ (si vous l'utilisez dans ce mode). Le commutateur DIL N° 4 n'est pas utilisé.

DIP Switch	Débit RS232	Adresse esclave I2C™
	2400	0
	4800	1
	9600	2
	19200	3
	28800	4
	38400	5
	57600	6
	115200	7

Vous ne devez utiliser qu'un seul port de communication sur l'afficheur (série ou I2C™)

Afficheurs numériques 7 segments à Leds série « CSG »

Cette gamme d'afficheurs spéciaux de type 7 segments à Leds vous permettra d'afficher très facilement et rapidement des nombres à l'aide des modules CUBLOC.



Habituellement l'utilisation d''afficheurs standards de type7 segments à Leds nécessite le recours à un demultiplexage dynamique qui peut s'avérer difficile à mettre en œuvre. Afin de vous simplifier la tâche, Comfile Technology a dévelopé une gamme de modules spécialisés appelés « CSG » lesquels se pilotent via un bus I2C™.

Ces afficheurs disponibles selon les modèles en 2 tailles différentes sont dotés de 4 digits. Au dos de ces derniers vous disposez de 2 connecteurs 4 points et d'un commutateur DIL 4 positions. Il vous sera très facile de les piloter via les modules CUBLOC™ au moyen des commandes présentes dans le tableau ci-dessous.

Commande	Explication	Exemple d'utilisation
CSGDEC SlaveAdr, Data	Affiche une valeur décimale.	CSGDEC 0, I
CSGHEX SlaveAdr, Data	Affiche valeur hexadécimale	CSGHEX 0,I
CSGNPUT SlaveAdr, Digit, Data	Contrôle l'affichage d'un digit	CSGNPUT 0,0,8
CSGXPUT SlaveAdr, Digit, Data	Permet le contrôle de chaque segment d'un digit	CSGNPUT 0,0,9

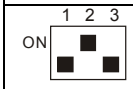
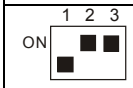
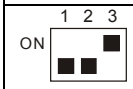
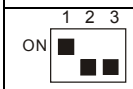
NOTA : Pour exploiter les commandes dédiées aux modules « CSG », il vous faudra impérativement utiliser en premier la commande setI2C.

Vous trouverez ci-dessous la description détaillée de ces commandes.

Csgdec

Csgdec slaveadr, Data
slaveadr : Adresse esclave du module « CSG »
Data : valeur décimale à afficher

Le paramètre slaveadr correspond à l'adresse esclave du module « CSG ». Cette adresse se configure au moyen du commutateur DIL présent au dos de l'afficheur. Il est ainsi possible d'utiliser 4 adresses différentes pour une ligne de bus I2C™.

DIP Switch	Adresse esclave
	0
	1
	2
	3

Pour afficher plus de 4 digits, il vous sera possible d'utiliser plusieurs modules « CSG » (sur des adresses esclaves différentes). Les modules seront chaînés en parallèle au moyen de leurs différents connecteurs 4 points.

Utilisez la commande **CSGDEC** pour afficher une valeur décimale sur un module « CSG ».

```
Const Device = cb280
Set I2c 9,8              ' Cette commande doit être appelée avant la commande Csgdec
b=8
Do
        Csgdec 0,b      ' Affichage sur le module GSG
        Delay 100
        b = b + 1
        If b=0 Then b=200
Loop
```

Csghex

CSGHEX slaveadr, data
slaveadr : Adresse esclave du module « CSG »
data : valeur hexadécimale afficher

Même commande que Csgdec mais pour l'affichage de valeur hexadécimale.

Csgnput

CSGNPUT slaveadr, digit, data
slaveadr : Adresse esclave du module « CSG »
digit : Position digit « de la gauche vers la droite » (0 ~ 3)
data : Donnée (&h30 à &h39, &h41 à &h46)

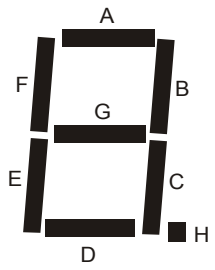
Permet d'afficher un chiffre (avec ou sans point décimal) ou une lettre sur un digit spécifié.
Data peut prendre les valeurs &H30 à &H39 pour afficher des chiffres de 0 à 9 (sans virgule).
Data peut prendre les valeurs &HB0 à &HB9 pour afficher des chiffres de 0 à 9 (avec virgule).
ou les valeurs &H41 à &H43 pour afficher respectivement les lettres : a b C d E F (sans virgule)
ou les valeurs &HC1 à &HC6 pour afficher respectivement les lettres : a b C d E F (avec virgule)
ou la valeur &H80 pour afficher seulement la virgule (sans chiffre)

```
CSGNPUT 0, 1, &H43          ' Affiche le C sur le digit 1
```

Csgxput

CSGXPUT slaveadr, digit, data
slaveadr : Adresse esclave du module « CSG »
digit : Position digit « de la gauche vers la droite » (0 ~ 3)
data : Donnée

Allume la LED à la position spécifiée. Cette commande est utilisée pour afficher autre chose que des chiffres en vous permettant de contrôler les digits séparément.



Bit	7	6	5	4	3	2	1	0
LED	H	G	F	E	D	C	B	A

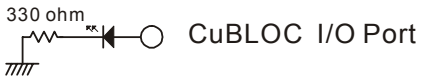
Pour afficher le caractère 'L', les Leds des positions D, E et F devront être allumée. Ainsi la valeur data devra être 0011 1000 (soit &H38 ou 0x38) et donc: CSGXPUT 0, 0, &H38

Chapitre 8.

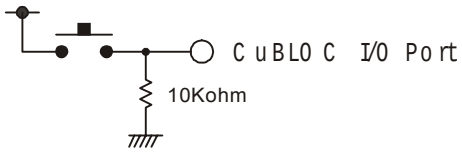
Interfaçage des modules CUBLOC™

Raccordement des Entrées / Sorties

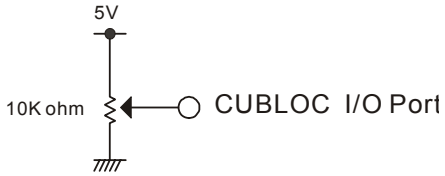
Comment connecter des Leds ?
Vous pouvez utiliser le schéma ci-dessous pour raccorder une Led sur un port configuré en sortie et sur lequel vous programmez un niveau logique haut pour allumer la Led. Le câble raccordant la Led au CUBLOC™ ne doit pas exéder une dizaine de centimètre.



Comment connecter un bouton-poussoir ?
Vous pouvez utiliser le schéma ci-dessous (le poussoir est relié au + 5 V) et configurer le port en entrée. Si vous n'utilisez pas de protection particulière, le fil reliant le bouton au CUBLOC™ ne devra pas exéder 3 – 4 cm afind'éviter les phénomènes de Latch-up. Lorsque le bouton est pressé, le CUBLOC™ recevra un niveau HAUT (et un niveau BAS en cas contraire).

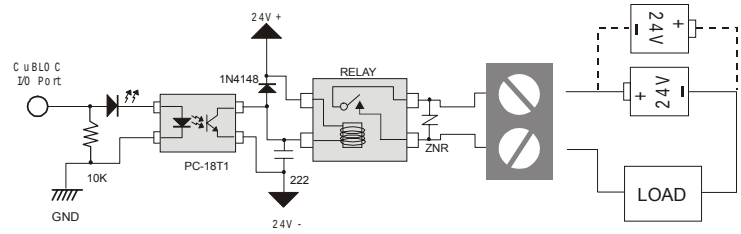


Comment connecter un potentiomètre ?
Vous pouvez utiliser le schéma ci-dessous. Le potentiomètre est relié entre le +5 V et la masse, tandis que son curseur est relié sur un port de conversion analogique/numérique du CUBLOC™ (lequel devra être configuré en entrée avant de pouvoir utiliser la commande ADIN pour lire la valeur de la tension présente sur le curseur). Si vous n'utilisez pas de protection particulière, le fil reliant le bouton au CUBLOC™ ne devra pas exéder 3 – 4 cm afin d'éviter les phénomènes de Latch-up.

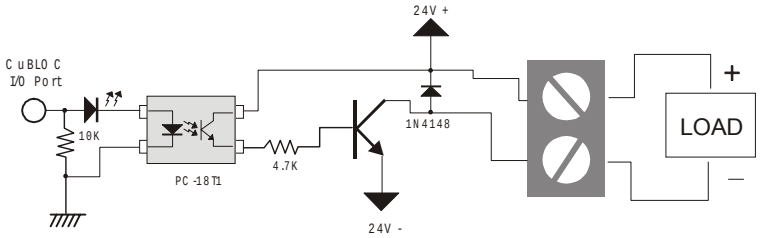


Dans tous les cas le module CUBLOC™ ne supporte que des niveaux de tension maximal de +5 V. Si vous devez utiliser des tensions de niveau supérieur, utilisez un régulateur de tension adapté.

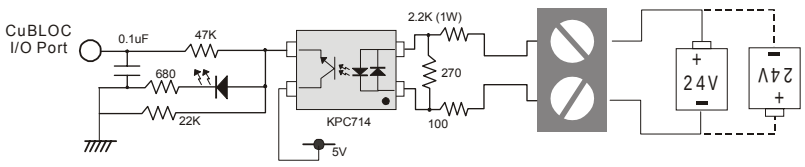
Comment connecter un relais sur le CUBLOC ?
Le schéma ci-dessous donne un exemple permettant de connecter un relais au CUBLOC™ en utilisant un photocoupleur afin de bénéficier d'une isolation vis à vis du système qui sera actionné par le relais. Le port du CUBLOC™ devra être configuré en sortie. Pour les applications dans lesquelles le système piloté n'est pas générateur de perturbation, il sera possible d'utiliser un montage simplifié dans lequel l'optocoupleur sera remplacé par un simple transistor de type NPN avec une résistance de 10 Kohms à sa base.



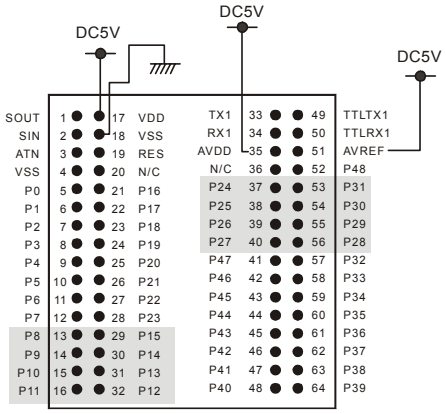
Comment connecter un transistor NPN sur un CUBLOC ?
Le schéma ci-dessous donne un exemple permettant de connecter un transistor de type NPN sur un CUBLOC™ au moyen d'un photocoupleur qui vous permettra de bénéficier d'une isolation vis à vis du système qui sera actionné par transistor.



Comment connecter une entrée en 24 Vcc sur un CUBLOC ?
Utilisez un photocoupleur à double polarité pour convertir le signal 24 Vcc en signal 5 Vcc. Dès lors, lorsque l'entrée est sollicitée, le module CUBLOC™ recevra un niveau HAUT (+ 5 V) sur le port d'E/S qu'il conviendra de configurer en entrée.

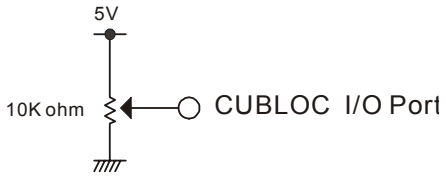


Comment raccorder une entrée analogique ?
Pour connecter une entrée A/N du module CUBLOC™ CB280, il vous faudra au préalable raccorder les broches AVDD et AVREF au +5 Vcc. La broche AVDD alimente l'étage du convertisseur A/N du module CUBLOC, tandis que la broche AVREF correspond à l'entrée de la tension de référence utilisée pour les conversions. Dès lors, si AVREF est connecté au + 5 V, le CUBLOC™ convertira les tensions d'entrée comprises entre 0 et 5 V en valeurs numériques et si par exemple AVREF est connecté au + 3 V, le CUBLOC™ convertira les tensions d'entrée comprises entre 0 et 3 V en valeurs numériques.

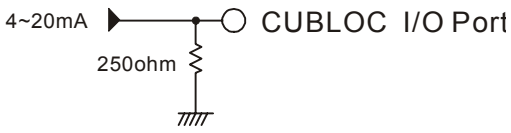


Sur le module CUBLOC™ CB220 les broches AVDD et AVREF sont directement reliées en interne au + 5 V CC.

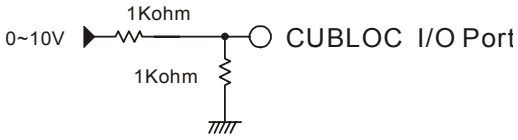
Le schéma ci-dessous montre comment relier un potentiomètre sur une entrée de conversion analogique/numérique d'un CUBLOC afin d'obtenir un nombre compris entre 0 et 1023. Si vous n'utilisez pas de protection particulière, le fil reliant le bouton au CUBLOC™ ne devra pas exéder 3 – 4 cm afind'éviter les phénomènes de Latch-up.



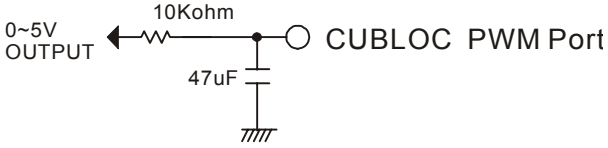
Le schéma ci-dessous montre un exemple d'utilisation simplifié (uniquement valable pour des tests et non pas pour un usage au sein d'une application finie) afin de permettre à une entrée de conversion « A/N » du CUBLOC de récupérer des signaux 4 – 20 mA. Vous pouvez utiliser une résistance de 230 Ohms en série avec une seconde résistance de 20 Ohm pour arriver à la valeur de 250 Ohms.



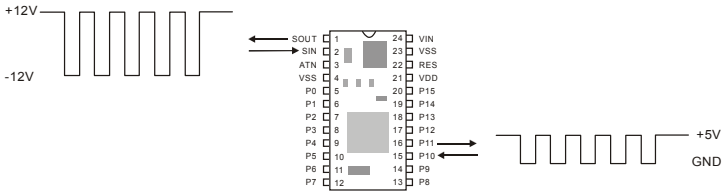
Pour mesurer un signal compris entre 0 et 10 V, utilisez un pont diviseur réalisé au moyen de 2 résistances de même valeur.



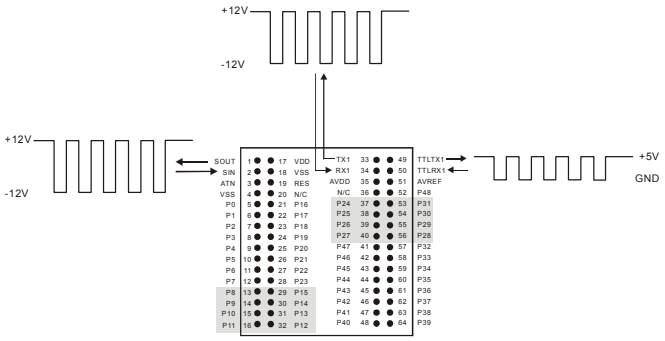
Comment générer une tension analogique à partir d'un signal PWM ?
Les modules CUBLOC™ disposent de 3 à 6 ports capables de générer des signaux PWM. Le schéma ci-dessous vous permettra de générer des tensions analogiques au moyen des sorties PWM. Ce schéma n'est valable de pour la réalisation de test (la tension ne devra pas être utilisée pour alimenter un dispositif externe sans quoi cette dernière risque de s'écrouler).



Comment raccorder un dispositif RS232 à un module CUBLOC™ ?
Par exemple, sur le module CUBLOC™ CB220, les broches 1 et 2 sont dédiées au canal 0 de communication série utilisé pour recevoir les signaux RS232 (avec niveaux +/- 12V) d'un PC afin de pouvoir programmer le CUBLOC™. Le CB220 dispose également d'un second port série (canal 1), lequel est doté de niveau logique 5 V.



Pour le module CB280, le canal 1 de communication série dispose à la fois de signaux accessibles en version avec niveaux logiques 5 V et en version avec niveau +/- 12 V.



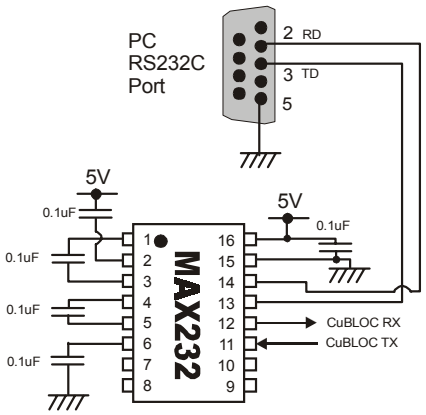
Les signaux du port 1 de communication série dotés de niveau logique 5 V pourront être utilisés par exemple pour dialoguer avec des modules microcontrôlés ou des périphériques alimentés en + 5 V.

Les signaux du port 1 de communication série dotés de niveaux logiques +/- 12 V pourront être utilisés pour dialoguer directement avec un dispositif RS-232 (tel qu'un PC par exemple).

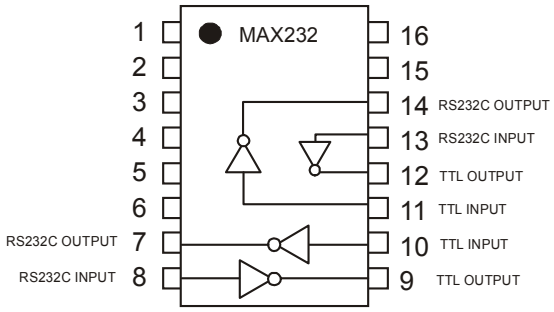
Il faudra cependant n'utiliser qu'un seul type de signaux à la fois (soit ceux dotés d'un niveau logique 5 V, soit ceux dotés d'un niveau logique +/- 12 V), mais jamais les 2 simultanément.

Il pourra également être intéressant d'avoir recours aux signaux de niveau logique 5 V afin de pouvoir piloter des composants de conversion externes susceptibles de pouvoir mettre les signaux du CUBLOC™ à niveau vis à vis de dispositifs externes de type RS422 ou RS485 ou RS232.

Le schéma ci-dessous donne un exemple de conversion mettant à profit un composant de type MAX232 (permettant de mettre à niveau des signaux série au niveau logique 5 V vers des signaux de niveaux logique +/- 12 V).



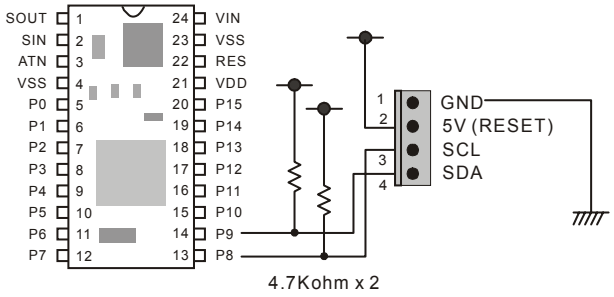
Ci-dessous vous trouverez la représentation interne du MAX232.



Les périphériques « CuNET » dédiés aux CUBLOC™

Les CUBLOC™ peuvent piloter des modules périphériques optionnels spéciaux appelés modules « CuNET » (afficheurs « CLCD », afficheurs 7 segments « CSG »…) via une liaison de type I2C™. Par exemple pour les afficheurs « CLCD », vous disposez d'un jeu d'instructions BASIC permettant de génerer directement des trames de données I2C™ que les afficheurs « CLCD » comprendons et traduirons en actions associées.

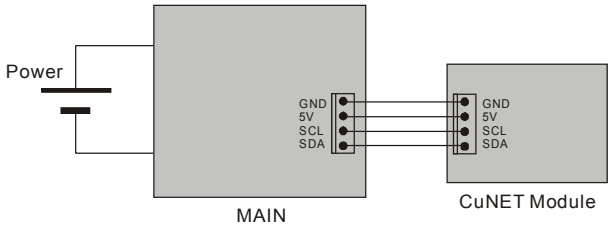
Pour piloter ces afficheurs, il vous faudra ajouter 2 résistances de tirage au niveau haut de 4.7 Kohms sur les lignes SCL et SDA. Ces lignes sont de type collecteur ouvert avec une réjection automatique des impulsions inférieures à 50 ns.



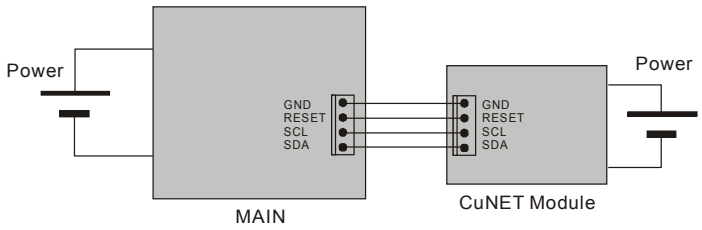
Les platines d'essais des modules CUBLOC™ disposent d'un connecteur standardisé 4 points dédié aux communications avec les modules « CuNET ». La broche 1 correspond à la masse, la broche 2 au + 5 V (ou Reset), la broche 3 au signal « SCL » et la broche 4 au signal « SDA ».

Lorsque vous utilisez un module « CuNET », le CUBLOC agira en tant que « maître » et les périphériques connectés sur la liaison en tant « qu'esclaves ». Tous les périphériques spécialisés (alors en mode attente) répondront au module CUBLOC™. De ce fait, les modules esclaves ne pourront pas initier eux même la communication avec le « maître ».

La broche 2 du connecteur dédié à la communication avec les modules « CuNET » permet de dupliquer et de ramener le + 5 V d'alimentation vers le module d'interface spécialisé « CuNET » :



Dans certains cas, cette borne + 5 V pourra être reliée à la borne RESET du module d'interface spécialisé « CuNET » afin que ce dernier s'initialise à la mise sous tension du module CUBLOC. Dans ce cas, le module d'interface spécialisé « CuNET » devra disposer de sa propre alimentation.



ATTENTION : la longueur des connexions des lignes vers le module « CuNET » ne devra pas dépasser une vingtaine de centimètres. Pour des liaisons de longueurs plus importantes, il vous faudra utiliser des composants amplificateurs de bus I2C™ (type P82B96 ou P82B715 par exemple). Dans tous les cas, on veillera également que les lignes de liaisons I2C™ (amplifiée ou non) ne cohabitent pas avec des câbles de signaux de puissance.

A propos des communications I2C™…
Les modules CUBLOC™ disposent d'un jeu complet de commandes qui leur permettront de pouvoir communiquer via le protocole I2C™. Ce protocole est très utilisé pour pouvoir piloter différents types de composants externes tels que des convertisseurs « A/N », des mémoires EEprom, des convertisseurs « N/A », des ports d'extension d'E/S…

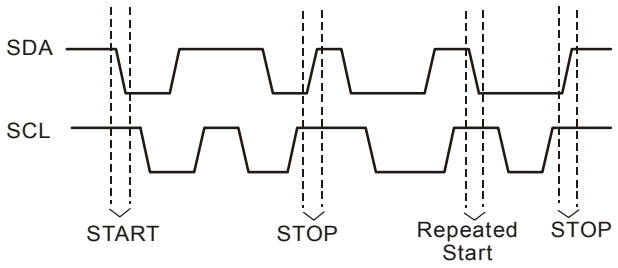
Une communication I2C™ nécessite l'utilisation de 2 ports (SDA et SCL) afin de pouvoir travailler en modes MAITRE (MASTER) ou ESCLAVE (SLAVE). Les modules CUBLOC™ peuvent uniquement travailler en mode MAITRE (MASTER).

Avant de pouvoir exploiter une communication I2C™ avec les modules CUBLOC™, il vous faudra au préalable déclarer le port I2C™ à l'aide de la commande **SET I2C**.

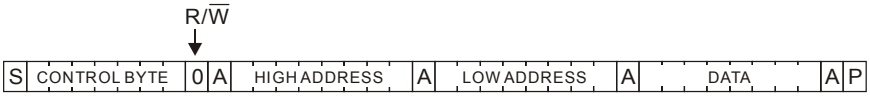
Conditions I2C™ START, STOP
Lorsque les signaux SCL(Clock) et SDA(Data) sont au niveau logique HAUT, le bus I2C™ est en mode attente. Si une commande **START** est exécutée durant cet état, la communication I2C™ commence alors.

Lorsque SCL et SDA sont tous les 2 au niveau logique BAS, le bus I2C™ est occupé. Si une commande **STOP** est exécutée durant cet état, la communication I2C™ est alors stoppée.

Il existe également une condition de répétition du mode Start en I2C™. Si la commande **START** est exécutée alors que le bus est occupé, la communication I2C™ démarre à nouveau.



Utilisation d'une mémoire EEPROM via le bus I2C™
La description qui suit montre comment réaliser une communication entre un module CUBLOC™ et une mémoire EEprom I2C™ externe de type 24LC32. Le schéma ci-dessous est extrait du data sheet de la mémoire EEPROM. Ce dernier montre comment envoyer des données dans la mémoire EEPROM.



S : Start
A : Acknowledge
P : Stop

Le premier bit correspond à la commande de **START**. Les 4 bits de poids fort de l'octet CONTROL BYTE doivent être configurés à 1010 et les 3 bits de poids faibles servent à sélectionner l'adresse « I2C™ » du composant. Cette adresse peut être modifiée suivant le niveau logique à appliquer sur certaines broches du composant. Il vous sera ainsi possible d'adresser plusieurs mémoires EEPROM via le bus I2C™ en utilisant des adresses « I2C™ » différentes pour chaque composant.

Pour une opération de lecture, le bit R/W devra être à 1 et à 0 pour une opération d'écriture.

Le bit A correspond à un bit d'acquittement retourné par le composant (pour indiquer au module maître – ici le CUBLOC™ - qu'il a bien reçu 8 bits de données.

Ensuite il faudra envoyer l'information de l'adresse mémoire de L'EEPROM sur laquelle on désire travailler (HIGH ADDRESS, LOW ADDRESS) et enfin les données à mémoriser dans la mémoire (DATA).

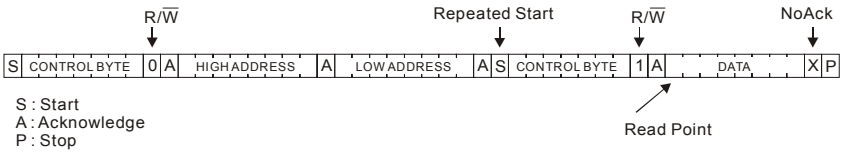
A chaque fois que la mémoire reçoit une série de 8 bits, ce dernier génère un bit d'acquittement et en fin de communication, il faudra envoyer une commande STOP pour indiquer que la communication est finie et que le bus I2C™ est libre.

A noter que l'opération d'écriture d'un octet en mémoire EEPROM nécessite environ 5 ms après que les commandes aient été reçues par le composant (tenez en compte dans votre programme).

L'exemple ci-dessous traduit en langage BASIC pour le CUBLOC™ la procédure d'écriture d'un octet dans une mémoire EEPROM I2C™:

```
Set I2c 8,9                    ' Configure les Port P8 en SDA et P9 en SCL
I2cstart
If I2cwrite(&H10100000) = 1 Then ERR_PROC    ' Adresse I2C™ composant = 0
If I2cwrite(ADR.BYTE1) = 1 Then ERR_PROC     ' Adresse en mémoire EEPROM
If I2cwrite(ADR.LOWBYTE) = 1 Then ERR_PROC
If I2cwrite(DATA) = 1 Then ERR_PROC          ' Ecriture d'un Octet
I2cstop
Delay 5                ' Attend que la mémoire EEprom se soit programmée
```

Dans l'étape ci-dessous en s'attardera à voir comment on peut lire un octet depuis la mémoire EEPROM. Bien que l'opération puisse paraître plus complexe que pour l'écriture d'un octet, vous vous apercevrez rapidement que l'opération et assez similaire.



