

# Aide ICNCStudio

## Table des matières

---

Interpreteur_PLC_Basic .....	6
Introduction .....	7
Généralités .....	8
Gestion des erreurs .....	9
Guide d'utilisation rapide .....	10
LES FONCTIONS .....	11
Fonctions PLC Basic spécifiques à l' InterpCNC .....	12
Accès aux Registres et Bits utilisateur en Lecture .....	13
Accès aux Registres et Bits système en Lecture seule .....	15
Variantes pour l'accès aux registres en lecture seule (Input Registers, V3) ...	16
Fonctions effectuant des calculs .....	18
Fonctions liées aux Mouvements d'Axes .....	18
Fonctions liées aux Timers .....	18
Fonctions liées aux Entrées et Sorties .....	19
Fonctions pour la détection de fronts sur Entrées ou Bits utilisateurs .....	20
Fonctions pour la manipulation des chaînes de caractères .....	21
Fonctions de gestion des interruptions et des signaux périodiques (V3) .....	25
LES COMMANDES .....	26
Commandes issues du Basic standard .....	26
Commandes PLC Basic spécifiques à l' InterpCNC .....	26
Accès aux Registres et Bits utilisateur en Ecriture (Holding Registers et Coils	28
Commandes spécifiques au Hardware de la carte InterpCNC .....	32
Commandes de gestion du programme .....	32
Commandes liées à la gestion de Recettes .....	33
Commandes liées aux Mouvements d'Axes .....	33
Commandes liées aux Timers .....	35
Commandes liées aux Entrées et Sorties .....	36
Commandes de gestion des interruptions et des signaux périodiques (V3) .....	38
CREATION D'UN GRAFCET AVEC L'INSTRUCTION « Select Case » .....	40
GESTION DES INTERRUPTIONS .....	41
UTILISATION DES ENTRÉES RAPIDES .....	44
COMMUNICATION VIA MODBUS AVEC DES DRIVERS OU VARIATEURS (V3) ...	47
COMMUNICATION_PAR_ETHERNET_V3 .....	49
UTILISATION EN MODE DMX .....	51
UTILISATION_DE_L'HORLOGE_RTC_V3 .....	56
ICNCStudio .....	59
Registres_Utilisateur .....	60
Registres_Sauvegardes .....	62
Fenêtres_axes .....	64
Moniteur .....	65
Digital_inputs .....	65
Digital_outputs .....	67
Coils .....	68
Editeur_de_texte .....	70
Chercher_Remplacer .....	73
Main .....	74
Tableau_des_parametres .....	77

Recettes .....	79
Variables_PLC .....	82
Donnees_personnalisees .....	83
Graphique_V3 .....	85
Analogique_compteurs .....	85
Firmware_update .....	87
ICNCStudio_update .....	89
Cryptage .....	89
Parametres .....	91
Configuration_generale .....	92
Axes .....	93
DIN .....	95
AIN .....	101
IN_ENA .....	102
Port_serie .....	104
Polling .....	105
Ethernet .....	105
Commande_numerique .....	106
Plasma_thc .....	106
Notice de l'InterpCNC V3 .....	109
Presentation .....	110
Installation .....	112
Raccordement .....	113
Notice de l'InterpCNC V2 .....	115
Présentation .....	116
Installation .....	123
Raccordement .....	125
Documentation_MODBUS_V3 .....	131
Introduction .....	132
Identification des PLCs InterpCNC connectés au réseau Ethernet.....	133
Identification des PLCs InterpCNC connectés en USB .....	133
Lecture / écriture des paramètres .....	134
Adresses des Bits en lecture seule (Input Bits) .....	134
Adresses des Registres en Lecture seule (Input registers) .....	136
Registres en Lecture/Écriture (Holding registers) .....	141
Généralités sur l'envoi des commandes Modbus .....	142
Commande 100 : Arrêt d'un axe .....	143
Commande 101 : Arrêt d'un ou plusieurs axes .....	143
Commande 102 : Déplacement d'un axe à une vitesse donnée .....	143
Commande 103 : Déplacement d'un axe vers une position cible .....	144
Commande 104 : Déplacement d'un axe du nombre de pas indiqué par rapport à la position actuelle .....	144
Commande 105 : Écriture du compteur de position actuelle (revient à écrire dans les registres de positions) .....	144
Commande 106 : Lancement du Homing d'un axe .....	145
Commande 107 : Lancement d'un Palpage sur une entrée .....	146
Commande 108 : Lancement d'un Palpage sur plusieurs entrées .....	147
Commande 109 : Lancement d'un Palpage sur seuil d'entrée analogique .....	147
Commande 110 : Forçage des entrées .....	150
Commande 200 : PLCBasic command .....	150

Utilisation des lectures/écritures indexées .....	150
Fonctions dédiées au pilotage CNC .....	152
1° partie : commandes bufferisées .....	152
Commande 1000 : Exécution d'une instruction Gcode .....	153
Commande 1001 : Définition de la vitesse d'usinage en mm/mn .....	154
Commande 1002 : Déplacement linéaire interpolé des axes vers des positions cibles (positions absolues) .....	154
Commande 1003 : Interpolation circulaire .....	155
Commande 1010: Action synchronisée (Sortie TOR, Sortie analogique, Registre) .....	155
Commande 1011 : Temporisation bufferisée .....	157
Commande 1012 : Attente d'un état ou d'un événement .....	157
2° partie : Commandes non bufferisées .....	159
Commande 1100 : Modification override Usinage et déplacement rapide ..	159
Commande 1101 : Pause de l'usinage en cours .....	159
Commande 1102 : Reprise d'un usinage interrompu .....	160
Commande 1110 : Exécution d'une séquence de homing machine .....	160
Commande 1111 : Exécution d'un déplacement manuel (Jog) .....	160
Commande 1200 : Exécution directe d'une commande .....	161
Utilisation de l'horloge interne RTC .....	162
Commande 112 : Réglage de la date sur l'horloge RTC .....	163
Commande 113 : Réglage de l'heure sur l'horloge RTC .....	163
Commande 114 : Réglage simultané date et heure sur l'horloge RTC .....	164
Documentation_MODBUS_V2 .....	164
Introduction .....	165
Lecture / écriture des paramètres .....	166
Adresses des Bits en lecture seule (Input Bits) .....	166
Adresses des bits en lecture / Écriture .....	168
Adresses des Registres en Lecture seule (Input registers) .....	168
Registres en Lecture/Écriture (Holding registers) .....	170
Généralités sur l'envoi des commandes Modbus .....	172
Commande 66 : Reset et Ré-armement .....	172
Commande 80 : Déplacement indépendant d'un axe .....	173
Commande 6 : Déplacement interpolés des axes avec profile de vitesse .....	174
Commande 71 : Prise d'origine des axes (ICNC_CMD_MACHINE_HOME) ....	174
Commande 78 : Prise d'origine d'un axe (ICNC_CMD_HOME_AXE) .....	175
Commande 40 : Déplacement jusqu'à un capteur (fonction de palpape ICNC_CMD_PROBE) .....	176
Commande 67 : Écriture de la position des axes (ICNC_CMD_WRITE_POSITION) 177	
Commande 96 : Changement vitesse d'un axe (ICNC_CMD_CHANGE_SPEED) .....	178
Commande 42 : Arrêt avec rampe des axes (ICNC_CMD_BREAKE_AXES_AND_CLEAR) .....	179
Commande 43: Arrêt immédiat sans rampe (ICNC_CMD_STOP) .....	179
Commande 41: Arrêt avec rampe d'un axe (ICNC_CMD_BREAKE_AXES) .....	179
Commande 13: Définition état sortie OUT bufferisée (ICNC_CMD_SET_OUTPUT_BUF) .....	179
Commande 14: Définition de l'état de toutes les sorties OUT bufferisée (ICNC_CMD_SET_OUTPUT_ALL_BUF) .....	180
Commande 15 : Définition de l'état d'une sortie analogique bufferisée	

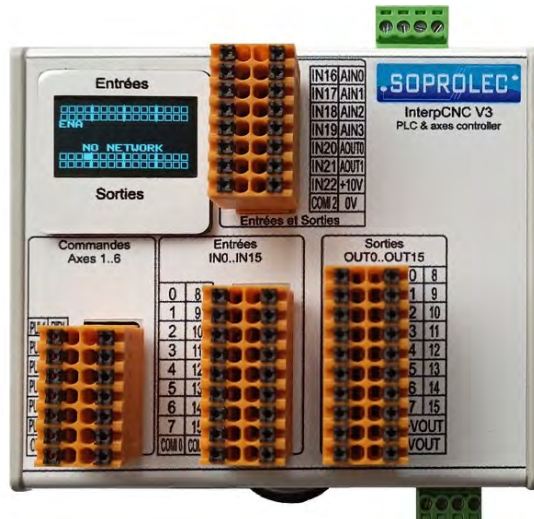
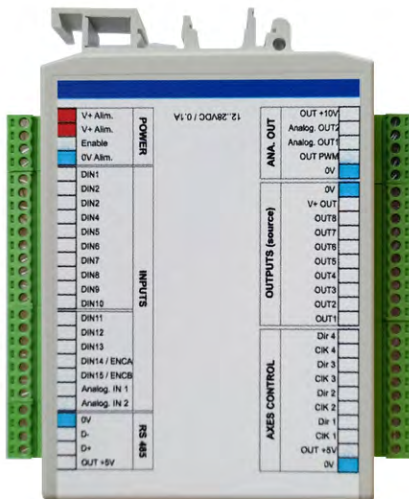
(ICNC_CMD_SET_ANALOG_BUF) .....	180
Commande 20 : Temporisation traitement buffer (ICNC_CMD_TEMPO_BUF)	180
Traitement des erreurs et diagnostic .....	180
Configurations_speciales_V3 .....	181
Protection par mot de passe .....	181
Module d'extension I/O .....	183
Les exportations et importations (V2 et V3) .....	184

## Interpreteur\_PLC\_Basic

**SOPROLEC**  
 ZAC DE L'EPINE  
 72460 SAVIGNE L'EVEQUE  
 Tél : +33 (0)2 4376 4476



## Cartes d'axes SOPROLEC InterpCNC V2 et V3



# Interpréteur langage PLC Basic intégré

## Introduction

Cette documentation traite à la fois des cartes **SOPROLEC InterpCNC V2** (V2.2B et V2.4) et **V3**.

Toutes deux sont équipées d'un processeur 32 bits :

- PIC32 pour les cartes **V2**
- STM32 H7 pour les cartes **V3**

C'est pourquoi, cette documentation étant commune, dans chaque rubriques certaines explications seront nuancées par les annotations: (**V2**) ou (**V3**).

Les cartes InterpCNC **V2** et **V3** disposent d'un puissant interpréteur de langage PLC Basic intégré. Cet interpréteur permet de développer des applications d'automatisme autonomes ou en association avec une interface homme/machine (IHM).

Dans l'ensemble, l'utilisation de l'interpréteur Basic est presque identique pour ces deux générations de cartes, ce qui a permis de concevoir un environnement de développement (IDE) commun: **lcncStudio**.

Toutefois, les différences hardware de chaque carte imposent 2 versions différentes de cet IDE:

- **lcncStudio** pour la **V3**
- **lcncStudio Light** pour la **V2**

L'interpréteur PLC Basic fonctionne en parallèle des autres fonctions de la carte. Il est donc possible d'utiliser les carte InterpCNC dans des applications de commandes numériques pilotées par un PC tout en exécutant le programme Basic pour des traitements d'actions particulières (par exemple, pupitre déporté de contrôle manuel de la machine).

Cet interpréteur travaille avec des variables qui peuvent être des chaînes de caractères ou des nombres réels.

Toutes les valeurs numériques sont de type réel codés sur 32 bits simple précision. La valeur maximale est de 17549435e-38 et la valeur minimale est de 3.40282347e+38.

Les nombres entiers pouvant être manipulés sans perte de précision doivent être compris dans l'intervalle de  $\pm 16777100$ .

L'interpréteur PLC Basic des cartes InterpCNC en fait l'une des cartes les plus accessibles sur le marché en termes de facilité de programmation.

## Généralités

D'un point de vue matériel, l'**InterpCNC V3** est une carte Automate dont les principales caractéristiques sont :

- Commande de 6 axes (Pulses / direction)
- 16 entrées compatibles NPN et PNP
- 16 sorties PNP 500 mA
- 4 entrées Analogiques 0 à 10V
- 2 sorties Analogiques 0 à 10V
- 7 entrées rapides
- Connectivité Ethernet, USB, et 2 x RS485
- 1 Ecran OLED 0,96"
- Microcontrôleur 32 bits
- Tension d'alimentation : 24V

L'**InterpCNC V2** se décline en 2 versions:

### 1) V2.2B

- Commande de 5 axes (Pulses / direction), sorties CK1 à CK5, D1 à D5 :  
Sortie TTL 5V/20mA maxi
- Sorties 1 à 4 : Relais à contact sec 250V/10A
- Sorties 5 à 12 : PNP 350mA (total de toutes les sorties)/24V
- Entrées 1 à 16 : 0 à 32V maxi. Niveau haut à partir de 3,5V. Filtre 1,5KHz.
- Entrée ENABLE : Fonction d'arrêt d'urgence. 0 à 32V maxi. Niveau haut à partir de 3,5V.
- 1 entrée codeur incrémental ou 2 entrées de comptage rapide de type TTL
- 4 Entrées analogiques AI1 à AI4 : 0 à 10V. Impédance 1M, Résolution 10 bits.
- 2 Sorties analogiques AO1 et AO2 : 0 à 10V. Résolution 10 bits.
- Tension d'alimentation : 24V

### 2) V2.4

- Commande de 4 axes (Pulses / direction), sorties CK1 à CK4, D1 à D4 :  
Sortie TTL 5V/20mA maxi
- Sorties 1 à 8 : PNP 350mA (total de toutes les sorties)/24V
- Entrées 1 à 13 : 0 à 32V maxi. Niveau haut à partir de 3,5V. Filtre 1,5 KHz
- Entrée ENABLE : Fonction d'arrêt d'urgence. 0 à 32V maxi. Niveau haut à partir de 3,5V.
- 1 entrée codeur incrémental ou 1 entrée de comptage rapide de type TTL
- 2 Entrées analogiques AI1 et AI2: 0 à 10V. Impédance 1M, résolution 10 bits.
- 2 Sorties analogiques AO1 et AO2 : 0 à 10V. Résolution 10 bits.
- 1 sortie analogique PWM (AO3) : 0 à 5V TTL (fréquence 10 KHz)
- Tension d'alimentation : 24V

Tous les connecteurs sont débrochables pour une mise en oeuvre simplifiée.

L'**InterpCNC V3** communique via une connexion Ethernet (protocole Modbus TCP ou Modbus UDP), ou Serie (RS485, protocole Modbus RTU), ou USB (port Com

virtuel, protocole Modbus RTU).

L'**InterpCNC V2** communique via une connexion serie (RS485, protocole Modbus RTU), ou USB (port Com virtuel, protocole Modbus RTU).

## Gestion des erreurs

S'agissant d'un langage interprété, les erreurs dans le programme sont détectées en cours de fonctionnement.

Il est donc important de pouvoir mettre en place un gestionnaire d'erreur pour interrompre d'éventuels mouvements lorsqu'une erreur survient.

L'interpréteur PLC Basic réalisera automatiquement un GoTo au label **OnError**: si une erreur d'exécution se présente.

Si ce label n'est pas présent dans le programme, aucun traitement d'erreur particulier ne sera réalisé.

Dans l'exemple qui suit, le programme principal est placé dans une boucle DO ... LOOP.

Si une erreur de traitement, survient, il y aura un GOTO automatique au label **OnError**:

Dans le traitement de l'erreur, on arrête tous les éventuels déplacements en cours et on désactive toutes les sorties. Le programme est ensuite interrompu.

Il est bien entendu possible d'ajouter dans le traitement d'erreur un saut de type GOTO pour retourner au début du programme ou reprendre l'exécution à un label particulier.

```
' Programme principal
Do
  ' Code de l'application
Loop

' Traitement d'erreur
OnError:
  StopAxes &H3F ' Arrêt de tous les axes (V3) / StopAxes &H1F (V2)
  OUTPort 0, 0 'Mise à 0 des sorties 0 à 7 (V3)
  OUTPort 1, 0 'Mise à 0 des sorties 8 à 15 (V3) / OutAll 0 (V2)
```

Vous pouvez donc mettre en place un tel gestionnaire dans votre programme PLC Basic à l'aide du label réservé **OnError**.

## Pour la carte V3 uniquement:

De la même manière, le label **OnStopPLC**: définit le code à exécuter dans le cas où le programme s'arrête.

Il peut être amené à s'arrêter soit par une commande **STOP** prévue dans son exécution, soit par une action volontaire (clic sur le voyant vert dans ICNCStudio).

Exemple:

```
OnStopPLC:
  ' Arrêt des interruptions
  SetTick 1, 0, OnTimerEtq1
  SetInputINT 4, -1, INPUT_INT_DFM_ONESHOT, OnCellObjet
```

```
' Arrêt de tous les Axes
SetAna OUT_ANA_VITESSE, 0
SetAna OUT_ANA_COUPLE, 0
StopAxes &h3F

? "OnStopPLC"
end
```

On peut exécuter la même portion de code dans les 2 cas, en cumulant **OnStopPLC**: et **OnError**: en un label commun

```
OnStopPLC:
OnError:
...
...
...
end
```

Le **end** n'est nécessaire que s'il y a encore du code après.

## Guide d'utilisation rapide

L'interpréteur PLC Basic de l'InterpCNC reprend bien entendu l'ensemble des expressions, instructions, opérateurs, commandes et fonctions, ainsi que les fonctions de manipulation des chaînes de caractères, existant de base dans tous les langages Basic.

Grands classiques de la programmation, leur explication ne sera reprise que pour quelques unes dans ce manuel.

Le tableau ci-dessous en fait la synthèse:

<p><b>Expressions Littérales / Variables Utilisateur</b></p> <p><b>Expression Littérales</b> Les chaînes de caractères sont contenues dans des guillemets, ex : "InterpCNC". Les nombres peuvent être décimaux ou représentés par: &amp;Hnn Hex Literal, ex : &amp;H3C (60) &amp;Bnn... Binaire Literal, ex : &amp;B00100011 (35) n.nE+n Scientifique, e.g. 1.6E+4 (16000)</p> <p><b>Variables Utilisateur</b> Les noms de variables commencent par un caractère</p>	<p><b>Formatage de chaînes</b></p> <p><b>% [flags] [width] [.prec] type</b> <b>flags</b>: - Justifie à gauche <b>0</b> Utilise 0 comme caractère de décalage (pas Espace). + Le signe + désigne des valeurs positives. <b>space</b> Espace comme signe, sauf si négatif.</p> <p><b>width</b>: nombre minimal de caractères en sortie, moins cause du décalage, plus cause de l'expansion. <b>.prec</b>: nombre de chiffres de fraction pour les types e, ou f, ou le max. de chiffres significatifs pour le type g. Doit être précédé par un</p>	<p>SELECT CASE Variable CASE Value CASE ValueFrom TO ValueTo CASE Value IS &lt; Limite CASE ELSE END SELECT</p> <p><b>Chaînes / Caractères</b> ASC (str\$) CHR\$ (nbr) FORMAT\$ (nbr [,format\$]) INSTR ([start,] search\$, pattern\$) LEFT\$ (str\$, nbr) LEN (str\$) LCASE\$ (str\$) MID\$ (str\$, start</p>
--	--	--

<p>alphanumérique ou un underscore et peuvent contenir n'importe quel caractère alpha ou numérique, point (.) et underscore (_); la longueur maximum est de 32 caractères. Les noms de chaînes de caractères sont terminés par le symbole. Les noms des variables numériques ne sont pas terminés par le symbole \$.</p> <h3>Opérateurs</h3> <h4>Arithmétiques</h4> <p>^ * / Exponentiation, Multiplication, Division MOD \ Modulus (reste), Division entière + + - Addition, Concaténation de chaîne, Soustraction</p> <h4>Logiques</h4> <p>NOT Inverse logique = &lt;&gt; Égalité, Inégalité &gt; &lt; Plus grand que, plus petit que &lt;= or =&lt; Inférieur ou égal à &gt;= or =&gt; Supérieur ou égal à AND OR Conjonction, Disjonction XOR Ou Exclusif</p>	<p>point(.) si utilisé. <b>Type: g ou G</b> format pour la meilleure présentation. <b>f ou F</b> Format décimal avec point décimal et chiffres <b>e ou E</b> Format exponentiel</p> <h3>Commandes / Déclarations</h3> <p>Déclaration de tableau : DIM variable(éléments...)</p> <h3>Contrôle d'exécution</h3> <p>CONTINUE DO &lt;déclarations&gt; LOOP DO WHILE expression &lt;déclarations&gt; LOOP DO &lt;déclarations&gt; LOOP UNTIL expression ELSE ELSEIF expression THEN ENDIF END EXIT EXIT FOR FOR compteur = début TO fin [STEP increment] GOSUB GOTO IF expression THEN IRETURN NEXT [compteur-variable] [, compteur-variable]... RUN STOP LIST NEW</p>	<p>[,nbr]) RIGHT\$ (str\$, nbr) SPACE\$ (nbr) SPC (nbr) STRING\$(nbr, val str\$) TAB(nbr) UCASE\$ (str\$) VAL (str\$) INKEY\$</p> <h3>Fonctions</h3> <h3>Maths / Nombres</h3> <p>ABS (nbr) ATN (nbr) CINT (nbr) COS (nbr) EXP (nbr) FIX (nbr) HEX\$ (nbr) INT (nbr) LOG (nbr) OCT\$ (nbr) RND (nbr) SGN (nbr) SIN (nbr) SQR (nbr) STR\$ (nbr)</p>
---	---	---

## LES FONCTIONS

Une **fonction** se différencie d'une commande en ce sens qu'elle peut retourner une valeur, un résultat.

Pour l'utiliser, tout comme pour une commande on lui passe en paramètres des arguments, dont le nombre est autant que de besoin.

Comme dans la plupart des langages de programmation, vous pouvez vous-même créer vos propres fonctions, à l'aide des instructions **function** et **end function**.

Pour se faire, la syntaxe est la suivante:

**function** NomFonction(Param1,Param2,Param3,...)

...  
... *' ici lignes de code utilisant les valeurs contenues dans les variables passées en paramètres*  
...

```

if ... then Retour=1
else Retour=0

```

**NomFonction**=Retour ' *le retour d'une valeur se fait par l'affectation de la variable portant le nom de la fonction*  
end fonction

Pour retourner un résultat nous utilisons le nom de la fonction comme une variable globale, que l'on affectera du résultat calculé ou de la valeur que l'on souhaite retourner.

## Fonctions PLC Basic spécifiques à l' InterpCNC

### Fonctions

GetMB(memo)  
GetMBit(memo)  
GetMW(adr)  
GetMDW(adr)  
GetMDWInv(adr)  
GetMI(adr)  
GetMDI(adr)  
GetMDIInv(adr)  
GetME(adr)  
GetMFIInv(adr)  
GetMBReg(adr)  
GetEEData8(adr)  
GetEEData16(adr)  
GetEEData32(adr)  
GetEEDataFloat(adr)  
GetUserMem(nbr, nbr)  
GetUserMemF((nbr, nbr)  
GetStr\$(adr, nbr)

StsBit(BitNbr)  
GetInputMB(BitNbr)  
GetInputMW(adr)  
GetPrm(nbr)

IsEEdataChanged  
IsRPCChanged  
IsMBPrmChanged

Min(nbr, nbr)  
Max(nbr, nbr)  
Limit(nbr, Min, Max)

GetPos(nbr)  
GetCapturePos(nbr)

GetTimer (str)  
Toc(nbr)  
Cyclestat(nbr)

**Timer**  
**Time\$**

**IN(nbr)**  
**GetInAll**  
**Ain(nbr)**  
**AinV(nbr)**  
**Out(nbr)**  
**GetOut**  
**GetOutAll**

**GetEncoder(nbr)**  
**GetCnt(nbr)**

**DF(nbr)**  
**DFM(nbr)**  
**DFD(nbr)**

**DFMBit(nbr, nbr)**  
**DFDBit(nbr, nbr)**

Accès aux Registres et Bits utilisateur en Lecture

## **GetMB(Memo)**

ou

## **GetMB(Registre, Bit)**

Lecture d'un Memo bit(Coil) dans l'espace des Memos (Coils utilisateurs), ou un bit particulier dans le domaine des registres.

Exemple :

if **GetMB**(*MBB\_ONOFF\_CONVOYEUR*) then ...

*'si le bit ON/OFF du convoyeur est à 1, alors...*

if **GetMB**(3100, 15) then...

*'si le 15ème bit de l'adresse 3100 (U16, ici en Ram) est à 1, alors...*

if **GetMB**(3100, 31) then...

## **GetMBit(Memo) -> cartes V2 uniquement**

Fonction devenue obsolète car remplacée depuis le firmware V5.39, par :

GetMB(Memo)

**NB: Les commandes et fonctions obsolètes** sont maintenues uniquement pour assurer la compatibilité des cartes avec d'anciens programmes. Il est toutefois recommandé de ne plus les utiliser.

## **GetMW(Registre)**

Lecture d'un registre 16 bit non signé (U16)

Exemple : ResolutionAxe2 = **GetMW**(EE\_RESO\_AXE2) /10

*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (U16) en EEprom*

## **GetMDW(Registre)**

Lecture d'un registre 32 bits non signé (U32)

Exemple :CompteurTotal = **GetMDW**(EE\_CPT\_TOTAL)*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (U32) en EEprom***GetMDWInv(Registre)**-> **carte V3** uniquement, firmware >=V1.41

Lecture d'un registre 32 bits non signé (U32), pour lequel les 2 registres 16 bits utilisés sont inversés

(selectionner U32\_INV pour l'affichage)

#	Value	Name	Type
3000			U16
3001			U16
3002			I16
3003			U32
3004			I32
3004			FLOAT
3005			U32_INV
			I32_INV
3006			FLOAT_INV

**GetMI(Registre)**

Lecture d'un registre 16 bits signé (I16)

Exemple :Offset = **GetMI**(RCP\_OFFSET\_BOUTEILLE) \*ResOrienteur/360**GetMDI(Registre)**

Lecture d'un registre 32 bits signé (I32)

Exemple :Cible = **GetMDI**(POSITION\_SPOT) + Offset**GetMDIInv(Registre)**-> **carte V3** uniquement, firmware >=V1.41

Lecture d'un registre 32 bits signé (I32), pour lequel les 2 registres 16 bits utilisés sont inversés

(selectionner I32\_INV pour l'affichage)

**GetMF(Registre)**

Lecture d'un registre 32 bits traité comme un Float (FLOAT)

Exemple :ResolConvoyeur = **GetMF**(EE\_RES\_CONVOYEUR)*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (FLOAT) en EEprom***GetMFInv(Registre)**-> **carte V3** uniquement, firmware >=V1.41

Lecture d'un registre 32 bits traité comme un Float (FLOAT) pour lequel les 2 registres 16 bits utilisés sont inversés

(selectionner FLOAT\_INV pour l'affichage)

**GetMBReg (Registre)** -> **cartes V2** uniquement

Adresse : 4096 à 6636 Valeur : 0 à 65535

Fonction devenue obsolète car remplacée depuis le firmware V5.39, par :

**GetMW (Registre)**

(voir plus haut).

**GetEEData8(Registre)**-> **cartes V2** uniquement

Registre = 0 à 1023. Fonction obsolète car remplacée par GetMW(Registre) depuis le firmware V5.39

Lecture d'un octet 8 bits dans la zone mémoire sauvegardée.

Exemple :

Valeur = GetEEData8(12)

**GetEEData16(Registre)**-> **cartes V2** uniquement

Registre = 0 à 511. Fonction obsolète car remplacée par GetMW(Registre) depuis le firmware V5.39

Lecture d'un mot 16 bits dans la zone mémoire sauvegardée.

Exemple :

Valeur = GetEEData16(3)

**GetEEData32(Registre)**-> **cartes V2** uniquement

Registre = 0 à 255. Fonction obsolète car remplacée par GetMDW(Registre) depuis le firmware V5.39

Lecture d'un mot 32 bits dans la zone mémoire sauvegardée.

Exemple :

Valeur = GetEEData32(9)

**GetEEDataFloat(Registre)**-> **cartes V2** uniquement

Registre = 0 à 255. Fonction obsolète car remplacée par GetMW(Registre) depuis le firmware V5.39

Lecture d'un nombre réel dans la zone mémoire sauvegardée.

Exemple :

Valeur = GetEEDataFloat(9)

**GetUserMem(Registre)**-> **cartes V2** uniquement

Registre = 0 à 9. Fonction obsolète car remplacée par GetMDW(Registre) depuis le firmware V5.39

Lecture de la valeur présente en mémoire utilisateur.

La lecture considère que la valeur est un entier 32 bits.

Voir SetUserMem pour plus de détail.

**GetUserMemF(Registre)**-> **cartes V2** uniquement

Registre = 0 à 9. Fonction obsolète car remplacée par GetMF(Registre) depuis le firmware V5.39

Lecture de la valeur présente en mémoire utilisateur.

La lecture considère que la valeur est nombre réel.

Voir SetUserMemF pour plus de détail.

### Accès aux Registres et Bits système en Lecture seule

**StsBit(BitNo)** -> **BitNo** : Numéro du bit de status de 0 à 359

Lecture de l'état d'un des bits de registre de la carte.

Exemple :

If not StsBit(256) then...

'Si l'Axe 1 n'est plus en mouvement, alors...