La lecture effective de la donnée en EEPROM débute à l'indication « Read Point » . L'ensemble des commandes précédentes sert à la configuration mémoire et à l'adressage du composant.

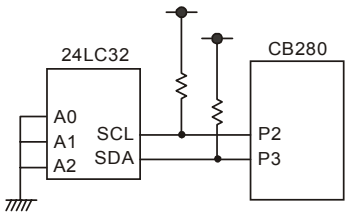
```
Set I2c 8,9                          ' Configure les broches SDA, et SCL
I2cstart
If I2cwrite(&H10100000) = 1 Then ERR_PROC    ' Adresse I2C™ composant = 0
If I2cwrite(ADR.BYTE1) = 1 Then ERR_PROC     ' Adresse en mémoire EEPROM
If I2cwrite(ADR.LOWBYTE) = 1 Then ERR_PROC
I2cstart                                     ' Nouvelle condition Start (Repeated Start)
If I2cwrite(&H10100001) = 1 Then ERR_PROC    ' Commande demandant la lecture.
DATA = I2cread(0)                            ' Récupération de l'octet de l'EEPROM.
I2cstop
```

Maintenant voyons comment lire une suite de données depuis la mémoire EEPROM. Sans utiliser la commande STOP, la lecture de la mémoire EEPROM pourra être continue du fait que celle-ci incrémente automatiquement son adresse mémoire. De ce fait, il suffira simplement de lui indiquer l'adresse mémoire de début pour ensuite lire les données les unes à la suite des autres.

```
Set I2c 8,9                          ' Configure les broches SDA, et SCL
I2cstart
If I2cwrite(&H10100000) = 1 Then ERR_PROC    ' Adresse I2C™ composant = 0
If I2cwrite(ADR.BYTE1) = 1 Then ERR_PROC     ' Adresse en mémoire EEPROM
If I2cwrite(ADR.LOWBYTE) = 1 Then ERR_PROC
I2cstart                                     ' Nouvelle condition Start (Repeated Start)
If I2cwrite(&H10100001) = 1 Then ERR_PROC    ' Commande demandant la lecture
For I = 0 To 10
      ADATA(I) = I2cread(0)                  ' Lectutre de 10 octets à la suite
                                             ' avec mémorisation dans tableau ADATA
Next
I2cstop
```

Exemple complet de communication I2C™
L'exemple qui suit montre comment interfacer un module CUBLOC™ CB280 avec une mémoire EEPROM 24LC32. Il vous permettra d'écrire une donnée à une adresse précise de la mémoire EPROM, puis de relire cette dernière et de l'afficher dans la fenêtre de DEBUG du PC sous l'environnement du CUBLOC Studio. Dans cet exemple la présence du bit d'acquittement n'est pas systématiquement vérifié.

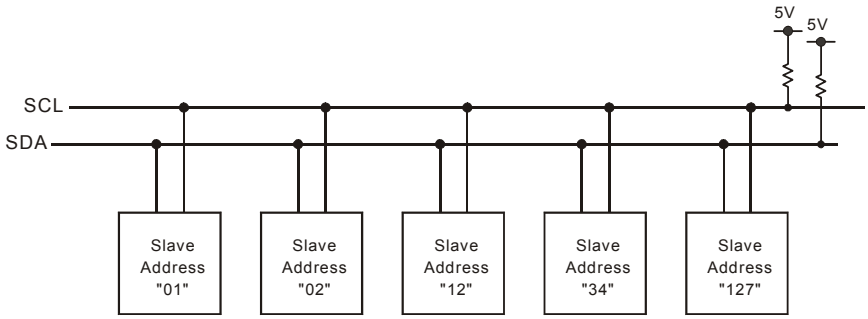
```
Const Device = cb280
Dim adr As Integer
Dim data As Byte
Dim a As Byte
data = &ha1                    ' Valeur de la donnée à mémorirer
adr = &h3                      ' Adresse mémoire en EEPROM
Set I2c 3,2                    ' Configure les broches SDA, et SCL
Do
    ‘ Ecriture d'un Octet en mémoire EEPROM
    I2cstart
    If I2cwrite(&b10100000)= 1 Then Goto err_proc
    a=I2cwrite(adr.byte1)
    a=I2cwrite(adr.lowbyte)
    a=I2cwrite(data)
    I2cstop
    Delay 1000                 ‘ Tempo pour programmation en EEPROM
    ‘ Lecture d'un Octet depuis la mémoire EEPROM
    I2cstart
    a=I2cwrite(&b10100000)
    a=I2cwrite(adr.byte1)
    a=I2cwrite(adr.lowbyte)
    I2cstart
    a=I2cwrite(&b10100001)
    a=I2cread(0)
    I2cstop
    ‘ Affiche la valeur lue à l'écran
    Debug Hex a,cr
    Delay 500
Loop

err_proc:
    Debug "Erreur communication I2C !"
    Do
    Loop
```



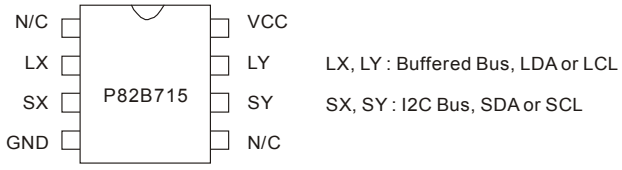
Utilisation avancée des communications I2C™ …
La plupart des ports des modules CUBLOC™ peuvent être configurés pour réaliser une communication I2C™ (au moyen d'une émulation « logiciel » via les commandes « I2CREAD ou I2CWRITE »). Ainsi il vous sera possible de déclarer et d'utiliser jusqu'à 24 Ports I2C™ différents sur un module « CB280 » !

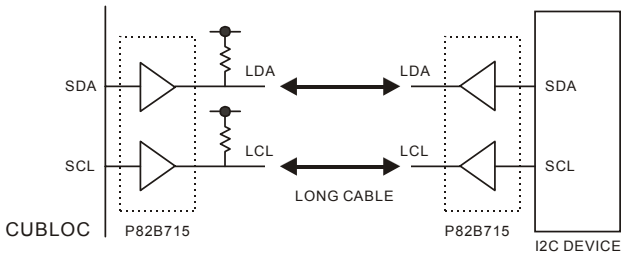
Seuls les ports pouvant être configurés à la fois en « entrée ou en sortie » doivent être utilisés pour les communications I2C™. Il ne faudra pas utiliser les ports uniquement utilisables en entrée ou en sortie pour les communications I2C™.

Les communications I2C™ générées par les modules CUBLOC™ sont de type « maître ». C'est à dire que seuls des périphériques I2C™ « esclaves » pourront être pilotés par les CUBLOC™.

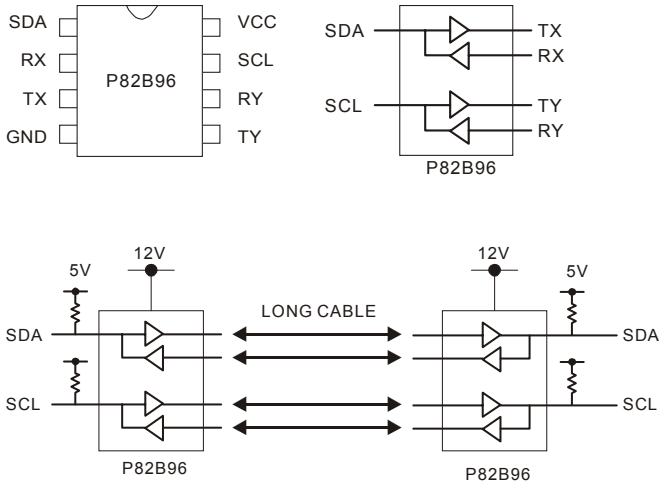


Rappelez-vous que les connexions des communications I2C™ ne peuvent pas s'étendre sur de longues distances. EN cas contraire, il vous faudra utiliser des circuits intégrés permettant d'étendre les distances (par exemple le circuit P82B715). Nous vous recommandons également le circuit P82B96 qui s'apparente à un buffer destiné à protéger les composants I2C™ des perturbations et surtensions.



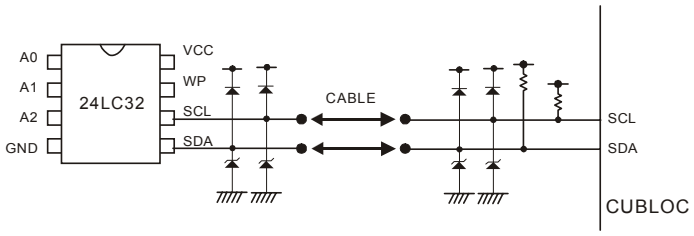


Le circuit « P82B96 » vous permettra d'isoler la commuincation I2C™ et d'améliorer la protection du bus de communication.



Consultez le site de Philips™ pour plus d'infos sur ces composants : http://www.standardics.philips.com/.

Si vous utilisez une communication I2C™ sur une grande distance sans circuit intégré de protection spécialisé, nous vous suggérons toutefois d'avoir recours à des diodes de protection comme indiqué ci-dessous :



Toutefois nous vous recommandons depréférence d'avoir recours aux circuits de protection décrits ci-avant.

Chapitre 9.

Communications MODBUS™

MODBUS™ est un protocole développé par la société MODICON™ pour aider à interfacer les périphériques avec des PLC.

Ce protocole est généralement utilisé pour les appareils tels que les écrans à touches tactiles, les modules HMI et les logiciels type SCADA. Nombreux sont ces périphériques qui supportent le protocole MODBUS.

Dans une communication de type MODBUS, il y a un mode maître et un mode esclave. Le maître envoie les données tandis que l'esclave les reçoit. L'esclave ne peut que recevoir les données et ne peut « appeler » le maître.

Chaque esclave dispose d'une adresse unique (appelée adresse esclave). Le maître ne peut dialoguer qu'avec un seul esclave à la fois (en fonction de son adresse).

Pour une connexion entre un maître et un esclave (1 vers 1), une liaison RS232 pourra être utilisée. Pour une connexion de type 1 vers N, il vous faudra utiliser une connexion de type RS485. Le maître envoie des messages sous la forme de « trames ». Chaque trame contient l'adresse esclave, des commandes, des données, des codes de contrôle d'erreur. L'esclave réceptionne la trame et l'analyse. Lorsqu'un esclave répond au maître, ce dernier lui renvoi également des trames.

En d'autres mots, au cours d'une communication MODBUS les informations envoyées et reçues se font sous la forme de trames.

Il existe 2 modes de communication MODBUS. Le mode ASCII et RTU. Le mode RTU peut être intégré en utilisant moins d'octets dans la communication. Le mode ASCII utilise des LRM pour les contrôles d'erreur, tandis que le mode RTU utilise des CRC.

Vous trouverez ci-après comment sont utilisés les modes ASCII et RTU:

Field	Hex	ASCII	RTU
Header		: (colon)	None
Slave Address	0X03	0 3	0X03
Command	0X01	0 1	0X01
Start Address HI	0X00	0 0	0X00
Start Address LO	0X13	1 3	0X13
Length HI	0X00	0 0	0X00
Length LO	0X25	2 5	0X25
Error Check		LRC (2 Bytes)	CRC(2 Bytes)
Ending Code		CR LF	None
Total Bytes		17 Bytes	8 Bytes

Le mode ASCII utilise 2 points (:) pour démarrer et un CR ou LF pour finir.

START	SLAVE ADR	FUNCTION	DATA	LRC	END
: (COLON)	2 Bytes	2 Bytes	n Bytes	2 Bytes	CR,LF

Le mode RTU ne nécessite aucun caractère spécial pour démarrer. Ce dernier utilise 4 octets « Blanc » pour indiquer le début ou la fin.

START	SLAVE ADR	FUNCTION	DATA	CRC	END
T1-T2-T3-T4	1 Byte	1 Byte	N Bytes	1 Byte	T1-T2-T3-T4

Les CUBLOC™ supportent les commandes et les adresses MODBUS™
Les CUBLOC™ supportent les commandes 1, 2, 3, 4, 5, 6, 15 et 16.

Commande	Nom de la commande
01, 02	Bit Read
03, 04	Word Write
05	1 Bit Write
06	1 Word Write
15	Multiple Bit Write
16	Multiple Word Write

En MODBUS™ il y a des notions d'adresses qui s'apparentent à des registres pour les CUBLOC™. Les registres P, M, F, C, T et D des CUBLOC™ peuvent être adressés en utilisant la table ci-dessous :

Bit Units		Word Units	
Addresse	Registre	Addresse	Registre
0000H	P		
1000H	M		
2000H	Not Used		
3000H	Not Used		
4000H	F		
		5000H	T
		6000H	C
		7000H	D
		8000H	WP
		9000H	WM
		0A000H	WF

Adresses des modules
Le tableau ci-dessous montre les adresses des modules MODBUS™. Ces adresses sont utilisées pour identifier les différents registres sur les CUBLOC™ ou les CUTOUCH™. La plupart des équipements hôtes (CUBLOC™, CUTOUCH, PC, HMI) devront utiliser les règles ci-dessous :

Adresse module	Adresse Modbus	Explication
1…10000	Adresse module – 1	Enlevez 1 unité pour obtenir l'adresse Modbus™
40001 … 50000	Adresse module – 40001	Enlevez 40001 unité pour obtenir l'adresse Modbus

Les adresse des modules après 40000 correspondent à des registres de type word accessibles en 16 bits).

Consultez la table d'adresse ci-dessous lorsque vous utilisez les communications Modbus™ avec les CUBLOC™ ou les CUTOUCH™. Les adresses des modules ci-dessous sont indiquées en décimale.

Accès bit (Coil, Etat entrée)	
Codes de fonction : 1,2,4,15	
Adresse module (décimal)	Données
1 à 128	Registres P
385 à 512	Registres F
4097 à 8192	Registres M

Accès Word (Holding/Registres d'entrées)	
Codes de fonction : 3,4,6,16	
Adresse module (décimal)	Données
40001 à 41000	Registres D
41001 à 42000	Registres T
42001 à 43000	Registres C
43001 à 44000	Registres WM

Adresses « flottantes »
Utilisez l'adresse des modules en fonction des No de registres disponibles.

Par exemple, le CUBLOC™ « CB280 » dispose de registres de données de D0 à D99. Il ne peut exister que des adresses de 40001 à 40099. Le reste de 400100 jusqu'à 41000 (les adresses « flottantes ») ne sont pas utilisables (elles seront exploitées pour les mises à jour futures du firmware). Donc… Ne les utilisez pas !

Fonction code 01 : Lecture état bobine
Fonction code 02 : Lecture de l'état d'une entrée
Cette fonction permet de lire le bit d'état des registres du PLC. L'exemple suivant montre la lecture des registres P20 à P56 depuis l'adresse esclave 3.

Requête:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X01	0 1	2
Start Address HI	0X00	0 0	2
Start Address LO	0X14	1 4	2
Length HI	0X00	0 0	2
Length LO	0X25	2 5	2
Error Check		LRC	2
Ending Code		CR LF	2

LRC est le 2's complément de la somme sur 8 bits de toutes les valeurs sauf Colon, CR et LF.
Pour la table ci-dessus : 0x03 + 0x01 + 0x13 + 0x25 = 0x3C.
Pour trouver le 2's complément de 0x3C, on peut l'écrire en premier en binaire.
0011 1100

Ensuite on inverse les bits.
1100 0011

Ensuite on ajoute 1, ce qui donne:
1100 0100 = 0xC4

LRC = 0xC4

ASCII	:	0	3	0	1	0	0	1	3	0	0	2	5	C	4	CR	LF
Hex	3A	30	33	30	31	30	30	31	33	30	30	32	35	43	34	13	10

La réponse à la requête est en page suivante.

Réponse:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X01	0 1	2
Byte Count	0X05	0 5	2
Data 1	0X53	5 3	2
Data 2	0X6B	6 B	2
Data 3	0X01	0 1	2
Data 4	0XF4	F 4	2
Data 5	0X1B	1 B	2
Error Check		LRC	2
Ending Code		CR LF	2

Vous pourrez constater que les bits 20 à 27 forment 1 octet. P20 correspond au bit LSB de la donnée 1 et P27 correspond au bit MSB de la donnée 1. Ainsi vous pourrez récupérer les informations de P20 à P56 tout en ne tenant pas compte des autres bits.

Fonction code 03 : Lecture registres maintenu
Fonction code 04 : Lecture regsitres d'entrée
Cette fonction permet de lire une donnée de type « Word » (sur 16 bits) – ce qui se évèle intérressant pour les Compteurs, Timers et Registres de données. Le tableau suivant montre comment lire les Registres D 0 à 2 depuis l'adresse esclave 3.

Requête:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X03	0 3	2
Start Address HI	0X70	7 0	2
Start Address LO	0X00	0 0	2
Length HI	0X00	0 0	2
Length LO	0X03	0 3	2
Error Check		LRC	2
Ending Code		CR LF	2

Une donnée « Word » tient en 2 octets : nous obtiendrons donc une réponse sur 6 octets.

Réponse:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X03	0 3	2
Byte Count	0X06	0 6	2
Data 1 LO	0X03	0 3	2
Data 1 HI	0XE8	E 8	2
Data 2 LO	0X01	0 1	2
Data 2 HI	0XF4	F 4	2
Data 3 LO	0X05	0 5	2
Data 3 HI	0X33	3 3	2
Length LO	0X03	0 3	2
Error Check		LRC	2
Ending Code		CR LF	2

Fonction Code 05 : Ecriture d'une « bobine »
Il est possible de piloter l'état des registres à distance d'un PLC via les bits grâce à cette fonction. L'exemple suivant montre comment le registre P1 sera activé (ON) via l'adresse esclave 3. Pour activer les registres (ON), FF 00 est envoyé et pour désactiver les registres (OFF) on envoi 00 00.

Requête:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X05	0 5	2
Start Address HI	0X01	0 1	2
Start Address LO	0X00	0 0	2
Length HI	0XFF	F F	2
Length LO	0X00	0 0	2
Error Check		LRC	2
Ending Code		CR LF	2

La réponse montre que les données on été correctement envoyée.
Vous DEVEZ utiliser FF 00 et 00 00 pour activer/désactiver (ON/OFF) les registres (toute autre valeur sera ignorée).

Réponse:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X05	0 5	2
Start Address HI	0X01	0 1	2
Start Address LO	0X00	0 0	2
Length HI	0XFF	F F	2
Length LO	0X00	0 0	2
Error Check		LRC	2
Ending Code		CR LF	2

Fonction Code 06 : Initialisation registre unique
Il est possible de piloter l'état des registres à distance d'un PLC via des Words au moyen de cette fonction. L'exemple suivant montre comment adresser le D1 via l'adresse esclave 3.

Requête:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X06	0 6	2
Start Address HI	0X70	0 1	2
Start Address LO	0X01	7 0	2
Length HI	0X12	1 2	2
Length LO	0X34	3 4	2
Error Check		LRC	2
Ending Code		CR LF	2

Réponse:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X06	0 6	2
Start Address HI	0X70	0 1	2
Start Address LO	0X01	7 0	2
Length HI	0X12	1 2	2
Length LO	0X34	3 4	2
Error Check		LRC	2
Ending Code		CR LF	2

Fonction Code 15: Ecritures de multiples « bobines »
Il est possible de piloter l'état des registres à distance d'un PLC via de multiples bits grâce à cette fonction. L'exemple suivant montre comment les registres P20 à P30 seront activés/désactivés via l'adresse esclave 3.

Requête:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X0F	0 F	2
Start Address HI	0X00	0 0	2
Start Address LO	0X14	1 4	2
Length HI	0X00	0 0	2
Length LO	0X0B	0 B	2
Byte Count	0X02	0 2	2
Data 1	0XD1	D 1	2
Data 2	0X05	0 5	2
Error Check		LRC	2
Ending Code		CR LF	2

La table ci-dessus montre comment répartir les données de la requête. P27 est placé en tant que MSB du premier octet envoyé et P20 est placé en tant que LSB du premier octet. Il y a un total de 5 octets envoyés avec cette méthode. Les autres bits peuvent être mis à zéro.

Bit	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1
Relay	P27	P26	P25	P24	P23	P22	P21	P20						P30	P29	P28

Réponse:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X0F	0 F	2
Start Address HI	0X00	0 0	2
Start Address LO	0X14	1 4	2
Length HI	0X00	0 0	2
Length LO	0X0B	0 B	2
Error Check		LRC	2
Ending Code		CR LF	2

Fonction Code 16 : Ecritures Multiples de registres
Il est possible de piloter l'état des registres à distance d'un PLC via de multiples words à la fois grâce à cette fonction. L'exemple suivant montre comment adresser les Registres D0 à D2 via l'adresse esclave 3.

Requête:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X10	1 0	2
Start Address HI	0X70	7 0	2
Start Address LO	0X00	0 0	2
Length HI	0X00	0 0	2
Length LO	0X03	0 3	2
Byte Count	0X06	0 6	2
Data 1 HI	0XD1	D 1	2
Data 1 LO	0X03	0 3	2
Data 2 HI	0X0A	0 A	2
Data 2 LO	0X12	1 2	2
Data 3 HI	0X04	0 4	2
Data 3 LO	0X05	0 5	2
Error Check		LRC	2
Ending Code		CR LF	2

Réponse:

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X10	1 0	2
Start Address HI	0X70	7 0	2
Start Address LO	0X00	0 0	2
Length HI	0X00	0 0	2
Length LO	0X03	0 3	2
Error Check		LRC	2
Ending Code		CR LF	2

Contrôle d'erreur
Si une erreur survient dans les données depuis le maître, l'esclave retournera un code d'erreur.

Field	Hex	ASCII	Bytes
Header		: (colon)	1
Slave Address	0X03	0 3	2
Function Code	0X81	8 1	2
Error Code	0X09	0 9	2
Error Check		LRC	2
Ending Code		CR LF	2

Voici les différents type d'erreurs:

Code	Nom erreur	Explication
01	ILLEGAL FUNCTION	Lorsque un code de fonction non supporté est reçu.
02	ILLEGAL DATA ADDRESS	Lorsque d'une adresse incorecte est reçue.
03	ILLEGAL DATA VALUE	Lorsque de mauvaise données sont reçue.
09	LRC UNMATCH	Lorsque le LRC est incorrect.

Le contrôle d'erreur est uniquement pour le Modbus™ ASCII, il n'y a pas de contrôle d'erreur en RTU. Le Modbus™ RTU utilise les CRC pour contrôler les erreurs de transmission.

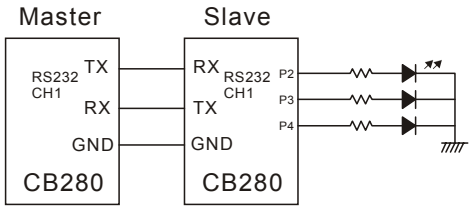
Mode MODBUS ASCII Maître

Il n'y a pas de commande spéciale pour placer le module CUBLOC en mode maître pour les communications de type MODBUS™. Le mode maître pourra simplement être mis en place au travers de la liaison série RS232 en utilisant les commandes usuelles telles que GET et PUT. L'exemple suivant montre l'implantation du mode maître dans le BASIC du CUBLOC™:

```
Const Device = cb280
        Dim RDATA As String * 80
        Dim a As Byte, ct As Byte
        Dim b As String * 17
        Dim Port As Integer
        Opencom 1,115200,3,80,80
        On Recv1 Gosub GETMODBUS      ' Routine d'interruption réception données
        Set Until 1,60,10             ' Quand donnée se termine par Code (10)
                                      ' sur canal 1 -> Création d'une interruption
        Do
                For Port=2 To 4
                        BitWrite Port, 1      ' Place P0, P1, P2 ON !
                        Delay 100
                Next
                For Port=2 To 4
                        BitWrite Port, 0      ' Place P0, P1, P2 OFF !
                        Delay 100
                Next

        Loop

GETMODBUS:
        If Blen(1,0) > 0 Then         ' Si le buffer est vide alors
                A=Blen(1,0)           ' Stock la taille sur buffer dans A !
                Debug "GOT RESPONSE: "
                B=Getstr(1,A)         ' Stock données reçues dans B
                Debug B
        End If
        Return
End
        Sub BitWrite(K As Integer, D As Integer)
                Dim LRC As Integer
                Putstr 1,":0305"
                Putstr 1,Hp(k,4,1)
                If D=0 Then
                        Putstr 1,"0000"
                        LRC = -(3+5+K.Byte1+K.Byte0)       'Calcule LRC
                Else
                        Putstr 1,"00FF"
                        LRC = -(3+5+K.Byte1+K.Byte0+0xFF) ' LRC
                End If
                Putstr 1,Hex2(LRC),13,10     ' Envoi

        End Sub
```

Mode MODBUS ASCII Esclave

```
' Source Esclave
        Const Device = cb280
        Opencom 1,115200,3,80,80
        set modbus 0,3
        Usepin 2, Out
        Usepin 3, Out
        Usepin 4, Out
        Set Ladder On
```



Lorsque l'esclave fini de recevoir les données envoyées par le maître, l'esclave continuera l'exécution du programme à l'étiquette GETMODBUS. On utilisera la commande SET UNTIL pour vérifier la fin du code LF (10).

Alors la commande **Getstr** sera utilisée pour sauvegarder l'arrivée de toutes les données dans RDATA. Les données présentes dans RDATA peuvent être analysées pour s'assurer que celles-ci sont correctes.

Lorsque l'esclave est déconnecté, le programme n'ira jamais exécuter la routine GETMODBUS.

Mode MODBUS RTU maître

L'exemple ci-dessous montre l'implémentation du mode RTU maître en BASIC pour écrire des valeurs flottantes dur 32 bits (2 registres 2 mots) vers un module esclave RTU 1 :

```
Const Device = CB280
#include "crctable.inc"
'_______________Open serial port for MODBUS_____________________________
'_________[Configure débit à 115200bps et 8-N-1]_____________________
'_________[et buffer réception avec 200 octes / buffer émission avec 100 octes]
Opencom 1,115200,3,200,100

'_________[Routine d'interruption en réception]_____________________________
On Recv1 Gosub GETMODBUS
'_______________[Efface tous les Buffers]_____________________________________
Bclr 1,2
'_______________[Utilise le Timer pour Timeout MODBUS]_____________________
On timer(1) Gosub MyClock

Debug "___________[Ecriture RTU MODBUS de la valeur point flottant]_____",Cr

'Test writing 32bit SINGLE to Register Address 0 of device 1
Debug "writing 3.14 and 6.99 Long value to register 0",Cr
writesingle 1,0,3.14
writesingle 1,0,6.99

' Exemple montrant comment envoyer de multiple variables à virgule flottante
' en faison une simple fonction WriteMultipleSingle()
SDataArray(0)=1.11
SDataArray(1)=2.22
SDataArray(2)=3.33
Debug "Writing multiple Single values to address 0",Cr
writemultiplesingle 1,0,3
'---------------------------------------------------------------
Do
Loop

'Routine réception Modbus
#include "ModbusRTUrecv.bas"
End

'Gestion Modbus
#include "ModbusRTULib016.bas"
```

* Consultez le Forum, www.cubloc.com sur internet pour d'avantages d'exemples de gestion de communications ASCII / RTU / MODBUS.

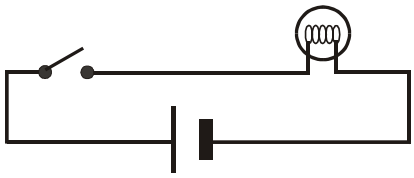
Chapitre 10.

Le LADDER des CUBLOC™…

Attention, si vous n'avez pas ajouté la commande **SET LADDER ON** en Basic, le programme écrit en LADDER ne s'exécutera pas !

Les bases du LADDER

Le schéma ci-dessous montre l'exemple d'une lampe et d'un interrupteur.

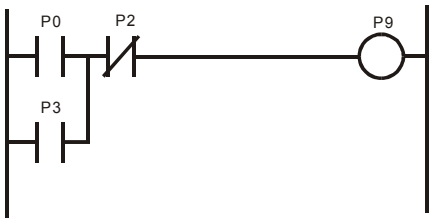


Si vous enlevez la source d'alimentation, le schéma se simplifie comme ci-dessous:

Si vous le traduisez en logique LADDER, vous obtenez le schéma ci-dessous :

Comme on peu le voir, le LADDER est une représentation simplifiée d'un circuit. L'interrupteur sera comparable au port P0 et le port P9 sera associé à la lampe.

En LADDER, il existe beacoup d'autres éléments tels que des timers, des compteurs, etc… Le schéma ci-dessous montre une connexion LADDER OR et AND:

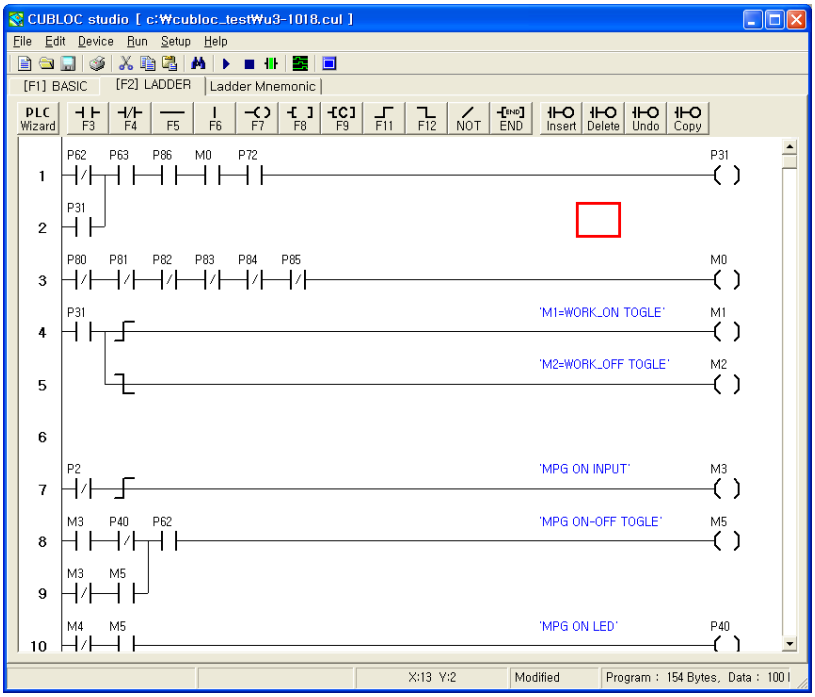


Dans ce diagramme P0 et P2 sont câblés en logique AND. P0 et P3 en logique OR (Ce qui veut dire que P0 ou P3 doivent être ON).

Le diagramme ci-dessus donne l'équivalence du circuit une fois transposé dans l'éditeur LADDER du CUBLOC STUDIO (on remarquera que la ligne de droite n'est pas visible). Dans le CUBLOC STUDIO, les éléments P0, P1, P2… sont appelés "Registres".

Développement en « LADDER »

La fenêtre ci-dessous montre un exemple de développement d'une application en langage « LADDER » via le CUBLOC STUDIO.

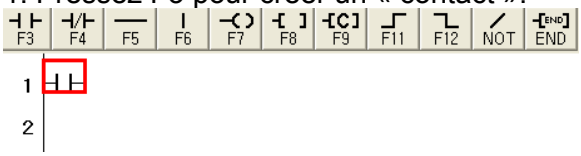


Le carré rouge représente le curseur de sélection du LADDER. Vous pouvez utiliser les touches de directions (haut/bas/droite/gauche) ou la souris pour déplacer ce curseur.

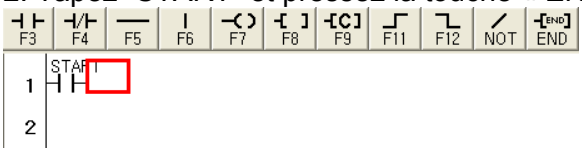
Une fois positionné à l'endroit voulu, vous pourrez utiliser les touches de fonctions F3 à F12 pour appliquer le symbole associé à l'écran.

Vous pourrez également saisir du texte pour les symboles qui le nécessitent.

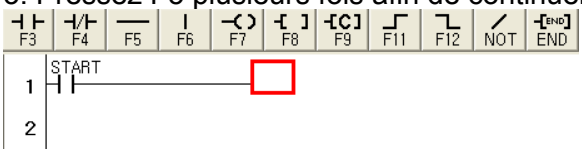
1. Pressez F3 pour créer un « contact ».



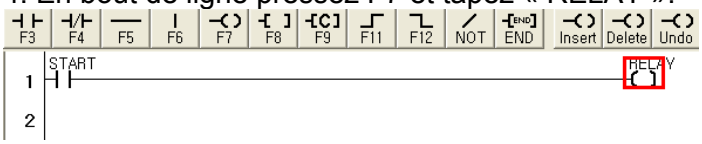
2. Tapez “START” et pressez la touche « ENTER ».



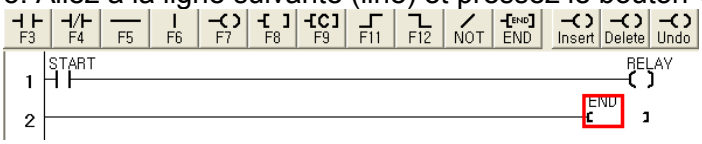
3. Pressez F5 plusieurs fois afin de continuer la ligne.



4. En bout de ligne pressez F7 et tapez « RELAY ».



5. Allez à la ligne suivante (line) et pressez le bouton « END ».



Pensez à presser la touche « ENTER » à la fin de chaque texte que vous aurez à saisir.

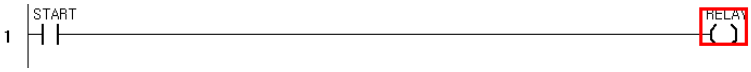
De même pour rappel, la fin de votre programme LADDER doit impérativement se finir par la commande « END ».

L'editeur du LADDER

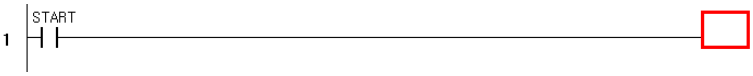
Modification de Texte

Pour modifier un TEXTE déjà existant, placez le curseur sur la position souhaitée et pressez la touche « ENTER ». Maintenant vous pouvez modifier librement le TEXTE comme vous le désirez.

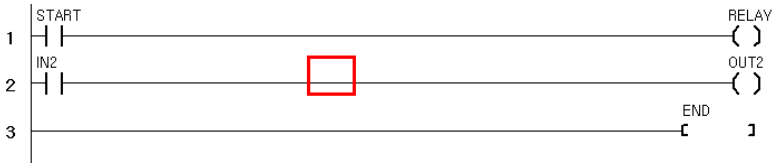
Effacer une cellule



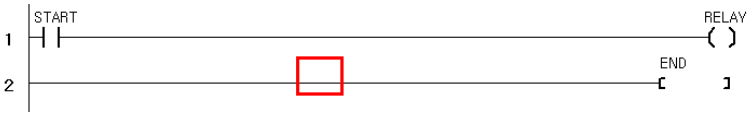
Sollicitez la touche ESPACE.



Effacer une ligne

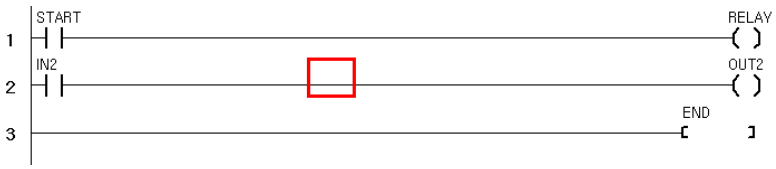


Vous pouvez presser la combinaison de touche CTRL-D pour effacer toute une ligne (cette ligne est stockée dans un buffer).

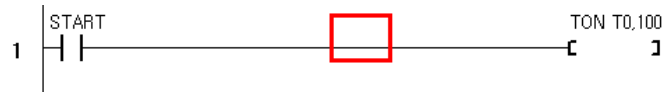


Récupérer une ligne

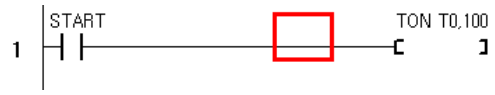
Pour récupérer une ligne effacée, tapez la combinaison CTRL-U.



Insérer et effacer une cellule



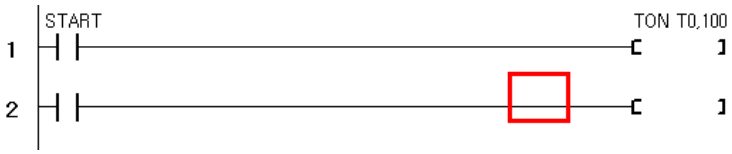
Si vous sollicitez la touche DEL, la cellule est effacée et le reste de la saisie de droite se décale vers la gauche.



Si vous pressez la touche INS, une cellule vide est crée et le reste de la saisie se décale vers la droite.

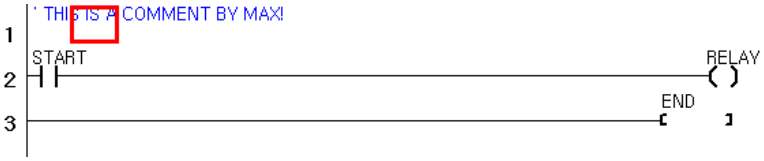
Copie de ligne

Si vous devez dupliquer une ligne, tapez la combinaison de touches CTRL-A.



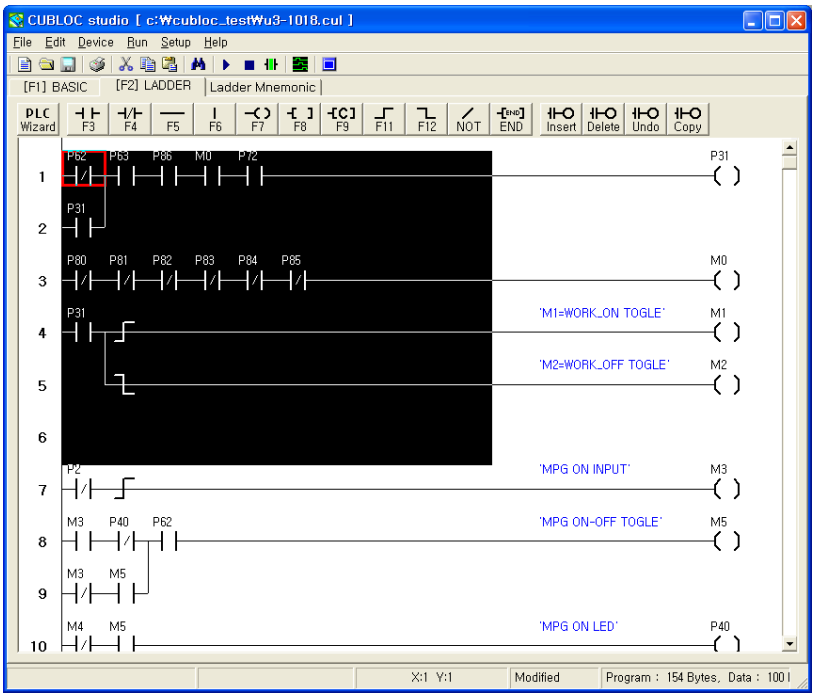
Commentaires

Vous pouvez saisir des commentaires en ajoutant une apostrophe (‘). Vous pouvez utiliser un point virgule (;) pour l’afficher à la ligne suivante.



Copier / Coller des blocs en LADDER

Vous pouvez faire une sélection d'un bloc afin de le copier et de le dupliquer à un endroit différent du LADDER.



Utilisez la souris pour effectuer la sélection de la partie à dupliquer (en restant appuyé sur le bouton gaucheet en déplaçant la souris). Sollicitez ensuite la combinaison de touches CTRL-C pour copier bloc puis duppliquez le à l'endroit voulu avec la combinaison de touches CTRL-V.

De la même façon que pour un éditeur de texte, vous pouvez presser la combinaison de touches CTRL-X pour couper (effacer) le bloc et le dupliquer avec CTRL-V.

* Attention la fonction UNDO n'est pas supportée en LADDER.

Monitoring en LADDER

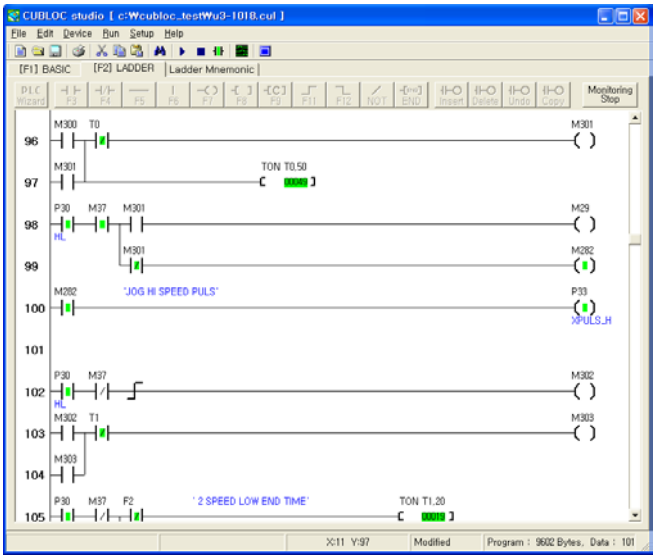
Le CUBLOC STUDIO supporte le monitoring temps réel du LADDER.

Pour ce faire, cliquez sur le bouton prévu à cet effet en haut de l'écran.

L'état des contacts actifs (ON) est affiché en **VERT**. Les valeurs des timers et compteurs seront affichées en valeurs décimales.

Vous pouvez modifier la vitesse du monitoring depuis le menu **Setup Menu -> Studio option -> Monitoring speed**. Si la vitesse du monitoring est trop rapide, il est possible que la communication du CUBLOC™ soit affectée (car le monitoring monopolise des ressources).

Nous recommandons une vitesse de monitoring de valeur 5.



*Assurez-vous impérativement que le monitoring est stoppé avant de modifier à nouveau votre programme ou de télécharger ce dernier dans le module CUBLOC™.

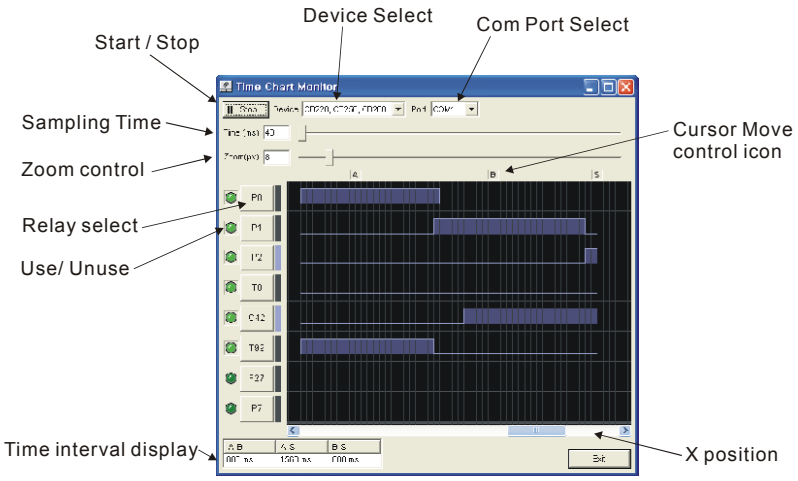
Monitoring avec affichage temporel

Pour ce faire, cliquez sur le bouton prévu à cet effet en haut de l'écran.

Avec le monitoring avec affichage temporel, vous pourrez suivre le déroulement des contacts du LADDER sur une échelle de temps graduée.

La durée minimale de la base de temps est de 40 ms. Vous pouvez utiliser la fonction de ZOOM afin de pouvoir effectuer des mesures de temps entres des impulsions après la phase d'acquisition.

Il est ainsi possible de faire du monitoring sur 8 « registres » à la fois.



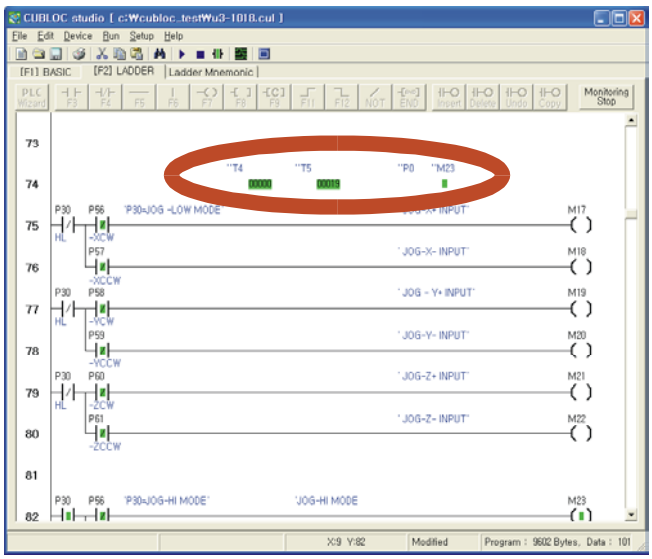
Pour utiliser le monitoring avec affichage temporel, vous devez impérativement désactiver le mode débug du BASIC. Pour faire ceci, ajoutez simplement la commande « Set Debug Off » à la toute première ligne de code BASIC.

```
Set debug off
```

Lorsque vous utilisez le monitoring avec affichage temporel, il n'est pas possible d'utiliser en même temps le monitoring en Ladder.

WATCH POINT

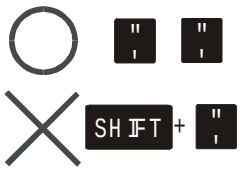
Vous pouvez utiliser 2 apostrophes (‘’) pour ajouter un WATCH POINT qui sera exploité lors de la phase de monitoring.



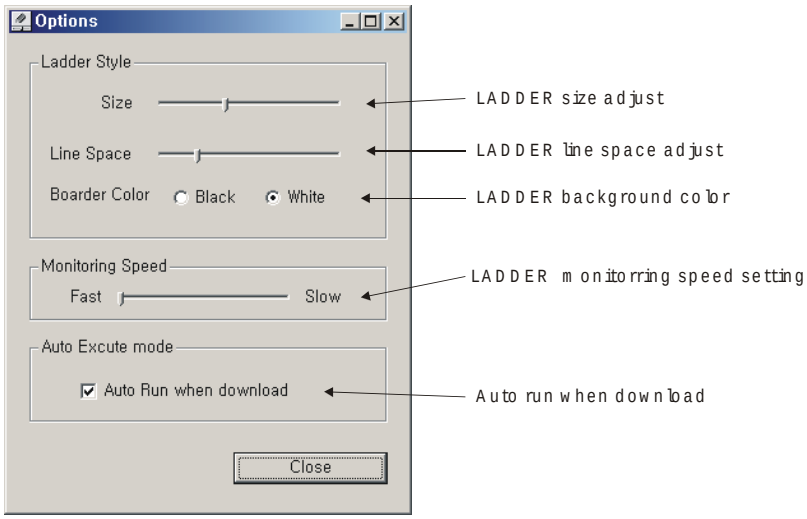
Ceci est utile par exemple si vous désirez surveiller l’état de P0 lors de la phase de monitoring ainsi que l’état d’autres « registres » qui se situent pourtant à l’opposé du programme et donc hors écran.

Exemples:
‘’P0 ‘’P1 ‘’D0

* Prenez garde de bien utiliser 2 APOSTROPHES(‘’) et non pas une QUOTATION (“).



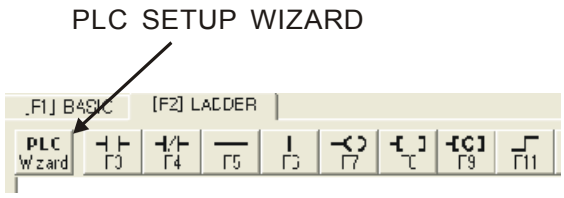
La fenêtre “Setup -> Cubloc Studio Options…”

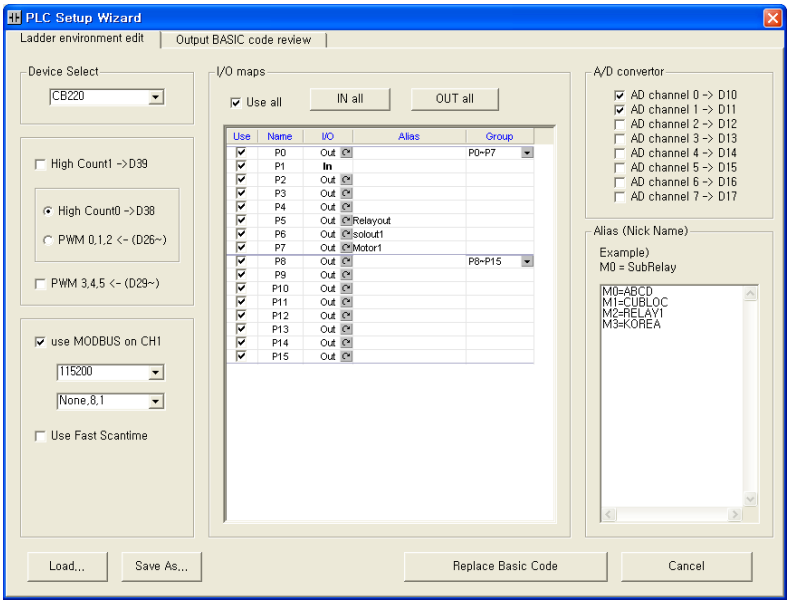


La sélection au bas de la fenêtre “Auto Run when download”, vous permet de configurer le module CUBLOC™ afin qu’il effectue automatiquement son programme après la phase de téléchargement. Vous pouvez si nécessaire désactiver cette fonction (il vous faudra alors effectuer un Reset du CUBLOC™ pour qu’il exécute son programme).

La fenêtre “Setup -> PLC Setup Wizard”

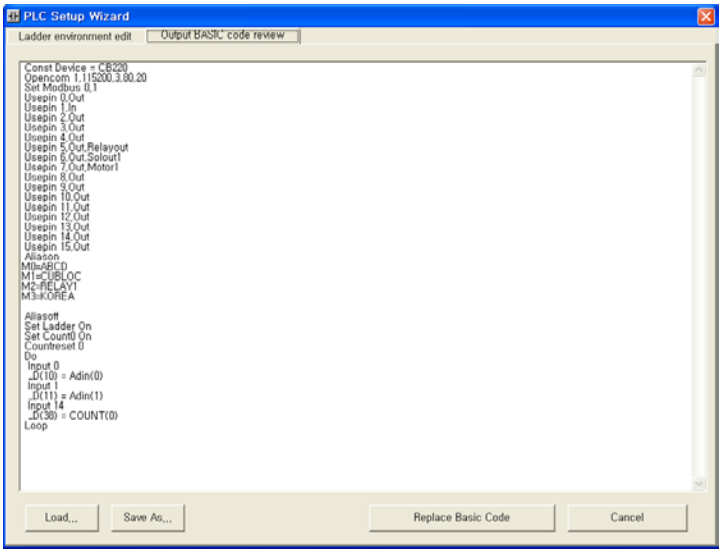
Afin de pouvoir utiliser un programme LADDER avec le module CUBLOC™, il vous faudra au préalable initialiser et déclarer certains paramètres dans l’entête du programme BASIC. Si cette opération n’est pas compliquée en soit, elle pourra toutefois monopoliser votre attention lors des premières utilisations. C’est la raison pour laquelle vous disposez d’une fenêtre spéciale qui vous permettra d’automatiser ces déclarations et initialisations.





Comme vous pouvez le constater, il vous sera possible au sein de cette fenêtre de configurer extrêmement simplement au moyen de simples clicks le nom du module, l’état des E/S, le nom que vous pourrez attribuer aux « Registres » du LADDER afin de faciliter la lisibilité de votre programme, la prise en compte des communication Modbus, etc…

Vous pourrez (en cliquant sur l'onglet « Output BASIC code review »), afficher une « visualisation préliminaire » des commandes BASIC qui seront à votre programme avec la configuration que vous aurez sélectionné.



Pour pouvoir utiliser les entrées de conversion « A/N », les sorties « PWM » ou les entrées de comptage « COUNT », vous pourrez simplement lire les résultats de ces derniers dans les « registres » D.

Pour ADC0, la valeur A/N sera stockée dans D(10). Il vous suffira alors simplement de lire le « Registre » D10 pour connaître la valeur de AD0.

Pour PWM3, il vous suffira d'écrire dans le « registres » D29 pour modifier la valeur PWM.

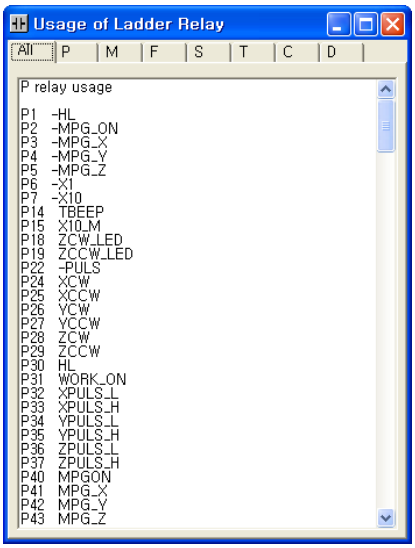
Pour le compteur rapide « HIGH COUNT1 », lisez simplement le « registre » D39.
Si vous le désirez, vous pourrez modifier le « regsitre » pour stocker ou écrire des valeurs en modifiant le code BASIC.

Cliquez sur le bouton [Replace Basic Code] lorsque vous désirez vraiment que votre programme BASIC soit initialisé avec la configuration que vous aurez sélectionné.

Vous pouvez également sauvegarder [SAVE AS..] ou récupérer [LOAD…] vos configurations préférées ou « typiques » sur le disque dur du PC.

Utilisation des “Registres” du LADDER

Ce menu accessible depuis : **Run -> View Register Usage** vous permet de visualiser la dénomination associée à chaque « registres » du LADDER afin que vous puissiez faciliter encore d’avantage la phase de développement de vos applications.