## GetInputMB(BitNo)

Lecture de l'état d'un des bits de registre de la carte (Input Bits, de 0 à 359, en lecture seule). Idem StsBit.

If not **GetInputMB**(256) then...

Exemple :

If not **GetInputMB**(256) then...

*'Si l'Axe 1 n'est plus en mouvement...*

## GetInputMW(RegisterNo)

Lecture de l'état d'un des registres de la carte (Input Registers, de 1000 à 1143, en lecture seule).

Exemple :

FirmVerHigh = **GetInputMW**(1115)

FirmVerLow = **GetInputMW**(1116)

? « Firmware Version : », FirmVerHigh, " ", FirmVerLow

*'Affiche la version du Firmware de la carte dans le Moniteur*

## GetPrm(ParameterNumber) -> ParameterNumber : ID paramètre de 0 à 999

Lecture de la valeur d'un paramètre de la carte en connaissant son identifiant.

Exemple :

? GetPrm (20) *'Affiche dans le moniteur, la valeur du paramètre 20.*

## IsEEdataChanged

Retourne 1 si le contenu de la mémoire utilisateur Sauvegardée a été modifié (NB : l'état de **IsEEdataChanged** est aussitôt remis à zéro après avoir été lu/testé).

Exemple :

If **IsEEdataChanged** then CalculParametres()

## IsRCPCChanged

Retourne 1 si le contenu de la mémoire utilisateur EEPROM dédiée aux recettes, a été modifié (NB : l'état de **IsRCPCChanged** est aussitôt remis à zéro après avoir été lu/testé).

Exemple :

If **IsRCPCChanged** then ? « Recette modifiée ! »

## IsMBPrmChanged

Retourne 1 si le contenu de la mémoire utilisateur EEPROM stockant les paramètres de la carte, a été modifié (NB : l'état de **IsMBPrmChanged** est aussitôt remis à zéro après avoir été lu/testé).

Exemple :

If **IsMBPrmChanged** then CalculParametres()

[Variantes pour l'accès aux registres en lecture seule \(Input Registers, V3\)](#)

## Cette variante d'accès s'applique aux registres système de la carte V3 uniquement:

Pour mémoire, les Input Registers sont situés entre les adresses 1000 et 2512, et entre les adresses 11000 et 12220 pour les buffers (voir la documentation Modbus pour la correspondance de ces registres).

L'utilisation des mêmes fonctions de lecture que pour les Holding Registers est possible, à condition d'ajouter 100000 à l'adresse du registre.

Exemples :

### **GetMW(Registre+100000)**

Lecture d'un Input Register 16 bit non signé (U16)

Exemple :

FirmVerLow = **GetMW**(101115)

FirmVerHigh = **GetMW**(101116)

*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (U16)*

### **GetMDW(Registre+100000)**

Lecture d'un Input Register 32 bits non signé (U32)

Exemple :

NbFramesDMXRecues = **GetMDW**(101998)

*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (U32)*

### **GetMI(Registre+100000)**

Lecture d'un Input Register 16 bits signé (I16)

Exemple :

I16Value = **GetMI**(1xxxxx)

*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (I16)*

### **GetMDI(Registre+100000)**

Lecture d'un registre 32 bits signé (I32)

Exemple :

PositionAxe1 = **GetMDI**(101030)

*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (I32)*

### **GetMF(Registre+100000)**

Lecture d'un Input Register 32 bits traité comme un Float (FLOAT)

Exemple :

FloatValue = **GetMF**(1xxxxx)

*'Affectation dans une variable, de la valeur stockée dans le registre correspondant (FLOAT)*

**NB :** GetInputMW(1115) est équivalent à GetMW(101115) mais dans un souci d'uniformité avec les autres commandes, il est préférable d'utiliser GetMW(Registre+100000)

## Fonctions effectuant des calculs

### **Min(Valeur1, Valeur2)**

Retourne la valeur minimale entre les valeurs (ou variables) Valeur1 et Valeur 2.

Attention : Valeur 1 et Valeur 2 sont traités comme des entiers.

Exemple :

TensionMin = **Min**(AinV(1), AinV(2))

*'TensionMin relève la plus petite valeur (tension) sur 2 entrées analogiques*

### **Max(Valeur1, Valeur2)**

Retourne la valeur maximale entre les valeurs (ou variables) Valeur1 et Valeur 2.

Attention : Valeur 1 et Valeur 2 sont traités comme des entiers.

Exemple :

TensionMax = **Max**(Ain(3), Ain(4))

*'TensionMax relève la plus grande valeur (points) sur les 2 entrées analogiques*

### **Limit(Valeur, Mini, Maxi)**

Retourne Valeur, bornée entre Mini et Maxi. Très utile pour tester une variable en évitant une succession de « if... then... else... »

Exemple :

**NewPosition = Limit(PositionActuelle, 10000, 30000)**

*'Quelle que soit la valeur de PositionActuelle, NewPosition restera limitée entre 10000 et 30000*

## Fonctions liées aux Mouvements d'Axes

### **GetPos(AxeNumber) -> AxeNumber** : Numéro de l'axe, 1 à 6

Lecture du compteur de position d'un axe. Le retour est un entier 32 bits signé.

Exemple :

PositionAxe\_1 = **GetPos**(1)

## Fonctions liées aux Timers

### **GetTimer(VariableTimer)**

Lecture d'un timer initialisé avec SetTimer. Retourne le temps restant sur une temporisation initialisée avec la commande SetTimer. Lorsque le timer est écoulé, cette fonction retourne 0.

Exemple :

SetTimer Tempo1, 100

if **GetTimer**(Tempo1) then

out 1,1

endif

## **Toc**(numéro instance de 0 à 15)

-> carte **V3** uniquement

Retourne la période de temps écoulée depuis la commande Tic (en micro-secondes)

Exemple :  
? **Toc**(1)

## **Cyclestat**(numéro instance de 0 à 15)

-> carte **V3** uniquement

Mise à jour de l'instance de mesure de temps.  
Sert à mesurer un temps de cycle, valeurs à récupérer aux 3 registres MW définis lors de la commande [CyclestatInit](#).  
Les temps sont donnés en 1/10 de ms

Exemple :  
**CycleStat**(1)  
*'Actualise les stats dans les 3 registres désignés*

## **Timer**

Retourne le temps écoulé (en ms) depuis la mise sous tension de la carte.

Exemple :  
Print Timer/1000 ; " secondes écoulées "

## **Time\$**

Retourne sous la forme d'une chaîne de caractères, le temps écoulé depuis la mise sous tension.

Format : hh:mn:s,ms

Exemple :  
? « Dernière mise en marche à : », Time\$

## **Fonctions liées aux Entrées et Sorties**

### **IN**(InputNumber) -> InputNumber : Numéro d'entrée 0 à 255

Lecture de l'état d'une entrée. Résultat = 0 ou 1.

Exemple :  
if not **IN**(8) then ? « Pression d'air insuffisante »  
*'Si l'entrée 8 est à 0, alors print « Pression d'air insuffisante »*

### **GetInAll** -> cartes V2 uniquement

Lecture de l'état des entrées.

Exemple :  
Entree = GetInAll  
if (Entree and &H7) = &H7 then  
  ' Code si les entrées 1 à 4 sont actives  
  ...  
Endif

**Ain(CanalNumber) -> CanalNumber : Numéro entrée analogique 0 à 3**

Lecture de l'état d'une entrée analogique en millivolts.

Exemple :

```
if Ain(1) > 5000 then
  Out 1, 1
else
  Out 1, 0
endif
```

**AinV(CanalNumber) -> CanalNumber : Numéro entrée analogique 0 à 3**

Lecture de l'état d'une entrée analogique en Volts

Exemple :

```
if AinV(1) > 3,30 then
  Out 1, 1
else
  Out 1, 0
endif
```

**Out(SortieNo) -> SortieNo = Numéro 0 à 15 de la sortie -> carte V3 uniquement****GetOut(SortieNo) -> SortieNo = Numéro 0 à 32 de la sortie -> cartes V2 uniquement**

Lecture de l'état actuel d'une sortie.

Exemple V3:

? **Out(15)**

'Affiche dans le moniteur, l'état de la sortie 15

Exemple V2:

' Si la sortie 1 est active, la remettre à 0

```
if GetOut(1)=1 then Out 1, 0
```

**GetOutAll -> cartes V2 uniquement**

Lecture de l'état actuel des sorties

**Fonctions pour la détection de fronts sur Entrées ou Bits utilisateurs****DF(n° ou nom de l'Entrée)**

Détection front montant ou descendant d'une entrée.

Cette fonction permet de détecter le passage de l'état 0 à l'état 1 entre deux appels à cette fonction.

Exemples :

```
if DF(2) then SetAlarme()
```

'Si l'entrée 2 change d'état, alors...

ou

```
if DF(IN_VERROU) then
```

? « Intrusion ! »

```
SetAlarme()
```

```

endif
'si l'entrée détectant le verrou change d'état, alors on exécute notre
fonction d'alarme

```

### **DFM(Numero ou nom de l'entrée)**

Détection front montant ou descendant d'une entrée.

Exemple :

```

if DFM(CELLULE_OBJET) then ? « Objet détecté » ...

```

### **DFD(Numero ou nom de l'entrée)**

Détection front descendant d'une entrée. Cette fonction permet de détecter le passage de l'état 1 à l'état 0 entre deux appels à cette fonction.

Exemple :

```

if DFD(PRESSOSTAT) then ClearAlarme()

```

### **DFMBit(Numero d'instance(1 à 64), Valeur du bit testé)-> carte V3**

### **DFMBit(Numero d'instance(1 à 32), Valeur du bit testé)-> cartes V2**

Détection d'un front montant sur un bit testé.

Exemple :

```

if DFMBit(2, GetMB(MBB_ALARME)) then ? « Interruption des cycles
sur Alarme » ...

```

Exemple 2 :

```

if DFMBit(3, AIN(1)) then
  ' Code exécuté lorsque l'entrée analogique AI1 devient > 0
endif

```

### **DFDBit(Numero d'instance(1 à 64), Valeur du bit testé)-> carte V3**

### **DFDBit(Numero d'instance(1 à 32), Valeur du bit testé)-> cartes V2**

Détection d'un front descendant sur un bit testé.

Exemple :

```

if DFDBit(3, GetMB(MBB_ALARME)) then ? « Acquiescement des
Alarmes »

```

Exemple 2 :

```

if DFDBit(3, AIN(1)) then
  ' Code exécuté lorsque l'entrée analogique AI1 n'est plus > 0
endif

```

## **Fonctions pour la manipulation des chaînes de caractères**

RAPPEL: La commande **PRINT** est l'un des fondamentaux de toutes les variantes de langages Basic.

Elle permet d'afficher le contenu de chaînes de caractères, de variables, ou d'un résultat retourné par une fonction, dans la console (Moniteur) de l'interpréteur Basic.

Elle peut être utilisée conjointement avec la virgule -> pour afficher plusieurs expressions séparées par un espace.

Exemple:

```
A$="Monsieur"
PRINT "Bonjour",A$
-> Affichera:
Bonjour Monsieur
```

Elle peut aussi être utilisée avec le point virgule -> pour afficher plusieurs expressions à la suite sans espace entre elles. Il s'agit de la **concaténation de chaînes de caractères**.

Exemple:

```
A$="Monsieur"
PRINT "Bonjour";A$
-> Affichera:
BonjourMonsieur
```

**La concaténation de chaînes de caractères peut également être obtenue à l'aide de l'opérateur +.**

Exemple:

```
A$="12345": B$="6789"
C$ = A$+B$ 'Concatenation des 2 chaînes
PRINT C$
-> Affichera alors 123456789
```

**Liste des fonctions (classées par ordre alphabétique):**

**ASC(chaîne\_de\_caractères)**

Retourne le code ASCII (décimal) du (ou du premier) caractère de la chaîne.

Exemple:

```
A$ = "Hello"
b = ASC(A$)
PRINT b
ou
PRINT ASC("Hello")
→ Retourne H
```

**BIN\$(nb\_entier)**

Retourne le poids binaire de la partie entière d'un nombre.

Exemple:

```
PRINT BIN$(15)
→ Retourne 1111
```

**CHR\$(code\_ascii)**

Retourne le caractère correspondant au code ASCII.

Exemple:

```
PRINT CHR$(68)
→ Retourne D
```

**GetStr\$(adresse Modbus, nombre de registres, [swapBytes = 0 ])**

Retourne la chaîne de caractères stockée à partir de l'adresse Modbus spécifiée, et pour la taille spécifiée (nombre de registres 16bits, et un registre = 2 caractères).

L'argument `swapBytes` est un paramètre optionnel booléen (1 ou 0) qui détermine si l'ordre des octets dans chaque registre Modbus doit être inversé.

Exemple:

```
PRINT GetStr$(3010, 5)
```

**NB:** Si le nombre de registres demandés est supérieur à la longueur réelle de la chaîne, la fonction retournera la chaîne dans sa longueur réelle.

S'utilise après la commande:

**SetStr "chaîne\_de\_caractères", adresse Modbus, nombre de registres, [swapBytes = 0 ]**

L'argument `swapBytes` est un paramètre optionnel booléen (1 ou 0) qui détermine si l'ordre des octets dans chaque registre Modbus doit être inversé.

Exemple:

```
SetStr "ABCDEFGHIJKLMN", 3010, 4
```

**NB:** Si votre chaîne est plus longue que la longueur indiquée, elle sera alors tronquée (=automatiquement raccourcie).

Si votre chaîne est plus courte que la longueur indiquée, elle sera complétée par des zéros "0"

### **HEX\$(nb\_entier)**

Retourne la valeur Hexadécimale correspondant à la partie entière du nombre.

Exemple:

```
PRINT HEX$(65535)
```

→ Retourne FFFF

### **INSRT(longueur, chaîne1, chaîne2)**

Recherche tout ou partie d'une chaîne de caractères dans une autre, et retourne la position du début.

Exemple:

```
A$ = "Learn PLC-BASIC"
```

```
B$ = "BAS"
```

```
C = INSTR(2, A$, B$)
```

```
PRINT "Le mot BA commence à la position "; C
```

→ Retourne 11

### **LCASE\$(chaîne\_de\_caractères)**

Retourne la chaîne de caractères transformée en lettres minuscules.

Exemple:

```
A$="COMPUTER"
```

```
PRINT LCASE$(A$)
```

Retourne computer

### **LEFT\$(chaîne\_de\_caractères, longueur)**

Retourne une portion de la chaîne de caractères de la longueur indiquée, à partir de la gauche.

Exemple:

```
A$ = "INTERPCNC"
```

```
L$ = LEFT$(A$, 6)
```

```
PRINT L$
```

→ Retourne INTERP

### **LEN(chaîne\_de\_caractères)**

Retourne la longueur de la chaîne de caractères.

Exemple:

```
A$ = "INTERPCNC"
L = LEN(A$)
PRINT L
→ Retourne 9
```

### **MID\$(chaîne\_de\_caractères, position, longueur)**

Retourne une portion de la chaîne de caractères à partir de la position indiquée, pour la longueur donnée.

Exemple:

```
A$ = "INTERPCNC V3"
S$ = MID$(A$, 7, 3)
PRINT S$
→ Retourne CNC
```

### **OCT\$(nb\_entier)**

Retourne la valeur octale de la partie entière d'un nombre.

Exemple:

```
PRINT OCT$(8)
→ Retourne 10
```

### **RIGHT\$(chaîne\_de\_caractères, longueur)**

Retourne une portion de la chaîne de caractères de la longueur indiquée, à partir de la droite.

Exemple:

```
A$ = "INTERPCNC"
L$ = RIGHT$(A$, 3)
PRINT L$
→ Retourne CNC
```

### **SPACE\$(nb\_entier)**

Retourne une chaîne de caractères avec le nombre d'espaces spécifié.

Exemple:

```
FOR i = 1 TO 6
A$ = SPACE$(i)
PRINT A$; i
NEXT i
```

### **STRING\$(répétition, caractère)**

Retourne une chaîne de caractères contenant le caractère spécifié répété n fois.

Exemple:

```
PRINT STRING$(2,68)
Ou
PRINT STRING$(2,"D")
→ Retourne DD
```

### **STR\$(nb\_reel)**

Retourne la représentation d'un nombre réel en chaîne de caractères.

Exemple:

```
N = 453.1
PRINT STR$(N)
→ Retourne 453.1
```

### **TIMES\$**

Retourne le temps écoulé depuis la mise sous tension de la carte.

Exemple:

(NB : L'affectation du type : TIME\$= « 00:00:00» ne peut modifier cette durée.)

```
PRINT "Sous tension depuis: "; TIME$
```

→ Retourne le temps,  
sous le format exemple: 00:25:30

**UCASE\$(chaîne\_de\_caractères)**

Retourne la chaîne de caractères transformée en lettres majuscules.

Exemple:

```
A$="computer"
```

```
PRINT UCASE$(A$)
```

→ Retourne COMPUTER

**VAL(chaîne\_de\_caractères)**

Retourne la valeur numérique située au début d'une chaîne de caractères.

Exemple:

Retourne 0 (zéro) s'il n'y en a pas.

```
N$="1234B5"
```

```
X=VAL(N$)
```

```
PRINT "Valeur attendue: "; X
```

→ Retourne 1234

**Fonctions de gestion des interruptions et des signaux périodiques (V3)****Clock.PeriodicSub**

Syntaxe : Clock.PeriodicSub( Periode\_ms, SubName\$)

Arguments :

Periode\_ms : période d'appel en millisecondes (float, recommandé  $\geq 1.00$  ms).

SubName\$ : nom de la sous-routine appelée périodiquement.

Description : Crée un appel périodique vers une sous-routine utilisateur et retourne un handle.

Exemple : Handle = Clock.PeriodicSub (100, "CheckInputs" ) → appelle

CheckInputs toutes les 100 ms.

Fonction interne plc : Extra(10,...)

**PulseTrain.Create**

Syntaxe : PulseTrain.Create(OUTNumber, Periode\_ms, Npulses,

[TON\_ms=Periode\_ms/2] ,[StatusBit])

Arguments :

OUTNumber : sortie ou coil cible.

Periode\_ms : période en millisecondes (nombre flottant, peut être inférieur à 1.00 ms , minimum 20µs)..

NPulses : nombre d'impulsions à générer.

TON\_ms (optionnel) : durée de l'état haut (défaut = Periode/2).

StatusBit (optionnel) : coil ou sortie de statut.

Description : Crée un train d'impulsions en millisecondes et retourne un handle.

Exemple : Handle = PulseTrain.Create(1, 2, 5) → génère 5 impulsions de 2 ms sur

OUT1.

Fonction interne plc : Extra(21,...)

**PWM.Create**

Syntaxe : PWM.Create(OUTNumber, Frequency\_hz, [ DutyCycle=50])

Arguments :

OUTNumber : sortie ou coil cible.

Frequency\_hz : fréquence du PWM (Hz).

DutyCycle (optionnel) : rapport cyclique en pourcentage (par défaut 50). Si

DutyCycle = 0 la sortie reste a 0 et si DutyCycle = 100 la sortie reste a 1.

Description : Crée un signal PWM sur une sortie et retourne un handle.

Exemple : Handle = PWM.Create(0, 2000, 30) → crée un PWM à 2 kHz et 30 % sur OUT0.

Fonction interne plc : Extra(30,...)

## LES COMMANDES

Une **commande** diffère d'une fonction, en ce sens qu'elle ne retourne aucune valeur, ni résultat.

Pour l'utiliser, nous lui passons des arguments sous forme de paramètres ou valeurs de réglages.

Lors de l'utilisation d'une commande, seule une action est attendue (écriture ou copie de bits ou de registres, déplacement d'axes, démarrage d'un timer, etc...)

### Commandes issues du Basic standard

#### Run

**Exécute** le programme PLC Basic

#### Stop

**Arrêt** de l'exécution du programme PLC Basic

#### List

**Affiche la liste** du programme PLC Basic présent en Ram, dans la fenêtre du moniteur

#### Print (ou ? )

**Affiche l'expression** entre guillemets ou la valeur d'une variable dans la fenêtre du **Moniteur**.

L'utilisation du point-virgule ajoutera plusieurs expressions dans la même commande Print.

L'utilisation d'une virgule séparera plusieurs expressions par des espaces.

Exemples:

? "Hello"; "World" -> affichera "HelloWorld"

Print "Production =",Compteur,"unités" -> affichera "Production = 33 unités"

### Commandes PLC Basic spécifiques à l' InterpCNC

#### Commandes

[SetMB](#) memo, nbr  
[SetMBit](#) memo, nbr  
[SetMW](#) adr, nbr  
[SetMDW](#) adr, nbr  
[SetMDWInv](#) adr, nbr  
[SetMI](#) adr, nbr  
[SetMDI](#) adr, nbr  
[SetMDIInv](#) adr, nbr  
[SetME](#) adr, nbr  
[SetMEInv](#) adr, nbr  
[IncMDW](#) adr[, nbr]  
[CopyReg](#) adr, nbr  
[SetMBReg](#) adr, nbr  
[SetEEData8](#) adr,nbr  
[SetEEData16](#) adr, nbr  
[SetEEData32](#) adr, nbr  
[SetEEDataFloat](#) adr, nbr  
[SetUserMem](#) nbr, nbr  
[SetUserMemE](#) nbr, nbr  
[SetStr](#) str, adr, nbr

[Unlock](#)  
[Lock](#)  
[ListFlash](#)  
[SRAAdd](#) ArrayName, nbr  
[SREFill](#) ArrayName, nbr  
[Msg](#)  
[SaveProgram](#)  
[LoadProgram](#)  
[SetPrm](#) nbr, nbr

[CopyRCP](#)  
[InsertRCP](#)  
[RemoveRCP](#)

[SetPos](#) nbr, nbr  
[MoveAxe](#) nbr, nbr, nbr, nbr  
[MoveSpeed](#) nbr, nbr, nbr  
[Home](#) nbr, nbr, nbr, nbr, nbr, nbr, nbr, nbr  
[MoveInterp](#) nbr, nbr, nbr  
[Probe](#) nbr, nbr, nbr, nbr, nbr, nbr, nbr  
[StopAxes](#) nbr  
[StopAxeID](#) nbr

[Pause](#) nbr  
[SetTimer](#) str, nbr

[Tic](#) nbr  
[CycleStatInit](#) nbr, adr, adr, adr

[SetIN](#) nbr, nbr  
[OUT](#) nbr, nbr  
[OUTALL](#) nbr  
[OUTPort](#) nbr, nbr  
[OUTTTL](#) nbr, nbr

[SetAna nbr, nbr](#)  
[SetAnaV nbr, nbr](#)

[SetEncoder nbr, nbr](#)  
[SetCnt nbr, nbr](#)

[SetTick nbr, nbr, str](#)  
[SetInputInt nbr, nbr, str](#)  
[SetCaptureInt nbr, nbr, str](#)  
[SetCaptureIDOnInputInt nbr, nbr](#)

## Accès aux Registres et Bits utilisateur en Ecriture (Holding Registers et Coils)

### **SetMB** Memo, Value

ou

### **SetMB** Registre, Bit, Value

Set ou Reset d'un Memo bit (Coil) dans l'espace des Memos (Coils utilisateurs), ou un bit particulier dans le domaine des registres.

Exemple :

**SetMB** *MBB\_ALARMES*, 1

*'mise à 1 du bit d'alarme*

**SetMB** 3100,15,0

*'mise à 0 du 16ème bit du registre 3100 (Ram)*

**SetMB** 3100,31,1

*'mise à 1 du 32ème bit du registre 3100 (Ram)*

### **SetMBBit** Memo, Value -> cartes V2 uniquement

Commande devenue obsolète car remplacée depuis le firmware V5.39, par :

SetMB Memo, Value

**NB: Les commandes et fonctions obsolètes** sont maintenues uniquement pour assurer la compatibilité des cartes avec d'anciens programmes. Il est toutefois recommandé de ne plus les utiliser.

### **SetMW** Registre, Value

Ecriture dans un registre 16 bits non signé (U16).

Exemple :

**SetMW** *G\_CYCLE*, 10 *'écrit la valeur 10, à l'adresse nommée G\_CYCLE*

### **SetMDW** Registre, Value

Ecriture dans un registre 32 bits non signé (U32)

Exemple :

**SetMDW** *EE\_COMPTEUR\_TOTAL*, **GetMDW**(*EE\_COMPTEUR\_TOTAL*)+1

*'incréméntation de 1 du registre EEprom stockant la valeur du compteur total*

### **SetMDWInv** Registre, Value -> carte **V3** uniquement, firmware >=V1.41

Ecriture dans un registre 32 bits non signé, pour lequel les 2 registres 16 bits utilisés sont inversés.

(selectionner *U32\_INV* pour l'affichage)

#	Value	Name	Type
3000			U16
3001			U16
3002			I16
3003			U32
3004			I32
3005			FLOAT
3006			U32_INV
			I32_INV
			FLOAT_INV

### SetMI Registre, Valeur

Ecriture dans un registre 16 bits signé (I16)

Exemple :

**SetMI** 3010, -32767

'écrit la valeur -32767, à l'adresse 3010 (ici définie en I16, en Ram)

### SetMDI Registre, Valeur

Ecriture dans un registre 32 bits signé (I32)

Exemple :

**SetMDI** 3010, -999999

'écrit la valeur -999999, à l'adresse 3010 (ici définie en I32, en Ram)

### SetMDIInv Registre, Valeur -> carte **V3** uniquement, firmware >=V1.41

Ecriture dans un registre 32 bits signé, pour lequel les 2 registres 16 bits utilisés sont inversés.

(selectionner I32\_INV pour l'affichage)

### SetMF Registre, Valeur

Ecriture d'un nombre de type FLOAT dans dans un registre 32 bits (FLOAT)

Exemple :

**SetMF** 3200, 3.14159

'écrit la valeur 3.14159, à l'adresse 3200 (ici définie en FLOAT, en Ram)

### SetMFIInv Registre, Valeur -> carte **V3** uniquement, firmware >=V1.41

Ecriture d'un nombre de type FLOAT dans dans un registre 32 bits, pour lequel les 2 registres 16 bits utilisés sont inversés.

(selectionner FLOAT\_INV pour l'affichage)

### IncMDW Registre[, +/-Valeur de l'incrément =1]

Permet d'incrémenter un registre 32 bits de la valeur indiquée ou de +1 si le second paramètre n'est pas précisé.

Exemple 1 : **IncMDW** 3020, 5 ' Incrément de 5

Exemple 2 : **IncMDW** 3030 ' incrément de 1

### CopyReg Adresse 1<sup>er</sup> registre Source (0 à 65535), Adresse Destination(Holding Register de 0 à 65535), Nombre de registres (0 à 65535)

Copie une zone de mémoire contenant le nombre de registres indiqués, vers une adresse de destination.

Exemple avec des Holding Registers :

**CopyReg** 3000, 3200, 16

'copie les registres 3000 à 3015, vers les registres 3200 à 3215'

Exemple avec des Input Registers :

**CopyReg** 101030, 3000, 12

'copie les registres de positions des 6 axes (Input Registers 1030 à 1041), vers les adresses 3000 à 3011 (Holding Registers)'

**SetMBReg** Registre, Valeur -> cartes V2 uniquement

Commande devenue obsolète car remplacée depuis le firmware V5.39, par :

**SetMW** Registre, Valeur  
(voir plus haut).

**NB: Les commandes et fonctions obsolètes** sont maintenues uniquement pour assurer la compatibilité des cartes avec d'anciens programmes. Il est toutefois recommandé de ne plus les utiliser.

**SetEEData8** Registre, Valeur -> cartes V2 uniquement

Registre = 0 à 1023. Valeur = 0 à 255.

Commande obsolète car remplacée depuis le firmware V5.39, par SetMW  
Registre, Valeur

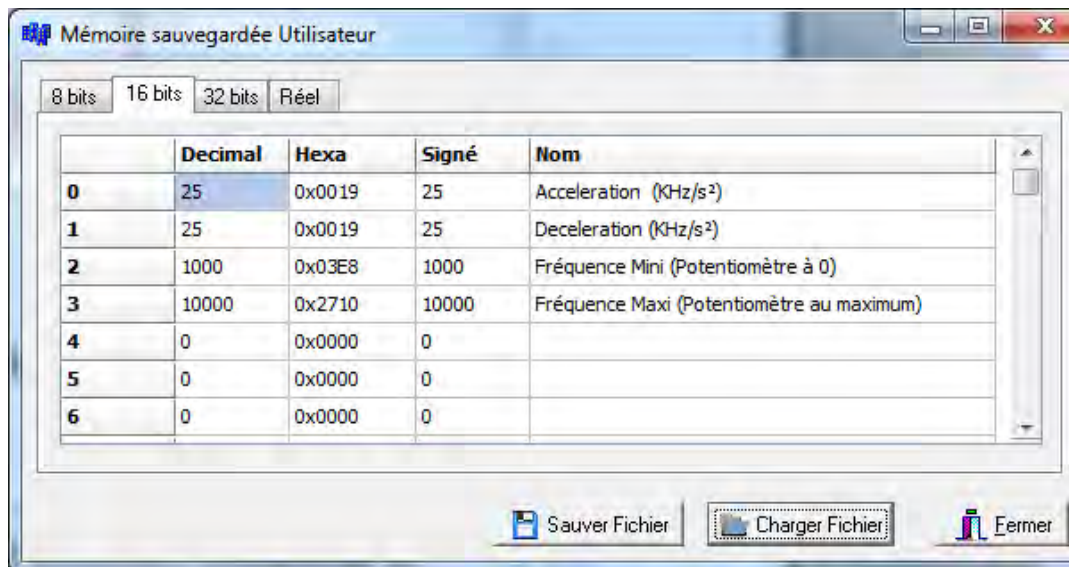
Écriture dans la zone de mémoire Utilisateur sauvegardée.