Liste des « Registres » utilisés par le LADDER

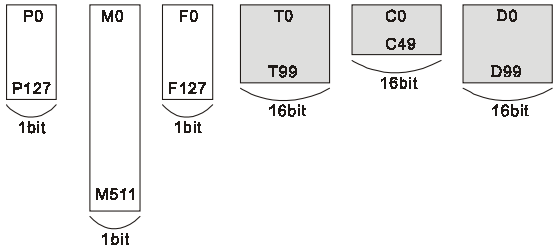
Le tableau ci-dessous donne la liste des « Registres » des modules CB220, CB280

Nom du “Registre”	Gamme	Unité	Possibilités
Registres P Entrée / Sortie	P0~P127	1 bit	Interface avec dispositifs externes
Registres internes M	M0~M511	1 bit	Registres internes
Registres spéciaux F	F0~F127	1 bit	Etat systèmes
Timer T	T0~T99	16 bits (1 Word)	Pour Timers
Copteur C	C0~C49	16 bits (1Word)	Pour Coumpteurs
Step Enable S	S0~S15	256 steps(1 octet)	For Step Enabling
Mémoire donnée D	D0~99	16 bits (1 Word)	Mémorisation de données

Les registres P, M et F sont sur 1 bit tandis que les registres T, C et D sont sur 16 bits (Word). Pour accéder aux registres P, M et F en mode 16 bits (word), vous pouvez utiliser WP, WM ou WF.

Nom “Registre”	Gamme	Units	Feature
WP	WP0~7	16 bits (1 Word)	Registre P (accès Word)
WM	WM0~WM31	16 bits (1 Word)	Registre M (accès Word)
WF	WF0~WF7	16 bits (1 Word)	Registre F (accès Word)

WP0 renferme P0 à P15. P0 est en partie LSB de WP0 et P15 est en partie MSB de WP0. Ces registres s’utilisent très facilement à l’aide de la commande WMOV.



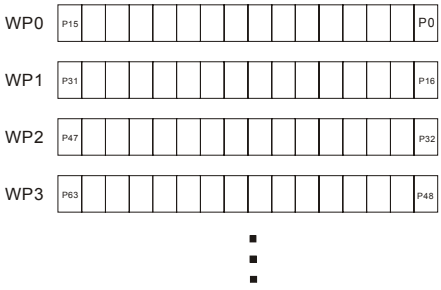
Le tableau ci-dessous dresse la liste des « Registres » du module CB290. Le module CUBLOC™ CB290 dispose de plus de « registres » de type M, C, T et D que les modules CB220 et CB280.

Nom du “Registre”	Gamme	Unité	Possibilités
Reègistres P Entrée / Sortie	P0~P127	1 bit	Interface avec dispositifs externes
Registres internes M	M0~M1023	1 bit	Registres internes
Registres spéciaux F	F0~F127	1 bit	Etat systèmes
Timer T	T0~T255	16 bits (1 Word)	Pour Timers
Copteur C	C0~C255	16 bits (1Word)	Pour Coumpteurs
Step Enable S	S0~S15	256 steps(1 octet)	For Step Enabling
Mémoire donnée D	D0~511	16 bits (1 Word)	Mémorisation de données

Les Registres P, M et F sont sur 1 bit tandis que les Registres T, C et D sont sur 16 bits (Word). Pour accéder aux Registres P, M et F en mode 16 bits (word), vous pouvez utiliser WP, WM ou WF.

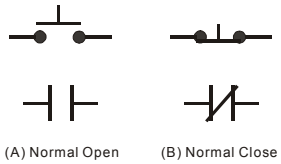
Nom “Registre”	Gamme	Units	Feature
WP	WP0~7	16 bits (1 Word)	Registre P (accès Word)
WM	WM0~WM63	16 bits (1 Word)	Registre M (accès Word)
WF	WF0~WF7	16 bits (1 Word)	Registre F (accès Word)

WP0 renferme P0 à P15. P0 est en partie LSB de WP0 et P15 est en partie MSB de WP0. Ces Registres s’utilisent très facilement à l’aide de la commande WMOV.



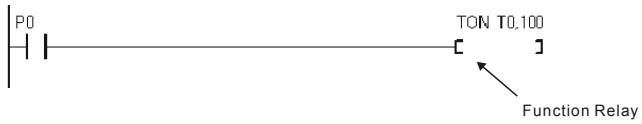
Les symboles du LADDER

Contact A, Contact B
Le contact A est “Normalement Ouvert” et se ferme lorsque le signal est reçu. A l’opposé, le contact B est “Normalement fermé” et s’ouvre lorsque le signal est reçu.

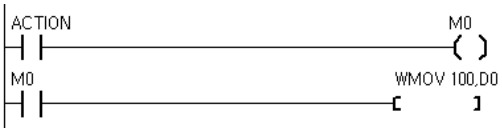


Symbole Registres Entrée, Sortie
Les registres d’Entrée/Sortie sont les symboles les plus simples en LADDER.

Registres Fonction
Parmi les Registres Fonction, on trouve : les timers, compteurs et autres Registres permettant de réaliser certaines opérations mathématiques.



Registres Internes
Les Registres internes (M) agissent simplement à l’intérieur du programme (à moins qu’ils ne soient connectés à un port externe). Vous pouvez utiliser les Registres M comme symbole d’entrée ou de sortie.



Les Registres P qui ne sont pas utilisés comme des ports d’E/S
Les CUBLOC™ disposent de Registres P répartis de P0 à P127. Ces Registres sont directement raccordés aux ports d’E/S. Toutefois, pour les CUBLOC qui disposent de moins de 128 E/S, il vous est possible d’utiliser les Registres P non connectés comme des Registres M.

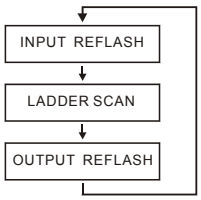
Utilisation des E/S

Les ports d'E/S des CUBLOC™ peuvent être utilisés à la fois par le programme BASIC et par le programme LADDER. Sans aucune déclaration particulière, tous les ports d'E/S sont attribués au programme BASIC. Pour utiliser des ports d'E/S en LADDER, vous devrez au préalable utiliser la commande "**Usepin**" afin de déclarer les ports qui seront utilisés en LADDER.

```
USEPIN 0,IN
USEPIN 1,OUT
```

Les commandes ci-dessus permettent e configurer les ports P0 en entrée et P1 en sortie afin qu'ils puissent être utilisés en LADDER.

Le mode de fonctionnement des CUBLOC™ implique que les commandes USEPIN sont flashées par le LADDER. Ce « Flashage » veut dire que le Ladder effectuera au préalable une lecture des E/S, puis les stockera leur état dans les Registres P. Après la réalisation du cycle du LADDER, ce dernier écriera à nouveau l'état des E/S dans les Registres P.



En BASIC, les commandes IN et OUT peuvent être utilisées pour contrôler les ports d'E/S. Cette méthode permet l'accès direct aux port d'E/S (que vous ayez besoin de les utiliser pour lire ou pour « écrire » sur ces derniers). Aussi, afin d'éviter les collisions entre les E/S lors de l'utilisation du BASIC et du LADDER, il conviendra de déclarer au préalable les ports à utiliser en LADDER.

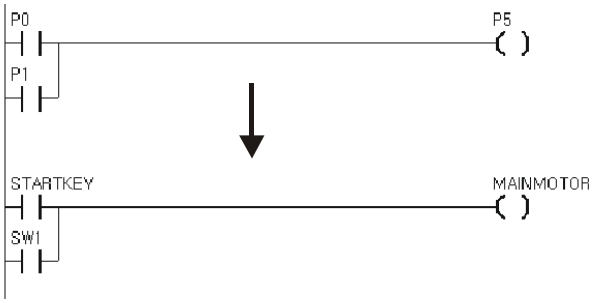
Une fois déclaré pour être utilisé par le LADDER à l'aide de la commande USEPIN, un port ne pourra alors plus être utilisé par le BASIC. Cette déclararion se fait en début de programme BASIC (il vous faudra donc écrire quelques lignes de commandes BASIC… même si vous ne programmez qu'en LADDER !).

```
USEPIN 0,IN, START
USEPIN 1,OUT, RELAY
```

Lors de la déclaration, il vous est également possible (et conseillé) d'utiliser des noms d'alias tels (START et RELAY dans l'exemple ci-dessus) afin de simplifier la lecture de votre programme et sa rédaction. Il est en effet plus facile ensuite dans votre programme d'utiliser START pour désigner une entrée de bouton-poussoir plutôt que P0 (voir explications ci-dessous).

Utilisation des « alias »

Lorsque vous concevez votre programme en LADER et que vous utilisez des Registres du type P0, P1 ou encore M0, il est plus simple de leur attribuer un nom (alias) qui vous permettra de faciliter la lecture et la rédaction de votre application.



Afin de pouvoir utiliser ces alias, vous devez les déclarer dans votre programme BASIC en « nommant » ainsi en « clair » les Registres en fonction de leur usage.

```
ALIAS M0 = MOTEUR1
ALIAS M2 = ETAT1
ALIAS M4 = MOTEURSTOP
```

Vous pouvez également utiliser la commande USEPIN (comme vu à la page précédente) pour déclarer vos alias.

Démarage du programme LADDER

A la mise sous tension, le programme BASIC des modules CUBLOC™ est toujours exécuté en premier. Vous pourrez alors dans ce programme BASIC ordonner l'exécution du programme LADDER à l'aide de la commande "**SET LADDER ON**". Lorsque cette commande est rencontrée, le LADDER sera exécuté en permanence avec un temps de cycle de 10 millisecondes.

Si vous ne placez pas la commande SET LADDER ON dans le programme BASIC, le programme LADDER ne sera pas exécuté !

```
SET LADDER ON
```

Déclaration du type de module CUBLOC™ utilisé

Il est TOUJOURS impératif de déclarer au début du programme BASIC (même si vous ne programmez qu'en LADDER) le type de module CUBLOC™ que vous utilisez à l'aide de la commande CONST DEVICE (voir exemples ci-dessous).

```
CONST DEVICE = CB220  ' Utilisation du module CB220.
```
ou
```
CONST DEVICE = CB280  'Utilisation du module CB280.
```

Cette commande est à placer à la première ligne du programme BASIC.

Si vous voulez utiliser le LADDER (sans le BASIC)

Dans l'éventualité ou votre application devra être développée uniquement langage LADDER, il vous faudra tout de même saisir quelques ligne de commandes en BASIC afin :

- De déclarer avec quel type de module CUBLOC™ vous allez travailler.
- De définir les ports d'E/S que vous utiliserez en LADDER.
- De déclarer éventuellement des noms d'alias pour les Registres.
- Et enfin pour demander le démarrage de l'exécution du programme LADDER.

Les lignes qui suivent donnent un exemple de la déclaration type :

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

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

Alias M0=RELAYSTATE             ' Déclaration des alias
Alias M1=MAINSTATE

Set Ladder On                   ' Activation du programme LADDER

Do
Loop                            ' Boucle sans fin du BASIC
```

Activation du mode « Turbo Scan Time » du LADDER

Lorsque vous exploitez à la fois une application en langage BASIC et en LADDER, vous obtenez un temps de cycle du LADDER de 10 ms. Il vous est possible d'activer un mode dit « Turbo Scan Time » (à condition de ne pas avoir à se servir du programme en BASIC). Pour ce faire, suivez la procédure ci-dessous.

Cette fonction est possible grâce à l'utilisation de la commande **LADDERSCAN** (laquelle devra être placée dans une boucle sans fin de type **DO…LOOP**.

La durée du temps de cycle pourra alors s'échelonner entre 500 uS à 1 ms pour des « petits » programmes LADDER comportant moins de 50 lignes (La durée est fonction de la taille du programme LADDER).

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

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

Alias M0=RELAYSTATE             ' Déclaration des alias
Alias M1=MAINSTATE

Do                              ‘ Boucle sans fin
      LadderScan
Loop
```

Le Registre F16 est un Registre spécial qui vous permettra de contrôler la durée courante du temps de cycle. Il vous suffira de le relier à un port ES et de visualiser le signal à l'aide d'un oscilloscope pour en connaître la valeur.

L'exemple ci-dessous permet d'exploiter le mode « Tubro Scan Time » en mode conditionné (uniquement lorsque le Registre M0 est ON).

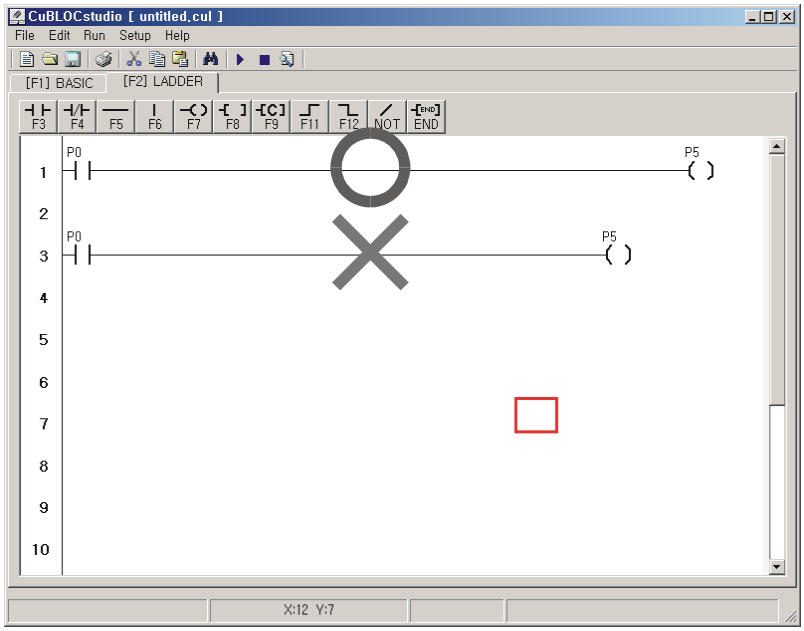
```
Do
    Set Ladder On           ’ Durée de cycle de 10 ms lorsque M0 est  OFF
      Do While _M(0) = 1
        LadderScan          ‘ Mode « Turbo Scan Time » uniquement quand M0 = ON
      Loop
Loop
```

Choses à se rappeler en LADDER

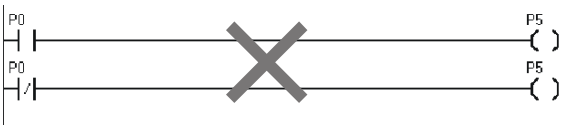
Le symbole « rond » indique ce qu'il faut faire – La croix… ce qu'il ne faut pas faire !

Les symboles d'entrées doivent être positionnés complètement à gauche en LADDER.

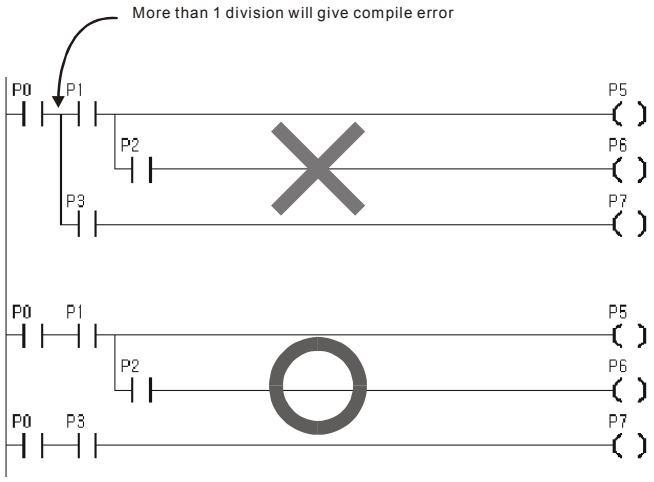
* Les symboles de doivent être positionnés complètement à droite en LADDER.



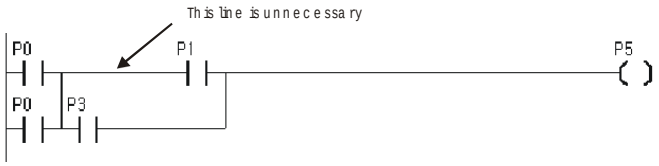
Il ne faut jamais utiliser les mêmes sorties en LADDER pour éviter les collisions.



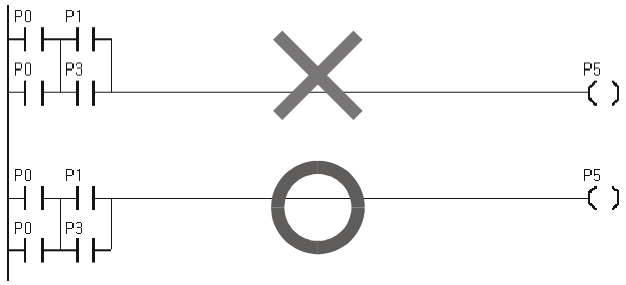
N’utilisez jamais plus d’une ligne verticale comme indiqué ci-dessous.



L’utilisation de bloc non nécessaire comme ci-dessous provoquera une erreur à la compilation.

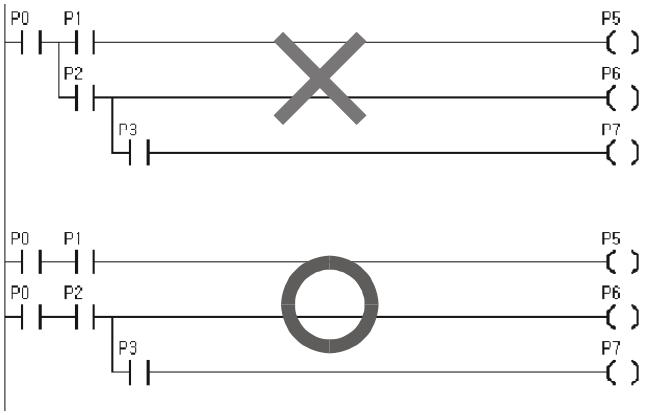


Le programme LADDER s’écrit du HAUT vers le BAS.



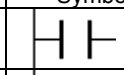
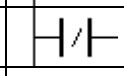
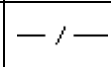
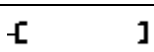
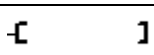
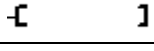
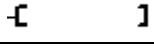
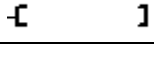
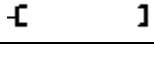
Les Registres Fonction ne doivent pas être positionnés à gauche en LADDER.

Lorsque qu'un programme LADDER est complexe, il est préférable de le fractionner afin de le rendre plus lisible (voir exemple ci-dessus).



Les instructions du LADDER

Instructions de « bas niveau »

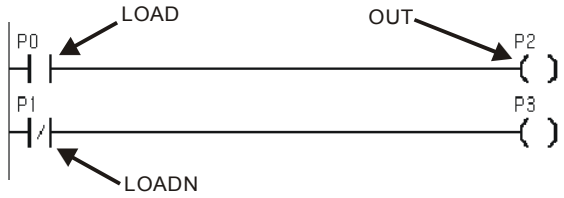
Commande	Symbole	Explication
LOAD		Contact A (Normalement ouvert)
LOADN		Contact B (Normalement fermé)
OUT		Sortie
NOT		NOT (Inverse le résultat)
STEPSET		Sortie Contrôleur Pas (Step Set)
STEPOUT		Sortie Contrôleur Pas (Step Out)
MCS		Contrôle Start Maitre
MCSCLR		Contrôle Stop Maitre
DIFU		Mis à ON pour 1 durée d'éxécution (scan time) lorsque signal HAUT reçu
DIFD		Mis à ON pour 1 durée d'éxécution (scan time) lorsque signal BAS reçu
SETOUT		Maintient la sortie à ON
RSTOUT		Maintient la sortie à OFF
END		Fin du programme LADDER
GOTO		Saut à une « étiquette »
LABEL		Déclaration « étiquette »
CALLS		Appel sous-routine
SBRT		Déclaration sous-routine
RET		Fin sous-routine
TND		Commande de « sortie » conditionnelle

Instructions de haut niveau

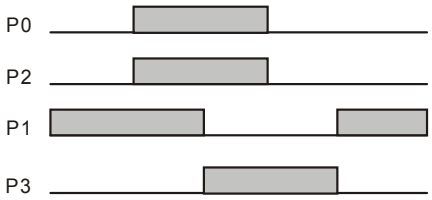
Commande	Paramètre	Explication
Commandes de transfert de données		
WMOV	s,d	Déplacement Donnée Word
DWMOV	s,d	Déplacement Donnée Double Word
WXCHG	s,d	Echange Données Word
DWXCHG	s,d	Echange Données Double Word
FMOV	s,d,n	Commande de remplissage de Données
GMOV	s,d,n	Commande déplacement Groupé
Commandes d'Incrémentation / Décrémentation		
WINC	d	Incrémente de 1 la donnée Word
DWINC	d	Incrémente de 1 la donnée Double Word
WDEC	d	Décrémente de 1 la donnée Word
DWDEC	d	Décrémente de 1 la donnée Double Word
Commandes Mathématiques		
WADD	s1,s2,d	Addition sur donnée Word
DWADD	s1,s2,d	Addition sur donnée Double Word
WSUB	s1,s2,d	Soustraction sur donnée Word
DWSUB	s1,s2,d	Soustraction sur donnée Double Word
WMUL	s1,s2,d	Multiplication sur donnée Word
DWMUL	s1,s2,d	Multiplication sur donnée Double Word
WDIV	s1,s2,d	Division sur donnée Word
DWDIV	s1,s2,d	Division sur donnée Double Word
Commandes d'opérations Logiques		
WAND	s1,s2,d	AND logique sur donnée Word
DWAND	s1,s2,d	AND logique sur donnée Double Word
WOR	s1,s2,d	OR logique sur donnée Word
DWOR	s1,s2,d	OR logique sur donnée Double Word
WXOR	s1,s2,d	XOR logique sur donnée Word
DWXOR	s1,s2,d	XOR logique sur donnée Word
Commandes de décalage de Bit		
WROL	d	Décalage à gauche 1 bit d'une donnée Word
DWROL	d	Décalage gauche 1 bit d'une donnée Double Word
WROR	d	Décalage droite 1 bit d'une donnée Word
DWROR	d	Décalage droite 1 bit d'une donnée Double Word

LOAD, LOADN, OUT

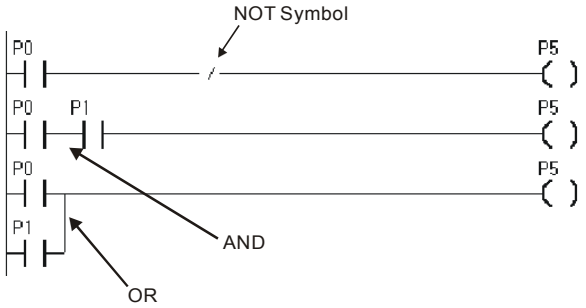
LOAD est pour les contacts Normalement Ouvert et **LOADN** pour les contacts Normalement fermés. **OUT** est permet de sortir la donnée.



Registres pouvant être utilisés	P	M	F	S	C	T	D	Constantes
LOAD LOADN	O	O	O	O	O	O		
OUT	O	O						



NOT, AND, OR

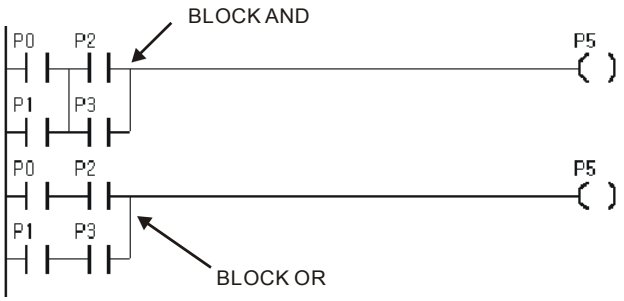


Le symbole **NOT** inverse le résultat. Si P0 est ON alors P5 passera en OFF.

Le symbole **AND** s'utilise lorsque 2 Registres sont placés horizontalement l'un de l'autre. Les 2 Registres P0 et P1 doivent être vrai (ON) pour que P5 soit vrai (ON).

Le symbole **OR** s'utilise lorsque 2 Registres sont placés verticalement sur une ligne séparée. Si P0 ou P1 est vrai (ON) alors P5 sera vrai (ON).

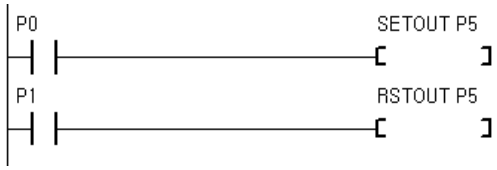
L'exemple ci-dessous montre l'utilisation de bloc AND et de bloc OR.



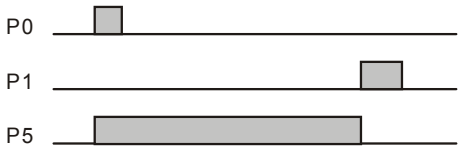
SETOUT, RSTOUT

SETOUT activera P5 en ON lorsque P0 sera ON et laissera P5 ON même si P0 retourne en OFF.

A l'inverse **RSTOUT** mettra P5 en OFF lorsque P1 sera ON et laissera P5 en OFF même si P1 retoorne en OFF.



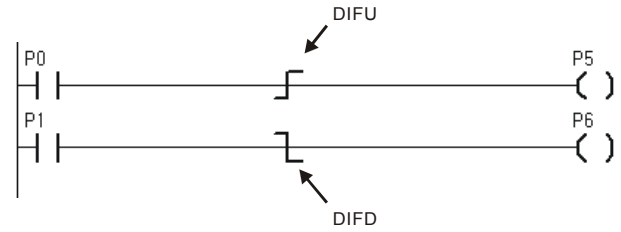
Registres pouvant être utilisés	P	M	F	S	C	T	D	Constantes
SETOUT	O	O	O					
RSTOUT	O	O	O					

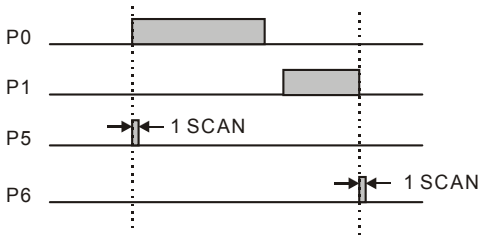


DIFU, DIFD

La commande **DIFU** active en ON la sortie 1 durant une durée de « time scan » si l'entrée passe du niveau OFF vers ON.

De la même façon, la commande **DIFD** passe la sortie 1 en ON une durée de « time scan » si l'entrée passe du niveau ON vers OFF.

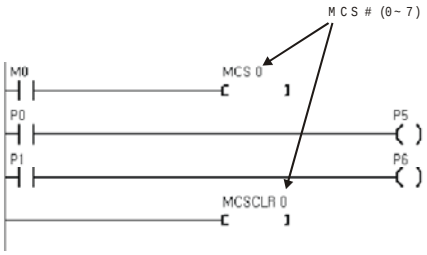




MCS, MCSCLR

Les commandes **MCS** et **MCSCLR** permettent l'exécution du programme LADDER placé entre MCS X et MCSCLR X lorsqu'elles sont activées (ON). Si MCS est OFF, le programme LADDER entre MCS X et MCSCLR X ne sera pas exécuté.

En utilisant cette commande, il est possible de contrôler l'exécution d'un bloc complet de LADDER.



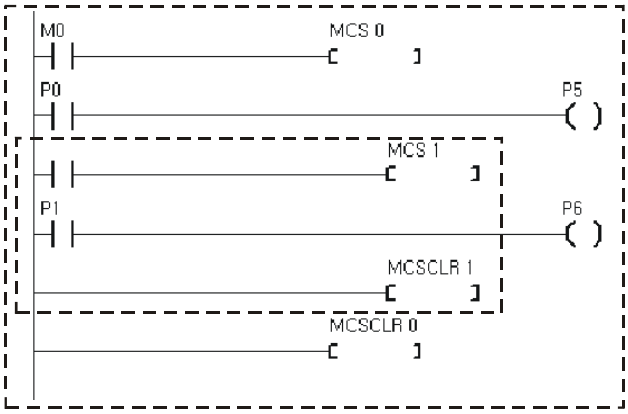
Dans l'exemple ci-dessus, lorsque M0 passe en ON, le programme LADDER entre MCS 0 et MCSCLR s'exécutera normalement. Si M0 est OFF, P5 et P6 seront désactivés.

Il est possible d'associer un chiffre de 0 à 7 avec la commande MCS. Le nombre associé à la commande MCS doit démarrer de 0, puis augmenter 1, 2, 3, etc… MCS 1 doit exister à l'intérieur de MCS 0 et MCS 2 doit exister à l'intérieur de MCS 0. De la sorte, il est possible d'utiliser 7 blocs MCS. Lorsque MCS 0 est OFF, tous les MCS à l'intérieur de MCS seront OFF.

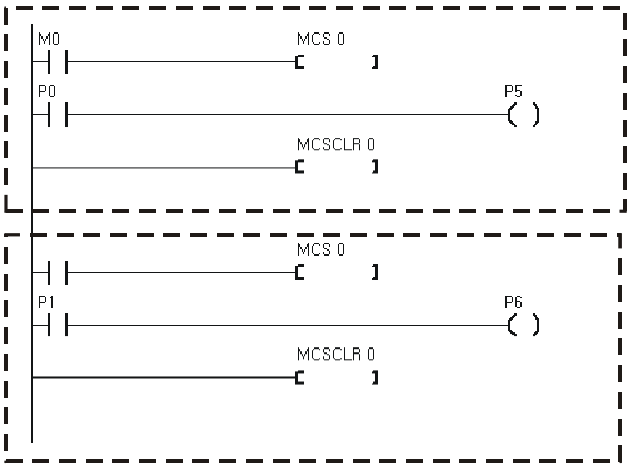
Lorsque MCS est désactivé OFF, toutes les sorties à l'intérieur des blocs MCS seront désactivés (OFF), les Timer seront effectueront un Reset et les compteurs seront stoppés.

Commande	Lorsque MCS est ON	Lorsque MCS est OFF
OUT	Opération Normale	OFF
SETOUT	Opération Normale	Maintien état après que MCS se soit placé en OFF
RSTOUT	Opération Normale	Maintien état après que MCS se soit placé en OFF
Timer	Opération Normale	Reset à la valeur par défaut
Compteur	Opération Normale	Maintien état après que MCS se soit placé en OFF
Autres Commandes	Opération Normale	Stop Opération

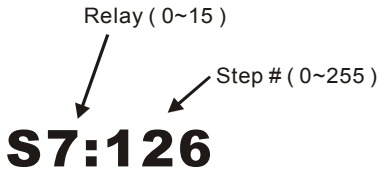
L'écran ci-dessous montre l'utilisation d'une commande MCS à l'intérieur d'une autre commande MCS.



*Vous pouvez simplement réutiliser MCS 0 si aucune autre commande MCS ne doit être placée à l'intérieur de la commande MCS.

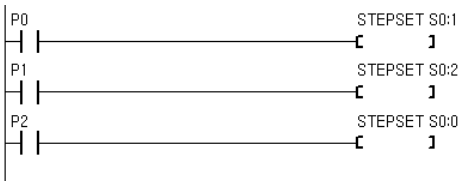


Contrôles de « pas » (Step Control)
Les Registres S sont utilisés pour le contrôle de « pas ». Vous trouverez ci-dessous le format correct à utiliser.



Pour les commande de type (Step Control), il y a des “pas normaux” et “pas inversés ”. Pour les « pas normaux », vous pouvez utiliser la commande **STEPSET**.

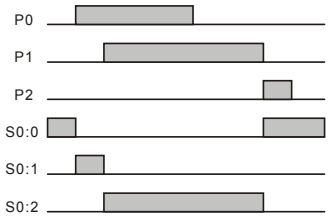
STEPSET



La commande **STEPSET** activera en ON le pas courant si le pas précédent était en ON. Comme on opère d’un pas à la fois, nous l’appelerons STEPSET.

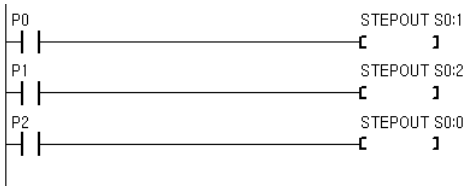
Dans l’exemple ci-dessous :
- Lorsque P1 est à ON, S0:2 passera en ON si S0:1 est déjà passé à ON. S0:1 passera à OFF.

- Lorsque P2 passe à ON, S0:0 passe à ON et les autres pas passe à OFF (S0:0 ou step 0 est utilisé pour le reset – ceci permet d’avoir une utilisation ordonnée).



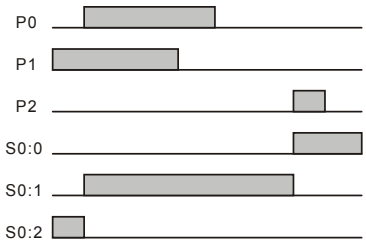
STEPOUT

La commande **STEPOUT** permettra l'activation d'un seul pas pour toutes les fois. Le dernier pas activé en ON pourra être remplacé par un autre pas à tout moment.



- Lorsque P1 passe en ON, S0:2 passe en ON.
- Lorsque P0 passe en ON, S0:1 passe en ON.

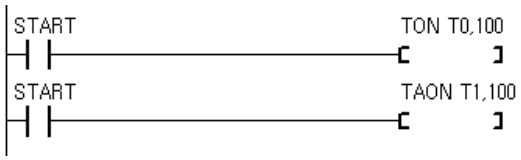
Un pas s'exécute jusqu'à ce qu'un autre prenne sa place et passe à ON.



TON, TAON

Lorsqu'une entrée passe à ON, la valeur d'un timer s'active et la sortie passe à ON lorsque le timer arrive à terme. Il existe 2 types de timers, le premier fonctionne avec des unités en multiples de 0.01 secondes et le secondes avec des unités multiples de 0.1 secondes.

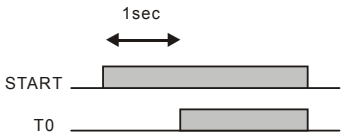
Type de Timer	Unité de temps	Durée maximale
TON	0.01 sec	655.35 sec
TAON	0.1 sec	6553.5 sec



Les commandes **TON** et **TAON** nécessitent 2 paramètres. Pour le premier paramètre, vous pouvez choisir entre T0 to T99 et pour le second paramètre, vous pouvez utiliser un nombre ou une mémoire données telle que D0.

Registres utilisables	P	M	F	S	C	T	D	Constantes
TON, TAON					O	O	O	O

Dans l'exemple ci-dessus, lorsque l'entrée START passe à ON, le Timer T0 démarre de zéro jusqu'à 100. Lorsque la valeur 100 est atteinte, T0 passe à ON. Ici, 100 est équivalent à 1 seconde pour TON et 10 secondes pour TAON.



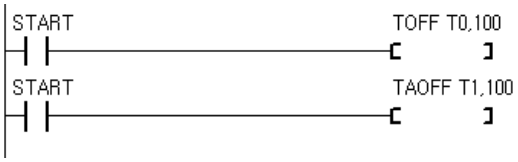
Lorsque l'entrée START repasse en OFF, le timer est réinitialisé à sa valeur initiale et T0 repasse aussi en OFF. Les commandes TON et TAON réinitialisent la valeur de leur timer lors d'une coupure d'alimentation. Pour utiliser des possibilités de sauvegarde par pile, vous devrez utiliser les commandes KTON et KTAON qui (au moyen d'une pile externe – sur le CB290 – pourront maintenir la valeur de leur timer suite à une coupure d'alimentation).

L'exemple ci-dessous montre comment réinitialiser TAON.

TOFF, TAOFF

Lorsqu'une entrée passe à ON, la sortie passe aussi à ON. Lorsque l'entrée repasse en OFF, la sortie reste à ON pendant qu'un timer s'active. Au terme de ce timer, la sortie repasse à OFF. Il existe 2 types de timers, le premier fonctionne avec des unités en multiples de 0.01 secondes et le secondes avec des unités multiples de 0.1 secondes.

Type de Timer	Unité de temps	Durée maximale
TOFF	0.01 sec	655.35 sec
TAOFF	0.1 sec	6553.5 sec

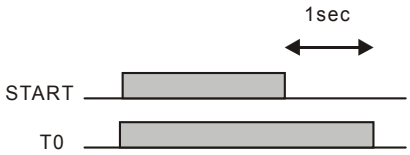


Les commandes **TOFF** et **TAOFF** nécessitent 2 paramètres. Pour le premier paramètre, vous pouvez choisir entre T0 to T99 et pour le second paramètre, vous pouvez utiliser un nombre ou une mémoire données telle que D0.

Registres utilisables	P	M	F	S	C	T	D	Constantes
TOFF, TAOFF					O	O	O	O

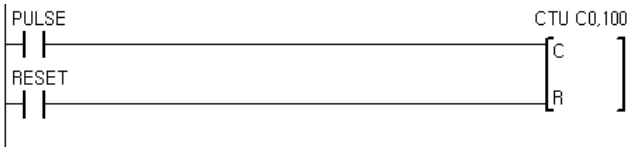
Dans l'exemple ci-dessus, lorsque l'entrée START passe à ON, la sortie T0 passe également à ON et le Timer s'initialise à la valeur 100. Lorsque l'entrée START repasse à OFF, le Timer commence à décompter. Une fois arrivé à zéro, la sortie T0 repasse à OFF.

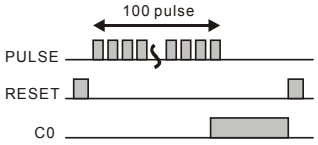
Ici, 100 est équivalent à 1 seconde pour TOFF et 10 secondes pour TAOFF.



CTU

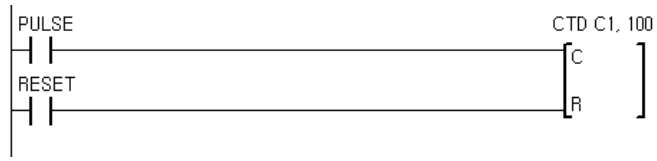
Cette commande est un compteur ascendant. Lorsque sa première entrée est sollicitée, le compteur est incrémenté d'une unité. Lorsque le compteur arrive à une certaine valeur, le Registre associé passera à ON. Le compteur dispose d'une seconde entrée permettant de bénéficier d'une fonction Reset si nécessaire.

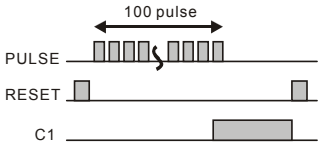




CTD

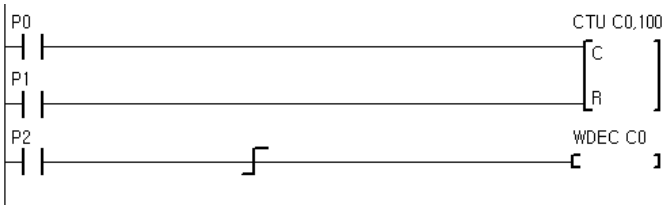
Cette commande est un compteur descendant. Lorsque sa première entrée est sollicitée, le compteur décrémenté d'une unité. Lorsque le compteur arrive à zéro, le Registre associé passera à ON. Le compteur dispose d'une seconde entrée permettant de bénéficier d'une fonction Reset si nécessaire.



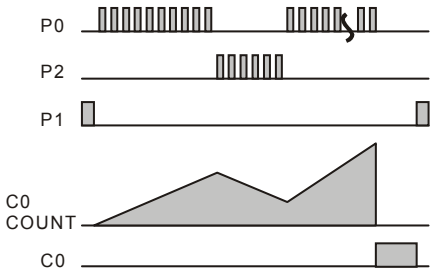


Compteur « UP/DOWN »

L'exmple ci-dessous montre une façon simple de réaliser un compteur « ascendant / descendant ».

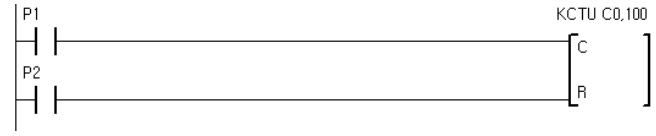


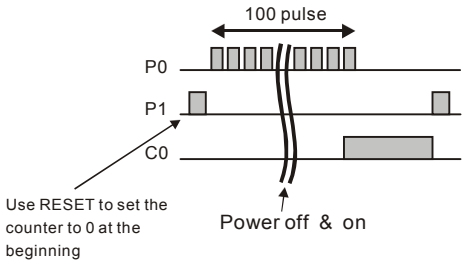
P0 sert au comptage ascendant, P2 au comptage descendant et P1 pour le reset. Lorsque le compteur arrive à 100, C0 est activé.



KCTU

Cette commande est identique à la commande CTU mise à part que la valeur du compteur pourra être mémorisée en cas de coupure d'alimentation (à condition de pouvoir disposer d'une option de sauvegarde par pile comme sur le module CB290. A l'inverse, la commande CTU perdra la valeur du compteur en cas de coupure d'alimentation.





Lorsque vous utilisez cette commande pour la toute première fois, il vous faudra impérativement effectuer un RESET du signal pour initialiser la valeur du compteur et ne pas travailler avec une valeur aléatoire.

KCTD

Cette commande est identique à la commande CTD mise à part que la valeur du compteur pourra être mémorisée en cas de coupure d'alimentation (à condition de pouvoir disposer d'une option de sauvegarde par pile comme sur le module CB290. A l'inverse, la commande CTD perdra la valeur du compteur en cas de coupure d'alimentation.

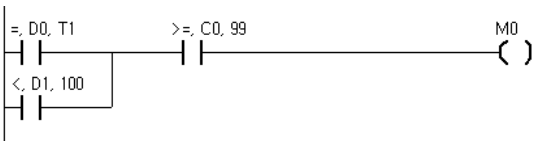
Les commandes KCTU et KCTD doivent être utilisée uniquement avec les modules capables de supporter les sauvegardes par piles (comme sur le CB290 par exemple).

Comparaisons Logiques

Ces commandes permettent de comparer 2 données de type Words(16 bits) ou 2 données de type Double Words (32 bits) afin de pouvoir activer une sortie lorsque la condition de test est remplie.

Commande de comparaison	Types données	Explications
=, s1, s2	Word (16 bits)	Lorsque s1 et s2 sont identique passe la sortie à ON.
<>, s1, s2	Word (16 bits)	Lorsque s1 et s2 sont différents passe la sortie à ON.
>, s1, s2	Word (16 bits)	Lorsque s1 > s2 passe la sortie à ON.
<, s1, s2	Word (16 bits)	Lorsque s1 < s2 passe la sortie à ON.
>=, s1, s2	Word (16 bits)	Lorsque s1 >= s2 passe la sortie à ON.
<=, s1, s2	Word (16 bits)	Lorsque s1 <= s2 passe la sortie à ON.
D=, s1, s2	Dword (32 bits)	Lorsque s1 et s2 sont identique passe la sortie ON.
D<>, s1, s2	Dword (32 bits)	Lorsque s1 et s2 sont différents passe la sortie à ON.
D>, s1, s2	Dword (32 bits)	Lorsque s1 > s2 passe la sortie à ON.
D<, s1, s2	Dword (32 bits)	Lorsque s1 < s2 passe la sortie à ON.
D>=, s1, s2	Dword (32 bits)	Lorsque s1 >= s2 passe la sortie à ON.
D<=, s1, s2	Dword (32 bits)	Lorsque s1 <= s2 passe la sortie à ON.

Vous pouvez mixer différentes comparaisons comme suit :

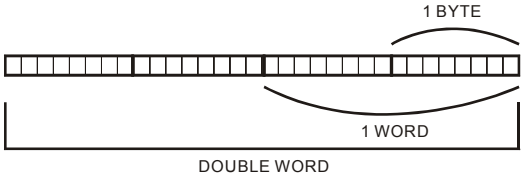


Lorsque l'un ou l'autre D0=T1 ou D1<100 et si C0>=99, alors M0 passera en ON.

En d'autres termes, D0 devra être équivalent à la valeur de T1 ou D1 est inférieur à 100 pendant que C0 devra être plus grand ou égal à 99.

Comment mémoriser des données de type Words et Double Words

Les Byte sont sur 8 bits, les Word sur 16 bits et les Double Word sur 32 bits.



Il existe 2 méthodes pour mémoriser les données de type Word issues d'un Double Word. Un Word ou Double Word peut être mémorisé à partir de l'octet de poid faible (LOW BYTE) ou de l'octet de poids fort (HIGH BYTE). Pour le CUBLOC™, il sera mémoriser à partir de l'octet de poid faible (LOW BYTE) ou LSB (Least Significant Byte).

Comme vous pouvez le voir, 1234H est mémorisé à l'adresse mémoire 0 et 12345678H est stocké à l'adresse mémoire 5. Dans chaque adresse mémoire, 1 octet de donnée est stocké.

0	34
1	12
2	
3	
4	
5	78
6	56
7	34
8	12
9	

Les Registres C, T et D sont des unités de Words. Pour mémoriser une donnée de type Double Word, un emplacement de 2 Word ser requis (soit l'emplacement de 2 Registres).

Vous voyez ci-dessous la mémorisation de la donnée Double Word, 12345678H. D1 renferme l'adresse 1234H et D0 referme 5678H.

D0	5678
D1	1234
D2	
D3	
D4	

Binaire, Décimal, Hexadécimal

Dans le cadre de votre programmation, vous aurez souvent à prendre en considération des nombres exprimés en décimal, en binaire ainsi qu'en hexadécimal. Le tableau ci-dessous vous donne la correspondance entre ces 3 types de représentation.

Décimal	Binaire	Héxadécimale
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Lors de la saisie de votre programme en LADDER, vous pourrez exprimer les nombre binaires et hexadécimal de la façon suivante:

Binaire: 00101010B
Hexadécimal: 0ABCDH

La lettre B est ajoutée à la fin de l'expression pour désigner un binaire et on ajoute H pour désigner un nombre hexadécimal. Pour clarifier l'identification pour un nombre de type valeur, il suffit d'ajouter 0 au début de l'expression numérique hexadécimale (Exemple : 0ABH, 0A1H, 0BCDH)

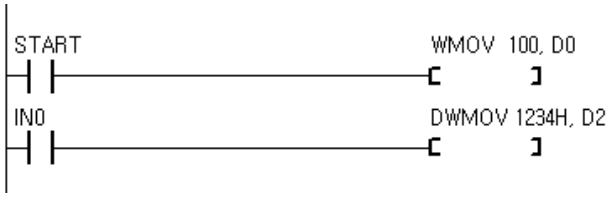
* En BASIC, c'est très légèrement différent du LADDER en ce sens qu'il vous est possible d'exprimer les nombre en binaire et en hexadécimal. Vous pouvez utiliser &B100010 ou &HAB pour exprimer ces 2 nombres.

WMOV, DWMOV

WMOV s, d
DWMOV s, d

La commande **WMOV** déplace une donnée de type 16 bits de « **s** » vers « **d** ». La commande DWMOV pourra être utilisée pour les données de type 32 bits.

Registre utilisables	P	M	F	S	C	T	D	Constantes
s (Source)					O	O	O	O
d (Destination)					O	O	O	



Lorsque l'entrée START passe à ON, D0 prendra la valeur 100. Lorsque IN0 passera à ON, D2 prendra la valeur 1234H.

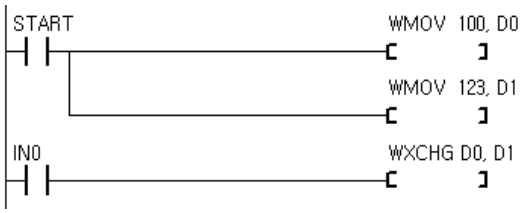
D0	100
D1	
D2	1234H
D3	0
D4	

WXCHG, DWXCHG

WXCHG s, d
DWXCHG s, d

La commande **WXCHG** échange les données entre « **s** » et « **d** ». WXCHG est pour l'échange de données de type Word et la commande DWXCHG est pour l'échange entre Double Word.

Registre utilisables	P	M	F	S	C	T	D	Constantes
s					O	O	O	
d					O	O	O	



Lorsque l'entrée START passe à ON, D0 prend la valeur 100 et D1 la valeur 123. Lorsque IN0 passe à ON, D0 et D1 échangent leur valeur (voir ci-dessous):

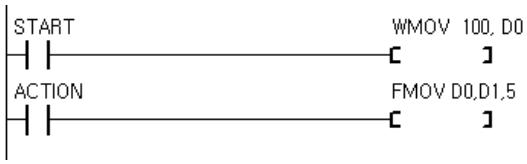
D0	123
D1	100
D2	
D3	
D4	

FMOV

FMOV s, d, n

Mémorise la valeur en « **s** » dans « **d** » et répète l'opération « **n** » fois aux adresses mémoires suivantes. Cette commande est généralement utilisée pour initialiser ou effacer la mémoire.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s					O	O	O	
d					O	O	O	
n								O



Le résultat du programme LADDER est visible ci-dessous:

D0	100
D1	100
D2	100
D3	100
D4	100
D5	100

*Nota: Configurez n avec un nombre inférieur à 255.

GMOV

GMOV s, d, n

Recopie les « **n** » valeurs démarrant en « **s** » vers l'emplacement mémoire en « **d** ». Vérifiez de ne pas dépasser les emplacements mémoires afin d'éviter les « collisions » entre les données.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s					O	O	O	
d					O	O	O	
n								O

Le résultat du programme LADDER est visible ci-dessous:

D0	12
D1	34
D2	56
D3	78
D4	90
D5	
D6	
D7	
D8	
D9	
D10	12
D11	34
D12	56
D13	78
D14	90
D15	
D16	

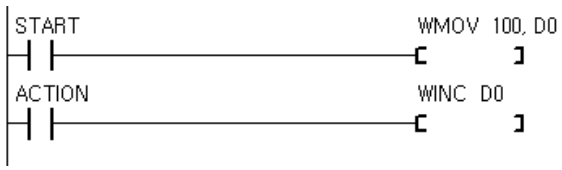
*Nota: n doit avoir une valeur inférieure à 255.

WINC, DWINC, WDEC, DWDEC

WINC d
DWINC d
WDEC d
DWDEC d

WINC incrémente une valeur de type Word en « **d** » d'une unité.
DWINC incrémente une valeur de type Double Word en « **d** » d'une unité.
WDEC décrémente une valeur de type Word en « **d** » d'une unité.
DWDEC décrémente une valeur de type Double Word en « **d** » d'une unité.

Registre utilisable	P	M	F	S	C	T	D	Constantes
d					O	O	O	



Le résultat du programme LADDER est visible ci-dessous:

D0	99
D1	
D2	
D3	

WADD, DWADD

WADD s1, s2, d
DWADD s1, s2, d

Ajoute « **s1** » et « **s2** » et mémorise le résultat en « **d** ».

WADD permet de travailler sur les valeurs de type Word et DWADD sur les Double Word.

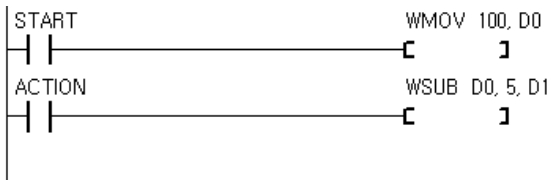
Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
d					O	O	O	

WSUB, DWSUB

WSUB s1, s2, d
DWSUB s1, s2, d

Soustrait « **s1** » de « **s2** » et mémorise le résultat en « **d** ».
WSUB permet de travailler sur les valeurs de type Word et DWSUB sur les Double Word.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
d					O	O	O	



D1 prend la valeur 95 dans ce programme LADDER.

WMUL, DWMUL

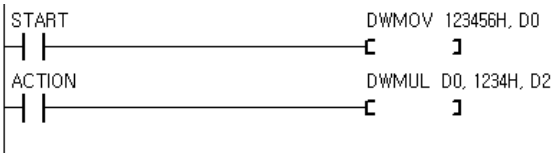
WMUL s1, s2, d
DWMUL s1, s2, d

Multiplie « **s1** » par « **s2** » et mémorise le résultat en « **d** ».
WMUL permet de travailler sur les valeurs de type Word et DWMUL sur les Double Word.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
d					O	O	O	

Le résultat de 1234H * 1234H est mémorisé en D1 en tant que double word de valeur 14B5A90H.

D0	1234H
D1	5A90H
D2	14BH



Le résultat de 123456H * 1234H est mémorisé en D2 avec la valeur 4B60AD78H

D0	3456H
D1	0012H
D2	0AD78H
D3	4B60H
D4	0
D5	0

WDIV, DWDIV

WDIV s1, s2, d
DWDIV s1, s2, d

Divise « **s1** » par « **s2** » et mémorise le résultat en « **d** » (le reste est dans d+1/).
WDIV permet de travailler sur les valeurs de type Word et DWDIV sur les Double Word.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
d					O	O	O	

D0	1234H
D1	
D2	3
D3	
D4	611H
D5	1

D0	5678H
D1	1234H
D2	7
D3	0
D4	0C335H
D5	299H
D6	5
D7	0

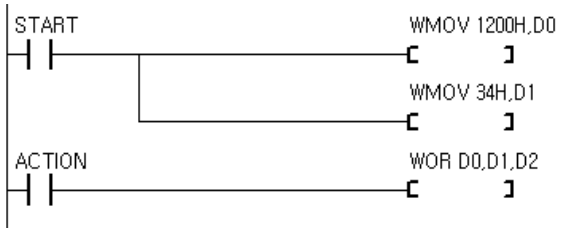
WOR, DWOR

WOR s1, s2, d
DWOR s1, s2, d

Effectue un « **OR** » logique entre « **s1** » et « **S2** » et mémorise le résultat en « **d** ».

WOR permet de travailler sur les valeurs de type et DWOR sur les Double Word.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
d					O	O	O	



Le résultat du programme LADDER est visible ci-dessous:

D0	1200H
D1	34H
D2	1234H

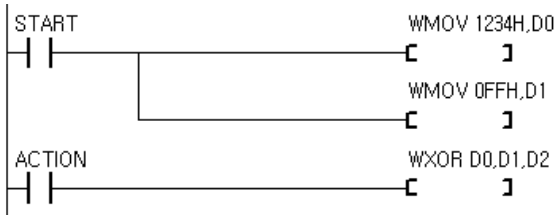
WXOR, DWXOR

WXOR s1, s2, d
DWXOR s1, s2, d

Effectue un « **XOR** » logique entre « **s1** » et « **S2** » et mémorise le résultat en « **d** ».

WXOR permet de travailler sur les valeurs de type et DWXOR sur les Double Word.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
d					O	O	O	



Le résultat du programme LADDER est visible ci-dessous:

D0	1234H
D1	0FFH
D2	12CBH

Lorsque vous désirez inverser un bit spécifique, vous pouvez utiliser une opération logique de type XOR.