Vous disposez d'une zone de 1024 octets accessibles sous forme d'octet, d'entiers 16 bit, d'entiers 32 bits ou de réels.

Cette zone mémoire est particulièrement adaptée à l'enregistrement de paramètres utilisateur et accessible également en Modbus ou par USB (Menu Édition/Éditeur EEPROM Utilisateur).

Attention, cette zone mémoire est accessible sous différents formats mais il s'agit d'une zone unique.

Donc, l'écriture de l'octet N°0 affectera l'entier 16 bit N°0, l'entier 32 bits N°0 et également le réel N°0



Exemple :

SetEEData8 0, 25

**SetEEData16** Registre, Valeur -> cartes V2 uniquement

Registre = 0 à 511. Valeur = 0 à 65535.

Commande obsolète car remplacée depuis le firmware V5.39, par SetMW

Registre, Valeur

Fonctionnement identique à la fonction SetEEData8 mais pour des entiers 16 bits

Exemple :

**SetEEData16** 2, 1000

**SetEEData32** Registre, Valeur -> cartes V2 uniquement

Registre = 0 à 255. Valeur = 0 à  $2^{32}-1$ .

Commande obsolète car remplacée depuis le firmware V5.39, par SetMDW

Registre, Valeur

Fonctionnement identique à la fonction SetEEData8 mais pour des entiers 32 bits

Exemple :

**SetEEData32** 9, 179400

**SetEEDataFloat** Registre, Valeur -> cartes V2 uniquement

Registre = 0 à 255. Valeur =  $1.17549435e-38$  à  $3.40282347e+38$

Commande obsolète car remplacée depuis le firmware V5.39, par SetMF Registre,

Valeur

Fonctionnement identique à la fonction SetEEData8 mais pour des nombres réels.

Exemple :

**SetEEDataFloat** 11, 3.141

**SetUserMem** Registre, Valeur -> cartes V2 uniquement

Registre = 0 à 9. Valeur à écrire (entier sur 32 bits).

Commande obsolète car remplacée depuis le firmware V5.39, par SetMDW

Registre, Valeur

Écriture dans la zone mémoire RAM utilisateur (10 registres 32 bits situés aux adresses 4198 à 4217).

Cette zone mémoire est accessible à l'aide de l'ancien programme TestCenter (Menu Édition/Mémoire utilisateur) et permet par exemple l'affichage de variable durant la phase de mise au point du programme ou pour l'échange de donnée entre application (Basic, Modbus ou USB).

**NB:** Depuis le firmware V5.39, les 10 registres UserMem ont été remapés aux adresses 7000 à 7018, zone dans laquelle écrivent à présent les commandes

**SetUserMem** et **SetUserMemF**.

Les cartes V2 bénéficient aussi de registres de RAM supplémentaires disponibles aux adresses 7019 à 7333 avec les autres commandes et fonctions d'accès aux Holding Registers.

**SetUserMemF** Registre, Valeur -> cartes V2 uniquement

Registre = 0 à 9. Valeur à écrire (nombre réel).

Commande obsolète car remplacée depuis le firmware V5.39, par SetMF Registre,

Valeur

Lecture de la valeur présente en mémoire utilisateur.

La lecture considère que la valeur est nombre réel.

**SRAdd** ArrayName, Valeur

Gestion d'un tableau en tant que registre à décalage.

La fonction SRAdd décale toutes les valeurs d'un tableau déclaré par DIM(x) et écrit Valeur à l'index 0 du tableau.

Exemple :

DIM tab(3)

tab(0)=0

tab(1)=1

```
tab(2)=2  
SRAdd tab, 5
```

Le tableau tab contient alors les valeurs :  
5 pour tab(0), 0 pour tab(1) et 1 pour tab(2)

## **SRFill** ArrayName, Valeur

Initialisation de l'ensemble des valeur d'un tableau déclaré par DIM(x) avec la valeur Valeur.

### Commandes spécifiques au Hardware de la carte InterpCNC

## **Unlock**

**Déverrouillage** de la carte.

Lorsque la carte est verrouillée, les commandes de déplacement ou d'activation des sorties sont inactives.

## **Lock**

**Verrouillage** de la carte.

Lorsque la carte est verrouillée, les commandes de déplacement ou d'activation des sorties sont inactives.

## **ListFlash**

Affiche la liste du programme PLC Basic présent en mémoire Flash, dans la fenêtre du moniteur.

## **SetPrm** Numéro, Valeur -> Numéro : ID paramètre de 0 à 999

**Écriture** de la valeur d'un paramètre InterpCNC en connaissant son identifiant.

Exemple :

**SetPrm** 20, 1300 ' Fréquence initiale des mouvements Axe 1, réglée sur 1300hz

## **Msg**

-> carte **V3** uniquement

**Affiche un message** sur l'écran Oled de la carte.

La longueur maximale est de 15 caractères (ce qui dépasse sera ignoré).

Le message est toujours centré sur la ligne.

Même utilisation que la commande **Print**.

Exemple:

**Msg** "Prod=",Compteur,"unités"

-> affichera "Prod = 33 unités"

### Commandes de gestion du programme

## **New**

**Efface** le programme PLC Basic en Ram

## **SaveProgram**

**Sauve le programme** PLC Basic actuellement en Ram, dans la mémoire Flash (même action que le click sur l'icône Carte SD)

## LoadProgram

**Charge le programme** présent en mémoire Flash pour le placer dans la RAM. Cette commande est appelée automatiquement à la mise sous tension si le paramètre « Démarrage PLC automatique » est actif. Elle sera alors suivie d'une commande RUN également appelée automatiquement.

## Commandes liées à la gestion de Recettes

### CopyRCP Recette Source, Recette Cible

**Fonction recopiant une Recette**, vers un autre emplacement Recette.

NB : **RCP\_SIZE** (à l'adresse 9995 -> nombre de registres alloués pour chaque recette) doit avoir été défini au préalable.

Exemple :

**CopyRCP** 0,1 *'Recopie la recette 0 dans la recette 1*

### InsertRCP Position Destination, Recette Source

**Insère la copie d'une recette** à l'emplacement précisé. Les recettes suivantes sont décalées.

Exemple :

**InsertRCP** 1,4 *'Insère une copie de la recette 4, en position 1*

### RemoveRCP NumeroRecette

**Supprime une recette**. Les recettes suivantes sont décalées.

Exemple :

**RemoveRCP** 1 *'Supprime la recette n°1*

## Commandes liées aux Mouvements d'Axes

### SetPos AxeNumber, Value

AxeNumber : Numéro de l'axe 1 à 5

Value : Valeur position

Écriture du compteur de position d'un axe. Cette fonction ne doit pas être appelée durant la rotation de l'axe. Le paramètre AxeNumber permet d'indiquer l'axe concerné par la commande.

Exemple :

**SetPos** 3, 1000 *'Écrire 1000 dans le compteur axe Y*

### MoveAxe AxelD, Accel, Vitesse, Decel, Position

AxelD : Identifiant de l'Axe (1 à 6)

Accel : Accélération en Hz/s (**V3**) ou kHz/s (**V2**)

Vitesse : Fréquence en Hz des pulses

Decel : Décélération en Hz/s (**V3**) ou kHz/s (**V2**)

Position : Cible, en pas moteur

Met l'axe en mouvement jusqu'à une position cible.

Exemple :

**MoveAxe** 1, 1500, 10000, 1500, 50936  
*'Déplacement axe X*

**MoveSpeed** AxelD, accel ou decel (selon la cible), vitesse (signée)

Rotation jusqu'à une vitesse spécifique (vitesse signée)

Exemple :

**MoveSpeed** 1, 1500, 10000

**Home** AxelD, InputNumber, InputState, Accel, HighSpeed, Decel, (+/-)MaxStep, LowSpeed, [HomePosition], [MoveStepAfterHome]

AxelD : Axe à initialiser (1 à 6)

InputNumber: Numéro de l'entrée utilisée pour le référencement

InputState : Etat de l'entrée lorsque le contact est activé (0 ou 1)

Accel : accélération mouvement rapide vers le capteur, en Hz/s (**V3**) ou kHz/s (**V2**)

HighSpeed : Vitesse rapide vers le capteur

Decel : décélération mouvement rapide vers le capteur, en Hz/s (**V3**) ou kHz/s (**V2**)

(+/-)MaxStep : donne le sens de déplacement et la course Maxi pour le

Homing

LowSpeed : vitesse dégagement du capteur

[HomePosition] valeur à laquelle est initialisé le compteur de position avant le dégagement

[MoveStepAfterHome] position cible pour le dégagement (relative à la home position)

Démarre une séquence de Homing. Plusieurs commandes peuvent être lancées simultanément sur différents axes.

Cette fonction est utilisée pour trouver une position d'origine à l'aide d'un capteur placé sur la course d'un axe.

Exemple :

**Home** 2, 2, 0, 25, 20000, 25, 1000000, 1000, 0

*'Origine axe 2 sur entrée N°2 type NC sur une course maxi de 1000000 pas. Position d'origine initialisée à 0*

**MoveInterp** AxesBit, Vitesse, Targets... -> **cartes V2** uniquement

AxesBit : Indique les axes à déplacer

Bit 0 = 1 pour déplacer l'axe X

Bit 1 = 1 pour déplacer l'axe Y

Bit 2 = 1 pour déplacer l'axe Z

Bit 3 = 1 pour déplacer l'axe A

Bit 4 = 1 pour déplacer l'axe B

Vitesse : Fréquence en Hz des pulses sur l'axe majeur

Targets : Cibles des axes à déplacer

Déplacement interpolé de 2 axes ou plus.

La Vitesse indiquée en Hz est celle de l'axe majeur (celui qui a le plus grand déplacement à réaliser).

Les vitesses des autres axes seront calculées pour que les axes arrivent en même temps aux cibles indiquées.

La fréquence Start/Stop utilisée est celle du paramètre InterpCNC N° 5.

L'accélération/décélération utilisée est celle du paramètres InterpCNC N° 6.

L'envoi de plusieurs commande de déplacement interpolés est possible. Une seule sera traité à la fois. Les commandes suivantes sont placées dans un buffer d'exécution et traitées les unes à la suite des autres.

Inconvénient de cette commande: ne peut être interrompue avant la fin de son exécution.

Exemple :

```
MoveInterp 3, 10000, 43567, 21237 ' Move X and Y
MoveInterp &H18, 5000, 12456, 14325 ' Move A and B
```

## **Probe** AxelD, InputNumber, InputState, Accel, Speed, Decel, (+/-)MaxStep

AxelD : Axe à initialiser (1 à 6)

InputNumber: Numéro de l'entrée utilisée pour le référencement

InputState : Etat de l'entrée lorsque le contact est activé (0 ou 1)

Accel : accélération mouvement vers le capteur, en Hz/s (**V3**) ou kHz/s (**V2**)

Speed : Vitesse vers le capteur

Decel : décélération mouvement vers le capteur, en Hz/s (**V3**) ou kHz/s (**V2**)

(+/-)MaxStep : donne le sens de déplacement (valeur signée) et la course

Maxi pour le Palpage

Lancement d'un Palpage.

Exemple :

```
Probe 1, 3, 0, 25, 50000, 25, 1000000
```

*'Palpage sur l'Axe 1, capteur sur l'entrée 3, de type NC, accélération et décélération de 25kHz, vitesse de 50000 Hz, sur 1000000 de pas Maxi '*

## **StopAxes** Valeur

Valeur : binaire (axe sélectionné=Bit à 1) ou Hexadécimal, ou décimal

Arrêt d'un ou plusieurs axes.

Exemple :

```
StopAxes &B111111 'Arrêt pour les 6 axes (sélection binaire des axes)
```

```
StopAxes &H3F 'Arrêt pour les 6 axes (valeur 3F en Hexa pour les bits sélectionnés)
```

```
StopAxes 63 'Arrêt pour les 6 axes (valeur décimale pour les bits sélectionnés)
```

## **StopAxelD** Valeur -> cartes **V3** uniquement

Arrêt d'un axe par son Identifiant (numéro ou nom).

Exemple :

```
StopAxesID 2
```

ou

```
StopAxesID AXE_ETIQUETTE
```

## Commandes liées aux Timers

### **Pause** ppp

ppp = durée en ms

Pause de ppp milliseconde dans l'exécution du programme.

Les traitement d'interruption ne sont en revanche pas interrompu durant une pause.  
Cette fonction peut également être utilisé dans les traitements d'interruption.

Exemple :

*'activation de la sortie 1 pendant une seconde:*

OUT 1,1 'Activation sortie 1

PAUSE 1000 'Pause 1000ms

OUT 1,0 'Désactivation sortie 1

## **SetTimer** NomVariable, Durée\_ms

Initialisation d'une variable timer. La durée est exprimée en milliseconde.

Commande à utiliser avec la fonction GetTimer.

Exemple :

**SetTimer** Tempo1, 500

## **Tic** numéro d'instance de 0 à 15 -> **carte V3** uniquement

Démarre une capture de temps, en micro-secondes.

Exemple :

**Tic** 1

## **CycleStatInit** N° d'instance (1 à 5), temps mini, temps actuel, temps maxi -> **carte V3** uniquement

Initialise une fonction mesure de temps. Les valeurs seront stockées dans les 3 registres MW (16bits) indiqués.

Les temps sont donnés en 1/10 de ms.

Exemple :

**CycleStatInit** 1, 3015, 3016, 3017

ou

**CycleStatInit** 1, TPLC\_MIN, TPLC\_ACTUEL, TPLC\_MAX

## Commandes liées aux Entrées et Sorties

### **SetIN** N° Entrée, Etat

Entrée : 0 à 255

Etat : -1 pas de forçage, forçage à 0, forçage à 1

Forçage de l'état d'une entrée.

Exemple :

**SetIN** 3, 1 'forçage à 1 de l'entrée 3

### **OUT** SortieNumero, Valeur

Numéro : 0 à 95

Valeur = 0 ou 1

Activation/désactivation d'une sortie tout ou rien (OUT 0 à OUT 15).

Les sorties OUT 0 à OUT 15 sont des sorties physiques. Les sorties 16 à 95 sont des sorties qui peuvent être utilisées via un dispositif externe (interfaces Modbus ou USB)

Exemple :

**OUT** 4,1 'Passe à 1 l'état de la sortie 4

**OUTPort NumPort, Valeur** -> **carte V3** uniquement

NumPort : 0 à 11

Valeur : 0 à 255

Sert à définir l'état (0 ou 1) sur un port de 8 sorties (8 bits).

Exemple :for i = 0 to 11 : **OUTPort** i, 0 : Next i *'Remise à zéro de toutes les sorties***OUTALL Valeur** -> **cartes V2** uniquement

Définition de l'état de toutes les sorties en même temps.

Exemple :

OUTALL &amp;H10 ' Activation sortie OUT5

OUTALL &amp;H12 ' Activation sortie OUT5 et OUT2

**OUTTTL Numéro, Valeur** -> **cartes V2** uniquementNuméro des sorties (Cartes 4 et 5 axes): **1** : CLK1, **2** : DIR1, **3** : CLK2, **4** : DIR2, **5** : CLK3, **6** : DIR3, **7** : CLK4, **8** : DIR4Numéro des sorties (Carte 5 axes uniquement): **9** : CLK5, **10** : DIR5

Valeur : 0 ou 1

Les sorties Pulses/Direction des axes peuvent être utilisées comme sorties TTL (0 ou 5V), ce qui peut être pratique dans le cas où vous avez besoin de plus de sorties physiques que celles normalement disponibles.

NB : Il est préférable de n'utiliser comme sorties TTL, que celles des axes non utilisés. (Éventuellement la sortie DIR d'un axes utilisé peut être employée, à condition d'être sûr de tourner toujours dans le même sens).

Exemples:

**OUTTTL** 2, 1*'Niveau haut sur sortie DIR1'***OUTTTL** 5, 0*'Niveau bas sur sortie CLK3'*

Il est possible d'inhiber l'effet des commandes d'axes (MoveAxe, Home, etc...) sur ces sorties pour ne pas interférer.

Pour ce faire, dans le registre système 45, un bit est attribué à chaque sortie (Bit 0 pour CK1, Bit1 pour DIR1, etc...)

→ à 1, la sortie répond aux commandes d'axes et à la commande **OUTTTL**→ à 0, les commandes d'axes sont inhibées, et la sortie ne répondra qu'à la commande **OUTTTL**.

Ainsi:

**SetSys** 45, 0 inhibe tous les mouvements d'axes sur toutes ces sorties(= **SetSys** 45, &b0000000000000000)**SetSys** 45, 975 inhibe les mouvements d'axes pour CK3 et DIR3(= **SetSys** 45, &b0000001111001111)**SetAna Canal, Valeur**

Canal = 0 ou 1 (AOUT0 ou AOUT1)

Valeur = 0 à 10000

Définit le niveau d'une sortie analogique en point mV (0 à 10000).

Exemple :

**SetAna** 1, 5000 '*Sortie analogique AOUT1 à 5V*

## **SetAnaV** Canal, Tension

Canal = 0 ou 1 (AOUT0 ou AOUT1)

Tension = 0 à 10V

Définit le niveau d'une sortie analogique en Volt (0 à 10v).

Exemple :

**SetAnaV** 1, 5.51 '*Sortie analogique AOUT1 à 5,51V*

## Commandes de gestion des interruptions et des signaux périodiques (V3)

### **Pulse.On**

Syntaxe : PulseOn OUTNumber, PulseLength\_ms

Arguments :

OUTNumber : numéro de la sortie ou du coil.

PulseLength\_ms : durée en millisecondes (nombre flottant, peut être inférieur à 1 ms , minimum 20µs).

Description : Active la sortie pendant PulseLength\_us puis la remet à 0. Si sortie inactive → pulse haut. Si sortie active → retour à 0 retardé.

Exemple : PulseOn 2, 10.5 → génère une impulsion haute de 10,5 ms sur OUT2.

Commande interne plc : Extra 17,...

### **Pulse.Off**

Syntaxe : PulseOff OUTNumber, PulseLength\_ms

Arguments :

OUTNumber : numéro de la sortie ou du coil.

PulseLength\_ms : durée en millisecondes (nombre flottant, peut être inférieur à 1 ms , minimum 20µs).

Description : Force la sortie à 0 pendant PulseLength\_us puis la remet à 1. Si sortie déjà à 0 → retour retardé. Sinon → pulse bas puis retour haut.

Exemple : PulseOff 3, 2 → génère une impulsion basse de 2 ms sur OUT3.

Commande interne plc : Extra 18,...

### **Clock.SetMode**

Syntaxe : Clock.SetMode TimerMode

Arguments :

TimerMode :

0 → exécute 1 callback si en retard (par défaut).

1 → exécute toutes les occurrences ratées (risque surcharge CPU).

2 → exécute 1 callback, avance la deadline par multiples de période (pas de dérive).

Description : Définit le mode global de gestion des timers périodiques.

Exemple : Clock.SetMode 2 → active le mode sans dérive temporelle.

Commande interne plc : Extra 9,...

### **Clock.Delete**

Syntaxe : Clock.Delete hClock

Arguments :

hClock : handle du timer à supprimer.

Description : Supprime un timer créé avec Clock.PeriodicSub.  
Exemple : Clock.Delete 5 → supprime le timer avec handle 5.  
Commande interne plc : Extra 30,...

### **Clock.SetPeriode**

Syntaxe : Clock.SetPeriode hClock, NewPeriode\_ms

Arguments :

hClock : handle du timer.

NewPeriode\_ms : nouvelle période en millisecondes (float, minimum 1ms).

Description : Change la période d'un timer existant avec prise en compte immédiate.

Exemple : Clock.SetPeriode 2, 0.5 → fixe la période du timer 2 à 0,5 ms.

Commande interne plc : Extra 31,...

### **Clock.Suspend**

Syntaxe : Clock.Suspend hClock

Arguments :

hClock : handle du timer.

Description : Suspend l'exécution d'un timer.

Exemple : Clock.Suspend 4 → met en pause le timer 4.

Commande interne plc : Extra 32,...

### **Clock.Resume**

Syntaxe : Clock.Resume hClock

Arguments :

hClock : handle du timer.

Description : Relance un timer suspendu.

Exemple : Clock.Resume 4 → relance le timer 4.

Commande interne plc : Extra 33,...

### **PulseTrain.Suspend**

Syntaxe : PulseTrain.Suspend hPulseTrain

Arguments :

hPulseTrain : handle du train d'impulsions.

Description : Suspend un train d'impulsions en cours. La sortie est réinitialisée, le compteur inchangé, le bit de statut reste actif.

Exemple : PulseTrain.Suspend 7 → suspend le train d'impulsions 7 et garde la sortie dans le même état pendant la durée de l'appel.

Commande interne plc : Extra 10,...

### **PulseTrain.Resume**

Syntaxe : PulseTrain.Resume hPulseTrain

Arguments :

hPulseTrain : handle du train d'impulsions.

Description : Reprend un train suspendu, à partir de l'endroit où il avait été arrêté.

Exemple : PulseTrain.Resume 7 → relance le train d'impulsions 7.

Commande interne plc : Extra 11,...

### **PulseTrain.Delete**

Syntaxe : `PulseTrain.Delete hPulseTrain`

Arguments :

hPulseTrain : handle du train d'impulsions.

Description : Supprime un train d'impulsions, réinitialise la sortie et efface le bit de statut.

Exemple : `PulseTrain.Delete 7` → supprime le train d'impulsions 7 remet la sortie à 0.

Commande interne plc : Extra 12,...

### **PWM.Delete**

Syntaxe : `PWM.Delete hPWM`

Arguments :

hPWM : handle du PWM.

Description : Supprime un signal PWM existant et réinitialise la sortie à 0.

Exemple : `PWM.Delete 3` → supprime le PWM avec handle 3.

Commande interne plc : Extra 20,...

### **PWM.Update**

Syntaxe : `PWM.Update hPWM, NewFrequency, [NewDutyCycle]`

Arguments :

hPWM : handle du PWM.

NewFrequency : nouvelle fréquence en Hertz.(float,minimum = 0.001 Hz max = 25KHz)

NewDutyCycle (optionnel): nouveau rapport cyclique en pourcentage (float entre 0–100).Si

DutyCycle = 0 la sortie reste à 0 et si DutyCycle = 100 la sortie reste à 1.

Description : Met à jour la fréquence et le rapport cyclique d'un PWM existant.

Exemple : `PWM.Update 2, 1000, 75` → configure le PWM 2 à 1 kHz et 75 % de duty cycle.

Commande interne plc : Extra 21,...

## **CREATION D'UN GRAFCET AVEC L'INSTRUCTION « Select Case »**

L'instruction « **Select Case** » permet aisément la création d'un **Grafcet**, et donne une plus grande lisibilité à votre programme, comparée à une séquence du type « If Etape=10 then... Else... Endif »

Les instructions associées disponibles sont :

**Select Case Variable**  
**Case Value**  
**Case ValueFrom to ValueTo**  
**Case Value is < Limite**  
**Case Else**  
**End Select**

Exemple d'utilisation:

Supposons que nous souhaitons créer un cycle automate de 4 étapes (étape 0, 10, 20, et 30) dans lequel nous allons activer/désactiver la sortie 1 selon l'état de l'entrée 1, puis activer/désactiver la sortie 2 avec une temporisation de 1000ms. Nous utiliserons par exemple une variable nommée « GCycle1 », qui prendra tour à tour la valeur d'étape en cours. Celle-ci sera réaffectée en fin de chaque étape, ce qui permettra de passer à la suivante au prochain tour de la boucle DO ... LOOP

```

GCycle1 = 0 ' Initialisation du Cycle à l'étape 0
DO
    Select Case GCycle1
    Case 0
        if IN(1) then
            Out 1,1
            GCycle1 = 10
        Endif

    Case 10
        if not IN(1) then
            Out 1,0
            GCycle1 = 20
        Endif

    Case 20
        Out 2,1
        SetTimer tempo1, 1000
        GCycle1 = 30

    Case 30
        if GetTimer(tempo1)=0 then
            Out 2,0
            GCycle1 = 0
        Endif

    Case Else
        ? "Erreur de programmation"
    End Select
LOOP

```

Ainsi, chaque instruction "Case .." permettra de traiter, pour chaque valeur d'étape du cycle GCycle1, le code à exécuter.

## GESTION DES INTERRUPTIONS

**Vous avez la possibilité de programmer trois types d'interruption.**

- Les interruptions périodiques
- Les interruptions liées à l'état des entrées
- Les interruptions sur positions d'axes (carte V3 uniquement)

Le temps de latence pour le traitement d'une interruption est  $< 5\mu\text{s}$ . ( $< 100\mu\text{s}$  pour les **cartes V2**)

Vous pouvez mettre en place jusqu'à 32 traitements d'interruption (16 pour les cartes V2) différents (périodiques ou liées aux entrées).

Le traitement d'interruption doit se terminer par l'instruction **iReturn**

Lorsqu'un traitement d'interruption n'a plus lieu d'être, vous pouvez le désactiver en indiquant :

Une période de 0 pour les interruption Periodique,

Un numéro d'entrée = 0 pour une interruption liée aux entrées.

La gestion des interruptions est exigeante en ce qui concerne la disponibilité du processeur. Nous vous conseillons donc d'utiliser cette fonction avec précaution avec les **cartes V2**.

## **Interruptions périodiques**

Mise en place à l'aide de la commande **SetTick**.

Un saut au label indiqué sera réalisé suivant la période précisée lors de la mise en place de l'interruption.

**SetTick** InterruptionNo, Période, Label

InterruptionNo : Numéro de l'interruption de 1 à 32 (1 à 16, **cartes V2**)

Période : Période d'interruption en milliseconde

Label : Label où trouver le traitement d'interruption.

Exemple d'interruption périodique :

```
SetTick 1, 500, OnInt1 ' Mise en place d'une interruption périodique de
500ms
do
... ' Code de l'application
Loop
OnInt1 :
Out 5, not Out(5) ' Changer l'état de la sortie 5
iReturn
```

Il est recommandé d'utiliser la fonction [Clock.PeriodicSub](#)

## **Interruptions Liées à l'état des entrées**

Afin d'alléger l'écriture de l'application et réagir rapidement à un changement d'état d'entrée, vous pouvez mettre en place un traitement d'interruption qui réalisera ce contrôle et exécutera le code souhaité.

Ce type de traitement peut prendre en charge :

- Le changement d'état d'une entrée (passage à 0 ou à 1),
- Le passage de l'état 0 à l'état 1 d'une entrée (Front montant),
- Le passage de l'état 1 à l'état 0 d'une entrée (front descendant).

La mise en place de ce traitement se fait à l'aide de la commande **SetInputInt**.

**SetInputInt :**

Syntaxe : InterruptionNo, EntreeNo, Type, Label

InterruptionNo : Numéro de l'interruption de 1 à 32 (ou 1 à 16, ou 1 à 15 cartes V2)

EntreeNo : Numéro de l'entrée à surveiller (0 à 255)

Type : Type de contrôle

1 pour contrôler tous les changements d'état (INPUT\_INT\_DF)

2 pour contrôler le passage de l'état 0 à l'état 1 de l'entrée (INPUT\_INT\_DFM)

3 pour contrôler le passage de l'état 1 à l'état 0 de l'entrée (INPUT\_INT\_DFD)

4 pour contrôler tous les changements d'état, 1 seule fois

(INPUT\_INT\_DF\_ONESHOT)

5 pour contrôler le passage de l'état 0 à l'état 1 de l'entrée, 1 seule fois

(INPUT\_INT\_DFM\_ONESHOT)

6 pour contrôler le passage de l'état 1 à l'état 0 de l'entrée, 1 seule fois

(INPUT\_INT\_DFD\_ONESHOT)

Label : Label où trouver le traitement d'interruption.

NB : pour annuler une interruption, il suffit de relancer exactement la même commande SetInputInt, mais en mettant « -1 » comme numéro de l'entrée.

Exemple : Gestion d'une entrée fin de course

*' Fin de course de type Normalement Fermé sur l'entrée 8*

*SetInputInt 1, 8, 3, OnFDC1*

*do*

*... 'Code de l'application*

*loop*

*' traitement entrée fin de course*

*OnFDC1:*

*StopAxeID 1 'Arrêt de l'axe 1 (V3) ou bien StopAxes 1 (V2)*

*iReturn*

## **Interruptions sur positions d' Axe**

**SetCaptureInt** : Interruption sur position axe atteinte. L'interruption est automatiquement effacée.

AxeNumber=0 pour annuler l'interruption.

Syntaxe : **SetCaptureInt** IntNumber, AxeNumber, CapturePosition, Label

IntNumber : numéro d'instance 1 à 32 (V3), 1 à 16 (V2)

AxeNumber (V3) : 1 à 6 (sur Axes motorisés), 7 à 9 (sur Entrées codeurs), 10 à 16 (sur

Compteurs sur entrées rapides)

AxeNumber (V2) : 1 à 5 (sur Axes motorisés), 6 (sur Entrées codeurs),

7 à 8 (sur

Compteurs A

et B sur entrées rapides)

CapturePosition : position cible (en pas moteur)

Label : Label de l'interruption

Exemple : **SetCaptureInt** 10, 2, CibleEtiquette, OnPosEtiquette

## **Et pour la carte V3 uniquement :**

**SetCaptureIDOnInputInt** : Configuration des captures de position sur interruption entrées rapides (16 à 22). S'utilise avec GetCapturePos(AxeID)

L'intérêt de cette commande est de permettre la capture de positions sur les axes, indépendamment de l'exécution du programme PLC Basic. Ainsi les captures étant prioritaires sur l'exécution du programme PLC Basic, celles-ci ont lieu avec une parfaite régularité sans variation dans les temps de réponse.