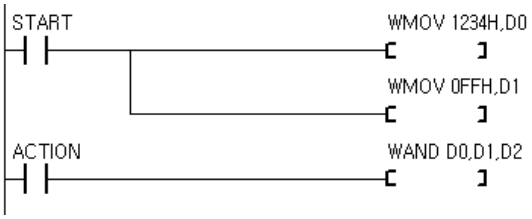
WAND, DWAND

WAND s1, s2, d
DWAND s1, s2, d

Effectue un « **AND** » logique entre « **s1** » et « **S2** » et mémorise le résultat en « **d** ».

WAND permet de travailler sur les valeurs de type et DWAND sur les Double Word.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s1					O	O	O	O
s2					O	O	O	O
D					O	O	O	



Le résultat du programme LADDER est visible ci-dessous:

D0	1234H
D1	0FFH
D2	34H

Vous pouvez utiliser une opération logique de type AND lorsque vous désirez utiliser seulement un bit spécifique.

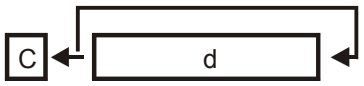
WROL, DWROL

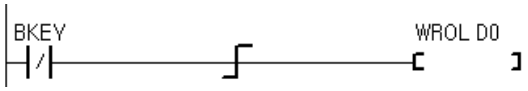
WROL d
DWROL d

Effectue une rotation du Registre « **d** » vers la gauche. Le bit sortant est mémorisé dans le « flag Carry »

WROL permet de travailler sur les valeurs de type et DWROL sur les Double Word.

Registre utilisable	P	M	F	S	C	T	D	Constantes
d					O	O	O	





Si D0 à la valeur 8421H, on obtient le résultat suivant:

D0	0843H
D1	

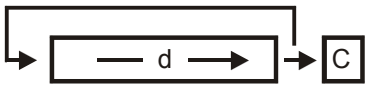
WROR, DWROR

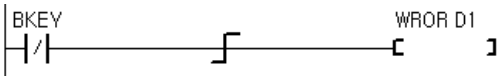
WROR d
DWROR d

Effectue une rotation du Registre « **d** » vers la droite. Le bit sortant est mémorisé dans le « flag Carry »

WROR permet de travailler sur les valeurs de type et DWROR sur les Double Word.

Registre utilisable	P	M	F	S	C	T	D	Constantes
d					O	O	O	





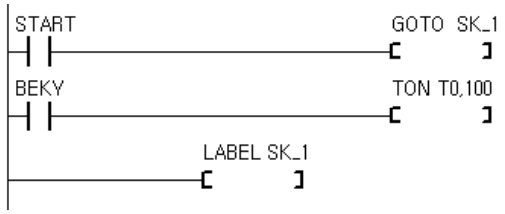
Si D1 à la valeur 8421H, on obtient le résultat suivant:

D0	
D1	0C210H

GOTO, LABEL

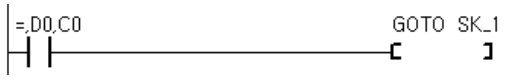
GOTO label
LABEL label

La commande GOTO permettra au LADDER de se rendre à une « étiquette » spécifique.
« **Label** » indique le nom de cette « étiquette ».



Lorsque l'entrée START passe à ON, le programme LADDER continuera son exécution à l' « étiquette » SK_1

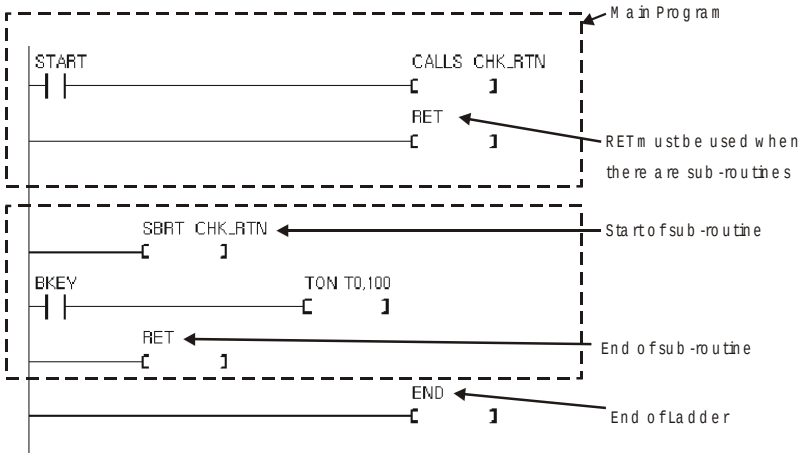
Dans ce second exemple, lorsque D0 est égal à C0, le programme continuera son exécution à l' « étiquette » SK_1.



CALLS, SBRT, RET

CALLS label
SBRT label

CALLS permet d'appeler une sous-routine.
SBRT est le point de départ de la sous-routine.
RET est le point d'arrêt de la sous routine.



Prenez garde lorsque vous ajoutez des sous-routines à votre programme à bien utiliser la commande RET à la fin du programme principal afin de le différencier des sous-routines

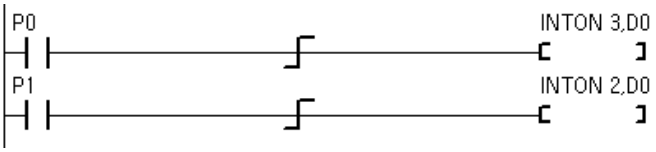
La commande END doit quand à elle être utilisée à la fin de votre programme principal et de vos sous-routines (si vous en avez).

INTON

INTON s,d

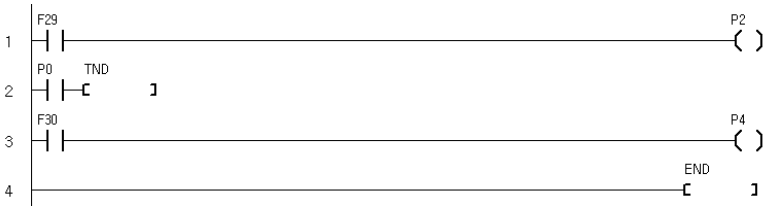
La commande **INTON** est identique à la commande WMOV à l'exeption que cette dernière peut être à l'origine d'une interruption en BASIC.

Registres utilisables	P	M	F	S	C	T	D	Constantes
s (Source)					O	O	O	O
d (Destination)					O	O	O	

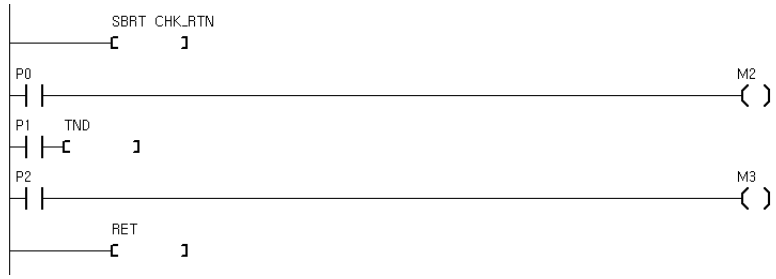


TND

TND est une commande se « sortie » conditionnelle. Lorsque l'utilisateur désire stopper un « scanning » du Ladder, il pourra avoir ecours à cette commande.



Lorsque P0 est ON, le Ladder sera stoppé.



Vous pouvez utiliser cette commande pour sortir de sous-routine lorsqu'une condition particulière est rencontrée. Dans l'exemple ci-dessus ce sera lorsque P1 sera ON.

« Registres » spéciaux

Vous pouvez utiliser ces Registres pour connaître l'état du CUBLOC™ ou pour réaliser des applications nécessitant un timing particulier.

Registre spécial	Explication
F0	Toujours OFF
F1	Toujours ON
F2	Passe en « on » durant un SCAN time à la mise sous tension (Set Ladder On).
F3	
F4	
F5	
F6	
F7	
F8	1 SCAN « On » chaque 10 ms
F9	1 SCAN « On » chaque 100 ms
F10	
F11	
F12	
F13	
F14	
F15	
F16	Répète cycle « ON/OFF » tous les 1 Scan time.
F17	Répète cycle « ON/OFF » tous les 2 Scan times.
F18	Répète cycle « ON/OFF » tous les 4 Scan times.
F19	Répète cycle « ON/OFF » tous les 8 Scan times.
F20	Répète cycle « ON/OFF » tous les 16 Scan times.
F21	Répète cycle « ON/OFF » tous les 32 Scan times.
F22	Répète cycle « ON/OFF » tous les 64 Scan times.
F23	Répète cycle « ON/OFF » tous les 128 Scan times.
F24	Répète cycle « ON/OFF » tous les 10ms
F25	Répète cycle « ON/OFF » tous les 20ms
F26	Répète cycle « ON/OFF » tous les 40ms
F27	Répète cycle « ON/OFF » tous les 80ms
F28	Répète cycle « ON/OFF » tous les 160ms
F29	Répète cycle « ON/OFF » tous les 320ms
F30	Répète cycle « ON/OFF » tous les 640ms
F31	Répète cycle « ON/OFF » tous les 1.28 secondes
F32	Répète cycle « ON/OFF » tous les 5.12 secondes
F33	Répète cycle « ON/OFF » tous les 10.24 secondes
F34	Répète cycle « ON/OFF » tous les 20.48 secondes
F35	Répète cycle « ON/OFF » tous les 40.96 secondes
F36	Répète cycle « ON/OFF » tous les 81.92 secondes
F37	Répète cycle « ON/OFF » tous les 163.84 secondes
F38	Répète cycle « ON/OFF » tous les 327.68 secondes
F39	Répète cycle « ON/OFF » tous les 655.36 secondes
F40	Appel « LADDERINT » en BASIC
F41	
F42	

* Si vous écrivez 1 en F40, vous pouvez générer un « LADDERINT » en BASIC. Consultez la description de la commande « ON LADDERINT GOSUB » pour davantage d'informations.

* F2 provoque un 1 Scan ON au moment de la commande BASIC « SET LADDER ON ».

* Les Registres « vierges » non expliqués sont réservés pour des usages futurs. Il conviendra de ne pas les utiliser.

Chapitre 11.

Tutorial : Premières applications en BASIC et LADDER

Préparation « matériel »

Alimentation :
La première chose à prévoir est avant tout une platine support qui vous permettra de relier le module CUBLOC™ à votre PC. Vous pouvez réalisez très facilement vous même cette platine. Le minimum requis est une alimentation **régulée et filtrée**.

Le module CUBLOC™ CB220 peut suivant l'entrée utilisée s'alimenter de 5,5 à 12 Vcc ou directement en + 5 Vcc (consultez les pages 35 pour plus d'information).

Les modules CUBLOC™ CB280, CB290 et CB405 ne peuvent s'alimenter qu'en 5 Vcc (consultez les pages 38, 39 et 43 pour le brochage).

N'utilisez JAMAIS de transformateur type bloc secteur dont la tension non régulée est souvent bien plus importante que celle indiquée sur le commutateur de réglage, vous risqueriez d'endommager irrémédiablement le module CUBLOC™. Utilisez un régulateur + 5 Vcc (associé à des condensateurs de découplage en entrée et en sortie) ainsi qu'un condensateur céramique de 47 nf à 0,1 uF à placer directement au plus près des broches d'alimentation du CUBLOC™.

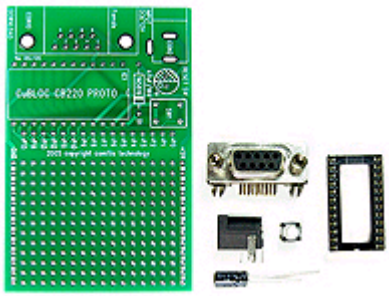
Sachez également qu'afin d'éviter d'avoir à souder les modules CUBLOC™, nous proposons des supports permettant de recevoir ces derniers :

- Référence « 18-AI » type support CI 24 broches pour le module CB220.
- Référence « CON-CB280 » jeu de 2 connecteurs pour le module CB280.
- Référence « CON-CB290 » jeu de 3 connecteurs pour le module CB290.
- Référence « CON-CB405 » jeu de 3 connecteurs pour le module CB290.

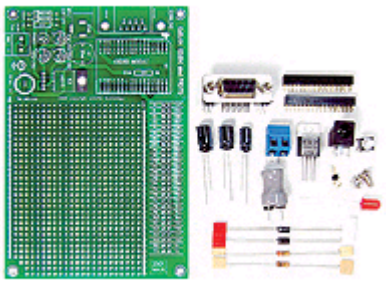
Entrée « Reset » :
A noter que l'entrée RESET des modules CUBLOC™ pourra rester « en l'air » ou recevoir de préférence une résistance de 10 Kohms de rappel au + 5 Vcc avec un bouton-poussoir relié entre la broche RESET et la masse. Ce dernier vous permettra de réinitialiser manuellement le CUBLOC™. La résistance et le bouton-poussoir devront être au plus près du module CUBLOC™ et la connexion ne devra pas dépasser quelques centimètres.

Connecteur de programmation :
Reste enfin à relier les broches de programmation du module CUBLOC™ au PC. Nous proposons pour ce faire un connecteur SUB-D 9 broches femelle coudé pour circuit imprimé (sous la référence « 16-W »). Vous trouverez les schémas relatifs à chaque modèles de CUBLOC™ en début de documentation (pages 34, 38, 41 et 43).

Afin de vous simplifier la tâche et de vous permettre de travailler plus rapidement, nous proposons des platines supports optionnelles qui permettront de recevoir les modules CUBLOC™.



« CB220-PROTO » Platine en kit pour le module CB220.



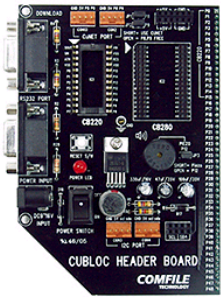
« MPROT-CB280 » Platine en kit pour le module CB280

« CB280-PROTO » Platine montée pour le module CB280.

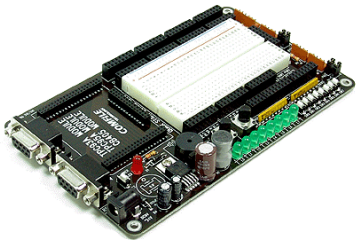
« CB290-PROTO » Platine montée pour le module CB290.

« CB220-I/O Cell Base » Platine montée montée pour le module CB220.

« CB280-I/O Cell Base » Platine montée pour le module CB280.



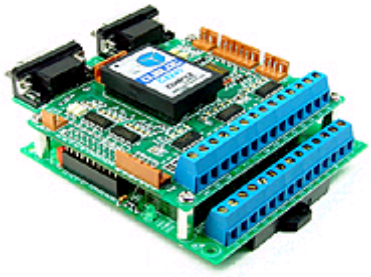
« Platine d'interface CUBLOC Header Board »



« Platine d'interface Quick-Start-Board »

« CUBASE-32M » Platine montée pour le module CB280 **(prochainement disponible – consultez-nous).**

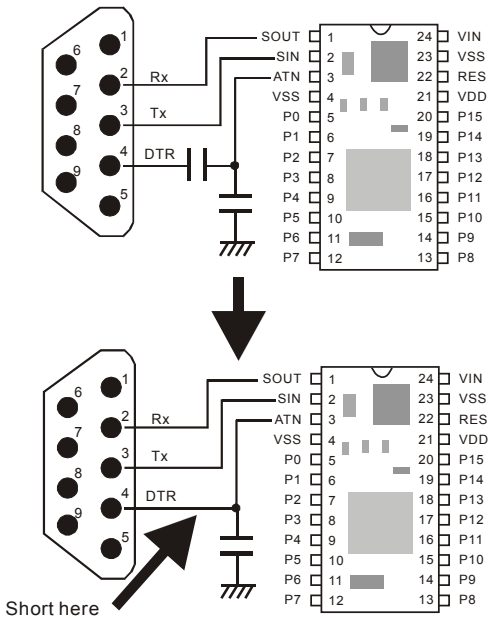
« CUBASE-64M » Platine montée pour le module CB290 **(prochainement disponible – consultez-nous).**



« CUSB-22D » Platine montée pour le module CB280

« CUBLOC-STUDY » Platine montée pour le module CB220 et CB280. Cette platine (décrite plus en détail ci-après) est spécialement conçue pour l'initiation et l'évaluation.

Note spéciale pour les utilisateurs des modules CUBLOC™ CB220 :
Comme indiqué en début de documentation, les modules CB220 sont compatibles broches à broches avec des modules 24 broches d'une autre marque également programmables en BASIC. Ceci pourra vous permettre de réutiliser vos platines existantes pour les modules CUBLOC™ CB220. Toutefois pour pouvoir y parvenir, il vous faudra shunter un condensateur présent sur vos platines au niveau du signal DTR (en effet cette capacité est pré-intégrée sur le module CUBLOC™ et si vous ne shuntez pas directement celle-ci sur la platine, le CUBLOC™ génèrera des messages d'erreurs lors des phases de programmation). Voir les schémas ci-dessous avec l'indication du condensateur à shunter.



Raccordement du CUBLOC™ au PC :
Si vous avez réalisé vous-même votre platine, n'insérez pas tout de suite le CUBLOC™ sur son support et vérifiez bien votre câblage ainsi que la présence du + 5 Vcc sur les entrées d'alimentation dédiées au CUBLOC™. Si tout est correct, coupez l'alimentation puis attendez quelques secondes avant d'insérer le module CUBLOC™ sur son support (ATTENTION AU SENS). Il ne vous reste plus qu'à relier la platine à un port série de libre du PC à l'aide d'un câble série mâle/femelle (nous proposon un modèle de câble sous la référence « CW014 »).

Si vous ne disposez pas de liaison RS232 de libre sur votre PC, il vous est possible d'utiliser un câble de conversion USB <> série.

Nous proposons à ce titre un modèle spécialement prévu à cet effet sous la référence « PCUSB6 ». Ce câble permet une fois son driver installé d'être « vu » comme un port COM série « virtuel » du côté du PC et de la platine du CUBLOC™ (tout en étant connecté sur le port USB de votre PC).

Note concernant l'installation du driver de ce câble :

- Insérez le CD-ROM livré avec le câble dans le lecteur de votre PC (sans relier le câble au port USB, ni à la prise SUB-D du module CUBLOC™). Laissez le programme d'installation démarrer et cliquez vers le bas de la fenêtre sur le câble bleu identique au votre. Le programme vous proposera alors de procéder à l'installation du driver.

- Fermez ensuite le programme d'installation et insérez alors le câble sur le port USB de votre PC (sans relier l'autre partie au CUBLOC™).

- Allez sur le bureau principal de Windows™. Réalisez un « click » droit de souris sur l'icône du poste de travail, puis cliquez sur « Propriétés ».

- Cliquez ensuite sur l'onglet « Matériel », puis sur la sélection « Gestionnaire de périphériques », puis double cliquez sur la sélection « Ports (COM et LPT).

- Double cliquez sur la sélection « Prolific USB-to-Serial Comm Port (COMx) » (ou x peut avoir différentes valeurs suivant les configurations). Si la valeur x correspond à une valeur comprise en 1 et 6 et que ceci correspond bien à un port COM non actuellement utilisé par un autre dispositif de votre PC, notez simplement le nombre afin de pouvoir configurer correctement le logiciel de programmation des CUBLOC™ (voir page 61) et refermez toutes les fenêtres.

- Si par contre le n° du port (x) est déjà exploité par un autre dispositif, cliquez sur l'onglet « Paramètres du port », puis sur la sélection « Avancée… ». A ce stade, vous aurez la possibilité de sélectionner au bas de la fenêtre un N° de port non utilisé (choisissez un N° compris entre 1 et 6). Validez la sélection par la touche « OK », puis notez le N° du port que vous avez affecté au câble et refermez toutes les fenêtres.

Note concernant l'utilisation du driver de ce câble :
Lorsque vous avez fini d'utiliser le CUBLOC™ et que vous éteignez votre PC, il conviendra de déconnecter le câble du port USB du PC. A la remise sous tension du PC, une fois le système d'exploitation opérationnel, reconnectez à nouveau le câble sur le port USB afin que ce dernier soit de nouveau correctement détecté (en cas contraire, si vous avez laissé le câble connecté sur le port USB et que le PC est allumé, il se peut que le câble ne soit pas automatiquement reconnu de nouveau – dans ce cas, déconnectez-le du port USB attendez quelques instants et reconnectez-le à nouveau).

Installation du logiciel «CUBLOC Studio » :
Consultez la feuille d'installation livrée avec votre module CUBLOC™ afin de savoir où trouver le logiciel sur le CD-ROM joint (la procédure est très simple et repose sur le décompactage du programme et à l'exécution d'un programme d'installation conventionnel qui pourra vous créer -- si vous le désirez - une icône sur votre bureau).

Le CD-ROM qui vous a été livré avec les modules CUBLOC™ contient généralement la dernière version en date du logiciel de développement « CUBLOC Studio ».

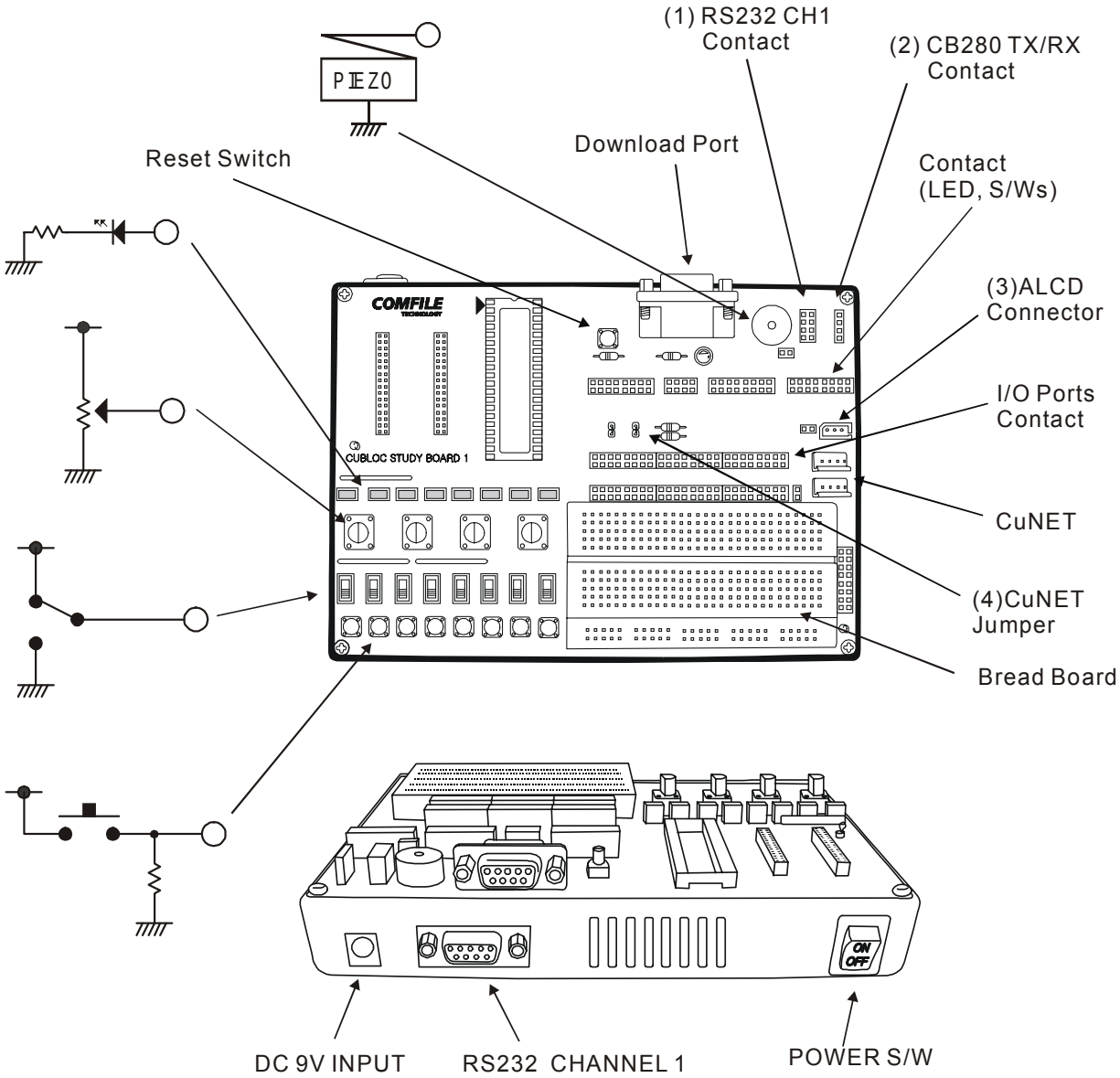
Toutefois de part les mises à jours régulières opérées par Comfile Technology, il vous faudra impérativement vérifier sur le site www.comfiletech.com (rubrique « Download ») que vous disposez bien de la dernière version en date du « CUBLOC Studio » avant d'installer et d'utiliser ce dernier lors de votre première utilisation.

Constatez-nous si vous ne disposez pas de connexion Internet afin que nous puissions vous adresser si nécessaire la dernière version à jour.

Pensez également à vous connecter régulièrement sur le site www.comfiletech.com (rubrique « Download ») pour télécharger les mises à jours.

La platine «CUBLOC Study Board » :
Les pages qui suivent vont vous permettent de réaliser respectivement votre premier programme en langage BASIC, votre premier programme en langage LADDER et votre première application qui exploitera à la fois une programmation en langage BASIC et LADDER. Pour ce faire, nous avons choisi d'utiliser la platine « CUBLOC Study Board » qui est idéalement conçue pour l'évaluation. Cette dernière est également utilisée dans la plupart des notes d'applications dédiées aux CUBLOC™ que vous pourrez retrouver sur le CD-ROM ou sur notre site Internet www.lextronic.fr.

Description de la platine :

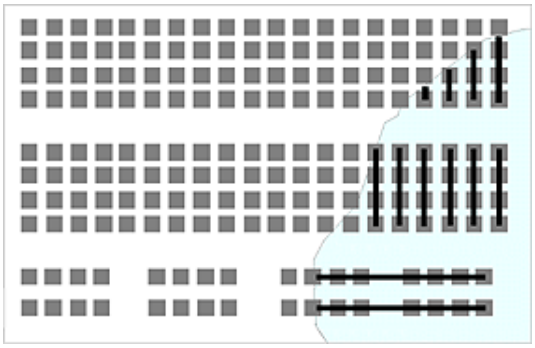


La platine « CUBLOC Study board » comprend :

- 8 boutons-poussoirs.
- 8 interrupteurs.
- 4 potentiomètres.
- 8 leds.
- 1 bouton de « Reset »
- 1 Buzzer Piezo.
- 2 connecteurs dédiés aux pilotage de module « CuNet » spécialisés.
- 1 connecteur pour liaison I2C™.
- 1 étage de régulation + 5 Vcc permettant d'alimenter la platine sous + 9 Vcc.
- 1 connecteur sub-D 9 Broches dédié au téléchargement de vos programmes.
- 1 connecteur sub-D 9 Broches dédié aux communications RS232.
- 2 supports (le modèle de gauche est destiné à recevoir un module CB280 – le modèle de droite est prévu pour un module CB220).
- 1 plaque de connexion sans soudure avec reprise possible du + 5 Vcc de la platine.