Syntaxe : **SetCaptureIDOnInputInt** EntréeNo, AxeID

Exemple : **SetCaptureIDOnInputInt** IN\_CEL\_ETIQUETTE, AXE\_ETIQUETTE

NB : Ces interruptions ne fonctionnent pas sur un forçage manuel des entrées, mais seulement sur un signal physique.

L'entrée doit également avoir été configurée en mode IT ou compteur (paramètres 220 à 226).

Une seule entrée peut être affectée à un axe. Toute réaffectation d'un axe sur une entrée, annule et remplace l'interruption précédemment déclarée.

**GetCapturePos**( : Lecture de la position capturée sur un axe pendant l'interruption. (Configurée avec SetCaptureIDOnInputInt).

Syntaxe : **GetCapturePos**(AxeID)

Exemple : ? "Position Capturée = ",

**GetCapturePos**(AXE\_ORIENTEUR\_ENTREE)

## **UTILISATION DES ENTRÉES RAPIDES**

### **UTILISATION DES ENTRÉES RAPIDES DE LA CARTE V3 (16 à 22)**

#### **En mode Codeur**

Nous pouvons disposer de 3 codeurs car 2 entrées rapides sont requises pour chacun des codeurs.

Canal 0 : entrées 21 et 22. Fréquence jusqu'à 1Mhz

Canal 1 : entrées 19 et 20. Fréquence jusqu'à 50 khz

Canal 2 : entrées 17 et 18. Fréquence jusqu'à 50 khz

Les paramètres 221 à 226 de la carte sont à configurer en Mode 4 (4X, tous les fronts sont pris en compte), ou en Mode 3 (2X, 1 front sur 2 est pris en compte).

Ces paramètres sont réglables directement depuis:

Paramètres / Configuration générale / Entrées/Sorties

**Entrées avancées DIN16 à DIN22**

**Entrée # 16**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur

**Entrée # 17**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur     Codeur 2X     Codeur 4X

**Entrée # 18**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur

**Entrée # 19**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur     Codeur 2X     Codeur 4X

**Entrée # 20**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur

**Entrée # 21**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur     Codeur 2X     Codeur 4X

**Entrée # 22**  
 Polarité inversée (contact type NC)     Standard     Interruption     Compteur

**GetEncoder(** : Renvoie la position d'une entrée codeur

Syntaxe : **GetEncoder**(Canal) 'Canal : de 1 à 3.

Exemple : SetMDW 3018, **GetEncoder**(3)

**SetEncoder** : Affectation d'une valeur à une entrée codeur

Syntaxe : **SetEncoder** Canal, Valeur 'Canal : de 1 à 3.

Exemple : **SetEncoder** 3, 0 'Mise à 0 de l'entrée codeur n°3

## En mode Compteur

On peut donc disposer de 7 compteurs. Les paramètres 220 à 226 de la carte sont à configurer :

En mode 0 (standard), le rafraîchissement du tableau stockant l'état des entrées a lieu toutes les millisecondes.

En mode 1 (sur interruption « IT »), le rafraîchissement du tableau stockant l'état des entrées a lieu à chaque interruption y accédant.

En mode 2 (Mode compteur), le rafraîchissement du tableau stockant l'état des entrées a lieu à chaque front (montant ou descendant, selon configuration de la polarité des entrées (paramètre 200 de la carte)).

La période entre 2 évènements peut être lue dans les registres modbus 32 bits (Input registers) 1130 à 1142.

**GetCnt(** : Lecture d'un des compteurs associés aux entrées rapides (16 à 22 pour la carte **V3**, **A et B** pour la **carte V2**).

Le résultat est un entier non signé.

Syntaxe : **GetCnt**(CompteurNo)

CompteurNo = 1 à 7 pour la **carte V3**

Exemple : SetMDW 3018, **GetCnt**(4) 'Ecrit à l'adresse 3018 la valeur lue au compteur n°4

**SetCnt** : Écriture dans les compteurs des entrées rapides 16 à 22. L'interpCNC dispose de 7 entrées rapides qui peuvent être utilisées comme entrées de

comptage.

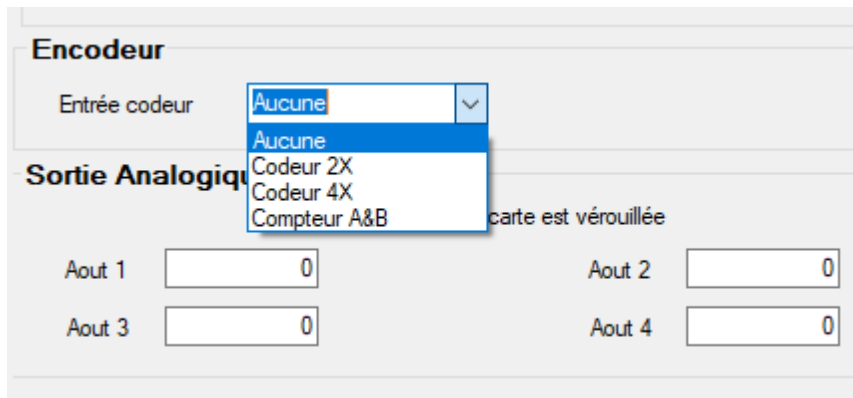
**Syntaxe :** **SetCnt** CompteurNo, Valeur  
Compteur No = 1 à 7  
Valeur = Valeur du compteur

**Exemple :** **SetCnt** 4, 0 'Remet à 0 le compteur 4

## UTILISATION DES ENTRÉES RAPIDES DES CARTES V2

**Avec les cartes V2**, le mode de fonctionnement des entrées rapides se règle dans:

Paramètres / Configuration générale / Entrées/Sorties :



### **GetEncoder**

Lecture du compteur associé aux entrées codeur A et B.

**Exemple :**

PositionCodeur = GetEncoder

? PositionCodeur

**GetCnt( CompteurNo )** -> CompteurNo = Numéro de l'entrée compteur 1 ou 2.

Résultat : Entier non signé

Lecture d'un des compteurs associés aux entrées rapides A et B.

Ces compteurs sont utilisables si le paramètre « Entrée codeur » est configuré en mode Compteur.

**SetCnt** CompteurNo, Valeur

CompteurNo : 0 pour écrire la position codeur / 1 pour écrire le compteur entrée A /

2 pour écrire le compteur entrée B

Valeur = Valeur du compteur

Écriture dans les compteurs d'entrée rapide A et B ou dans le compteur codeur.

L'InterpCNC dispose de 2 entrées rapides qui peuvent être utilisées comme entrées de comptage.

**Sur la carte V2.2**, deux entrées de comptage rapide (2\*- A1 et B1) sont disponibles.

Elles peuvent être utilisées indépendamment l'une de l'autre (2 compteurs indépendants) ou en entrée codeur (quadrature).

Sur l'emplacement-connecteur dédié à l'entrée codeur 2\*, vous disposez également du +5V et du 0V qui peuvent servir à alimenter le codeur.



**Sur la carte V2.4**, deux entrées de comptage rapide (A1 et B1) sont disponibles. Elles correspondent aux entrées 14 et 15. Elles sont nommées respectivement ENCA et ENCB sur le bornier "INPUT B". Elles peuvent être utilisées en tant que compteur ou en entrée codeur (quadrature). Vous pouvez si besoin utiliser les broches +5V et du 0V du connecteur voisin (RS485), pour alimenter le codeur.

- soit en mode codeur ou compteur selon le réglage sélectionné (Codeur 2X, 4X, compteur A&B)

La sélection du mode Compteur A&B permet à l'InterpCNC V2.4 de disposer uniquement d'un compteur A.

- quel que soit le mode de fonctionnement sélectionné pour les entrées **ENCA** et **ENCB** celles-ci fonctionnent aussi comme des entrées TOR classiques, accessibles en permanence en tant que **IN14** et **IN15**.

## COMMUNICATION VIA MODBUS AVEC DES DRIVERS OU VARIATEURS (V3)

Chacun des ports COM1 et COM2 peut être utilisé pour le pilotage en **MODBUS**, de drivers ou variateurs. Dans ce cas, la carte **InterpCNC V3** sera « **Maître** ». (NB: Les cartes V2 ne disposant pas du mode **Modbus Master**, ne peuvent piloter des appareils externes).

Au préalable il sera nécessaire d'étudier la documentation de votre driver/variateur, pour connaître les adresses Modbus de ses principaux registres à utiliser:

### **Communication de base**

→ Mode, Baud Rate, Bits de données, Parité, Bits de Stop, ID en tant qu'esclave

En principe, il faudra définir ces réglages soit depuis le clavier en façade du variateur, soit à l'aide d'un logiciel de paramétrage et connexion au PC. D'autres réglages peuvent être nécessaires, tels que mode de fonctionnement des entrées, sens de rotation, etc...

Il faudra bien évidemment saisir des paramètres identiques du côté de la carte InterpCNC V3 (registres 400 à 405 pour l'utilisation du COM1, registres 420 à 425 pour l'utilisation du COM2).

Paramètres supplémentaires :

COM1 :           registre 408 (délai mini entre chaque trame)  
                   registre 409 (nombre de tentatives avant retour d'une erreur de communication)

COM2 :       registre 426 (délai mini entre chaque trame)  
               registre 427 (nombre de tentatives avant retour d'une erreur de communication)

### **Utilisation des Commandes MBRTU**

La commande **MBRTU.Send** enverra une trame Modbus via le port COM1 ou 2, destinée à l'ID du variateur (Esclave), avec un type de donnée spécifié (exemple : Holding Register).

La valeur à envoyer sera lue depuis un registre ou bit source (ou plusieurs consécutifs) chez le Maître (la carte InterpCNC V3), pour être écrit(s) vers le registre ou bit spécifié (ou plusieurs consécutifs) de l'esclave.

Ainsi par exemple, on pourra écrire au registre de vitesse du variateur pour faire varier cette dernière.

Structure de la commande **MBRTU.Send** :

(les membres d'une commande peuvent être soit des variables, soit des valeurs directes)

**MBRTU.Send** COMPort, JobType, SlaveID, MasterAddress, SlaveAddress, DataSize, MIRegisterNumber

**COMPort** est le numéro du port COM utilisé (1 ou 2)

**JobType** est le type d'action (Read ou Write) selon le type d'accès de la donnée (Lecture/Ecriture, Lecture seule):

***MB\_RTU\_WRITE\_HOLDING\_REG*** ou «5», **et**  
***MB\_RTU\_READ\_HOLDING\_REG*** ou «2»  
***MB\_RTU\_WRITE\_COIL*** ou «4», ***MB\_RTU\_READ\_COIL*** ou «0»  
***MB\_RTU\_READ\_INPUT*** ou «1», ***MB\_RTU\_READ\_INPUT\_REG*** ou «3»

**MasterAddress** est l'adresse source chez le Maître

**SlaveAddress** est l'adresse de destination chez l' esclave

**DataSize** est le nombre de données consécutives (registres ou bits)

**MIRegisterNumber** est un registre dans lequel sera retourné un code erreur (0 = pas d'erreur)

Il est à noter que le mode d'accès aux Input Registers ou Input Bits en ajoutant +100000 à l'adresse, fonctionne également.

Il peut permettre par exemple de les lire directement sur le Maître dans le mode ***MB\_RTU\_READ\_HOLDING\_REG*** (au lieu de ***MB\_RTU\_READ\_INPUT\_REG***) afin de copier directement la valeur d'un Input Register chez le Maître, vers un Holding Register chez l'esclave. Exemple :

```
const INPUTS_SHADOWS = 3022
const RESULT_COM_VAR5 = 3036
```

```
MBRTU.Push 2, MB_RTU_WRITE_HOLDING_REG, 5, 101000, INPUTS_SHADOWS,
2, RESULT_COM_VAR5
```

Cette commande copiera donc via le COM2 , l'état des entrées physiques 0 à 31 de la carte InterpCNC vers les registre 3022 et 3023 de l'esclave. (Le résultat de la communication sera stocké chez le Maître en 3036).

La commande **MBRTU.Push** est identique, à ceci près que les commandes successives se trouveront dans une file d'attente, exécutées dans l'ordre d'arrivée.

**MBRTU.Push** COMPort, JobType, SlaveID, MasterAddress, SlaveAddress, DataSize, MIRegisterNumber

Exemple :

**MBRTU.Push** COMVariateurs, MB\_RTU\_WRITE\_HOLDING\_REG, VAR\_CONVOYEUR, HZ\_TAPIS\_CONVOYEUR, REGISTRE\_SPEED\_VAR, 1, RESULT\_COM\_VAR5  
pourrait aussi s'écrire

**MBRTU.Push** 2, 5, 5, HZ\_TAPIS\_CONVOYEUR, 3, 1, 3030

La commande **MBRTU.Flush**(PortCOM) permet de vider la file d'attente des commandes **MBRTU.** non encore traitées.

Exemple :

**MBRTU.Flush**(1)

La fonction **MBRTU.Count**(PortCOM) permet de retourner le nombre de commandes actuellement dans la file d'attente.

Exemple :

```
if MBRTU.Count(COMVariateurs)>25 then
  ? "Erreur envoi commande variateur"
endif
```

Elle peut être utilisée par exemple pour détecter un problème de communication. En effet, un nombre de commandes non traitées qui s'accumulent dans la file d'attente, sont révélateurs.

## COMMUNICATION\_PAR\_ETHERNET\_V3

La carte **InterpCNC V3** est équipée d'un port Ethernet, qui lui permet non seulement d'être utilisée à distance par ICNCStudio, mais également de communiquer entre elles.

L'intérêt peut-être par exemple de constituer un réseau de plusieurs postes automates ou contrôleurs d'axes sur une ligne de production, n'ayant besoin que d'un seul programme sur la carte dite « Maître » (ou « serveur »).

En Ethernet, 2 protocoles coexistent : l'UDP et le TCP.

L'UDP est le plus simple à utiliser. Le TCP quant à lui, s'il est plus lourd à mettre en œuvre, est beaucoup plus fiable et plus sécurisé.

De ce fait c'est le TCP qui a été retenu pour la communication entre cartes InterpCNC.

Ainsi nous pourrons accéder aux registres, en lecture ou lecture/écriture, de chacune des cartes connectées, et même leur transmettre des commandes Modbus.

### Utilisation, commandes et fonctions

Voici un programme de démonstration simple :

```

1  ' SOPROLEC InterpCNC V3 PLCBasic
2
3  ' Copie état des entrées ICNC vers registre robot adresse 0
4  ' Copie Entrées robot adresse 10000 vers sorties ICNC
5  const ROBOT_OUTPUT_ADDR = 0      ' Adresse pour écriture des sorties sur robot
6  const ROBOT_INPUT_ADDR = 10000  ' Adresse pour lecture des entrées robot
7  const SCAN_RATE=1              ' Période de rafraichissement (ms)
8
9  const ICNC_INPUT_ADDR=1000      ' Adresse modbus de l'état des entrées 0..15 de l'ICNC (Input Register)
10 const ICNC_OUTPUT_ADDR=2160    ' Adresse modbus de l'état des sorties OUT0..15 de l'ICNC (Holding register)
11
12 const IP_ROBOT="192.168.10.11"
13
14 ' Pour statistique communication (Adresses de registres internes ICNC)
15 const ETHERNET_RX_FPS= 1198
16 const ETHERNET_TX_FPS= 1199
17 const ETHERNET_TCP_CLIENT_ERROR=1201
18 const ETHERNET_TCP_CLIENT_SUCCESS=1203
19
20 ' Création socket de communication
21 SockRobot = MBClient.Open(IP_ROBOT, 502, MBB_SOCK_BUSY)
22
23 do
24 ' Quand le socket est disponible
25 if not getMB(MBB_SOCK_BUSY) then
26 ' Copy IN0..15 dans holding Register
27 SetMW REG ICNC_INPUT_COPY, GetInputMW(ICNC_INPUT_ADDR)
28 ' Envoie entrées ICNC 0..15 sur sorties robot
29 MBClient.WriteVar SockRobot, MB_WRITE_HOLDING_REG_TO_HOLDING_REG, REG_ICNC_INPUT_COPY, 1, ROBOT_OUTPUT_ADDR, 0
30 'Copie entrées robot vers sorties 0..15 de ICNC INPUT_ADDR
31 MBClient.ReadVar SockRobot, MB_READ_HOLDING_REG,ROBOT_INPUT_ADDR, 1, ICNC_OUTPUT_ADDR, SCAN_RATE
32 endif
33
34 ' Pour statistique communication
35 SetMW REG ETHERNET_RX_FPS, GetInputMW(ETHERNET_RX_FPS)
36 SetMW REG ETHERNET_TX_FPS, GetInputMW(ETHERNET_TX_FPS)
37 SetMW REG TCP_CLIENT_ERROR, GetInputMW(ETHERNET_TCP_CLIENT_ERROR)
38 SetMW REG_TCP_CLIENT_SUCCESS , GetInputMW(ETHERNET_TCP_CLIENT_SUCCESS)
39
40 loop
41

```

Nous supposons ici que notre carte Serveur s'adresse à un Robot équipé d'une carte Client dont l'adresse IP est 192.168.10.11

→ `const IP_ROBOT="192.168.10.11"`

Nous définissons tout d'abord en tant que constantes les adresses des zones mémoire où écrire et lire sur la carte Client (`const ROBOT_OUTPUT_ADDR = 0` et `const ROBOT_INPUT_ADDR = 10000`), de même pour la carte Serveur (`const ICNC_INPUT_ADDR=1000` et `const ICNC_OUTPUT_ADDR=2160`).

Les constantes qui suivent ne sont pas indispensables, elles servent ici juste à calculer des statistiques de connexion.

### 1) Ouvrir un « Socket »

Prérequis : choisir un bit utilisateur (coil), qui fera état de la disponibilité du socket pour recevoir les requêtes.

Exemple : MBB\_SOCK\_BUSY, à l'adresse Modbus 96

`SockRobot = MBClient.Open(IP_ROBOT, 502, MBB_SOCK_BUSY)`

La fonction **MBClient.Open** permet d'ouvrir un Socket, et retourne un n° de Socket que l'on stockera dans une variable (ici : SockRobot)

→ paramètres : adresse IP du client, n° port, nom du bit d'Etat

### 2) Envoyer des requêtes en lecture ou écriture

**NB** : Le Socket est disponible pour recevoir des requêtes en lecture ou écriture, lorsque son bit d'état (ici, MBB\_SOCK\_BUSY), est à 0.

→ `if not getMB(MBB_SOCK_BUSY) then`

**REG\_ICNC\_INPUT\_COPY** est un registre utilisateur 16 bits, disons à l'adresse 3000,

et **ICNC\_INPUT\_ADDR** (adresse 1000 en lecture seule) correspond au mappage sur 16 bits des entrées numériques 0 à 15 de la carte Serveur).

Ainsi :

SetMW **REG\_ICNC\_INPUT\_COPY**, GetInputMW(**ICNC\_INPUT\_ADDR**)  
va recopier l'état des entrées 0 à 15 du Serveur, à l'adresse 3000.

Requête en écriture :

MBClient.WriteVar SockRobot,  
**MB\_WRITE\_HOLDING\_REG\_TO\_HOLDING\_REG,**  
**REG\_ICNC\_INPUT\_COPY, 1, ROBOT\_OUTPUT\_ADDR, 0**

La fonction **MBClient.WriteVar** permet d'envoyer une requête en écriture au Client :

→ paramètres :

adresse IP du client, la commande

**MB\_WRITE\_HOLDING\_REG\_TO\_HOLDING\_REG**, adresse de début à copier (côté Serveur), nombre de registres à copier, adresse de destination (côté Client), temps alloué en ms.

Requête en lecture :

MBClient.ReadVar SockRobot, **MB\_READ\_HOLDING\_REG,**  
**ROBOT\_INPUT\_ADDR, 1, ICNC\_OUTPUT\_ADDR, SCAN\_RATE**

La fonction **MBClient.ReadVar** permet d'envoyer une requête en lecture au Client :

→ paramètres :

adresse IP du client, la commande **MB\_READ\_HOLDING\_REG**, adresse de début à copier (côté Client), nombre de registres à copier, adresse de destination (côté Serveur), temps alloué en ms.

NB : On peut empiler jusqu'à 10 requêtes (Read ou Write).

Le Bit d'état **MBB\_SOCKET\_BUSY** retombe à zéro après les avoir envoyées.

### **3) Fermer un Socket**

On utilise la fonction **MBClient.Close**

→ paramètres : n° de socket

exemple : **MBClient.Close** SockRobot

NB : Les Sockets sont automatiquement fermés lors de l'arrêt du programme automate.

## **Introduction :**

Le DMX est un protocole de communication très répandu dans le monde du spectacle, permettant le pilotage de différents appareils (jeux de lumière, actionneurs motorisés (patiences, tournettes, etc...).

Le DMX 512 existe en mode Device (=slave) (pour les variateurs de lumières ou des projecteurs) ou en mode Master pour piloter les Devices (exemple : console de lumières).

L'interpCNC **V3** vous autorise les 2 modes de fonctionnement :

Le mode Device disponible sur le port COM1

Le mode Master disponible sur le port COM2.

L'interpCNC **V2** autorise sur son port COM uniquement le mode Device.  
 Pré-requis: les cartes V2 doivent être mises à jour avec la version DMX de leur firmware, téléchargeable depuis leur page article sur le site SOPROLEC :  
<https://www.soprolec.com/shop/category/controleur-d-axes-1>

En mode Device, vous pouvez donc piloter les sorties ou des moteurs à partir d'une console DMX.

En mode master, vous pouvez programmer des séquences de contrôle d'appareil en interagissant avec les entrées de l'automate.

## **Raccordement :**

Connecter votre console DMX512 à la carte InterpCNC est très simple. Il suffit de raccorder les signaux D+ (fil vert) et D- (fil jaune) issus de la prise DMX de votre console, sur les entrées D+ et D- du port COM de la carte InterpCNC.



## **UTILISATION DU DMX AVEC LA CARTE V3**

Dans le tableau des paramètres, il s'agit de configurer le port COM1 en mode DMX :

Paramètres	
<b>Port COM1</b>	
Mode :	DMX Slave
Communication settings	
Baud rate :	256 000 bauds
Data bits :	8 bits
Parity :	None (Aucune)
Stop bit :	1 bit
Modbus Slave settings	
Slave ID :	1
Modbus master settings	
Inter frame delay (ms) :	0
Max retry :	3
DMX Settings	
Base address :	1
Channels used :	512

L'adresse Base est le n° du 1<sup>er</sup> canal DMX à partir duquel on veut travailler, et le nombre de canaux utilisés indiquera la plage des canaux utilisés depuis l'Adresse Base.

Le paramètre « Base address » permet de modifier l'adresse DMX de l'automate sans renuméroter les canaux utilisés dans le programme automate.

**I) Utilisation du DMX Device** dans votre programme :

Les fonctions disponibles sont :

**StsBit(STS\_DMX\_CONNECTED)** → Ce bit de status indique si une liaison DMX est établie (Si à 1).

**IsDMXReceived** → Indique qu'une trame DMX vient d'être reçue (Si à 1). Il est automatiquement effacé après sa lecture.

Exemples :

```
if IsDMXReceived then
    ...
endif
```

ou encore

```
SetMB DMXReceived, IsDMXReceived
(→ recopie de l'état du bit système, dans un bit utilisateur nommé
DMXReceived)
```

Lorsque ce bit est à 1, c'est le moment de lire les canaux à l'aide des fonctions suivantes :

**GetDMX(n° canal (1 à 512))** → Récupère la valeur entre 0 et 255 présente sur ce canal.

Remarque : Le numéro de canal réellement exploité dépend du paramètre « Base address ».

Si le paramètre vaut 1, la commande GetDMX(1) retourne bien la valeur du premier canal DMX.

Si le paramètre « Base address » vaut 10, la commande GetDMX(1) retournera en réalité la valeur du canal 10.

Exemples :

```
vitesse = GetDMX(2)
ou
if GetDMX(1)>127 then
    Out Sortie1 = 1
else
    Out Sortie1 = 0
endif
```

ou encore

**GetDMX16(n° du 1er canal (1 à 511))** → Récupère les valeurs de 0 à 255 de 2 canaux consécutifs,

et reconstitue une valeur de 0 à 65535

Formule : valeur du 1er canal + valeur du 2e canal\*256

**II) Utilisation du DMX Master** dans votre programme :

**SetDMX** N° canal, ValeurDMX → Ecriture d'une valeur 8 bits sur un canal DMX

**SetDMX16** N° canal, ValeurDMX → Ecriture d'une valeur 16 bits sur 2 canaux DMX consécutifs.

**SetDMXMaster** MasterLevel → Pondération des l'ensemble des canaux DMX.

Exemple :

SetDMXMaster 255 ' les canaux DMX seront transmis tels que donnés par les commandes

SetDMX ou SetDMX16

SetDMXMaster 0 ' Les canaux seront envoyés avec une valeur 0 (black out)

SetDMXMaster 127 ' Les canaux seront envoyés avec une valeur divisée par 2

L'ensemble des canaux et également, la valeur de DMX Master sont accessibles en modbus dans les registres en lecture/écriture des adresses 5000 à 5512.

## **UTILISATION DU DMX DEVICE AVEC LA CARTE V2**

Les fonctions introduites par ce Firmware sont :

IsDMXReceived → Si à 1, indique qu'une trame DMX vient d'être reçue. Dans ce cas, c'est le moment de lire les canaux, par :

GetDMX(n° canal) → Récupère la valeur de 0 à 255 présente sur ce canal

Exemples :

```
vitesse = GetDMX(2)
```

```
ou
```

```
if GetDMX(1)>127 then
```

```
    Out Sortie1 = 1
```

```
else
```

```
    Out Sortie1 = 0
```

```
endif
```

### **Programme simple de démonstration**

Supposons ici, que nous souhaitons utiliser les canaux 1, 2 et 3 d'une console DMX, pour piloter :

- une sortie numérique tout ou rien (Canal 1 → OUT 1)
- un moteur (Axe1) dont on peut régler la vitesse (Canal 2) et la position (Canal 3)

Nous fixerons l'accélération et la décélération à 10 Khz, et prévoyons le cas d'un Time Out au bout duquel on pourrait faire une action particulière si la connexion DMX était interrompue pendant par exemple plus de 2 secondes.

Après avoir défini nos constantes et initialisé nos variables de vitesse, de position, et l'état des sorties à 0, tout se situe dans le Do... Loop

NB : L'adresse 7000 est le premier des registres utilisateurs en Ram. Nous l'utiliserons pour compter le nombre de trames reçues depuis le 1<sup>er</sup> démarrage du programme.

Pour faire un exemple simple de démonstration, la vitesse sera la valeur du canal

DMX2 \* 10, soit de 0 à 2550 Hz  
et la position cible sera la valeur du canal DMX3 \* 100, soit de 0 à 255000 pas.

Dans un projet réel, nous calculerons et utiliserons plutôt la résolution de l'axe motorisé.

```

1  ' Declaration des constantes
2  const ACCEL = 10 'Accel/decel des mouvement moteur (KHz/s)
3  const TIMEOUT_DMX = 2000 'Timeout reception DMX provoquant l'arrêt des mouvements et l'activation du frein (ms)
4
5  ' Adresses DMX
6  const DMX_ON_OFF = 1 ' Adresses Marche/arrêt sortie OUT1
7  const DMX_V1 = 2 ' Adresses Vitesse moteur 1
8  const DMX_POS1 = 3 ' Adresses Position moteur 1
9
10 'Sauvegarde position et vitesse moteur 1
11 V1=0
12 POS1=0
13
14 ' *****
15 Initialisation:
16 OUTALL 0
17 Pause 300
18 Unlock

```

```

19
20 ' Boucle programme principal
21 if stsb(8)=1 then goto Initialisation ' L'entree ENABLE a ete coupee (ex : arrêt d'urgence)
22
23 ' Gestion timeout reception DMX
24 if IsDMXReceived then
25   SetMDW 7000, GetMDW(7000)+1 ' Compteur reception de trames DMX
26   DMXReceived = 1
27   TimerTimeoutDMX = Timer + TIMEOUT_DMX
28   TimeoutDMX = 0
29 else
30   DMXReceived = 0
31 endif
32
33 ' Detection d'un timeout de reception DMX
34 if Timer > TimerTimeoutDMX and not TimeoutDMX then
35   ? "Timeout DMX"
36   TimeoutDMX = 1
37   ' Action à faire si pas de reception DMX pendant plus de TIMEOUT_DMX(ms)
38 endif
39
40 if DMXReceived then ' Si nouvelles commandes DMX
41   ' Si canal DMX ON OFF > 127, activer sortie 1, sinon, désactiver sortie 1
42   if GetDMX(DMX_ON_OFF) > 127 then
43     OUT 1, 1
44   else
45     OUT 1, 0
46   endif
47
48   ' Traitement des canaux 2 et 3 correspondant à Vitesse et position de l'axe 1
49   newV1 = GetDMX(DMX_V1)
50   newPOS1=GetDMX(DMX_POS1)
51   if NewV1 <> V1 or newPOS1 <> POS1 then
52     if V1 = 0 then
53       StopAxes 1
54     else
55       frequence = NewV1 * 10 ' transformation Valeur DMX 0..255 en fréquence de 0 à 2550Hz
56       Position = NewPOS1 * 100 ' transformation Valeur DMX 0..255 en pas moteur de 0 à 255000pas
57       moveaxe 1, ACCEL, frequence, ACCEL, Position
58     endif
59     V1 = NewV1
60     POS1 = NewPOS1
61   endif
62 endif

```

**Ligne 24 :**

Si une trame DMX est reçue, alors on incrémente le compteur de trames, et on calcule le moment du prochain Time Out, à partir du Timer qui court depuis la mise sous tension de la carte.

**Ligne 34 :** Traitement de l'éventualité d'un Time Out.

**Ligne 40 à 62 :** Traitement de la trame DMX reçue :

- La sortie 1 passe à 1 si la position du curseur du canal 1 est > 127, sinon elle passe à 0 (lignes 52 à 56)

- Seulement si la vitesse ou la position ont changé depuis la trame précédente (lignes 59 à 61), alors si la vitesse est nulle on arrête le mouvement (ligne 62 et 63), sinon on calcule la vitesse et la position cible (lignes 64 à 66), puis on lance le mouvement du moteur (ligne 67).

- On mémorise la vitesse et la position courante (lignes 69 et 70), afin de pouvoir détecter lors de la trame suivante si vitesse ou position ont changé.

Fin du programme.

## UTILISATION\_DE\_L'HORLOGE\_RTC\_V3

### Fonctions Real Time Clock (RTC)

L'interpCNC dispose d'une horloge interne permettant de gérer la date et l'heure. Cette horloge n'est cependant pas sauvegardée lors de la mise hors tension. Il convient donc de l'initialiser avant d'en exploiter les fonctions.

L'initialisation peut se faire par :

- Les commandes modbus 112, 113 et 114 (détaillées dans Modbus InterpCNC)

- Le programme automate à l'aide de la commande RTC

- Automatiquement via un serveur SNTP si l'interpCNC dispose d'un accès internet.

#### **Synchronisation automatique :**

Pour la mise à jour automatique par STNP, les paramètres 546 et 547 doivent être correctement

réglés. Le serveur SNTP utilisé est « sntp.pool.org ».

L'horloge sera alors initialisée en tenant compte de fuseau horaire indiqué dans le paramètre 547 et

de l'heure d'été/hiver si le bit b1 du paramètre 547 est actif.

Vous disposez de 2 bits de status qui permettent de déterminer l'état de la synchronisation :

stsBit(**STS\_RTC\_SYNCHRONIZED**) qui indique que l'horloge a été réglée,

stsBit(**STS\_Sntp\_CONNECTED**) qui indique qu'une connection SNTP est établie.

La synchronisation par serveur SNTP, si elle est activée, est renouvelée automatiquement toutes les heures.

#### **Commandes et fonctions PLCBasic :**

Vous disposez de 3 instructions pour exploiter l'horloge RTC. Pour chacune d'entre elles, plusieurs

codes de fonctions seront détaillés ci-dessous :

RTC **RTC\_SubCommandCode**, ... pour les commandes qui ne retournent pas de réponse,

RTC(**RTC\_SubFunction**, ...) pour les fonctions qui retournent une valeur numérique,

RTC\$(**RTC\_SubFunction**, ...) pour les fonctions qui retournent une chaîne de caractères

Les codes de fonctions sont des constantes pré-définies dans ICNCStudio (System\RTC)

Notez également que plusieurs des fonctions ci-dessous détaillées utilisent des variables de type

UnixTime. Il s'agit d'un format interne qui ne peut donc pas être manipulé

directement par des fonction arithmétiques (addition, soustraction). Pour les opérations sur ces variables, veuillez utiliser les fonctions proposées.

Les informations gérées par la RTC (date et heure) sont également disponibles dans les registres modbus (Input registers) 1987 à 1995. Vous pouvez donc y accéder depuis le PLC avec la commande GetMW( aux adresses 101987 à 101995

### Détails de la commande RTC :

Code de commande RTC (Syntaxe : RTC RTC\_SubCommand, .....

RTC **RTC\_SetTime**, hh, mn, ss

=> Réglage manuel de l'heure de l'horloge. Si la synchronisation SNTP est active, cette opération n'est pas nécessaire car l'heure sera automatiquement synchronisé.

RTC **RTC\_SetDate**, jj, mm, aa

=> Réglage manuel de la date de l'horloge. Si la synchronisation SNTP est active, cette opération n'est pas nécessaire car la date sera automatiquement synchronisé.

RTC **RTC\_SetAlarmA**, hh, mn, s, dayOfMonth\_or\_DayOfWeek[, d\_meaning=0][, mask=0]

=> Programmation de l'alarme A. Le déclenchement de l'alarme A provoque un appel à la sub

routine onAlarmA()

qui doit exister dans votre programme.

- dayOfMonth\_or\_DayOfWeek peut être un jour dans le mois de 1 à 31 si

d\_meaning = **RTC\_DAY\_OF\_MONTH**

ou un jour de la semaine de 1 à 7 si d\_meaning = **RTC\_DAY\_OF\_WEEK**

- mask permet une programmation avancée de l'alarme.

bit 0 de mask permet de masquer les secondes de l'heure

bit 1 de mask permet de masquer les minutes de l'heure

bit 2 de mask permet de masquer les heure de l'heure

bit 3 de mask permet de masquer le jour de la date

La mise à 1 d'un bit de masque permet de déclencher l'alarme quelque soit la valeur du champs

concerné.

exemple :

mask = &b1000, jour est masqué => l'alarme se déclenchera tous les jours à hh:mn:ss (quelque soit jj)

mask = &b1100, jour et heure sont masqués => Alarme périodique toutes les heures à mn:ss

(quelque soit jj, hh)

mask = &b1110, jour, heure et mn sont masqués => Alarme périodique toutes les minutes à ss

(quelque soit jj, hh, mn)

mask = &b1111, jour, heure et mn et seconde sont masqués => Alarme périodique toutes les

secondes (quelque soit jj, hh, mn, s)

RTC **RTC\_SetAlarmA**, hh, mn, s, dayOfMonth\_or\_DayOfWeek[, d\_meaning=0][, mask=0]

=> indentique à l'alarme A mais appel à une sub routine appelée onAlarmB

RTC **RTC\_StopAlarmA**

=> Désactiver l'alarme A

Les alarmes sont automatiquement désactivées lorsque le PLC passe en

mode STOP

**RTC RTC\_StopAlarmeB**

=> Désactiver l'alarme B

Les alarmes sont automatiquement désactivées lorsque le PLC passe en mode STOP

**RTC RTC\_SetAlarmAutime**, utime[, mask=0]

=> fonctionnement identique à RTC\_SetAlarmeA mais la date et l'heure d'alarme sont donnés par

une instance de type UnixTime.

Les instance de type UnixTime sont obtenue avec les fonction RTC(...) et permettent de faire

des opération simples sur les temps.

**RTC RTC\_SetAlarmButime**, utime[, mask=0]

=> indentique à l'alarme A mais appel à une sub routine appelée onAlarmB

**RTC RTC\_CalcSunPosition**, utime, latitude, longitude, MF\_AZIMUTH,

MF\_ELEVATION =>

Calcul la position du soleil à un instant donné. Les résultats sont donnée dans des registres de type

Float (numéros de registres donnés dans MF\_AZIMUTH et MF\_ELEVATION

Exemple :

```
const MY_LATTITUDE = 48.064422
```

```
const MY_LONGITUDE = 0.28127
```

```
Actualutime = RTC(RTC_GetActualUTime) ' Date et heure actuelle
```

```
RTC RTC_GetSunPosition, Actualutime, MY_LATTITUDE, MY_LONGITUDE,
SUN_AZIMUT, SUN_ELEVATION
```

### Détails de la fonction RTC(

utime = RTC(**RTC\_GetActualUTime**)

=> retourne une instance de type UnixTime représentant la date et l'heure actuels

utime = RTC(**RTC\_GetMakeUTime**, d,m,y,h,mn,s)

=> création d'une instance de type UnixTime correspondant à la date et heure indiqués

Delta\_s = RTC(**RTC\_DiffTime**, utime1, utime2)

=> retourne l'écart de temps en seconde entre deux instances de type UnixTime.

Soit, utime2-

utime1 en seconde.

sum\_utime = RTC(**RTC\_AddToUtime**, utime, +/-seconde

=> retourne une instance de type UnixTime correspondant à un décallage temporaire de l'instance

utime. Le décallage est exprimé en seconde et peut être positif ou négatif.

DayLen = RTC(**RTC\_DayLength**, day, month, year, lon, lat[, phase=0]) =>

Retourne la durée en

heure du jour.

sunriseHour = RTC(**RTC\_SunRise**, day, month, year, long, lat[, phase=0][,

do\_adjustDST=1]) => Heure de levé du soleil à la longitude et la latitude indiquées (en heure)

sunsetHour = RTC(**RTC\_Sunset**, day, month, year, long, lat[, phase=0][,

do\_adjustDST=1]) =>

Heure de couché du soleil (en heure) à la longitude et la latitude indiquées

SunRiseUtime = RTC(**RTC\_SunRiseUTime**, UTime, longitude, latitude[, type=RTC\_SUNSET\_TIME][,do\_adjustDST=1]) => Heure de levé du soleil au format UnixTime.

Peut être utilisé pour réaliser des calculs sur les heures,

SunSetUTime = RTC(**RTC\_SunsetUTime**, UTime, longitude, latitude[, type=RTC\_SUNSET\_TIME][,do\_adjustDST=1]) => Heure de couché du soleil au format

UnixTime. Peut être utilisé pour réaliser des calculs sur les heures.

### Détails de la fonction RTC\$(

RTC\$(**RTC\_GetTimeStr**)

=> Retourne une chaîne de caractère représentant l'heure RTC actuelle au format "hh:mn:ss "

Exemple : ? RTC\$(RTC\_GetTimeStr); "Message d'information horodaté"

RTC\$(**RTC\_GetDateStr**)

=> Retourne une chaîne de caractère représentant l'heure RTC actuelle au format "jj/mm/aa "

Exemple : ? RTC\$(RTC\_GetDateStr); "Message d'information horodaté"

RTC\$(**RTC\_GetDateTimeStr**)

=> Retourne une chaîne de caractère représentant l'heure RTC actuelle au format "jj/mm/aa hh:mn:ss "

Exemple : ? RTC\$(RTC\_GetDateTimeStr); "Message d'information horodaté"

## ICNCStudio

---

### ICNCStudio - Logiciel de développement et de diagnostic

La carte **InterpCNC V3** est livrée avec le logiciel **ICNCStudio**, conçu spécifiquement pour le développement et le diagnostic, dans la mise au point de vos programmes PLCBasic.

Le hardware étant différent les cartes **InterpCNC V2** s'utiliseront avec le logiciel **ICNCStudio Light**.

Il permet l'accès à l'ensemble des fonctions de la carte et aux différents paramètres. Attention, ce logiciel et en particulier les fonctions de déplacement doit être réservé aux utilisateurs avertis, ayant des connaissances avancées en programmation et/ou automatisation.

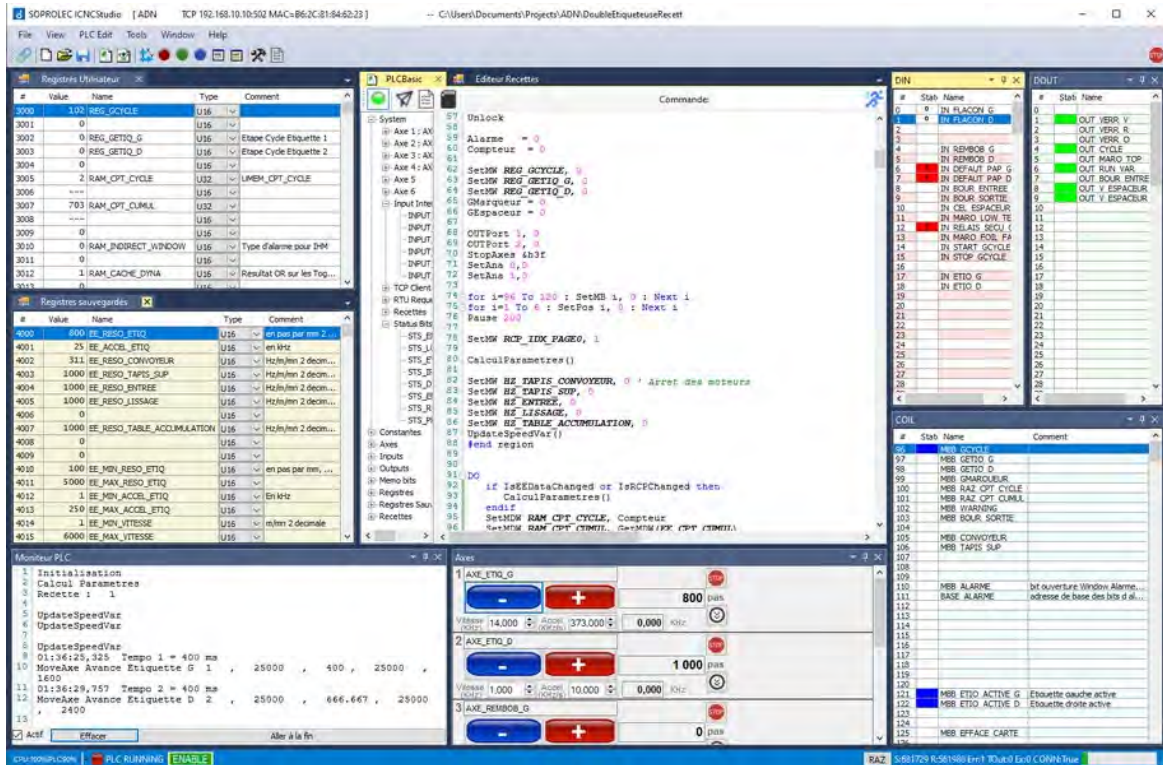
Plus qu'un outil, ICNCStudio est un véritable studio de développement logiciel (d'où son nom) pour votre carte InterpCNC.

Contrairement à la génération précédente des logiciels « Test Center », il regroupe en un seul écran l'ensemble des fenêtres dédiées à chaque type de paramètres.

Ces fenêtres ne sont plus flottantes et indépendantes car font partie d'une plus grande fenêtre commune.

Elles restent toutefois redimensionnables et déplaçables à votre convenance sur

l'espace de l'application, et vous pouvez « Verrouiller la disposition des fenêtres » (voir Onglet « Tools »).



Une aide contextuelle intégrée est disponible en appuyant sur la touche "F1", à tout endroit où se trouve votre curseur, ou bien pour le texte sélectionné dans l'éditeur (exemple: recherche de la syntaxe pour l'utilisation d'une commande ou d'une fonction).

## Registres\_Utilisateur

– Fenêtre des **Registres Utilisateur** (RAM)

Ces registres correspondent aux adresses Modbus 3000 à 3999 (V3) ou bien aux adresses Modbus 7000 à 7333 (V2)

Pour chaque Registre Utilisateur, vous définissez directement l'affichage du typage dans la colonne « type » du registre, comme suit :

#	Value	Name	Type	Co ^
3000	0	ENABLE	U16	
3001	0	GCYCLE	U16	
3002	0	GHABILLAGE	U16	
3003	0	GAVANCE_COLLERETTE	I16	
3004	0	GCOLLERETTE	U32	
3005	0	GETIQUETTE	I32	
3006	0	GCONTRE_ETIQUETTE	Float	
3007	0	GMEDAILLON	U16	

**U16 et U32** : entiers 16 et 32 bits (pf, pF)

**I16 et I32** : entiers signés 16 et 32 bits (pf, pF)

**Float** : nombres flottants (pf, pF)

Et pour la carte **V3** uniquement, en plus :

**U32\_INV** : entiers 32 bits, inversion des 2 registres 16 bits utilisés (pF, pf)

**I32\_INV** : entiers signés 32 bits, inversion des 2 registres 16 bits utilisés (pF, pf)

**Float\_INV** : nombres flottants, inversion des 2 registres 16 bits utilisés (pF, pf)

NB: pf=poids faible, pF=poids Fort

#	Value	Name	Type
3000			U16
3001			U16
3002			I16
3003			U32
3004			I32
3005			Float
3006			U32_INV
			I32_INV
			Float_INV

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier une unité de mesure, etc...)

NB: Si vous renommez un registre depuis son tableau (**Registre Utilisateur**, **Registre Sauvegardé**, ou **Recette**), une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour lesquelles des occurrences du programme Basic, le renommage devra être appliqué.

Vous pouvez aussi rechercher les occurrences du nom dans votre programme. Voir également la fonction [Chercher-Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer le registre dans son tableau.

Un ou plusieurs registres peuvent également être ajoutés aux données personnalisées pour surveiller l'évolution de leur état.

### **Déplacement des registres par glisser-dépose :**

Le glisser-déposer permet de déplacer la **déclaration** d'un registre (colonnes *Nom/Type/Commentaire*) sans toucher à sa **Valeur** afin de réassigner proprement une adresse. On peut **prendre** une ligne uniquement si son **Nom n'est pas vide** ; si le **Type** est un **type "2 lignes"** (*U32, I32, FLOAT*), la prise englobe la ligne et **celle du dessous**. On peut **déposer** seulement sur une ligne **au Nom vide** (et pour un type "2 lignes", la ligne suivante doit aussi être vide). Pendant le survol, la colonne "#" affiche  si le dépôt est possible ou  sinon. Au dépôt, le système **coupe** *Nom/Type/Commentaire* de la source et les **colle** sur la cible, en gardant les valeurs et index internes cohérents.

La sélection multiple est supportée : plusieurs lignes peuvent être déplacées ensemble.

### **Les Menus Contextuels :**

- **Sur les tableaux de Registres Utilisateur:**  
« Insert Line » et « Remove Line »

## **Registres\_Sauvegardes**

- Fenêtre des **Registres sauvegardés** (=RAM non volatile)  
Ces registres correspondent aux adresses Modbus 4000 à 4999 (**V3**)  
ou bien aux adresses Modbus 5120 à 5631 (**V2**)

NB : Pour la carte V3, cette zone de mémoire sauvegardée ne correspond pas à une classique EEprom, mais à de la mémoire RAM Ferromagnétique. Cette dernière, d'une technologie bien plus avancée, permet des cycles d'écritures presque illimités, et des temps d'accès nettement plus rapides.

### **Attention !! :**

**Pour les cartes V2, cette zone de mémoire supporte 1 million d'écritures dans chacun des registres.**

**Il faudra alors bien veiller à ne jamais faire d'écriture en boucle sur les mêmes adresses, dans cette zone de mémoire.**

Pour chaque Registre Sauvegardé, vous définissez directement l'affichage du type dans la colonne « type » du registre, comme suit :

Registres sauvegardés				
#	Value	Name	Type	Commer
4000		EE_R_COLLERETTE	FLOAT	pulse...
4001			U16	
4002		EE_R_ETIQUETTE	FLOAT	pulse...
4003			U16	
4004		EE_R_CONTRE_ETIQUETTE	FLOAT	pulse...
4005			U16	
4006		EE_R_MEDAILLON	FLOAT	pulse...
4007			U16	
4008		EE_R_ORIENTEUR_ENTREE	FLOAT	pulse...
4009			U16	
4010		EE_R_CONVOYEUR	FLOAT	pulse...
4011			U16	
4012		EE_V_ORIENT_ETIQ_INIT	U16	Hz ...
4013		EE_AD_ORIENT_ETIQ_INIT	U16	Hz ...

**U16 et U32** : entiers 16 et 32 bits (pf, pF)

**I16 et I32** : entiers signés 16 et 32 bits (pf, pF)

**FLOAT** : nombres flottants (pf, pF)

Et pour la carte **V3** uniquement, en plus :

**U32\_INV** : entiers 32 bits, inversion des 2 registres 16 bits utilisés (pF, pf)

**I32\_INV** : entiers signés 32 bits, inversion des 2 registres 16 bits utilisés (pF, pf)

**FLOAT\_INV** : nombres flottants, inversion des 2 registres 16 bits utilisés (pF, pf)

NB: pf=poids faible, pF=poids Fort

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier une unité de mesure, etc...)

NB: Si vous renommez un registre depuis son tableau (Registre Utilisateur, **Registre Sauvegardé**, ou Recette), une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour lesquelles des occurrences du programme Basic, le renommage devra être appliqué.

Voir également la fonction [Chercher-Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer le registre dans son tableau.

Un ou plusieurs registres peuvent également être ajoutés aux données personnalisées pour surveiller l'évolution de leur état.

### **Déplacement des registres par glisser-dépose :**

Le glisser-déposer permet de déplacer la **déclaration** d'un registre (colonnes *Nom/Type/Commentaire*) sans toucher à sa **Valeur** afin de réassigner proprement une adresse. On peut **prendre** une ligne uniquement si son **Nom n'est pas vide** ; si le **Type** est un **type "2 lignes"** (*U32, I32, FLOAT*), la prise englobe la ligne et **celle du dessous**. On peut **déposer** seulement sur une ligne **au Nom vide** (et pour un type "2 lignes", la ligne suivante doit aussi être vide). Pendant le survol, la

colonne “#” affiche  si le dépôt est possible ou  sinon. Au dépôt, le système **coupe** Nom/Type/Commentaire de la source et les **colle** sur la cible, en gardant les valeurs et index internes cohérents.

La sélection multiple est supportée : plusieurs lignes peuvent être déplacées ensemble.

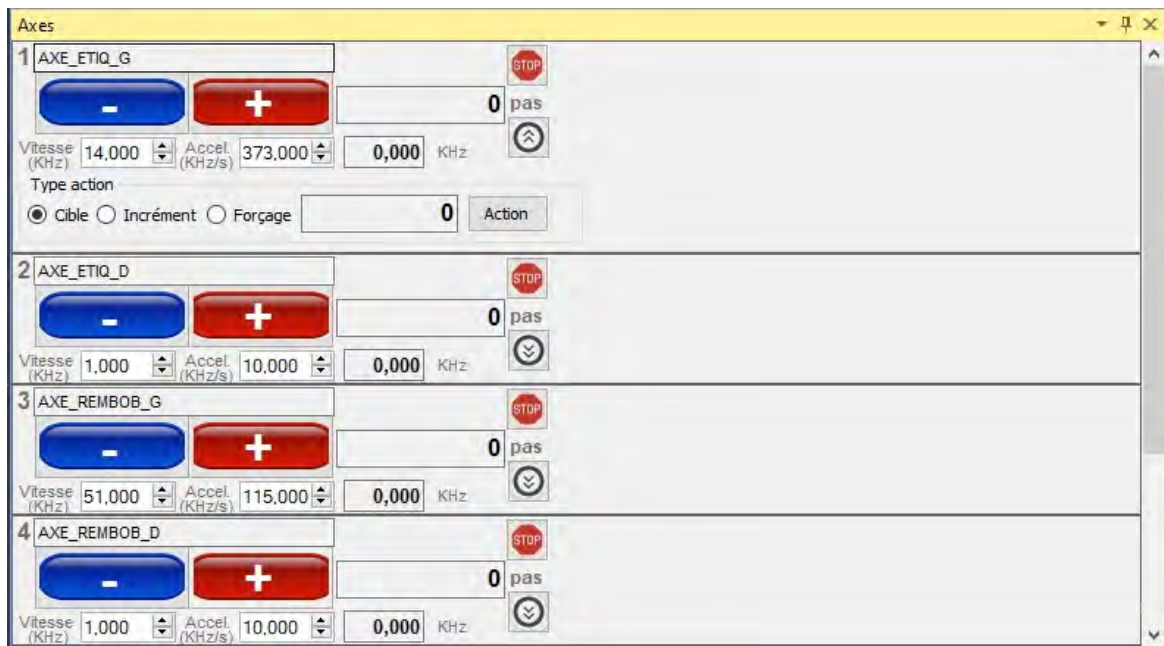
### Les Menus Contextuels :

- **Sur les tableaux de Registres Sauvegardés:**  
« Insert Line » et « Remove Line »

## Fenêtres\_axes

**Fenêtres Axes**, permettant de commander manuellement et faire des tests de mouvements sur les 6 axes.

La carte doit être au préalable en mode Enable, et Unlock.



L'icône sous le mot "pas" permet d'afficher une ligne d'options.

1) Choisir l'option

Il est ainsi possible d'aller:

- A une position cible (absolue)
- A une position incrémentée depuis la position actuelle (relative)
- Le "Forçage" permet tout simplement de forcer la valeur du compteur de position pour l'axe.

2) Entrer une valeur (en pas, i.e en pulses)

3) Cliquer sur "**Action**"

Le bouton STOP permet d'arrêter immédiatement le mouvement en cours.

NB: Si vous renommez un Axe depuis la fenêtre ci-dessus, une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour quelles occurrences du programme Basic, le renommage devra être appliqué.

Voir également la fonction [Chercher et Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer l'Axe dans son tableau.

## Moniteur

- **Fenêtre du Moniteur (=Console)**  
Dans cette fenêtre s'affichent vos Print (« ? ») en cours d'exécution du programme, permettant ainsi le débogage, ou de suivre le bon déroulement des étapes.

```

Moniteur PLC
1 UpdateSpeedVar
2 Initialisation
3
4 Calcul Parametres
5 Recette : 1
6
7 UpdateSpeedVar
8 UpdateSpeedVar
9 Acquitement Alarmes
10 -- > ALARME 13
11 GCYCLE= 0
12 GETIQ_G= 0
13 GETIQ_D= 0
14 INTERRUPTION DES CYCLES sur ALARME
15
 Actif   Effacer   Aller à la fin

```

## Digital\_inputs

### ***Fenêtre des Entrées Numériques (DIN= Digital IN)***

Cette fenêtre est en réalité des tableaux rafraîchis presque en temps réel quand la connexion est établie entre ICNCStudio et votre carte.

Dans ce tableau, vous pouvez saisir et attribuer un nom aux Bits correspondants, et forcer leur état (0 ou 1).

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier le rôle de cette entrée...)

#	Stat	Name	Comment
0		IN FLACON G	Presence Obiet G
1		IN FLACON D	Presence Obiet D
2			
3			
4		IN REMBOB G	
5		IN REMBOB D	
6		IN DEFAULT PAP G	entree default pa...
7		IN DEFAULT PAP D	entree default pa...
8		IN BOUR ENTREE	bourrae entree
9		IN BOUR SORTIE	bourrae sortie
10		IN CEL ESPACEUR	Cellule Espaceur ...
11		IN MARO LOW TEMP	marqueur tempe...
12		IN RELAIS SECU OK	
13		IN MARO FOIL FAULT	fin ruban marqueur
14		IN START GCYCLE	
15		IN STOP GCYCLE	
16			
17		IN ETIO G	Cellule Etiquette G
18		IN ETIO D	Cellule Etiquette D
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			

Supprimer forçages (R)

---

Forcer à 0 (0)

Forcer à 1 (1)

---

Insert Line

Remove Line

NB: Si vous renommez un bit depuis son tableau (Entrée **DIN**, sortie DOUT, ou un Coil), une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour lesquelles des occurrences du programme Basic, le renommage devra être appliqué.

Vous pouvez aussi rechercher les occurrences du nom dans votre programme. Voir également la fonction [Chercher-Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer le Bit dans son tableau.

Une ou plusieurs entrées peuvent également être ajoutées aux données personnalisées pour surveiller l'évolution de leur état.

### Déplacement des registres par glisser-dépose :

Le glisser-déposer dans le tableau permet de déplacer la déclaration (colonnes **Nom** et **Commentaire**) sans modifier sa **Valeur**, afin de réassigner proprement une ligne. Une ligne ne peut être prise que si son **Nom** n'est pas vide, et le dépôt n'est autorisé que sur une ligne dont le **Nom** est vide. Pendant le survol, la colonne "#" affiche  si le dépôt est possible ou  sinon, en indiquant le nombre de lignes déplacées lorsqu'il s'agit d'une sélection multiple. Au moment du dépôt, le système coupe les informations **Nom** et **Commentaire** des lignes sources pour les coller sur les lignes cibles, tout en préservant les valeurs et la cohérence interne des index.

La sélection multiple est également supportée : plusieurs lignes peuvent être déplacées ensemble

## Les Menus Contextuels :

- **Sur le tableau des Entrées (DIN) :**  
« Supprimer Forçage(R) », « Forcer à 0 », « Forcer à 1 », « Insert Line », « Remove Line », « Recherche dans le programme », « Ajouter aux données personnalisées »

## Digital\_outputs

### *Fenêtres des Sorties Numériques (DOUT = Digital OUT)*

Cette fenêtre est en réalité des tableaux rafraîchis presque en temps réel quand la connexion est établie entre ICNCStudio et votre carte.

Dans ce tableau, vous pouvez saisir et attribuer un nom aux Bits correspondants, et forcer leur état (0 ou 1).

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier le rôle de cette sortie...)

#	State	Name	Comment
0			
1		OUT VERR V	Verrine verte. active en mode cvde
2		OUT VERR R	Verrine rouge. active en mode cvc...
3		OUT VERR O	Verrine orange. active sur un war...
4		OUT CYCLE	idem Verrine Verte
5		OUT MARO TOP	Top maraueur 50ms
6		OUT RUN VAR	Controle ON/OFF moteurs asvndh...
7		OUT BOUR ENTREE	
8		OUT V ESPACEUR 1	
9		OUT V ESPACEUR 2	
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			

NB: Si vous renommez un bit depuis son tableau (Entrée DIN, sortie **DOUT**, ou un Coil), une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour lesquelles des occurrences du programme Basic, le renommage devra être appliqué.

Vous pouvez aussi rechercher les occurrences du nom dans votre programme. Voir également la fonction [Chercher-Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer le Bit dans son tableau.

Une ou plusieurs sorties peuvent également être ajoutées aux données personnalisées pour surveiller l'évolution de leur état.