Chaque connexion des périphériques de la platine (bouton-poussoir, buzzer, Leds, etc..) ainsi que chaque port d'entrée/sortie du module CUBLOC™ sont accessibles sur des connecteurs femelles « doubles » au pas de 2,54 mm. Il vous sera ainsi très facilement possible de relier les ports de sorties du CUBLOC™ aux Leds, les ports d'entrées aux interrupteurs, etc…

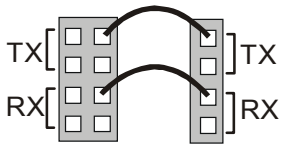
Le fait de disposer de connecteurs à double contact vous permettra par exemple de positionner une sonde d'oscilloscope en parallèle afin de pouvoir observer l'évolution d'un signal par exemple (des reprises pour la masse (GND) sur des petites boucles sont à ce titre également disponibles).



La platine intègre également une plaque de connexion sans soudure qui vous permettra d'utiliser des composants additionnels, sans avoir besoins de souder ces derniers grâce à l'existence de raccordements internes préexistants sur la plaque de connexion. Des reprises d'alimentation + 5 Vcc et masse (GND) sont également à votre disposition à droite de la plaque de connexion afin que vous puissiez alimenter les composants additionnels.

La platine « CUBLOC Study board » intègre un étage de mise à niveau interne via un circuit intégré MAX232. Il vous suffira alors de relier les signaux TX / RX du CUBLOC™ présentant un niveau logique 5 V sur les connecteurs en haut à droite TX / RX RS232C afin que vous puissiez rendre ces signaux compatibles avec une liaison série au niveau +/- 12 V sur la prise sub-D 9 broches présente à l'arrière de la platine.

Si vous utilisez un module CB280, il vous suffira alors de réaliser les pontages ci-dessous pour mettre à niveau les signaux RS232.



Si vous désirez piloter un afficheur spécialisé via une liaison série (au niveau logique 5 V), il vous suffira de relier la broche TX (niveau logique 5 V) du CUBLOC™ sur une des broches du connecteur femelle « RS232 LCD connect -> » et de relier le câble de l'afficheur sur le petit connecteur blanc 3 points.

Si vous désirez utiliser des modules spécialisés « CuNET », il vous faudra faire un pontage des plots en JP1 et JP2 à l'aide des cavaliers.

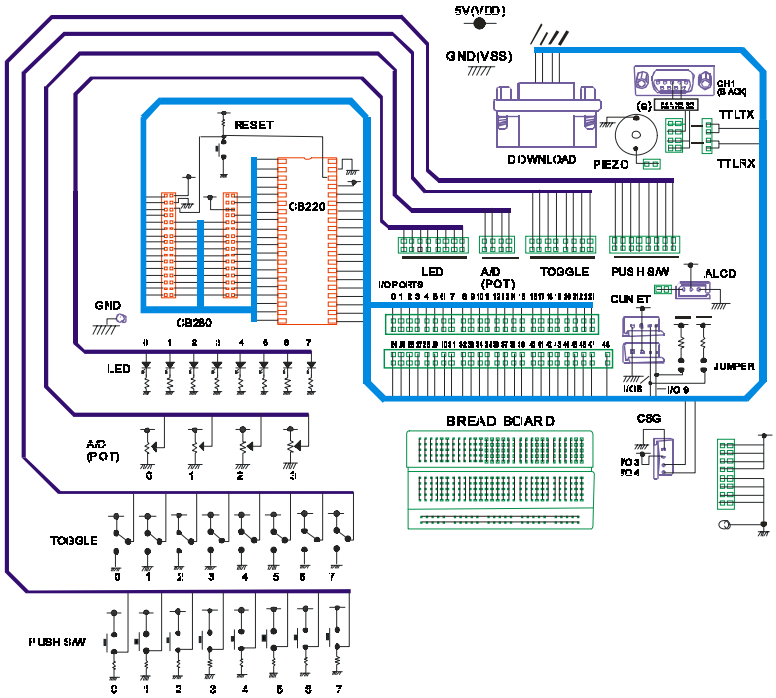


Schéma théorique de la platine « CUBLOC Study Board »

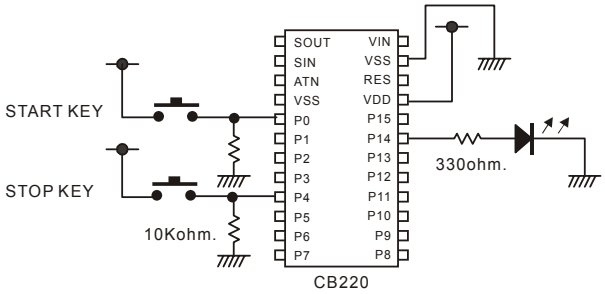
Réalisation de votre premier programme « BASIC »

Commencez en premier lieu par vérifier que l'interrupteur au dos de la platine soit sur la position « OFF ». Insérer ensuite avec précaution le module CUBLOC™ CB220 sur le haut de son support 40 broches en utilisant comme repère le marquage « 24 PIN ». Vérifiez également le sens du module CUBLOC™ (le repère avec le demi rond blanc doit être vers le haut de la platine comme indiqué sur le support). Si vous utilisez un CB280 vous disposez d'un ergot en haut à gauche servant de repère.

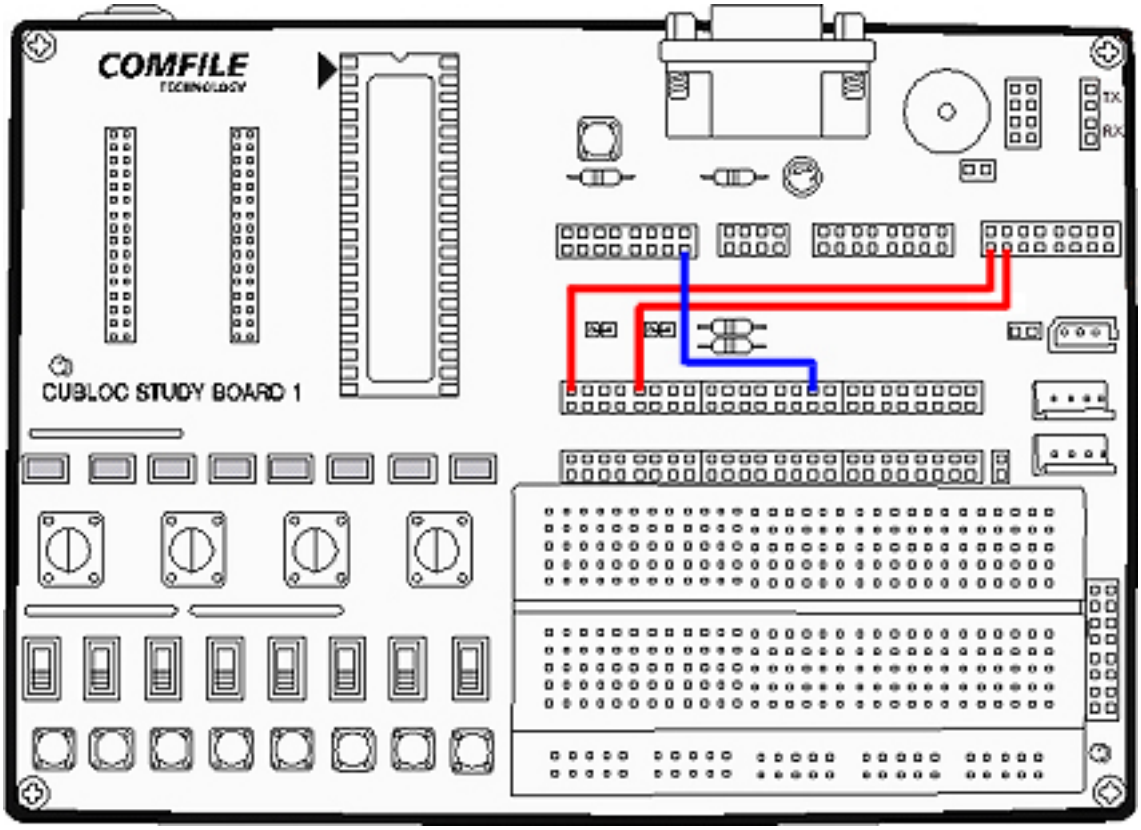
« Allumez » ensuite votre PC et raccordez le câble de programmation sur le connecteur sub-D 9 broches du dessus (celui avec le marquage « DOWNLOAD »). Alimentez alors la platine à l'aide d'une tension régulée et filtrée de +9 Vcc (la polarité de la tension n'a pas d'importance). Il vous faut par contre utiliser impérativement et uniquement une tension de + 9 Vcc (nous proposons un bloc tout indiqué sous la référence PSU10R si nécessaire).

Commutez enfin l'interrupteur au dos de la platine sur la position « ON » (la Led « PWR LED » doit alors s'allumer).

Votre premier programme sera volontairement très simple. Il consistera à réaliser l'application ci-dessous dans laquelle on utilisera 2 ports (le port 0 et le port 4) du module CUBLOC™ en entrée afin d'y relier des poussoirs que l'on appellera « START » et « STOP » ainsi qu'un port en sortie (le port 14) sur lequel on connectera une Led.



Ainsi la « retranscription » du montage ci-dessus sur la platine « CUBLOC Study Board » nécessitera que vous réalisiez les pontages suivants à l'aide de fils électriques.



Les 2 premiers boutons-poussoirs en partant de la gauche correspondront donc respectivement aux boutons « START » et « STOP » et la Led L7 sera utilisée.

- Avant de « lancer » le programme du « CUBLOC Studio » sur votre PC, effectuez la manipulation indiquée en page 100 de ce manuel (celle-ci vous permettra de réaliser une configuration nécessaire au bon fonctionnement du logiciel – même si l'opération demandée ne concerne pas l'exemple du tutorial, elle aura le mérite d'être faite une fois pour toute et de ne pas gêner vos développements futurs).

- Exécutez ensuite le programme « CUBLOC Studio » sur le PC.

- Commencez par aller dans le menu « Setup », puis sélectionnez « PC interface setup… ». Sélectionnez alors le N° du port COM du PC sur lequel vous avez connecté le cable série de programmation du CUBLOC™.

  Si vous utilisez un cable de raccordement USB, sélectionnez alors ne N° de port COM virtuel préalablement choisi (voir page 346). Validez la sélection avec le bouton « Ok ».

- Allez enfin dans le menu « File » et sélectionnez « New ».
  Saisissez alors le programme ci-dessous :

```
Const Device = cb220

Dim a As Byte
Do
        If In(0) = 1 Then a = 1
        If In(4) = 1 Then a = 0
        Out 14,a
Loop
```

Pour rappel, les lignes ne doivent être « collées » à fond vers la gauche, il vous faut en effet laisser quelques espaces par rapport au bord gauche de l'écran (utiliser une ou deux fois la touche « TAB » par exemple pour obtenir un espacement constant).

Une fois le programme saisie, cliquez sur le menu « Run » (ou sur le petit triangle bleu de la barre d'outils du haut de l'écran). A ce stade, un nom de fichier vous est demandé. Ce dernier correspondra au nom que vous voudrez donner à votre application. Choissez par exemple « BASIC » et validez. A ce stade, une barre de progression au bas de l'écran doit s'afficher, indiquant ainsi que le module CUBLOC™ est en cours de programmation.

Si un message du type « Invalid RS232C Port ! ! ! » s'affiche, c'est que vous avez mal configuré le N° du port série à utiliser ou que le câble de programmation n'est pas relié au PC ou au CUBLOC™ ou si vous utilisez un câble USB, que ce dernier n'à pas été correctement détecté (déconnectez-le et reconnectez le alors sur le port USB).

Si un message d'erreur s'affiche toujours malgré vos vérifications lors de votre tentatives de programmation, essayez de remettre à jour le Firmware de votre module CUBLOC™. Il faut en fait savoir que le « Firmware » des modules est mémorisé dans le logiciel « CUBLOC Studio » (d'où l'intérêt de télécharger la dernière version en date sur le site www.comfiletech.com afin de pouvoir bénéficier des dernières possibilités de votre module). Consultez la F.A.Q du chapitre 12 (page 366) pour connaître la procédure.

Note : Il se peut également que lors d'une première utilisation, le « CUBLOC Studio » vous affiche directement un message relatif au « firmware » en vous indiquant qu'un « Firmware » plus récent est présent et en vous invitant à remettre à jour ce dernier dans le module CUBLOC™. Dans ce cas acceptez la mise à jour qui se par la même procédure que celle décrite dans la F.A.Q du chapitre 12 (page 366).

Dans tous les cas, patientez jusqu'à la fin de l'opération de remise à jour du Firmware. **A noter également que la remise à jour du « Firmware » efface tout programme potentiellement présent dans le CUBLOC™.**

Rappel important : Dans tous les cas, ne déconnectez **JAMAIS** le câble de programmation et/ou ne coupez **JAMAIS** l'alimentation du module CUBLOC™ lorsque ce dernier est en cours de programmation ou que vous remettez à jour son « Firmware ».

Interprétation du programme :
Ce programme très simple se déroule au sein d'une boucle « sans fin » du type :

```
Do
 .
 .
 .
Loop
```

La ligne de programme **If In(0) = 1 Then a = 1** test l'état de l'entrée du port P0.
Si le bouton-poussoir qui est relié dessus est sollicité, le port passe au niveau logique 0 et selon la condition de test, la variable « a » prendra la valeur 1.

La ligne de programme **If In(4) = 1 Then a = 0** test l'état de l'entrée du port P4
Si le bouton-poussoir qui est relié dessus est sollicité, le port passe au niveau logique 0 et selon la condition de test, la variable « a » prendra la valeur 0.

La ligne de programme **Out 14,a** applique sur le port P14 le niveau logique défini par la variable « a ».

Donc lorsque vous sollicitez alternativement les 2 boutons poussoirs de gauche de la platine (le modèle le plus à gauche fera passer la variable « a » à 1, tandis que le 2ème bouton fera passer la variable « a » à 0, ce qui se traduira selon les cas par l'allumage ou l'extinction de la Led L7.

Attention rappelez-vous que lorsque l'on travaille avec les ports d'entrée/sortie, on fait référence au **N° du port** et non pas au N° de la broche du CUBLOC™.

Il est intéressant de s'attarder sur une caractéristique propre à la plupart des boutons-poussoirs. En effet lors de leurs manipulations, il est fréquent de constater que les contacts générés par ceux-ci ne sont pas « francs » et sont à l'origine de « rebonds » comme le montre la figure ci-dessous.

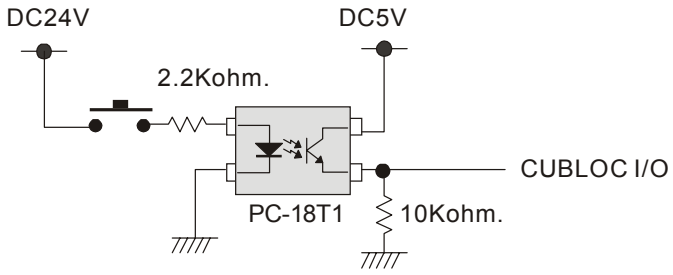
Ces rebonds peuvent interférer avec le fonctionnement de votre application qui au lieu de détecter un passage « franc » du niveau « bas » vers le niveau « haut », sera alors en présence d'une succession de passage « bas/haut ». Si dans le petit exemple ci-dessus, le phénomène n'aura aucune conséquence, il peut en être autrement pour d'autres applications.

Pour pallier ce type de phénomène, on peut avoir recours à un filtrage via une résistance et un condensateur sur l'entrée du module CUBLOC™, mais également à une instruction spéciale du CUBLOC™ : **KEYINH()** en remplacement de **IN()** afin de gérer ce « problème » de rebond par une sorte de filtrage « logiciel » comme le montre l'exemple de programme ci-dessous.

```
Const Device = cb220

Dim a As Byte
Do
        If Keyinh(0,20) = 1 Then a = 1
        If Keyinh(4,20) = 1 Then a = 0
        Out 14,a
Loop
```

Le 2ème paramètre de l'instruction KEYINH(0, 20) permet de définir la durée qui permettra d'éliminer l'effet de rebond de la touche. Dans notre exemple, l'état de la touche ne sera pris en compte qu'au bout de 20 ms.

Dans le cadre d'une application en milieu perturbé, il vous faudra utiliser un montage similaire à celui présenté ci-dessus afin de pouvoir d'une part utiliser des tensions supérieures à 5 Vcc en entrée et surtout pour pouvoir bénéficier d'une isolation entre le CUBLOC™ et le bouton-poussoir.



Mise en œuvre du mode « DEBUG »
Saisissez maintenant le programme suivant et exécutez-le.

```
Const Device = cb220

Dim a As Byte
Do
        If Keyinh(0,20) = 1 Then
           a = 1
           debug dec a, "  Led allumée",Cr
        end if
        If Keyinh(4,20) = 1 Then
           a = 0
           debug dec a, "  Led étteinte",Cr
        end if
        Out 14,a
Loop
```

Vous verrez alors apparaître une fenêtre sur le PC qui affichera les messages : **1 Led allumée** ou **0 Led éteinte** lorsque vous solliciterez les boutons-poussoirs de la platine. Ces messages sont générés par les commandes **debug** présentes dans le programme qui afficheront la valeur décimale de « a », associée à un message (Led allumée ou led éteinte) – la commande Cr en bout permet de revenir à la ligne suivante dans la fenêtre de débug du PC. Le recours aux commandes Debug sera extrêmement utile pour vous aider à développer vos applications plus rapidement et plus simplement. Vous pourrez ainsi vérifier que votre programme « passe » bien à un endroit particulier de votre listing en plaçant une commande debug à l'endroit en question ou encore pour vérifier aisément la valeur de vos variables à un moment précis de cotre application.

A noter qu'avant de pouvoir programmer à nouveau votre module CUBLOC™, il vous faudra au préalable fermer la fenêtre de Debug au niveau du PC.

Il conviendra par contre d'éviter d'utiliser une commande debug de façon répétée dans une boucle comme dans l'exemple ci-dessous par exemple :

```
Do
  a = a + 1
  Debug dec a, Cr
loop
```

Dans ce cas, l'incrémentation de la valeur décimale de « a » s'affichera de façon permanente et continue dans la fenêtre Debug. Le problème est que dès lors, le port de communication série initialement dédié au téléchargement du programme dans le CUBLOC™ sera monopolisé pour la « remontée » de la valeur de « a » dans la fenêtre de Debug et lorsque vous essayerez à nouveau de programmer le CUBLOC™, ce dernier vous affichera dans la plupart des cas un message d'erreur signalant qu'il n'arrive pas à télécharger son programme. Dans ce cas, il vous faudra à chaque fois fermer la fenêtre Debug et essayer à nouveau de programmer le CUBLOC™ jusqu'à temps que ce dernier arrive à entrer en liaison avec le PC ! Dans certains cas extrêmes, lors de ces tentatives de programmation et de conflits de données présentes sur le port série, le « CUBLOC Studio » pourra même considérer que le CUBLOC™ a changé de « Firmware » et vous demander si vous désirez le reprogrammer.

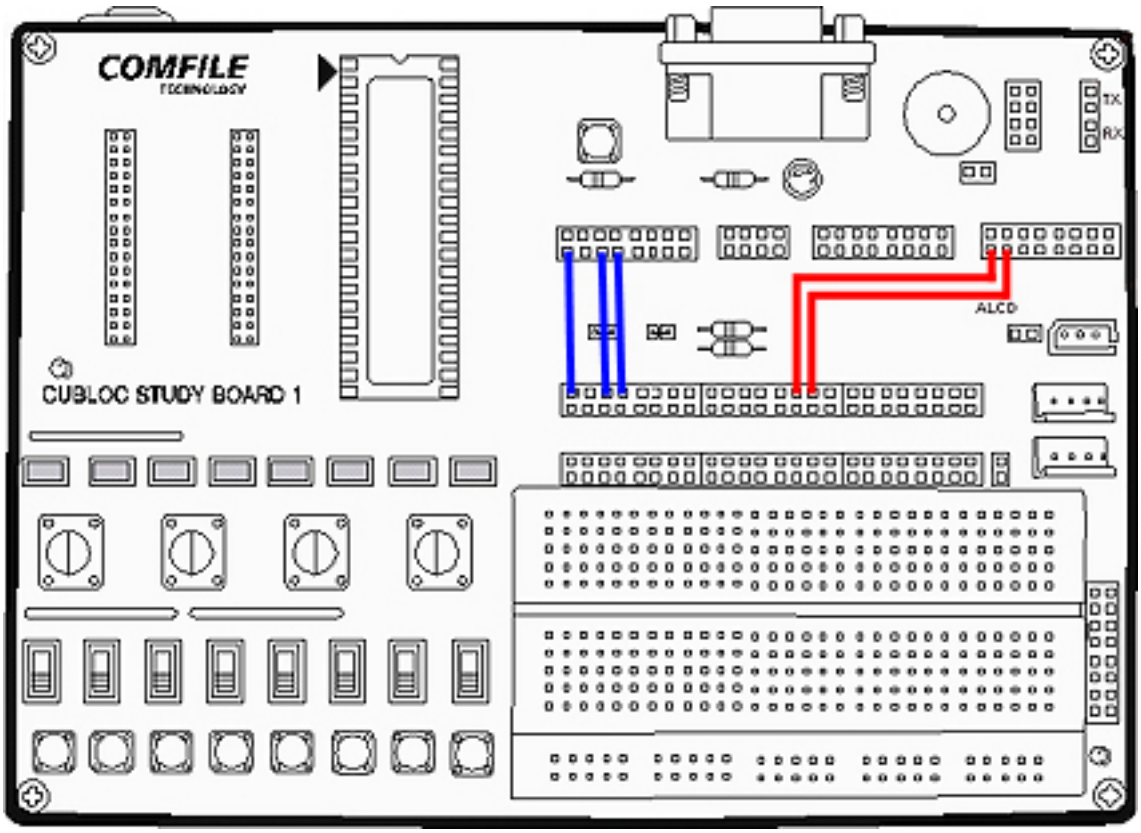
Usez donc des commandes Debug… sans en abuser ! Dans les cas où vous devez utiliser une commande dans une boucle pour surveiller l'évolution d'une variable, il conviendra alors (quitte à ralentir arbitrairement votre programme) à placer une commande de temporisation après la commande Debug afin que le « CUBLOC Studio » puisse trouver plus facilement une période pendant laquelle le port série n'est pas exploité pour entrer en communication avec le module CUBLOC™.

Vous connaissez maintenant les principes de base de la programmation d'une application en langage BASIC et nous vous invitons à développer vous-même vos propres programmes. N'hésitez pas tout de même à relire le chapitre de la page 47 relatifs aux précautions d'usages. Ainsi gardez à l'esprit de ne jamais connecter un port du CUBLOC™ utilisé en sortie sur un bouton-poussoir ou un interrupteur par exemple (reconfigurez donc IMPERATIVEMENT vos ports en entrées avant de connecter des signaux dessus).

Si nécessaire, les programmes ci-dessus sont présents (sous les noms : essai1 et essai2) sur notre site internet : www.lextronic.fr. Consultez également les très nombreuses notes d'applications qui vous apporteront une aide précieuse.

Réalisation de votre premier programme « LADDER »

Dans cette description, nous considèrerons que vous avez déjà inséré le module CUBLOC sur son support comme indiqué dans la procédure au bas de la page 351 et que vous avez déjà configuré le N° du port série utilisé pour la programmation comme indiqué en haut de la page 346. Commencez par vérifier que l'interrupteur au dos de la platine soit sur la position « OFF » et réalisez alors le câblage ci-dessous.

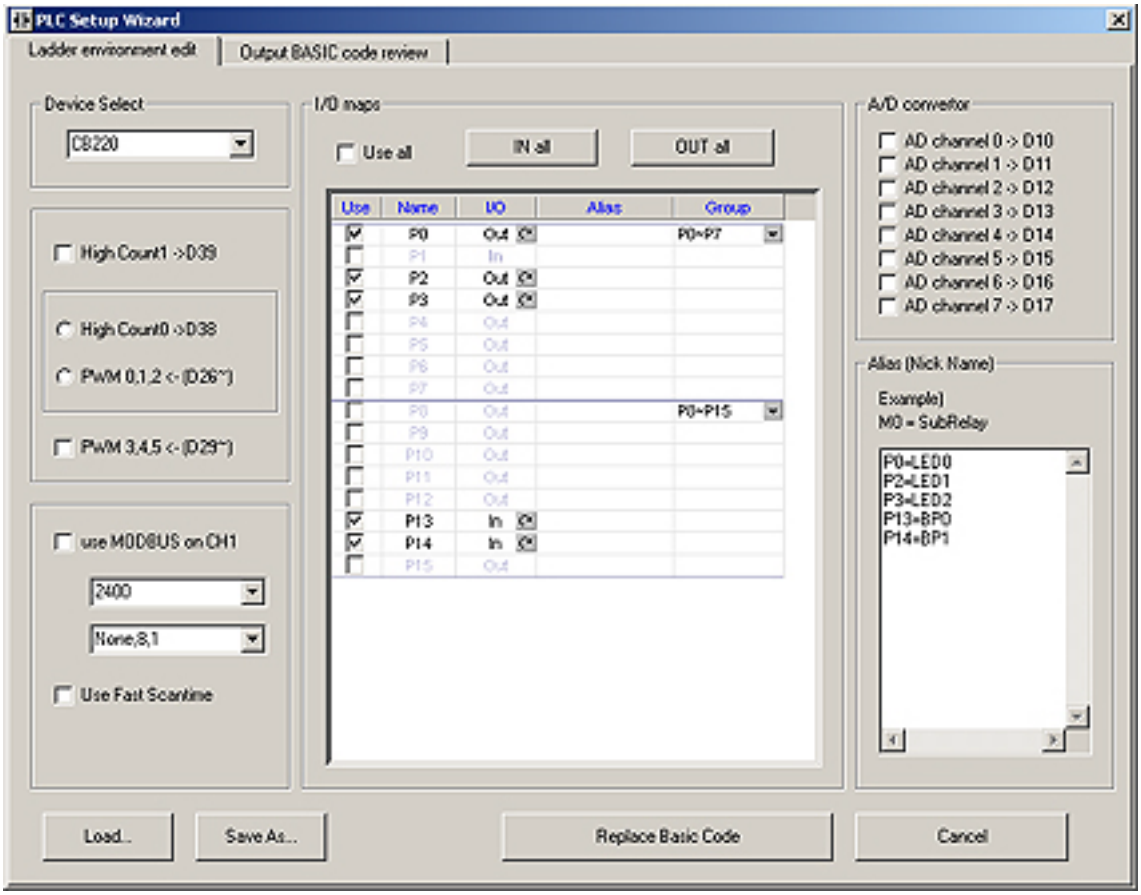


- Exécutez alors le programme « CUBLOC Studio » sur le PC.

- Allez enfin dans le menu « File » et sélectionnez « New ».

- Comme indiqué dans ce manuel, la programmation en LADDER nécessite toutefois de déclarer au préalable certains paramètres par le biais d'instructions au sein d'un mini programme BASIC. Vous pouvez effectuer cette déclaration manuellement ou utiliser un utilitaire spécialement dédié à cet usage comme nous allons le faire ci-dessous.

- Pour ce faire, allez ensuite dans le menu « Setup », puis sélectionnez « PLC setup Wizard… ». A ce stade une nouvelle fenêtre s'affiche. Commencez par sélectionner le modèle de CUBLOC™ que vous utilisez dans la sélection « Device Select ». Puis indiquez les ports du CUBLOC™ qui devront être utilisés en « LADDER » ainsi que leur état (entrée « in » ou sortie « ou » en cliquant sur la petite flèche arrondie pour modifier l'état – pour rappel, P1 ne peut être utilisé qu'en entrée). Reportez-vous à la copie d'écran ci-dessous pour réaliser la configuration.



- La section des Alias vous permettra lors de la saisie dans le LADDER d'utiliser les désignations telles que LED0, BP1… plutôt que les noms de « Registres » (ce qui rend l'écriture et la compréhension de votre programme plus simple). Lors de la compilation du programme le « CUBLOC Studio » fera automatiquement le lien entre le nom d'Alias et son « Registre » associé). Une fois terminé, cliquez sur l'onglet « Output BASIC review » afin d'avoir une idée sur la nature du programme BASIC d'initialisation que vous auriez du saisir manuellement si vous n'aviez pas utilisé l'utilitaire spécialisé. Afin de pouvoir commencer la programmation en « LADDER », cliquez sur « Replace Basic Code » au bas de la fenêtre. Le programme « CUBLOC Studio » vous demandera alors si vous voulez sauvegarder le programme BASIC potentiellement déjà présent dans l'éditeur avant de le remplacer par le petit programme d'initialisation ainsi créée. Dans notre cas, du fait que nous n'avons aucun programme BASIC en cours de réalisation, il ne sera pas nécessaire de procéder à une sauvegarde (une seconde confirmation vous sera toutefois demandée – mais ici en encore, il ne sera pas nécessaire de sauvegarder l'ancien programme BASIC).

- A ce stade, le petit programme BASIC d'initialisation du LADDER réalisé de façon automatique prendra place dans l'éditeur du BASIC. Cliquez alors sur l'onglet **[F2] LADDER** ou tapez la touche de fonction **F2** pour afficher la fenêtre de l'éditeur du LADDER et saisissez le programme présenté en page suivante (aidez-vous de la description des commandes en page 282 si nécessaire).

Le programme ci-dessous est présent (sous le nom : essai3) sur notre site internet : www.lextronic.fr.

Une fois le programme saisie, alimentez la platine et cliquez sur le menu « Run » (ou sur le petit triangle bleu de la barre d'outils du haut de l'écran). A ce stade, un nom de fichier vous est demandé. Ce dernier correspondra au nom que vous voudrez donner à votre application. Choissez par exemple « LADDER » et validez. A ce stade, une barre de progression au bas de l'écran doit s'afficher, indiquant ainsi que le module CUBLOC™ est en cours de programmation.

Si un message du type « Invalid RS232C Port ! ! ! » s'affiche, c'est que vous avez mal configuré le N° du port série à utiliser ou que le câble de programmation n'est pas relié au PC ou au CUBLOC ou si vous utilisez un câble USB, que ce dernier n'à pas été correctement détecté (déconnectez-le et reconnectez le alors sur le port USB).

Consultez également la note concernant l'apparition potentiel d'un message à propos du « Firmware » du CUBLOC™ au bas de la page 366.

Si la programmation du CUBLOC™ se passe correctement, la Led 2 doit se mettre à clignoter.

Interprétation du programme :
Ce programme très simple permet de mettre en œuvre quelques « Registres » et fonctions spéciales des CUBLOC™.

Le « Registre » F22 génère ainsi automatiquement des impulsions toutes les 64 « scan times ». Ce « Registre » est directement appliqué sur la Led2 (le port P3). Il s'en suit que la Led Led2 clignote automatiquement.

Le « Registre » F21 génère ainsi automatiquement des impulsions toutes les 32 « scan times ». Ce « Registre » inséré en série avec le bouton poussoir N° 2 (du port P14) est appliqué sur la Led1 via le port P2). Ainsi en absence de sollicitation du poussoir N° 2, la Led1 est éteinte et elle se met à clignoter 2 fois plus vite que la Led2 lorsque le bouton-poussoir N° 2 est utilisé.

La commande TAOFF est activée sur sollicitation du bouton-poussoir n° 0 (relié sur l'entrée P13). Ceci a pour action d'initialiser le « Registre » Timer T1 avec une temporisation de 5 s et d'allumer par la même la Led0 (le timer étant injecté sur le port P0). Au terme de la durée du Timer T1, La led0 s'éteint.

Mise en œuvre du mode « Monitor » :
Le « CUBLOC Studio » dispose d'une puissante option qui vous permettra de visualiser l'exécution de votre programme et des différents « Registres » utilisés dans celui-ci.

Pour activer ce mode, sélectionnez le menu « Run », puis « Ladder Monitor off ».
Vous pouvez également utiliser la combinaison de touche « Ctrl + F7 » ou cliquer sur le bouton représentant un carré vert entouré de 2 barres noires dans la boîte à outils du haut de l'écran.

A ce stade, la fenêtre du mode Debug fait apparaître votre programme LADDER dans lequel vous pourrez suivre l'évolution des valeurs de vos « Registres » Timers, entrées, sorties en quasi temps réel (sauf pour les « Registres » dont le temps d'évolution est trop rapide).

Cliquez sur le bouton « Monitoring Stop » pour sortir de ce mode (ce qui est impératif si vous désirez programmer à nouveau votre module CUBLOC™). Afin de pouvoir visualiser l'évolution des « Registres » les plus rapides, il vous est possible d'avoir accès à un deuxième outil pré-intégré dans le « CUBLOC Studio » (voir ci-dessous).

Mise en œuvre du mode « Time Chart Monitor… » :
Cette outil fait office en quelque sorte de mini analyseur logique dédié aux « Registres » des modules CUBLOC™. Pour activer ce dernier, il vous suffira (à condition que la fenêtre du moniteur ne soit pas activée), de sélectionnez le menu « Run », puis « Time Chart Monitor…».
Vous pouvez également utiliser les petits signaux carrés verts dans la boîte à outils du haut de l'écran.

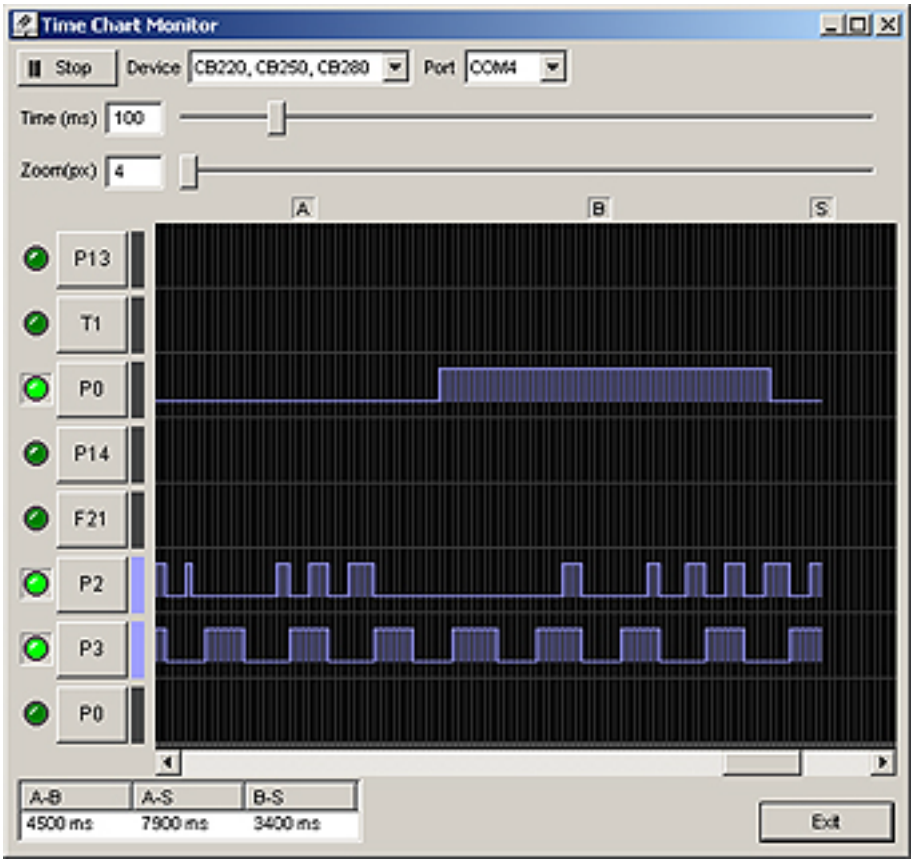
A ce stade, une nouvelle fenêtre apparaît dans laquelle vous pourrez sélectionner le modèle de CUBLOC™ que vous utilisé (dans la sélection Device) et le N° du port utilisé pour la communication entre le PC et le CUBLOC™.

Le curseur Time (ms) permet de régler la fréquence d'échantillonnage (réglez temporairement cette dernière sur 100) et choisissez un Zoom de valeur 4.

Sur le côté gauche de l'écran vous avez à votre disposition 8 boutons qui vous permettront de choisir le nom des « Registres » dont vous voudrez faire la surveillance (vous pouvez choisir des « Registres » de type mémoire, Timer, Sortie…).

En cliquant sur le voyant vert à gauche des touches représentant les « Registres », vous allez pouvoir activer (voyant allumé) ou désactiver (voyant éteint) la mesure des signaux des « Registres ». Ne validez dans un premier temps qu'un seul « Registres » (P3 par exemple), puis cliquez sur le bouton « Start » (en haut à gauche de la fenêtre).

Après quelques instant, cliquez sur le bouton « Stop ». Dès lors, la barre d'ascenseur horizontale au bas de l'écran vous permettra de vous déplacer dans l'ensemble du signal acquis, tandis que le curseur du « zoom » vous permettra de modifier l'échelle d'affichage. Il vous sera enfin possible de déplacer 3 curseurs dont les durées d'intervalles s'affichent automatiquement en bas à gauche de l'écran.

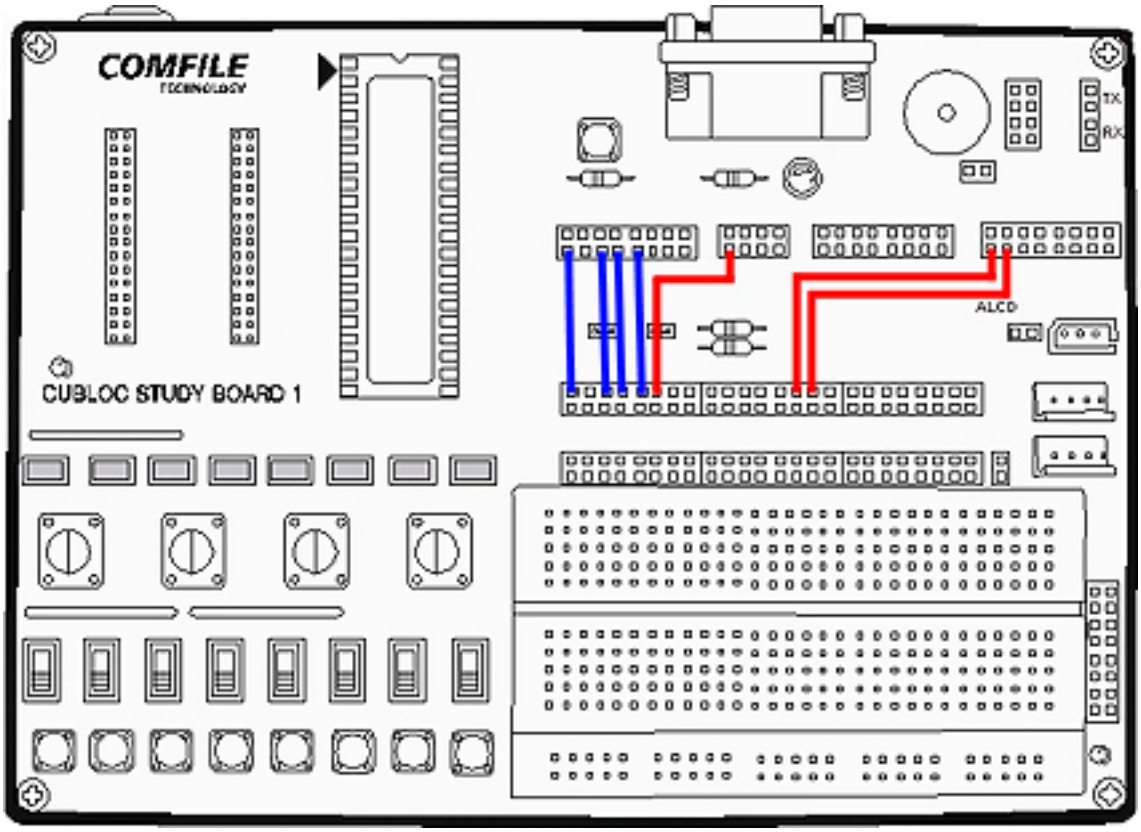


A noter qu'il vous est possible à tout moment lors de la phase d'acquisition d'activer ou de désactiver un des signaux par l'intermédiaire des voyants verts de gauche. A noter toutefois que si plusieurs signaux sont sollicités en même temps ou si des signaux de nature différentes sont sélectionnés, l'affichage de ces derniers pourra être « altéré » et « haché ». Il conviendra donc de sélectionner les signaux avec attention afin de pouvoir afficher des données exploitables à l'écran.

Cliquez sur le bouton « Exit » pour sortir de ce mode (ce qui est impératif si vous désirez programmer à nouveau votre module CUBLOC™).

Réalisation de votre premier programme « BASIC et LADDER »

Cette petite application vous permettra de mettre en lumière les possibilités de fonctionnement multitâches des modules CUBLOC™ par le biais d'un programme BASIC et LADDER qui fonctionneront simultanément. Dans cette description, nous considèrerons que vous avez déjà inséré le module CUBLOC sur son support comme indiqué dans la procédure au bas de la page 351 et que vous avez déjà configuré le N° du port série utilisé pour la programmation comme indiqué en haut de la page 346. Commencez par vérifier que l'interrupteur au dos de la platine soit sur la position « OFF » et réalisez alors le câblage ci-dessous.



Le montage reprend exactement le même câblage que pour l'exemple du programme « LADDER » de la page 266 en y ajoutant une liaison entre le premier potentiomètre de gauche de la platine et le port P5 ainsi qu'une seconde liaison entre le port P4 et la Led 3 de la platine. Une fois le cablâge terminé alimentez la platine et exécutez le programme « CUBLOC Studio » sur le PC.

- Allez ensuite dans le menu « File » et sélectionnez « New ».

- Cliquez sur la touche **F2** et saisissez alors le programme « LADDER » de la page 282 (sans utiliser le logiciel de configuration automatisé du programme de déclaration du LADDER)

- Cliquez sur la touche **F1** et saisissez ensuite le programme « BASIC » présenté ci-dessous.

Cette application complète (programme BASIC et LADDER) est présent (sous le nom : essai4) sur notre site internet : www.lextronic.fr.

```
Const Device = cb220
Usepin 0,Out,LED0
Usepin 2,Out,LED1
Usepin 3,Out,LED2
Usepin 13,In,BP0
Usepin 14,In,BP1
Set Ladder On
Dim pot As Integer
Input 5                          ‘ Configure le Port P5 en entrée
Do
        Pot = Adin(5)            ‘ Lecture de la tension du curseur du potentiomètre
        If pot > 500 Then
          Out 4,1                ‘ Allume la Led 3
          Debug Dec pot,Cr       ‘ Affiche la valeur dans la fenêtre Debug
        Else
          Out 4,0                ‘ Eteint la Led 3
        End If
Loop
```

Une fois le programme saisie, alimentez la platine et cliquez sur le menu « Run » (ou sur le petit triangle bleu de la barre d'outils du haut de l'écran). A ce stade, un nom de fichier vous est demandé. Ce dernier correspondra au nom que vous voudrez donner à votre application. Choissez par exemple « MIXTE » et validez. A ce stade, une barre de progression au bas de l'écran doit s'afficher, indiquant ainsi que le module CUBLOC™ est en cours de programmation.

Tournez alors le potentiomètre de gauche de la platine et observez la led 3 ainsi que la fenêtre de Debug du PC. Manipulez les boutons-poussoirs et observez également les Leds comme indiqué dans l'application du programme « LADDER » seul de la page 360.

Interprétation des programmes :
Le programme en « LADDER » n'apporte aucun commentaire puisqu'il fonctionne exactement de la même manière que celui décrit en page 360 (c'est normal… il s'agit su même programme !).

Le programme BASIC commence par la déclaration du type de module CUBLOC™ utilisé, suivi de la déclaration des ports utilisés par la partie « LADDER » du programme. Au passage on remarquera que l'on déclare ici à la fois le N° du port, sa fonction (entrée ou sortie) et le nom de l'Alias qui lui sera associé sur une seule ligne. Cette façon de déclarer est plus compacte que celle qui est générée par l'utilitaire de configuration automatique du « CUBLOC Studio ».
Le programme BASIC s'intègre ensuite dans une boucle sans fin de type Do… Loop dans laquelle on fait l'acquisition de la valeur analogique présente sur le port P5 (qui aura au préalable été déclaré en tant que port d'entrée via la commande **Input 5**). Une commande conditionnelle test ensuite si la valeur lue est supérieure à 500. Si tel est le cas, la Led 3 est allumée et on affichage la valeur décimale de la valeur lue dans la fenêtre de Debug du PC. Si la valeur est inférieure à 500, la Led 3 s'éteint et les valeurs ne s'affichent plus dans la fenêtre de Debug du PC.

Vous pourrez ainsi constater que le programme BASIC et LADDER s'exécutent simultanément en mode multitâche. A noter que si vous désirez programmer à nouveau votre module CUBLOC™, il vous faudra tourner le potentiomètre de telle sorte qu'aucune valeur ne s'affiche

dans la fenêtre de DEBUG et que le port série de programmation du CUBLOC™ soit « libre » (voir explications en page 58).

Enfin, vous pourrez également constater qu'il n'est pas possible d'activer les modes « Monitor » ou « Time Chart Monitor…» du LADDER pour la simple et bonne raison que ces modes partages les ressources du port série de programmation avec le mode Debug du BASIC. Si nécessaire, retirez la ligne de la commande Debug dans le programme BASIC pour avoir de nouveau accès aux modes « Monitor » ou « Time Chart Monitor…» du LADDER.

Consultez le CD-ROM des CUBLOC™ ainsi que le site : www.lextronic.fr et www.cubloc.com pour télécharger de très nombreuses notes d'applications sur les modules CUBLOC™.

Chapitre 12.

F.A.Q CUBLOC™

Vous trouverez ci-après une liste de questions/réponses vous permettant de résoudre la plupart des cas de problèmes de fonctionnement que vous pourrez être amené à rencontrer lors des premières utilisations de vos modules « CUBLOC™ ».

Q : Un message du type « Invalid RS232C Port ! ! ! » s'affiche lorsque j'essai de programmer mon module CUBLOC™ ?

R : Vous avez probablement mal configuré le N° du port série reliant le PC au CUBLOC™ ou le câble de programmation n'est pas relié au PC et/ou au CUBLOC™. Si vous utilisez un câble USB, vérifiez que ce dernier a été correctement détecté (déconnectez-le et reconnectez si nécessaire sur le port USB). Vérifiez également que votre système d'exploitation soit Windows XP™ (édition familiale) – Les autres versions ne sont pas compatibles.

Q : Un message d'erreur s'affiche lorsque j'essai de programmer mon module CUBLOC™ ?

R : Vous avez probablement utilisé des commandes Debug qui « monopolisent » l'usage du port série de programmation du module CUBLOC™. Consultez le bas de la page 263 pour comprendre ce qui se passe et y remédier.

Q : Je n'arrive toujours pas (ou plus) à « dialoguer » avec le module CUBLOC™ ?

R : Votre version du « CUBLOC Studio » est probablement plus récente (ou plus ancienne) que le « Firmware » du module CUBLOC™.

Téléchargez et installez la dernière version à jour du logiciel de développement des CUBLOC™ (CUBLOC Studio) sur le site Internet : www.comfiletech.com (rubrique Download) – Contactez-nous si vous ne disposez pas d'une connexion Internet – Ce logiciel est peut être disponible sur le CD joint avec les CUBLOC™. Une fois le logiciel installé, remettez à jour le « Firmware » du CUBLOC™ via le menu « Setup », puis « Firmware download ». Le programme vous demande si vous être prêt à réaliser la mise à jour, validez par « Oui ». Une barre de progression s'affiche alors au bas à gauche de l'écran (l'opération peut prendre plusieurs minutes).

Ne coupez PAS l'alimentation de la platine et ne retirez pas la liaison série entre le PC et le module CUBLOC™ pendant cette opération. Laissez cette opération aller jusqu'à terme.

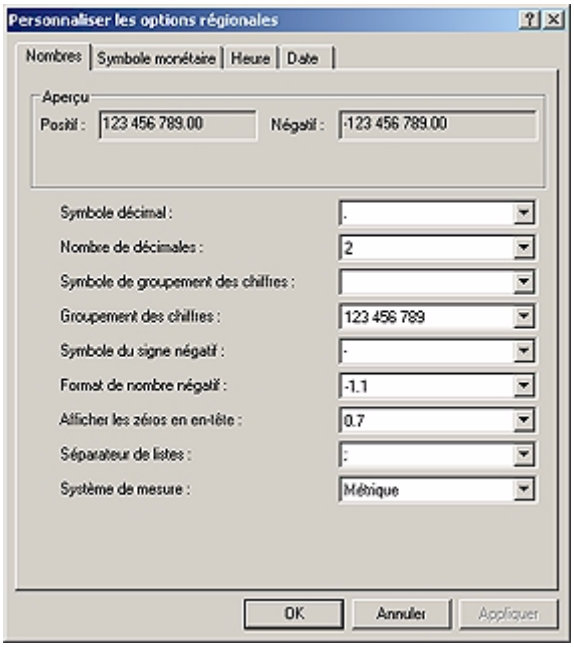
ATTENTION ! Cette opération efface tout programme potentiellement présent dans votre module CUBLOC™.

Une fois l'opération terminée, essayer à nouveau de transférer votre programme au sein de votre module CUBLOC™.

Q : Je n'arrive pas à utiliser les fonctions mathématiques de mes CUBLOC™ ?
(les résultats semblent erronés)

R : Avant de pouvoir utiliser correctement les fonctions mathématiques des CUBLOC™, il vous faut impérativement (sous votre système d'exploitation Windows XP™) sélectionner le menu « Démarrer » -> « Paramètres » - > « panneau de configuration ».

Réalisez alors un « double click » sur l'icône « Options régionales… ».



Cliquez ensuite sur le bouton « Personnaliser… » et sélectionnez le « . » (point) comme symbole décimal. Puis validez les modifications en cliquant à chaque fois sur le bouton « OK » pour referner les fenêtres.

Si vous ne procédez pas à cette modification, vous ne pourrez pas utiliser correctement les fonctions et calculs mathématiques des CUBLOC™ (les résultats retournés seront erronés).

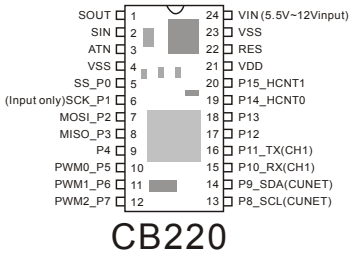
Chapitre 13.

Appendice CUBLOC™

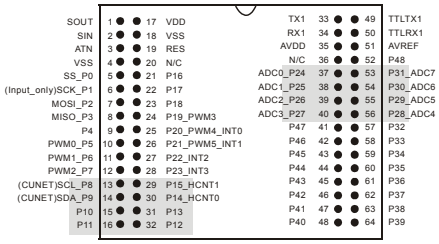
Table de codes « ASCII »

Code	char.	Code	char.	Code	char.	Code	char.
00H	NUL	20H	SPACE	40H	@	60H	`
01H	SOH	21H	!	41H	A	61H	a
02H	STX	22H	“	42H	B	62H	b
03H	ETX	23H	#	43H	C	63H	c
04H	EOT	24H	$	44H	D	64H	d
05H	ENQ	25H	%	45H	E	65H	e
06H	ACK	26H	&	46H	F	66H	f
07H	BEL	27H	‘	47H	G	67H	g
08H	BS	28H	(	48H	H	68H	h
09H	HT	29H	)	49H	I	69H	I
0AH	LF	2AH	*	4AH	J	6AH	j
0BH	VT	2BH	+	4BH	K	6BH	k
0CH	FF	2CH	,	4CH	L	6CH	l
0DH	CR	2DH	-	4DH	M	6DH	m
0EH	SO	2EH	.	4EH	N	6EH	n
0FH	SI	2FH	/	4FH	O	6FH	o
10H	DLE	30H	0	50H	P	70H	p
11H	DC1	31H	1	51H	Q	71H	q
12H	DC2	32H	2	52H	R	72H	r
13H	DC3	33H	3	53H	S	73H	s
14H	DC4	34H	4	54H	T	74H	t
15H	NAK	35H	5	55H	U	75H	u
16H	SYN	36H	6	56H	V	76H	v
17H	ETB	37H	7	57H	W	77H	w
18H	CAN	38H	8	58H	X	78H	x
19H	EM	39H	9	59H	Y	79H	y
1AH	SUB	3AH	:	5AH	Z	7AH	z
1BH	ESC	3BH	;	5BH	[	7BH	{
1CH	FS	3CH	<	5CH	\	7CH	|
1DH	GS	3DH	=	5DH	]	7DH	}
1EH	RS	3EH	>	5EH	^	7EH	~
1FH	US	3FH	?	5FH	_	7FH	DEL

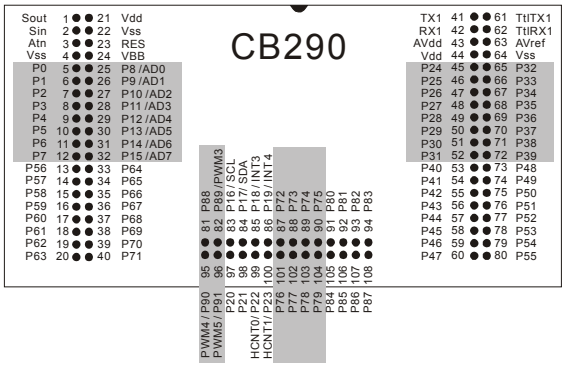
Rappel du brochage des modules « CUBLOC™ »

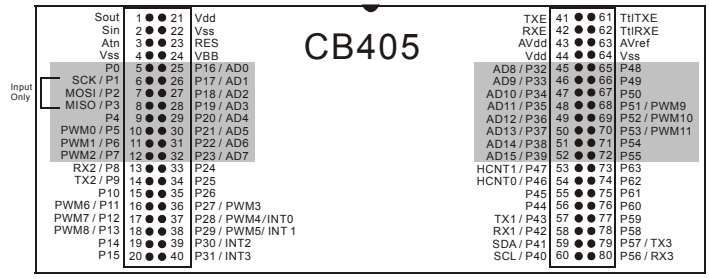


CB220



CB280





Les informations présentes dans ce manuel sont données à titre indicatif. Les caractéristiques techniques et possibilités des modules CUBLOC™ ainsi que de leur logiciel de développement « CUBLOC STUDIO » peuvent changer à tout moment sans aucun préavis dans le but d'améliorer les possibilités de ces produits.