**Déplacement des registres par glisser-dépose :**

Le glisser-déposer dans le tableau permet de déplacer la déclaration (colonnes **Nom** et **Commentaire**) sans modifier sa **Valeur**, afin de réassigner proprement une ligne. Une ligne ne peut être prise que si son **Nom** n'est pas vide, et le dépôt n'est autorisé que sur une ligne dont le **Nom** est vide. Pendant le survol, la colonne “#” affiche  si le dépôt est possible ou  sinon, en indiquant le nombre de lignes déplacées lorsqu'il s'agit d'une sélection multiple. Au moment du dépôt, le système coupe les informations **Nom** et **Commentaire** des lignes sources pour les coller sur les lignes cibles, tout en préservant les valeurs et la cohérence interne des index. La sélection multiple est également supportée : plusieurs lignes peuvent être déplacées ensemble

**Les Menus Contextuels :**

- **Sur le tableau des Sorties (DOUT) :**  
 « Mettre à 0 », « Mettre à 1 », « Insert Line », « Remove Line », « Recherche dans le programme », « Ajouter aux données personnalisées »

**Coils*****Fenêtre des Bits Utilisateurs (COILS, appelés aussi aussi Memo Bits)***

Cette fenêtre est en réalité des tableaux rafraîchis presque en temps réel quand la connexion est établie entre ICNCStudio et votre carte.

Dans ce tableau, vous pouvez saisir et attribuer un nom aux Bits correspondants, et forcer leur état (0 ou 1).

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier le rôle de ce bit...)

Bits Modbus n° 96 à 511 (**V3**)

Bits Modbus n° 320 à 415 (**V2**)

#	State	Name	Comment
96		MBB GCYCLE	
97		MBB GETIO G	
98		MBB GETIO D	
99		MBB GMARQUEUR	
100		MBB RAZ CPT CYCLE	
101		MBB RAZ CPT CUMUL	
102		MBB WARNING	
103		MBB BOUR SORTIE	
104			
105		MBB CONVOYEUR	
106		MBB TAPIS SUP	
107			
108			
109			
110		MBB ALARME	bit ouverture Window Alarme IHM
111		BASE ALARME	adresse de base des bits d alarme
112		Mettre à 0	(0)
113			
114		Mettre à 1	(1)
115			
116		Insert Line	
117			
118		Remove Line	
119			
120		Recherche dans le programme	
121		MBB ETIO ACTIVE G	Etiquette gauche active
122		MBB ETIO ACTIVE D	Etiquette droite active
123			
124			
125		MBB EFFACE CARTE	

NB: Si vous renommez un bit depuis son tableau (Entrée DIN, sortie DOUT, ou un **Coil**), une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour lesquelles des occurrences du programme Basic, le renommage devra être appliqué.

Vous pouvez aussi rechercher les occurrences du nom dans votre programme. Voir également la fonction [Chercher-Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer le Bit dans son tableau.

Un ou plusieurs Coils peuvent également être ajoutés aux données personnalisées pour surveiller l'évolution de leur état.

### **Déplacement des registres par glisser-dépose :**

Le glisser-déposer dans le tableau permet de déplacer la déclaration (colonnes **Nom** et **Commentaire**) sans modifier sa **Valeur**, afin de réassigner proprement une ligne. Une ligne ne peut être prise que si son **Nom** n'est pas vide, et le dépôt n'est autorisé que sur une ligne dont le **Nom** est vide. Pendant le survol, la colonne "#" affiche  si le dépôt est possible ou  sinon, en indiquant le nombre de lignes déplacées lorsqu'il s'agit d'une sélection multiple. Au moment du dépôt, le système coupe les informations **Nom** et **Commentaire** des lignes sources pour les coller sur les lignes cibles, tout en préservant les valeurs et la cohérence interne des index.

La sélection multiple est également supportée : plusieurs lignes peuvent être déplacées ensemble

### **Les Menus Contextuels :**

- **Sur le tableau des COILS (= Memo Bits) :**  
« Mettre à 0 », « Mettre à 1 », « Insert Line », « Remove Line », « Recherche dans le programme », « Ajouter aux données personnalisées »

## Editeur\_de\_texte

- **Fenêtre du Programme PLCBasic**, avec à sa gauche la **Liste des Définitions** de tous les paramètres utilisés par votre programme Basic : Corps du programme, vos Fonctions et Sous-routines, Système (bit et registres), Constantes, Axes, Entrées, Sorties, Bits Utilisateurs (Memo), Registres Utilisateurs, Registres Sauvegardés, et Recettes.

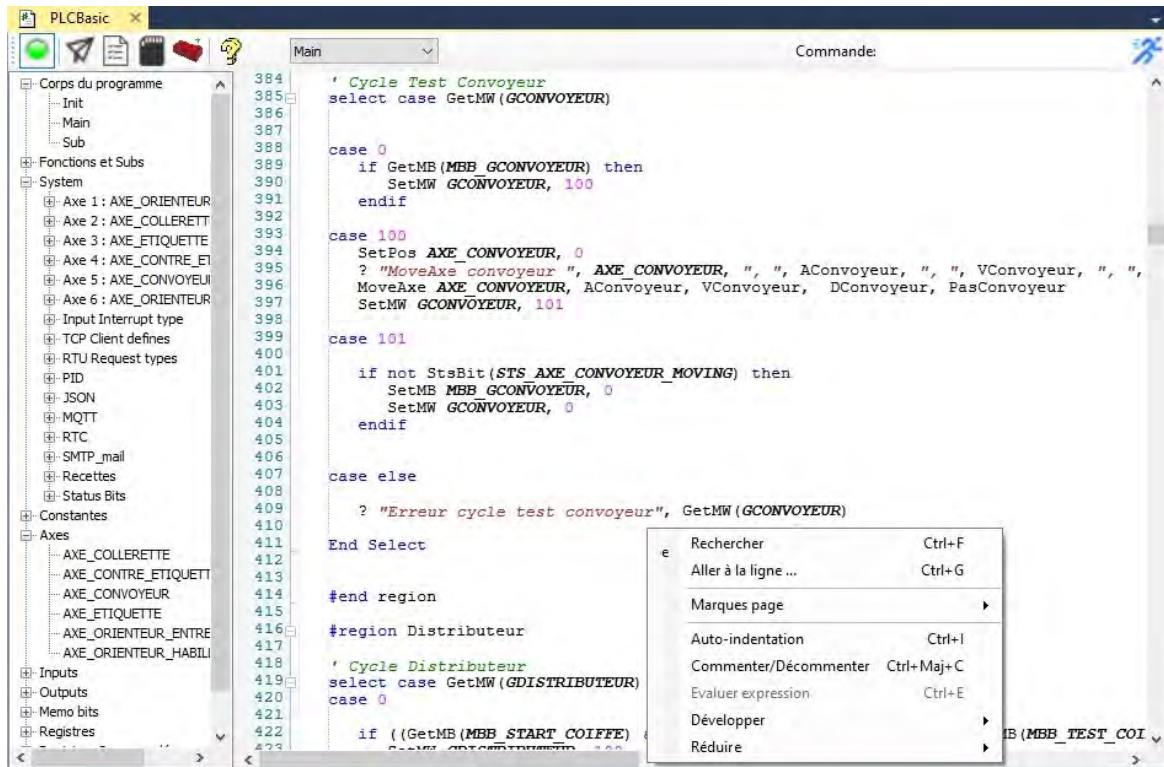
**NB :** La **Liste des définitions** est automatiquement alimentée par vos déclarations de constantes dans le programme, ainsi que vos nommages dans les tableaux. Elle est déroulable (clics sur +/-).

Depuis cette liste, vous pouvez également faire un "Drag and drop" du nom d'une variable, constante, registre, entrée, sortie ou d'un paramètre système, vers la zone de saisie de l'éditeur de texte pour l'y copier directement, ce qui constitue une forme d'autocomplétion.

Un clic sur le **voyant Vert**, permet de d'exécuter ou arrêter le programme Basic actuellement en Ram ou dans la mémoire non volatile de la carte.

- **La barre de Commande de l'éditeur**  
Celle-ci fait partie de la fenêtre du programme Basic. Lorsque le programme n'est pas en cours d'exécution (donc à l'arrêt), vous pouvez lancer une commande en Basic.  
NB : les variables utilisées dans le programme sont disponibles pour la ligne de commande.  
Exemple 1 : ? CompteurProd  
→ affichera dans le moniteur la dernière valeur contenue dans votre variable CompteurProd  
Exemple 2 : for i= 3000 to 3511:SetMW i,0 : Next i 'carte V3  
ou for i= 7000 to 7333:SetMW i,0 : Next i 'carte V2  
→ commande effaçant tous les registres utilisateur de la Ram.

La fenêtre centrale est commune au programme Basic, à la liste des Paramètres de la carte, et à l'Éditeur de Recettes. Les onglets permettent de basculer de l'une ou l'autre.



## Une lisibilité optimisée de vos programmes

Tous les : nom de Registre, nom de Bit, nom d'Entrée, et nom de Sortie, utilisés deviennent plus lisibles (Majuscules+Italiques) s'ils sont déjà déclarés dans le tableau correspondant.

Ainsi, en l'absence de Maj+Italiques, vous savez que vous êtes sûrement en présence d'une faute de frappe.

- Les fonctions, commandes et instructions du Basic apparaissent en **Bleu**.
- Les valeurs numériques apparaissent en **Rose**.
- Les commentaires (après quote ou entre quotes : ' ou ' ') apparaissent en **Vert**.
- Les contenus pour la commande « print » (« ? ») sont en **Rouge**.
- Lorsque vous cliquez sur une variable, nom de Bit, nom de Registre, nom d'Entrée ou de Sortie, l'ensemble des occurrences est montré par une surbrillance **Orange**.
- La carte étant connectée, lorsque vous laissez votre curseur sur une constante associée à un registre, ses informations contextuelles apparaîtront (adresse Modbus, type de registre (MW, MDW, etc...))

## Structure d'un programme Basic

Par défaut, le **Corps du programme** comprend un bloc "Init", ainsi qu'un bloc "Main".

Le bloc **Init** doit contenir toutes vos déclaration de constantes, ainsi que le code servant comme son nom l'indique à l'initialisation de votre programme.  
Le principe est que le code contenu dans ce bloc sera exécuté une seule fois au démarrage du programme, puis l'exécution enchainera sur le bloc **Main**.

Le bloc **Main** contiendra votre code principal (code combinatoire, cycles automates, etc...).

Tout le code contenu dans ce bloc sera répété en boucle, exactement comme s'il se trouvait entre des balises **DO... LOOP**

Vous pouvez créer autant de blocs supplémentaires que nécessaire, en cliquant sur l'icone symbolisant une **brique rouge**. Leur nommage est libre.  
L'idéal est de placer dans ce type de bloc vos fonctions, vos sous-routines, ainsi que le code correspondant aux labels de vos interruptions.  
En fin de compte, il s'agit ici du code exécuté occasionnellement, c'est à dire appelé par le programme principal (depuis le bloc **Main**).

## Les régions

Dérivée d'autres langages, la notion de région a également été implémentée dans ICNCStudio, car elle contribue également à la lisibilité de votre programme.  
Les balises **#region Nom** et **#end region**, permettent de délimiter certaines portions de programme, que vous pouvez ensuite masquer (clic sur -) ou afficher à nouveau (clic sur +).  
Ainsi vous avez moins de lignes à parcourir si vous masquez des zones déjà terminées et testées, ou qui ne concernent pas une fonctionnalité que vous êtes en train de développer.

## L'aide intégrée

En cliquant sur le point d'interrogation jaune "?", ou bien en appuyant sur la touche **F1**, vous faites apparaître les rubriques d'aide que vous pouvez parcourir ou dans laquelle vous pouvez saisir une recherche (mot clé).

Si vous avez au préalable fait une sélection dans l'éditeur (d'une instruction, d'une commande, d'une fonction, etc...), alors l'aide vous proposera soit de choisir une occurrence s'il y en a plusieurs, soit pointera directement sur la rubrique correspondante.

## Les Raccourcis clavier

**Ctrl+A** : Sélectionner Tout

**Ctrl+B** : Ajouter Marque Page

**Ctrl+Maj+B** : Supprimer Marque Page

**Ctrl+C** : Copier la sélection

**Ctrl+Maj+ C** : Commenter/Dé-commenter

**Ctrl+E** : Évaluer une expression → Si ICNCStudio est connecté à la carte, retourne la valeur actuelle de l'expression

**Ctrl+F** : Rechercher une expression (Find)

**Ctrl+G** : Aller à la ligne n° xxx (Go to line)

**Ctrl+H** : Rechercher et remplacer (Find and Replace)

**Ctrl+I** : Auto-indentation de la sélection

**Ctrl+N** : Marque Page suivant

**Ctrl+Maj+N** : Marque Page précédent

**Ctrl+O** : Quitter l'application

**Ctrl+S** : Sauvegarder les modifications

**Ctrl+U** : Met la sélection en Majuscules (Uppercase)

**Ctrl+V** : Coller la sélection

**Ctrl+X** : Couper la sélection

**Ctrl+Z** : Annuler la dernière action

**Ctrl+Maj+Z** : Rétablir la dernière action

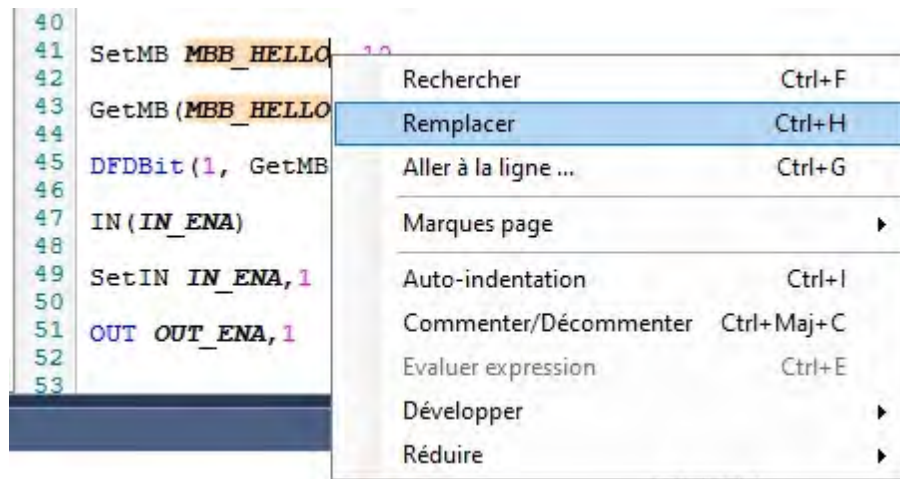
**Shift+Ctrl+C** : Commenter la zone sélectionnée

## Chercher\_Remplacer

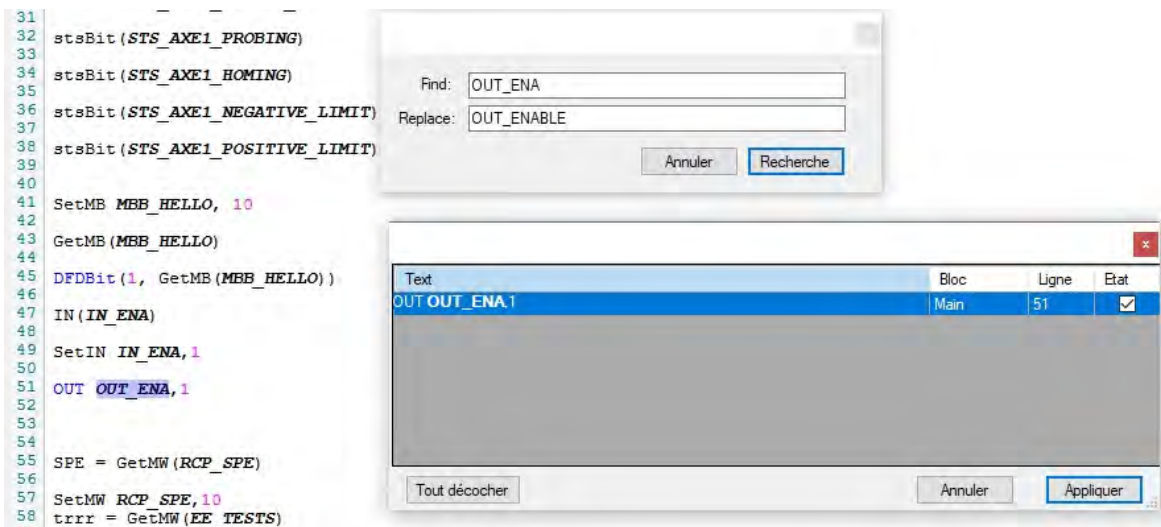
La fonction **Rechercher et Remplacer** (Find and Replace) d' ICNCStudio est très puissante, car elle permet de substituer automatiquement le nom d'une variable de votre programme PLC Basic, ou celui d'un Registre, ou encore celui d'un Bit, par un autre nom et ce pour toutes les occurrences et dans toutes les zones de votre projet PLC.

Ainsi ce renommage, demandé depuis l'éditeur de texte, s'effectuera automatiquement non seulement dans tous les blocs du programme (Init, Main, et autres créés par vous...), mais aussi dans tous les tableaux concernés (registres Utilisateurs, registres Sauvegardés, DIN, DOUT, COILS, Axes, Recettes, ...

La fenêtre pour **Rechercher et Remplacer** se lance soit avec un clic droit sur l'expression visée, puis choisir "**Remplacer**"



ou bien avec le raccourci clavier **Ctrl + H** :



**NB:** Toutes les occurrences rencontrées apparaissent alors dans une fenêtre telle que celle ci-dessus.

Il suffit ensuite de garder cochées les occurrences pour lesquelles vous souhaitez appliquer le renommage (et décocher les autres).

**NB:** La recherche s'applique à tous les cas de la casse (majuscules, et minuscules), en revanche la casse pour le remplacement de l'expression sera exactement identique à ce qui est saisi dans le champ "Replace".

## Main

d' Outils :

Aperçu de la barre de Menu et la barre



### **La barre de Menu:**

- **Fichier (File)** : Ouvrir Projet, Ouvrir Projet récent, Sauver Projet, Sauver Sous, Options Projet, Nouveau Projet, Sauvegarde Configuration InterpCNC, Restaurer Configuration InterpCNC, Export programme PLC au format HTML, Export **Declaration Kinco Address Tag**, Exit.
- **Edit** : Chercher, Remplacer, Aller à la ligne..., Marques page, Auto-indentation, Commenter/Décommenter, Évaluer l'expression, Développer ou réduire, Naviguer vers l'avant ou vers l'arrière  
Évaluer l'expression permet d'afficher dans la console le résultat recalculé à la demande, de l'expression sélectionné (qui peut être une variable, une fonction, ...)
- **Affichage (View)** : choix des fenêtres ou tableaux à afficher, affichage des barres.  
Ce menu permet l'accès au même fenêtres que les icônes de la barre d'outils.

La restauration du Layout fenêtres permet de recharger la disposition par défaut, tandis que la sauvegarde Layout fenêtres permet de sauvegarder votre disposition personnalisée.

L'option « **Toutes variables PLC** » est une autre version du tableau des paramètres système de la carte. Elle permet d'accéder à ces paramètres non plus depuis leur numéro de paramètre, mais plutôt avec leur adresse Modbus.

L'intérêt est également de bien identifier leur format (32 bit, 16 bit, signé ou non, bit, etc.), et d'avoir des possibilité de tri pour les rechercher.

#### **Pour la carte V3 uniquement:**

L'option « Graphique personnalisé » permet de tracer sur des axes Temps (X) et Valeur (Y), un graphique rafraîchi à la fréquence souhaité (200ms par défaut), en sélectionnant parmi:

Position des axes, vitesses des axes, position cibles des axes, état des entrées, état des sorties, ou "Custom".

"Custom" permet de tracer un graphique représentant l'évolution de la valeur contenu dans un registre donné, en précisant son adresse, son type, et son format.

- **PLC** (Editeur programme PLC, Constantes programme, Moniteur PLC, Liste des définitions visible, Carte Programme visible, Export programme analysé).  
Sert à réouvrir toute fenêtre décrite par la ligne, qui aurait été fermée.  
La carte programme visible : cette fonctionnalité donne un aperçu très condensé de la structure générale de votre programme. Elle fait appel à la mémoire visuelle.  
Ainsi vous pouvez vous rendre directement à un endroit dont vous reconnaissez visuellement la structure.  
Exemple : déclaration de constantes, déclarations de variables, Select... case, boucles, etc...

Le parcours sur le programme se fait avec la molette de la souris, ou clic gauche + montée ou descente sur la carte programme visible.

- **Outils (Tools)** : Mise à jour Firmware, Thème, Style de Document, Affichage inversé, Exportation de fichier de Paramètres.  
La fenêtre de Mise à jour Firmware permet également d'afficher des informations spécifiques à votre carte V3 : n° de série (CPUID), Adresse MAC, Version du Firmware, etc...
- **Fenêtre (Window)** : sélection d'une fenêtre parmi les onglets déjà existants dans la fenêtre centrale.
- **Aide / Help (A propos)**

### **La barre d'Outils:**

- **Icône de connexion** :  
Elle permet d'appeler la fenêtre permettant de sélectionner et/ou configurer le type de connexion à établir avec la carte (Serie (USB ou RS485) ou Ethernet (TCP ou UDP, +port)).
- **Icônes des fonctions habituelles** :  
Nouveau Projet, Ouvrir un projet, Enregistrer le projet en cours, Enregistrer Sous...
- **Icônes d'affichage des différentes fenêtres** (si masquées, sinon rappel) :  
Affichage du Moniteur, affichage des tests de commande d'axes, affichage des Entrées, affichage des Sorties, affichage des COILS, affichage des Registres Utilisateurs, affichage des Registres Sauvegardés.

Dans la fenêtre centrale : affichage du Programme PLC, affichage du tableau de constantes, affichage du Tableau de paramètres (réglages) de la carte, ou affichage de l'éditeur de Recettes.

### **Les Barres d'Etat :**

- **La barre supérieure** indique:  
→ le type de connexion établie avec la carte (USB, TCP IP, ...), ainsi que l'emplacement du fichier programme en cours.
- **La barre inférieure** indique:  
→ le taux de d'occupation du processeur en tant réel (% CPU disponible), ainsi que le taux de disponibilité de la carte.  
→ si un programme est en cours d'exécution (PLC RUNNING/PLC STOPPED), avec commande Marche/Arrêt  
→ Si l'entrée ENABLE de la carte est active (ENABLE/ARRET)  
→ Si le projet est cripté ou non cripté  
→ Les échanges de données avec la carte (paquets envoyés (S:), paquets reçus (R:), Erreurs de transmission (Err:), et l'état de connexion (CONN : True ou False).



## Tableau\_des\_parametres

Il existe 2 façons d'accéder aux paramètres de la carte.

Soit depuis l'icone paramètres:



Soit depuis le menu:

**View -> Paramètres InterpCNC**

## Pour la carte V3

### Description des paramètres les plus utiles

Les paramètres portent un numéro situé en 20 et 1515.

Parmi les paramètres de réglages les plus utiles, il faut distinguer :

- **La fréquence initiale des mouvements des Axes** → paramètres 20 à 25  
Appelé aussi « Frequence Start » ce paramètre permet de réduire le temps nécessaire à la phase d'accélération (sans en augmenter la vitesse) en ne démarrant pas à la fréquence 0.  
Attention toutefois certaines applications peuvent ne pas supporter certains réglages (bouteilles sur convoyeur, etc...)
- **L'inversion du sens de rotation** → paramètre 30  
0 : sens par défaut. 1 : sens inversé. 1 bit par axe.
- **La Polarité des Entrées** → paramètres 200, 201 et 202  
Il s'agit de 3 registres 32 bit, mappant l'état de fonctionnement de chaque entrée (1 bit par entrée).  
Valeur : 0 ou 1.  
Ce réglage peut par exemple vous permettre d'utiliser en entrée un niveau de signal opposé à celui prévu dans le programme, sans avoir à modifier le programme existant.  
Sur 0 (mode normal) → un niveau bas donne une lecture à 0, un niveau haut donne une lecture à 1  
Sur 1 → un niveau haut donne une lecture à 0, un niveau bas donne une lecture à 1
- **La configuration des Entrées 0 à 15** → registres 210 à 213 :  
Voir au chapitre des paramètres DIN de la carte.
- **La configuration des Entrées 16 à 22 (les Entrées Rapides)** → registres 220 à 226  
Ces 7 entrées sont configurables comme suit :  
0 : rafraîchissement à chaque ligne du programme.  
1 : rafraîchissement à chaque interruption accédant à cette entrée.  
2 : mode compteur

- 3: mode encodeur 2X
- 4: mode encodeur 4X

- **La configuration des entrées Analogiques**

**Le gain** : Réglage à 1, donc directement la résolution du convertisseur.  
La valeur par défaut 20.07979 est un ajustement tenant compte des résistances d'entrée.

**L'offset** : permet de régler la valeur 0 (car on peut mesurer de -10V à +10V)

**L'échelle de mesure** :

- à 5 (valeur par défaut), la plage de mesure s'étend de -10V à +10V
- à 4, la plage de mesure s'étend de -5V à +5V ce qui permet aussi d'améliorer la précision.

- **La configuration de Communication (COM1 et COM2)**

- Les paramètres 400 et 420 définissent le mode de fonctionnement des ports COM1 et 2 :

0=désactivé, 1=Esclave, 2=Maître, 3=Mode DMX (voir manuel de l'interpréteur Basic)

- Les paramètres 401 à 404 et 421 à 424 sont de grands classiques de réglage des ports de communication série (Vitesse en bauds, nombre de bits de données, parité, bits de stop)

- Les paramètres 405 et 425 définissent l'ID de la carte lorsqu'elle est configurée en tant qu'esclave (voir paramètres 400 et 420)

- Les paramètres 408 et 426 définissent le délai minimum en millisecondes entre l'envoi de 2 trames Modbus (certains variateurs par exemple nécessitent au moins 5 ms entre 2 trames reçues)

- Les paramètres 409 et 427 définissent le nombre de tentatives infructueuses avant de retourner une erreur de transmission (visible dans « Toutes les Variables PLC », registres Modbus RTU master error (ou success)).

- **L'État des sorties 0 à 31 au boot** → Registre 270.

Il s'agit d'un registre 32 bits. Ce paramètre permet de définir (forcer) l'état des sorties numériques à la mise sous tension de la carte (rappel : les sorties 0 à 15 sont des sorties physiques, les sorties 16 à 31 sont des sorties virtuelles (par exemple pour un module externe)).

- **L'État des sorties analogiques (AOUT0 et AOUT1) sur perte de l'ENable**  
→ Registres 330 et 331.

Ces sorties seront forcées aux tensions correspondant à ces valeurs, lorsque la liaison de l' ENA est interrompue (cas d'arrêt d'urgence).

- **Le paramètre « PLC Basic AutoRUN »** → Paramètre 510

Valeur : 0 ou 1.

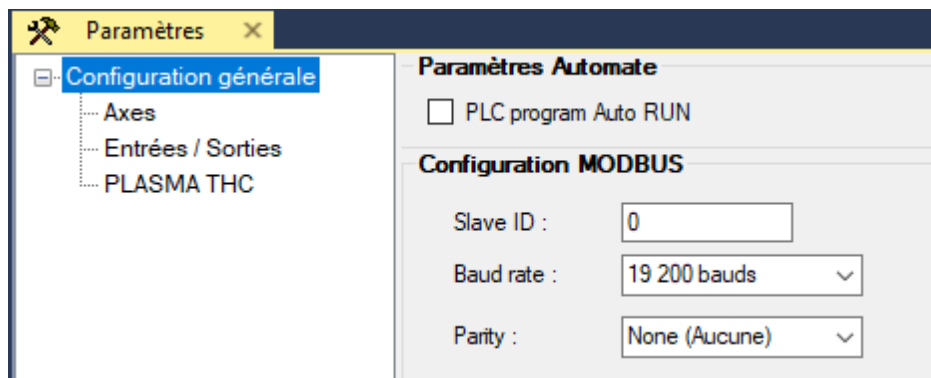
Sur 1, le programme PLCBasic présent dans l'EEPROM de la carte s'exécutera automatiquement à l'allumage.

- **Les paramètres de Configuration Réseau** → Paramètres 520 à 544  
Ici vous pourrez donner un nom Netbios à la carte sur votre réseau, choisir le mode DHCP (à 1), ou bien définir l'adresse IP de votre carte, masque de sous réseau, passerelle, port, etc...

## Pour le cartes V2

Un onglet Paramètres apparait dans la fenêtre centrale, faisant apparaître 4 pages de réglages:

- Configuration Générale
  - Axes
  - Entrées / Sorties
  - PLASMA THC



## Recettes

### *Principe des recettes*

La création de recettes vous permet de prévoir pour votre programme, différents « jeux » de paramètres utilisables selon le type de production à réaliser (variantes).

Exemple : des paramètres pour étiquetage d'objets de taille différente, paramètres pour pose d'étiquettes de taille différentes, etc...

L'**Éditeur de Recettes** vous permet d'affecter pour chaque registre :  
→ un nom, un type, une valeur, un commentaire.

Vous disposez de 4 pages (0 à 3) permettant de travailler simultanément sur 4 recettes.

Pour chaque page, vous disposez d'un **index** permettant de sélectionner la recette active.

#	Value	Name	Type	Comment
10000	28514	RCP_RECETTE_NOM	U16	Nom de la recette (10 caractères maxi)
10001	29813		U16	
10002	26981		U16	
10003	27756		U16	
10004	8293		U16	
10005	0	RCP_ONOFF_ORIENT_ENTREE	U16	0/1 , avec/sans detection spot 0=sans
10006	0	RCP_ONOFF_COIFFE	U16	
10007	0	RCP_ONOFF_SPOT_ETIQUETAGE	U16	
10008	0	RCP_ONOFF_ETIQUETTE	U16	
10009	0	RCP_ONOFF_CONTR_ETIQUETTE	U16	
10010	0	RCP_ONOFF_MEDAILLON	U16	
10011	0	RCP_ONOFF_COLLERETTE	U16	
10012	0	RCP_ONOFF_TABLE_ENTREE	U16	
10013	0	RCP_ONOFF_ETOILE	U16	
10014	0	RCP_ONOFF_DISTRIBUTEUR	U16	
10015	0	RCP_ONOFF_MAGASIN	U16	
10016	0	RCP_ONOFF_MARQUEUR	U16	
10017	65535		U16	
10018	2652	RCP_R_ORIENT_ETIQ	FLOAT	pulses/tr 0 decimale , resolution orienteur poste...
10019	---		U16	
10020	16,2	RCP_R_PUL_MM_COLLERETTE	FLOAT	pulses/mm 1 decimale , resolution en pulses/mm...
10021	---		U16	
10022	9,5	RCP_R_PUL_MM_ETIQUETTE	FLOAT	pulses/mm 1 decimale , resolution en pulses/mm...
10023	---		U16	
10024	1	RCP_R_PUL_MM_MEDAILLON	FLOAT	resolution en pulses/mm au niveau du medaillon
10025	---		U16	
10026	138	RCP_CIRCONF_COLLERETTE	U16	mm 0 decimale , circonference bouteille au ni...
10027	276	RCP_CIRCONF_ETIQUETTE	U16	mm 0 decimale , circonference au niveau etiq...
10028	65535		U16	
10029	5	RCP_AV_COLLERETTE	U16	mm 0 decimale , avance collerette

## UTILISATION

### Pour la carte V3:

Il convient tout d'abord de définir le nombre de paramètres dont vous aurez besoin pour vos recettes. NB : chaque paramètre occupe par défaut 1 registre 16 bits (U16).

Pour chaque registre, vous devrez en préciser le type (U16, U32, I16, I32, FLOAT). Dans le cas d'un registre sur 32 bits (DWORD ou bien FLOAT), le registre de 16 bits suivant sera automatiquement réservé, et son champ valeur deviendra alors non éditable puisque la valeur 32 bits figurera sur la ligne précédente.

La zone mémoire dédiée au stockage des recettes (EEprom) est un espace unique de 16 Ko (soit 8192 registres 16 bits).

La taille d'une recette est limitée à 1000 registres.

Vous pouvez donc par exemple gérer plus de 40 recettes de 200 registres ( $8192 / 200 = 20+$ ) ou bien encore plus de 100 recettes de 80 registres ( $8192 / 80 = 100+$ ).

**Pour les cartes V2:**

La taille des recettes est limitée à 250 registres et le stockage est limité à 5000 registres.

Vous pouvez donc par exemple gérer 20 recettes de 250 registres ( $5000 / 250 = 20$ ) ou bien encore, 50 recettes de 100 registres ( $5000 / 100 = 50$ ).

L'adresse 5632 est le premier registre utilisé par la zone des recettes.

5632 : Paramètre de taille des recettes

La taille des recettes s'exprime en nombre de registre 16 bits par recette (de 1 à 250).

5633 à 5636 : Index des pages 0 à 3 pour les recettes

La première recette se trouve avec un index à 0.

5637 -> 5886 : Page 0

5887 -> 6136 : Page 1

6137 -> 6386 : Page 2

6387 -> 6636 : Page 3

La taille des recettes est limitée à 250 registres et le stockage est limité à 5000 registres.

Vous pouvez donc par exemple gérer 20 recettes de 250 registres ( $5000 / 250 = 20$ ) ou bien encore, 50 recettes de 100 registres ( $5000 / 100 = 50$ ).

La première recette se trouve avec un index à 0.

**Déplacement des registres par glisser-dépose :**

Le glisser-déposer permet de déplacer la **déclaration** d'un registre (colonnes *Nom/Type/Commentaire*) sans toucher à sa **Valeur** afin de réassigner proprement une adresse. On peut **prendre** une ligne uniquement si son **Nom n'est pas vide** ; si le **Type** est un **type "2 lignes"** (*U32, I32, FLOAT*), la prise englobe la ligne et **celle du dessous**. On peut **déposer** seulement sur une ligne **au Nom vide** (et pour un type "2 lignes", la ligne suivante doit aussi être vide). Pendant le survol, la colonne "#" affiche  si le dépôt est possible ou  sinon. Au dépôt, le système **coupe** *Nom/Type/Commentaire* de la source et les **colle** sur la cible, en gardant les valeurs et index internes cohérents.

La sélection multiple est supportée : plusieurs lignes peuvent être déplacées ensemble.

**Fonctionnement****En lecture :**

Selon le numéro de recette renseigné à l' Index d'une page, le système fera apparaître de façon transparente, le bloc des données correspondant à la recette aux adresses de début de la page :

Pour la carte **V3** : 10000, 11000, 12000 ou 13000

Pour les cartes **V2** : 5637, 5887, 6137 ou 6387

Ainsi, vos registres (nommés par convention RCP\_...) prendront tour à tour des valeurs différentes, juste en changeant la valeur de l' Index.

**En écriture :**

Les valeurs écrites à ces premières adresses, seront (toujours de façon transparente) effectivement stockées dans la zone de mémoire pointées par l' Index (=numéro de recette).

**Avantage avec un IHM** : depuis un écran dédié à la saisie de paramètres de recettes, vos champs saisissables sont lus et écrits toujours aux mêmes adresses Modbus, et en changeant seulement la valeur de l'Index ils seront effectivement stockés dans la zone de mémoire attribuée au numéro de recette.

NB: Si vous renommez un registre depuis son tableau (Registre Utilisateur, Registre Sauvegardé, ou **Recette**), une fenêtre apparaîtra vous demandant de cocher (sélectionner) pour lesquelles des occurrences du programme Basic, le renommage devra être appliqué.

Voir également la fonction [Chercher et Remplacer](#), qui permet de renommer des occurrences sélectionnées dans le programme Basic, mais sans renommer le Registre dans son tableau.

## Variables\_PLC

A l'onglet **View -> Toutes variables PLC**

Cette page permet d'accéder et visualiser en temps réel tous les registres systèmes, sous la forme d'un tableau et avec des possibilités de recherche et de tri.

Il est possible d'afficher ou non ces registres, selon leur type et leur méthode d'accès (cocher) :

**Input Registers** (en lecture seule), **Input Bits** (bits en lecture seule)

**Holding Registers** (en lecture/écriture), et **Coils** (bits en lecture/écriture).

Il peut être également pratique d'ouvrir plusieurs fois « Toutes les variables PLC » (= plusieurs instances), afin d'accéder séparément à des registres selon leur type.

Pour une recherche, vous pouvez saisir un nom, un mot-clé, une adresse, une valeur, un format, etc... On peut aussi affiner en limitant la recherche sur un type de registre et/ou une colonne.

Vous pouvez bien entendu entrer des valeurs directement dans les Holding Registers et Coils, depuis le tableau (colonne Value).

Section	Address	Name	Value	Comment	Format
Buffer commandes Modbus	2141	CMD_Buffer 2141	0	CMD[141]	U16
Buffer commandes Modbus	2142	CMD_Buffer 2142	0	CMD[142]	U16
Buffer commandes Modbus	2143	CMD_Buffer 2143	0	CMD[143]	U16
Buffer commandes Modbus	2144	CMD_Buffer 2144	0	CMD[144]	U16
Buffer commandes Modbus	2145	CMD_Buffer 2145	0	CMD[145]	U16
Buffer commandes Modbus	2146	CMD_Buffer 2146	0	CMD[146]	U16
Buffer commandes Modbus	2147	CMD_Buffer 2147	0	CMD[147]	U16
Buffer commandes Modbus	2148	CMD_Buffer 2148	0	CMD[148]	U16
Buffer commandes Modbus	2149	CMD_Buffer 2149	0	CMD[149]	U16
Analog outputs	2150	Analog output 0	0	Analog OUT[0]	I16
Analog outputs	2151	Analog output 1	0	Analog OUT[1]	I16
Analog outputs	2152	Analog output 2	0	Analog OUT[2]	I16
Analog outputs	2153	Analog output 3	0	Analog OUT[3]	I16
Analog outputs	2154	Analog output 4	0	Analog OUT[4]	I16
Analog outputs	2155	Analog output 5	0	Analog OUT[5]	I16
Analog outputs	2156	Analog output 6	0	Analog OUT[6]	I16
Analog outputs	2157	Analog output 7	0	Analog OUT[7]	I16
Digital outputs words	2160	Digital output word 0	0	DOUT as word	U16
Digital outputs words	2161	Digital output word 1	0	DOUT as word	U16
Digital outputs words	2162	Digital output word 2	0	DOUT as word	U16
Digital outputs words	2163	Digital output word 3	0	DOUT as word	U16
Digital outputs words	2164	Digital output word 4	0	DOUT as word	U16
Digital outputs words	2165	Digital output word 5	0	DOUT as word	U16

L'intérêt principal est donc de pouvoir accéder le facilement possible au contenu des bits et registres systèmes, afin par exemple de tester le bon fonctionnement de votre programme Basic, et sa mise au point.

(Sans cela, il faudrait arrêter le programme en cours d'exécution et lancer un Print depuis la ligne de commande, en ayant recherché au préalable l'adresse du registre en question...)

## Donnees\_personnalisees

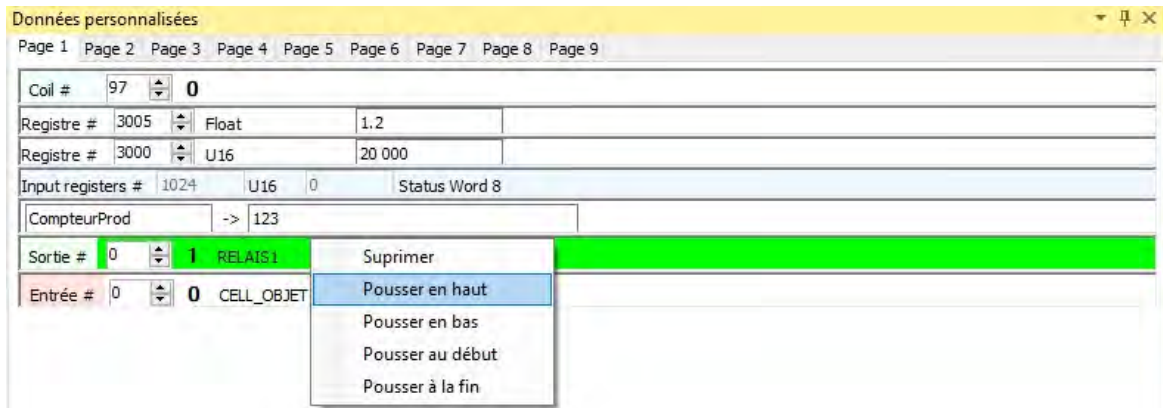
Pour déboguer un programme automate en cours de développement, ICNCStudio offre un outil permettant de faire une sélection "à la carte" des registres, bits (Coils, Entrées, ou Sorties), variables PLC et variables du programme dont vous souhaitez suivre l'évolution en cours d'exécution.

Il s'agit de la **liste des Données Personnalisées**.

Celle-ci est disponible depuis l'onglet **View** -> **Données personnalisées**, ou

depuis l'icone 

La fenêtre des Données Personnalisées apparaîtra par défaut à côté de la fenêtre des Entrées DIN, mais peut être glissée et déplacée à loisir dans tout autre zone de l'écran.



La fenêtre des Données Personnalisées compte 9 pages distinctes (1 à 9), qui peuvent être constituées des registres, bits et variables insérés à votre guise. Cela est possible depuis les fenêtres **Registres Utilisateurs**, **Registres Sauvegardés**, **DIN**, **DOUT**, **COIL**, **Editeur de Recettes**, et **Toutes Variables PLC**.

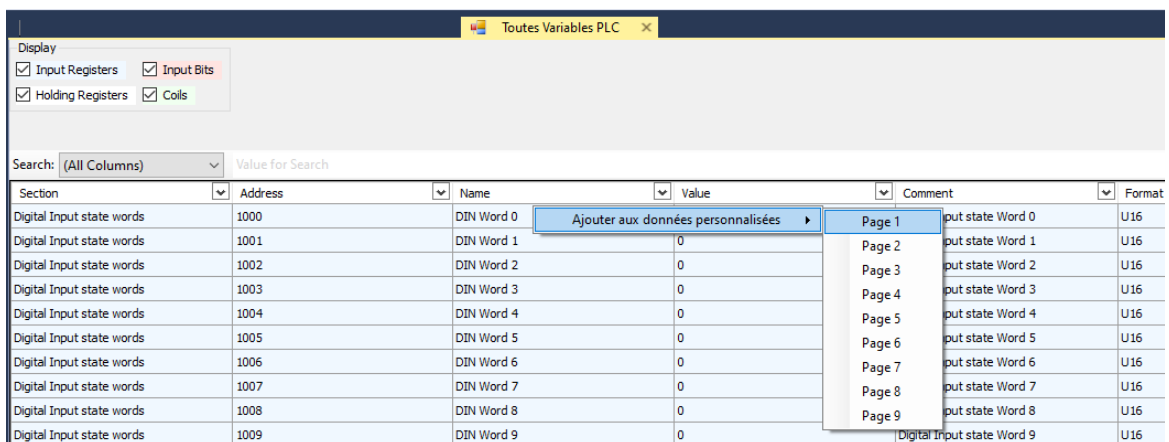
L'intérêt de disposer de différentes pages peut être par exemple d'avoir des listes distinctes de registres, bits, et variables, selon le cycle automate en cours, selon le mode de fonctionnement en cours, selon la fonction (Ethernet, Modbus, etc...)

Chacune de ces 9 pages peut être renommée (**clic droit** sur l'onglet -> **Renommer l'Onglet**).

La liste de ces données se compose dans l'ordre des ajouts. Toutefois il est possible de changer cet ordre: **clic droit**, puis

- > Supprimer
- > Pousser en haut
- > Pousser en bas
- > Pousser au début
- > Pousser à la fin

L'ajout d'un registre, bit, ou variable sur l'une des 9 pages se fait par un: **clic droit** -> **Ajouter aux données personnalisées** -> **Page x** depuis la ligne du registre, bit ou variable PLC concernée.



Les valeurs de ces registres, bits et variables peuvent être changées en temps réel directement depuis la fenêtre des Données Personnalisées.

Pour les Coils, et les sorties DOUT, la valeur du booléen peut être changé

directement par un double-clic sur la ligne concernée, dans la page de la fenêtre des Données Personnalisées.

## Graphique\_V3

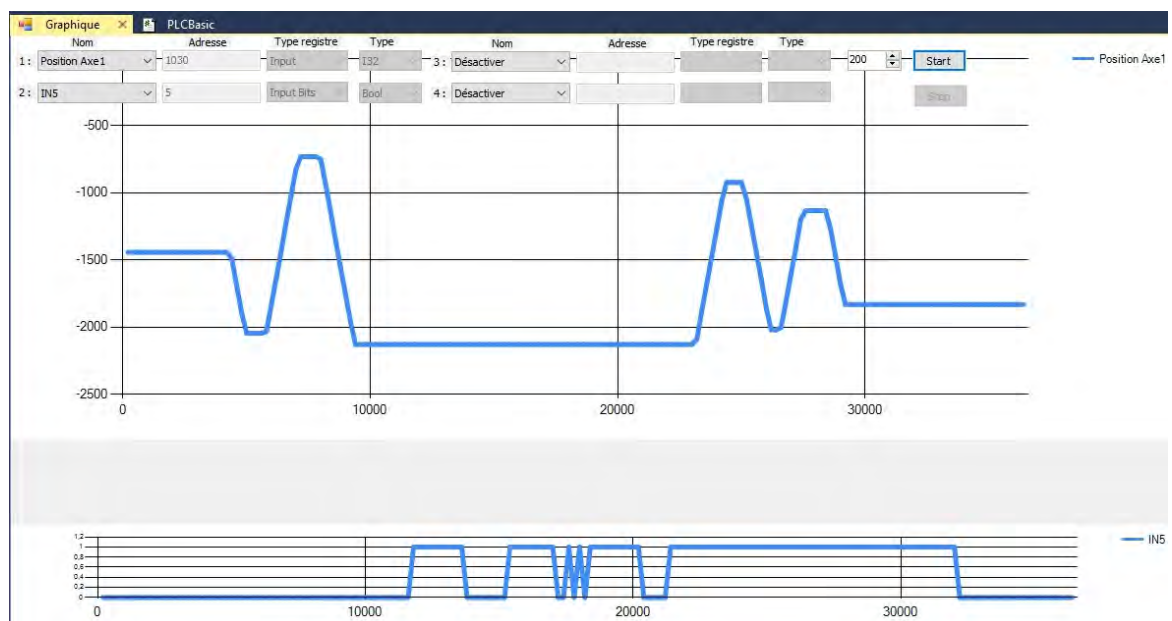
Cette fenêtre permet de monitorer sur une échelle de temps, l'évolution de jusqu'à 4 valeurs, qui peuvent être:

- Position courante d'un ou plusieurs Axes, ou vitesse, ou position cible
- Etat d'entrées ou de sorties digitales (Booléen)

Le choix peut se faire à l'aide des préselections des menus déroulants, ou bien par la sélection de "**Custom**" qui permet de monitorer le contenu d'un registre, en précisant son adresse, son Type, et son Format

Exemple: Adresse 3010 (Ram), Holding Register, U16

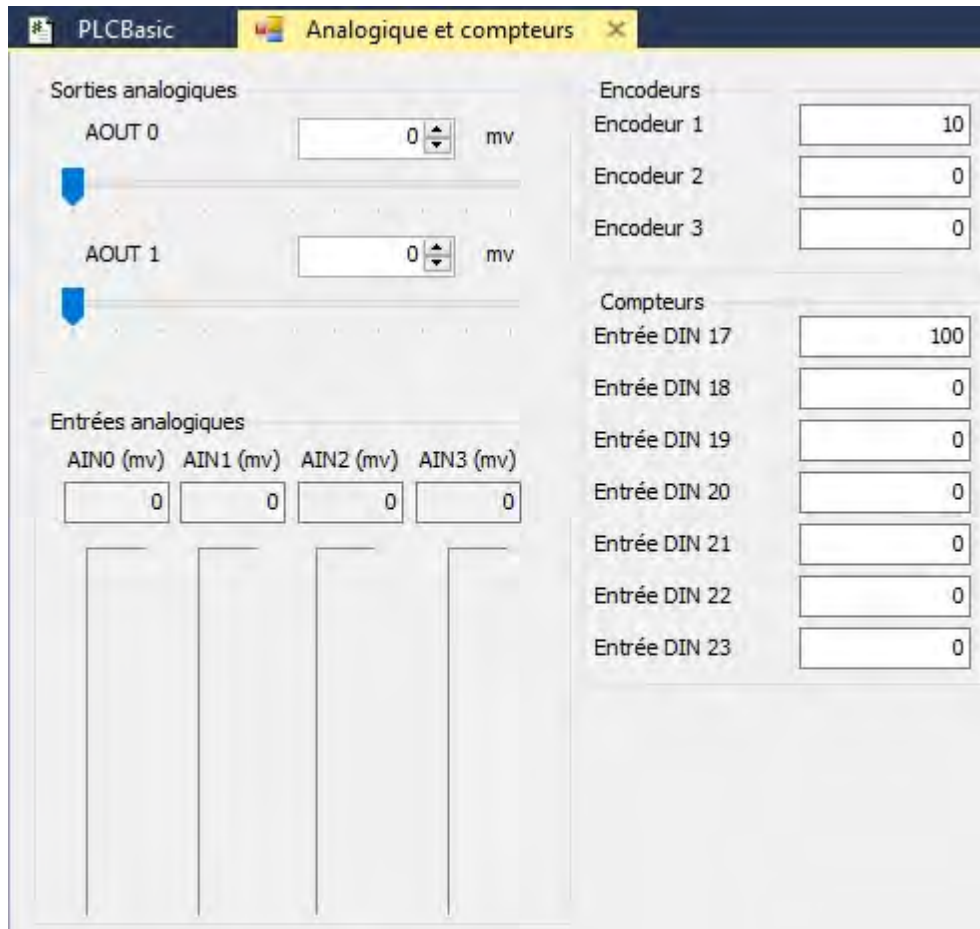
Le délai de rafraichissement est réglable en millisecondes. Réglé à 200ms ici:



Les axes apparaîtrons toujours sur le même graphique. L'échelle est auto-ajustable au fil du temps.

Les booléens apparaitrons toujours sur un graphique différent de celui des axes, et ce pour une question d'échelle.

## Analogique\_compteurs

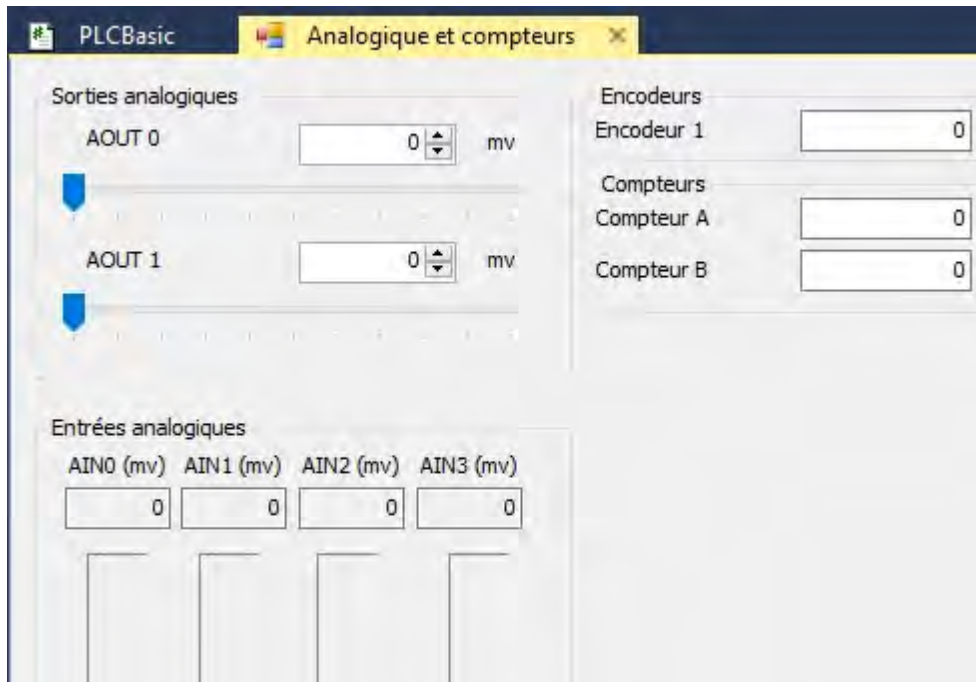
**Pour l'InterpCNC V3**

Cette fenêtre permet la lecture instantanée des 4 entrées analogiques, ainsi que le réglage manuel de la tension des 2 sorties analogiques (via curseur, ou bien en saisissant la valeur souhaitée).

Elle permet également la lecture des entrées rapides, pour:

- les positions des 3 codeurs
- les valeurs des compteurs

**Pour l'InterpCNC V2**



Cette fenêtre permet la lecture instantanée des 4 entrées analogiques, ainsi que le réglage manuel de la tension des 2 sorties analogiques (via curseur, ou bien en saisissant la valeur souhaitée).

Elle permet également la lecture des entrées rapides, pour:

- les positions du codeur
- les valeurs des compteurs A et B

## Firmware\_update

### *Mise à jour du firmware*

#### *Définition*

Le firmware est le logiciel système embarqué de la carte. Il permet le bon fonctionnement de tous ses composants, entre eux et dans leur mise en œuvre optimale. Il est aussi responsable de toutes les fonctionnalités offertes par la carte.

Il est conseillé de mettre à jour le firmware chaque fois qu'une nouvelle version est disponible.

Cela permet de bénéficier dans vos applications des derniers correctifs ou améliorations des performances, ainsi que des éventuelles nouvelles fonctionnalités développées.

**NB** : si vous avez utilisé la carte InterpCNC pour un projet qui n'est pas susceptible d'évoluer et qui fonctionne déjà parfaitement en l'état, il n'est alors pas recommandé de faire une mise à jour.

**Avertissement** : Ne jamais interrompre l'alimentation +24V de la carte, ni débrancher sa connexion réseau ou USB pendant une mise à jour

(Malgré le dispositif mis en place il pourrait subsister un risque de programmation incomplète, ce qui aurait pour effet de rendre la carte non fonctionnelle, et nécessiterait alors son retour pour dépannage en atelier).

### **Procédure de mise à jour de la carte InterpCNC V3**

La mise à jour du firmware se fait à la demande :

→ Onglet Tools / Mise à jour Firmware

Remarque : Avant de pouvoir détecter si une nouvelle version du firmware est disponible, ICNCStudio doit être à jour.

ICNCStudio doit bien évidemment être au préalable connecté à la carte.  
La fenêtre de mise à jour Firmware vous informe de la version actuelle.  
La sélection de la version se fait juste au dessous (bouton [...] pour parcourir).  
Le bouton **[Release Notes]** vous permet de consulter les notes de version de la version sélectionnée et précédentes.

Une fois la version sélectionnée, cliquer sur [Envoyer] :

→ la procédure démarre (barres de progression) et le nouveau firmware est transféré à la carte.

Cliquer ensuite sur **[Reboot and Update]**

→ le firmware s'installe (voir progression sur l'afficheur de la carte).

A 100 %, la mise à jour est terminée et la carte est opérationnelle.

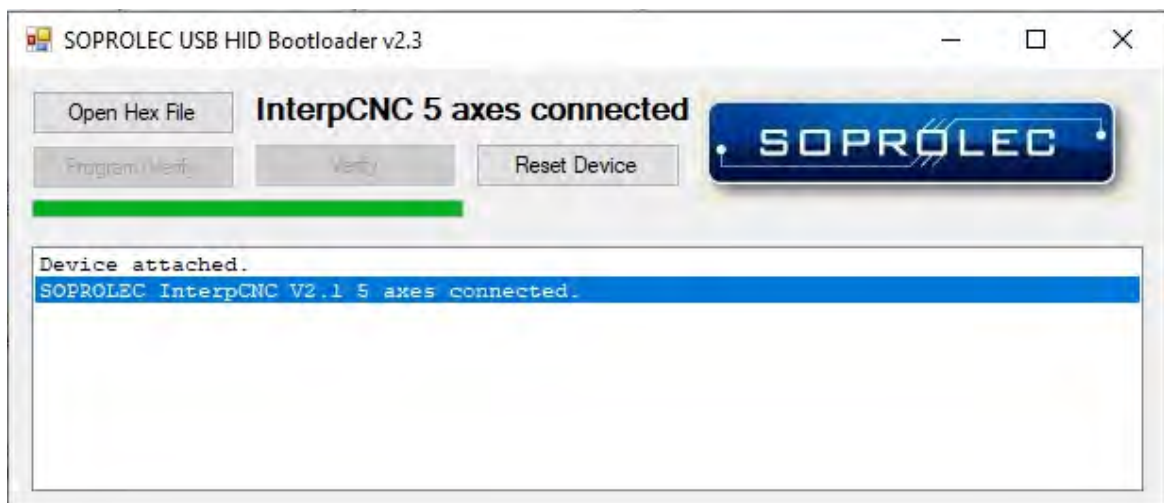
### **Mise à jour du firmware de la carte InterpCNC V2**

La mise à jour du firmware se fait à la demande :

→ Onglet Tools / Mise à jour Firmware

Un petite fenêtre vous informe alors que vous devez être connecté à la carte (USB) pour poursuivre -> cliquer sur OK.

L'utilitaire HID Bootloader s'ouvre alors:



Cliquer sur **Open Hex File** pour parcourir jusqu'au fichier .hex du firmware que vous souhaitez mettre en place dans la carte.

Puis cliquer sur **Program/Verify** et la mise en jour s'effectue.  
A la fin de la barre de progression, cliquer sur **Reset Device**. La carte quitte alors le mode Bootloader, et redémarre en mode normal avec le nouveau firmware.

## Mise à jour d' ICNCStudio

ICNCStudio est doté d'un système de mise à jour automatique en ligne. Périodiquement, au démarrage de l'application, la version d'ICNCStudio installée compare son numéro de version avec la dernière disponible, et vous informera si une mise à jour est disponible.

Vous pouvez faire le choix ou non de l'installer. Dans l'affirmative, la mise à jour sera alors téléchargée et s'installera automatiquement (suivre les indications à l'écran).

Vous pouvez également lancer à votre initiative une recherche de mise à jour :  
→ Onglet Tools : Mise à jour ICNC Studio

La version en cours d'utilisation est consultable à l'onglet Help/About ICNCStudio.

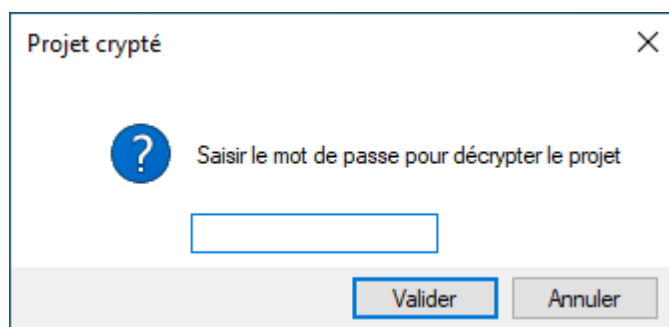
## Cryptage

Au delà de la protection par mot de passe, et toujours afin de protéger votre programme PLC Basic contre la contrefaçon, ICNCStudio intègre aussi un système de cryptage.

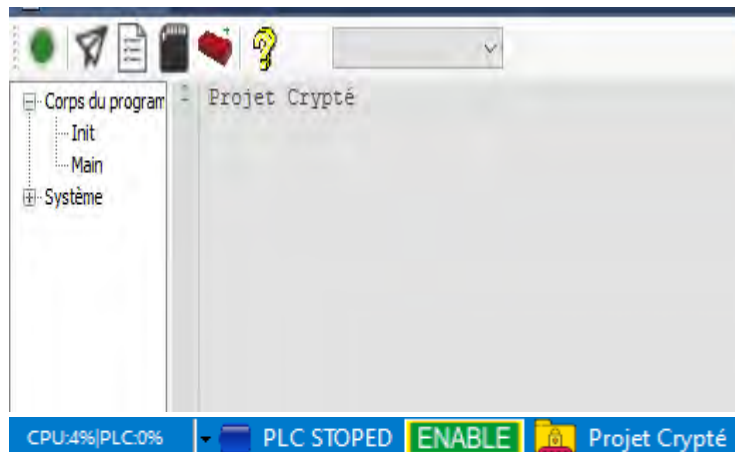
Ainsi un utilisateur avancé qui aurait pu se procurer le fichier source de votre programme automate, ne pourra si vous l'avez crypté, ni en voir le contenu dans ICNCStudio, ni même reconnaître des portions de code avec un éditeur de texte.

Pour accéder aux options de Cryptage/Décryptage du projet, cliquer sur:  
**File -> Options Projet**  
ou bien sur la barre d'état en bas: sur **Projet Crypté** ou **Projet Non Crypté**

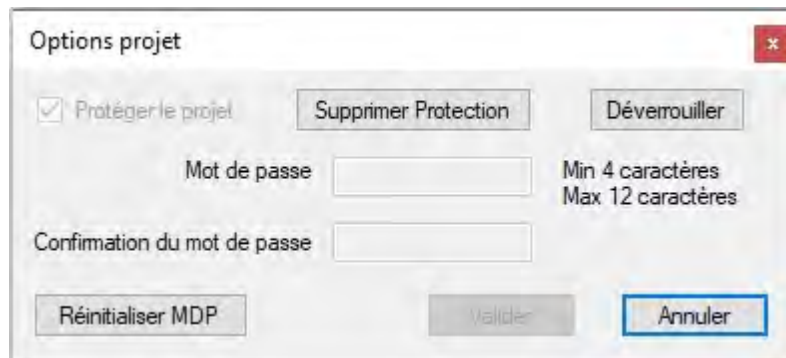
Si vous ouvrez un programme crypté, cette fenêtre de saisie apparaît :



Si vous annulez la saisie du mot de passe, le programme est chargé, mais l'éditeur de texte apparaît comme suit:



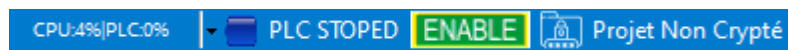
Cliquer sur le logo **jaune/rouge** (ou bien aller dans **File -> Options Projet**) fera apparaître la fenêtre de saisie du mot de passe de décryptage :



Cliquer sur "**Déverrouiller**" et saisir le bon mot de passe permettra de voir et éditer le contenu du projet, en revanche il sera toujours crypté lorsque vous enregistrerez les modifications.

Cliquer sur "**Supprimer Protection**" fera revenir la fenêtre initiale de saisie de mot de passe.

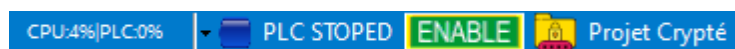
Le contenu du programme devient ensuite totalement accessible:



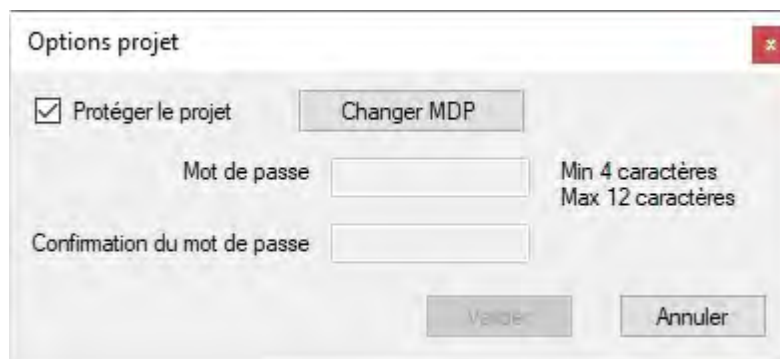
Cliquer sur le logo devenu gris (ou bien aller dans **File -> Options Projet**), permet de crypter le projet non crypté :



-> Cocher "**Protéger le projet**", choisir et confirmer le mot de passe, puis Valider. Vous pouvez alors toujours visualiser le programme que vous avez crypté, mais si vous le sauvegardez ce sera sous forme cryptée :



Si vous cliquez à nouveau sur le logo **jaune/rouge** (ou bien aller dans **File -> Options Projet**), le bouton "**Changer MDP**" vous permet de changer le mot de passe en cours :



## Paramètres

Ce chapitre décrit les différents écrans de configuration accessibles depuis l'icône des paramètres.



A la différence du tableau accessible depuis **View -> Paramètres InterpCNC**, seuls les principaux paramètres sont proposés, mais la lisibilité et la sélection des réglages est facilitée par des menus déroulants, champs de saisie et les "check box".

## Configuration\_generale

### Pour l'InterpCNC V3

### La fonction verrouillage

Protège votre programme PLC Basic embarqué, contre la lecture et/ou l'écrasement, grâce à la définition d'un mot de passe.

Ainsi, il est impossible de cloner la machine dont vous avez conçu le programme automate, ou encore aux utilisateurs trop curieux de l'effacer par mégarde.

**Lecture protégée:** Le programme ne pourra être lu, ni depuis la Ram, ni depuis l'Eprom.

**Écriture protégée:** Un programme peut être envoyé en Ram et être exécuté (Run), mais ne pourra être écrit dans l'Eprom (l'écrasement du programme déjà en place n'est donc pas autorisé).

Pour plus de détail voir les explications au chapitre des Configurations.

### Le paramètre « PLC Basic AutoRUN »

Valeur : 0 ou 1.

Sur 1, le programme PLCBasic présent dans l'EEprom de la carte s'exécutera automatiquement à la mise sous tension.

### Horloge temps réel (= RTC)

La carte InterpCNC V3 ne comportant pas de batterie ni de pile interne, elle peut se mettre à l'heure automatiquement (à la mise sous tension, ou sur commande), en se connectant à un serveur SNTP.

Note: SNTP = "Simple Network Time Protocol"

Dans ce cas activer "Synchro sur serveur SNTP"). Votre fuseau horaire et l' "Heure été/hiver automatique" sont les paramètres permettant d'ajuster la mise à l'heure à votre position géographique.

### **Rotation 180°**

Permet une lecture plus facile de l'écran LCD selon le sens de montage de la carte sur un rail DIN.

### **Sortie PUL/DIR en TTL**

Permet à l'automate, lorsque l'on n'utilise pas les commandes d'axes, d'utiliser les sorties pulses et direction en sorties TOR 0-5V, lesquelles seront remappées en tant que sorties n° 16 à 27.

## ***Pour l'InterpCNC V2***

A l'onglet **Configuration générale**, le paramètre « **PLC Basic AutoRUN** » est identique à la carte V3.

Pour la communication Modbus, renseigner l'ID, la vitesse de communication, et la parité.

### **A l'onglet Entrées / Sorties**

Il est possible de réaffecter la fonction Arrêt d'urgence à une autre entrée que l'Enable, et d'en choisir la polarité.

Il est également possible de forcer l'état des sorties 1 à 8 (ou 1 à 16) au démarrage de la carte,

de même qu'il est possible de définir une valeur de tension pour les sorties analogiques au au démarrage de la carte.

Pour plus d'informations, voir également la rubrique "[DIN](#)".

## **Axes**

### ***Pour l'InterpCNC V3***

**Axe 1**

Inverser sens de rotation  Pulses actifs à 1

Fréquence initiale des mouvement (Hz)

Fin de course

Action sur fin de course  Décélération rapide (Hz)

Entrée sens négatif		Entrée sens positif	
Numéro entrée	Polarité	Numéro d'entrée	Polarité
<input type="text" value="Aucun"/>	<input type="text"/>	<input type="text" value="Aucun"/>	<input type="text"/>

### ***La fréquence initiale des mouvements des Axes***

Appelée aussi « Fréquence Start » ce paramètre permet de réduire le temps nécessaire à la phase d'accélération

(sans en augmenter la vitesse) en ne démarrant pas à la fréquence 0.

Attention toutefois certaines applications peuvent ne pas supporter certains réglages (bouteilles sur convoyeur, etc...)

### ***Fin de course***

Il s'agit de déclarer ici l'entrée utilisée pour chaque capteur de fin de course, selon le sens de l'axe, ainsi que sa polarité.

Le type d'action sur fin de course peut-être l'arrêt rapide de l'axe (prenant en compte la valeur de du champ de Décélération rapide) ou bien son arrêt immédiat. Cet arrêt peut aussi porter sur tous les axes.

## ***Pour l'InterpCNC V2***

Configuration générale

- Axes
- Entrées / Sorties
- PLASMA THC

### Commun

Generale

Fréquence initiale des mouvement (Hz)  Accel/Decel(KHz/s)

Duplication axes

Origine

Fréquence initiale des mouvement (Hz)  Vitesse Lente (Hz)

Reset position automatique

### Axe 1

Inverser sens de rotation

Fin de course

Action sur fin de course

Entrée sens négatif

Numéro entrée  Polarité

Entrée sens positif

Numéro d'entrée  Polarité

Origine

Numéro entrée  Type Capteur  Sens  Accel (KHz)  Vitesse Rapide (Hz)

### Axe 2

A l'onglet **Axes**, il est possible de spécifier la fréquence de démarrage commune à tous les axes, ainsi qu'une fréquence d'accélération qui sera la même pour la décélération. De même pour la prise d'origine. Une duplication d'axe est possible de X vers A, ou de X vers B (carte V2.2 seulement).

Pour chaque axe, il est possible d'inverser le sens de rotation par défaut, et de paramétrer la prise d'origine ou l'action sur fin de course.

## DIN

### Pour l'InterpCNC V3

#### Configuration des Entrées standards 0 à 15

Entrées standards DIN0 à DIN15

Entrée # 0

Polarité inversée (contact type NC)    Filtre anti-rebond     Filtre

#### – La Polarité des Entrées

Ce réglage peut par exemple vous permettre d'utiliser en entrée un niveau de signal opposé à celui prévu dans le programme, sans avoir à modifier le programme existant.

Sur 0 (mode normal) → un niveau bas donne une lecture à 0, un niveau haut donne une lecture à 1

Sur 1 → un niveau haut donne une lecture à 0, un niveau bas donne une lecture à 1

– **Réglage du filtre Anti-Rebond pour chaque entrée (temps en ms)**

1) Délai de filtrage →

4 valeurs de réglage sont possibles, soit 2 bits par entrée.

00 : Délai de 0 ms → Filtre désactivé

01 : Délai de 10ms

10 : Délai de 30 ms

11 : Délai de 100 ms

Ainsi toute variation de front pendant ce délai, il sera ignorée.

2) Mode de filtrage

Ces réglages sont basés sur le délai anti-rebond sélectionné ci-dessus.

3 valeurs de réglage sont possibles, soit 2 bits par entrée :

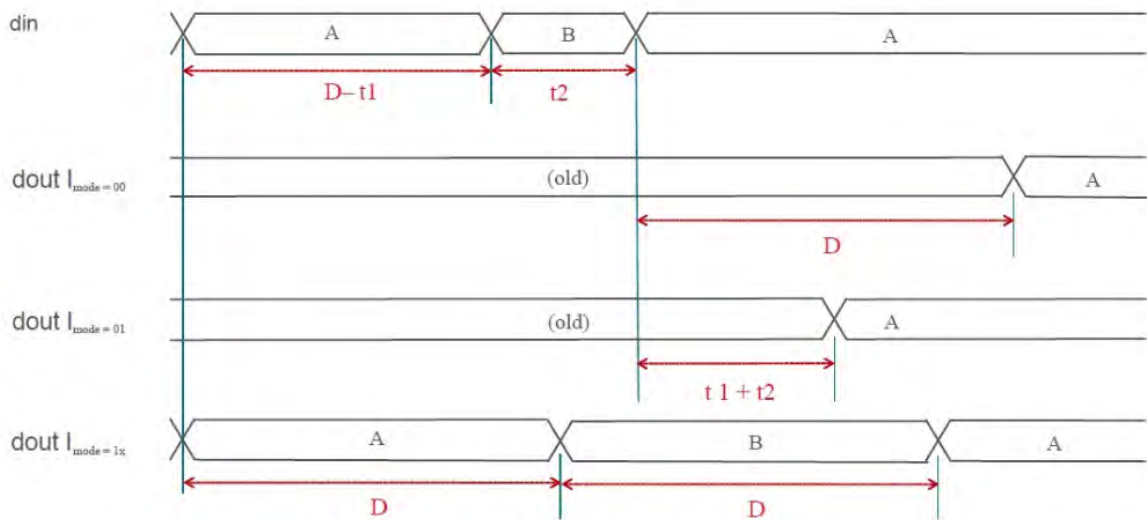


Figure 2.7. Debounce Filter Modes Timing Diagram

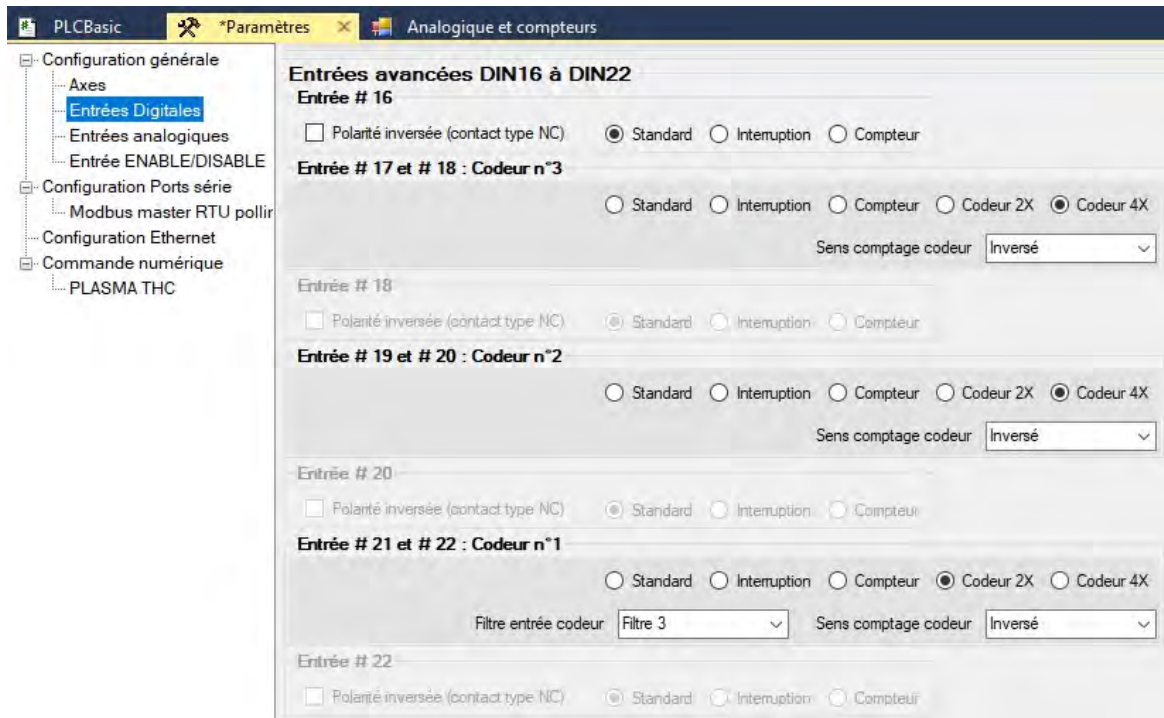
00 : Pas de filtre. Ce mode ne reporte que les changement d'état dont la durée est supérieure à X ms (tel que définis par l'anti-rebond).

01 : Filtre Passe-Bas. Ce mode ne reporte que les changement d'état dont la durée est supérieure à :

(temps manquant pour aller aux X ms + durée du rebond).

10 ou 11 : Ce mode réalise un échantillonnage tous les X ms (tel que définis par l'anti-rebond), et reporte l'état relevé à cet instant précis.

## Configuration des Entrées 16 à 22 (les Entrées Rapides)



Ces 7 entrées sont configurables comme suit :

- 0 : rafraîchissement à chaque ligne du programme (Standard).
- 1 : rafraîchissement à chaque interruption accédant à cette entrée.
- 2 : mode compteur
- 3 : mode encodeur 2X
- 4 : mode encodeur 4X

Les modes codeurs 2X et 4X utilisent 2 entrées consécutives.

Ainsi les entrées 21 et 22 formeront le codeur n°1,  
les entrées 19 et 20 formeront le codeur n°2,  
et les entrées 17 et 18 formeront le codeur n°3.

Un réglage de filtrage est possible pour le codeur n°1 (valeur de 1 à 15 selon le niveau du bruit à l'entrée).

La polarité de chacune de ces entrées peut également être inversée ici.

## 1 Utilisation des entrées en Mode Codeur (17 à 22)

Nous pouvons disposer de 3 codeurs car 2 entrées rapides sont requises pour chacun des codeurs.

Canal 0 : entrées 21 et 22. Fréquence jusqu'à 1Mhz

Canal 1 : entrées 19 et 20. Fréquence jusqu'à 50 khz

Canal 2 : entrées 17 et 18. Fréquence jusqu'à 50 khz

Les paramètres 221 à 226 de la carte sont à configurer en Mode 4 (4X, tous les fronts sont pris en compte), ou en Mode 3 (2X, 1 front sur 2 est pris en compte).

**GetEncoder**( : Renvoie la position d'une entrée codeur

Syntaxe : **GetEncoder**(Canal) 'Canal : de 1 à 3.

Exemple : SetMDW 3018, **GetEncoder**(3)

**SetEncoder**: Affectation d'une valeur à une entrée codeur

Syntaxe : **SetEncoder** Canal, Valeur 'Canal : de 1 à 3.

Exemple : **SetEncoder** 3, 0 'Mise à 0 de l'entrée codeur n°3

## 2 Utilisation des entrées en mode compteur (16 à 22)

On peut donc disposer de 7 compteurs. Les paramètres 220 à 226 de la carte sont à configurer :

En mode 0 (standard), le rafraîchissement du tableau stockant l'état des entrées a lieu toutes les millisecondes.

En mode 1 (sur interruption « IT »), le rafraîchissement du tableau stockant l'état des entrées a lieu à chaque interruption y accédant.

En mode 2 (Mode compteur), le rafraîchissement du tableau stockant l'état des entrées a lieu à chaque front (montant ou descendant, selon configuration de la polarité des entrées (paramètre 200 de la carte)).

**GetCnt** : Lecture d'un des compteurs associés aux entrées rapides 16 à 22. Le résultat est un entier non signé.

Syntaxe : **GetCnt**(CompteurNo)  
CompteurNo = 1 à 7

Exemple : SetMDW 3018, **GetCnt**(4) 'Ecrit à l'adresse 3018 la valeur lue au compteur n°4

**SetCnt** : Écriture dans les compteurs des entrées rapides 16 à 22. L'interpCNC dispose de 7 entrées rapides qui peuvent être utilisées comme entrées de comptage.

Syntaxe : **SetCnt** CompteurNo, Valeur  
Compteur No = 1 à 7  
Valeur = Valeur du compteur

Exemple : **SetCnt** 4, 0 'Remet à 0 le compteur 4

Idem pour les entrées virtuelles, qui peuvent être:

- soit des entrées analogiques utilisées en tant qu'entrées tout ou rien (TOR) et remappée comme entrées numérotées de 23 à 26
- soit des entrées digitales

**Entrées analogiques DIN23 à DIN26**

- |  |  |
|--|--|
| <input type="checkbox"/> DIN23 - Polarité inversée (contact type NC) | <input type="checkbox"/> DIN25 - Polarité inversée (contact type NC) |
| <input type="checkbox"/> DIN24 - Polarité inversée (contact type NC) | <input type="checkbox"/> DIN26 - Polarité inversée (contact type NC) |

**Polarité entrées virtuelles**

- |                             |                             |                             |                             |                             |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| <input type="checkbox"/> 32 | <input type="checkbox"/> 48 | <input type="checkbox"/> 64 | <input type="checkbox"/> 80 |                             |
| <input type="checkbox"/> 33 | <input type="checkbox"/> 49 | <input type="checkbox"/> 65 | <input type="checkbox"/> 81 |                             |
| <input type="checkbox"/> 34 | <input type="checkbox"/> 50 | <input type="checkbox"/> 66 | <input type="checkbox"/> 82 |                             |
| <input type="checkbox"/> 35 | <input type="checkbox"/> 51 | <input type="checkbox"/> 67 | <input type="checkbox"/> 83 |                             |
| <input type="checkbox"/> 36 | <input type="checkbox"/> 52 | <input type="checkbox"/> 68 | <input type="checkbox"/> 84 |                             |
| <input type="checkbox"/> 37 | <input type="checkbox"/> 53 | <input type="checkbox"/> 69 | <input type="checkbox"/> 85 |                             |
| <input type="checkbox"/> 38 | <input type="checkbox"/> 54 | <input type="checkbox"/> 70 | <input type="checkbox"/> 86 |                             |
| <input type="checkbox"/> 39 | <input type="checkbox"/> 55 | <input type="checkbox"/> 71 | <input type="checkbox"/> 87 |                             |
| <input type="checkbox"/> 40 | <input type="checkbox"/> 56 | <input type="checkbox"/> 72 | <input type="checkbox"/> 88 |                             |
| <input type="checkbox"/> 41 | <input type="checkbox"/> 57 | <input type="checkbox"/> 73 | <input type="checkbox"/> 89 |                             |
| <input type="checkbox"/> 42 | <input type="checkbox"/> 58 | <input type="checkbox"/> 74 | <input type="checkbox"/> 90 |                             |
| <input type="checkbox"/> 27 | <input type="checkbox"/> 43 | <input type="checkbox"/> 59 | <input type="checkbox"/> 75 | <input type="checkbox"/> 91 |
| <input type="checkbox"/> 28 | <input type="checkbox"/> 44 | <input type="checkbox"/> 60 | <input type="checkbox"/> 76 | <input type="checkbox"/> 92 |
| <input type="checkbox"/> 29 | <input type="checkbox"/> 45 | <input type="checkbox"/> 61 | <input type="checkbox"/> 77 | <input type="checkbox"/> 93 |
| <input type="checkbox"/> 30 | <input type="checkbox"/> 46 | <input type="checkbox"/> 62 | <input type="checkbox"/> 78 | <input type="checkbox"/> 94 |
| <input type="checkbox"/> 31 | <input type="checkbox"/> 47 | <input type="checkbox"/> 63 | <input type="checkbox"/> 79 | <input type="checkbox"/> 95 |

***Pour l'InterpCNC V2***

**Arrêt d'urgence**

Vérouillage / Dévérouillage

N° entrée AU  Type

N° entrée dévérouillage  Type

---

**Set CNC prête**

1    2    3    4    5    6    7    8  
 9    10    11    12    13    14    15    16

---

**Reset CNC prête**

1    2    3    4    5    6    7    8  
 9    10    11    12    13    14    15    16

---

**Set CNC non prête**

1    2    3    4    5    6    7    8  
 9    10    11    12    13    14    15    16

---

**Reset CNC non prête**

1    2    3    4    5    6    7    8  
 9    10    11    12    13    14    15    16

---

**États des Sorties Digitales au Démarrage**

1    2    3    4    5    6    7    8  
 9    10    11    12    13    14    15    16

---

**Encodeur**

Entrée codeur

---

**Sortie Analogiques**

Etat initial ou quand la carte est vérouillée

Aout 1       Aout 2

Aout 3       Aout 4

**Set CNC prête** permet de forcer à 1 l'état des sorties sélectionnées une fois la carte réarmée après un arrêt d'urgence (permet d'avoir un signal CNC prête).

**Set CNC non prête** permet de forcer à 1 l'état des sorties sélectionnées lorsque survient un arrêt d'urgence (permet d'avoir un signal CNC non prête).

**États des Sorties Digitales au Démarrage** permet de forcer l'état des sorties à la mise sous tension de la carte.

**Encodeur** permet de choisir le mode de fonctionnement du codeur sur les entrées rapides : 2X, 4X, ou compteur A&B

Les **Sorties Analogiques** peuvent être forcées à la valeur souhaitée (de 0 à 10V) à la mise sous tension ou lorsque la carte est verrouillée.

## AIN

### *Pour l'InterpCNC V3 seulement :*

#### – La configuration des entrées Analogiques

**Le gain** : Réglage à 1, donc directement la résolution du convertisseur. La valeur par défaut 4,885198 est un ajustement tenant compte des résistances d'entrée.

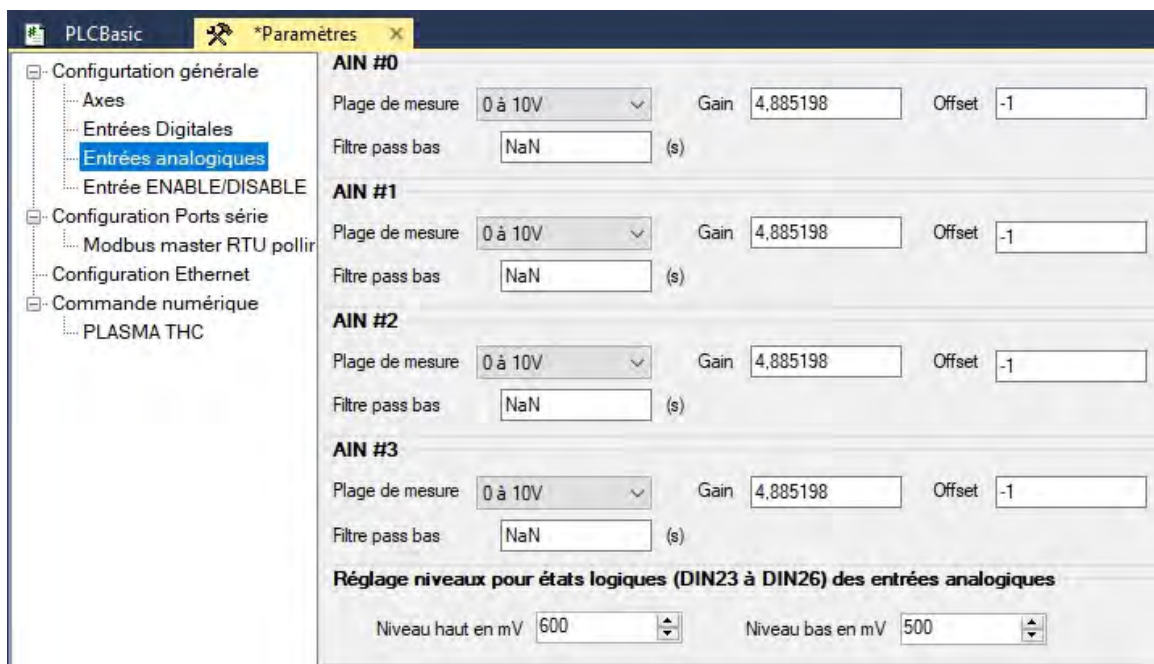
**L'offset** : permet de régler la valeur 0 (car on peut mesurer de 0V à +10V).

**L'échelle de mesure** :

- à 5 (valeur par défaut), la plage de mesure s'étend de 0V à +10V
- à 4, la plage de mesure s'étend de 0V à +5V ce qui permet aussi d'améliorer la précision.

#### – Utilisation des entrées analogiques comme entrées digitales

En cas de besoin de plus d'entrées digitales "tout ou rien" que les 24 disponibles, il est également possible d'utiliser les entrées analogiques. Celles-ci seront mappées en tant que DIN23 à DN26.



Un paramètre (**312**) permet de définir le niveau à partir duquel on considère un niveau haut (600 mV par défaut)

Un paramètre (**313**) permet de définir le niveau en dessous duquel on considère un niveau bas (500 mV par défaut)

Les valeurs sont directement saisies en mV (de 0 à 10000).

**La polarité de ces entrées peut également être inversée** par exemple pour l'utilisation de contacts de type Normalement Fermés.

Pour ce faire, cocher l'option correspondante sur la page "[Entrées Digitales](#)" du menu "**Paramètres**".

## IN\_ENA

### ***Pour l'InterpCNC V3 seulement :***

Dans le **menu Paramètres**, la page de configuration de l' **Entrée ENABLE/DISABLE** donne accès à la configuration de l'état des sorties 0 à 31, qui sera forcé en fonction de l'état physique (0 ou 1) de l'entrée **ENA** (Enable).

NB: Eviter les sélections contradictoires (activer et désactiver à la fois pour le même état) qui ne sont pas gérées.

- **L'État des sorties 0 à 31 au boot** (Registre 270)

Il s'agit d'un registre 32 bits. Ce paramètre permet de définir (forcer) l'état des sorties numériques à la mise sous tension de la carte

(rappel : les sorties 0 à 15 sont des sorties physiques, les sorties 16 à 31 sont des sorties virtuelles (par exemple pour un module externe)).

**Etat des sorties au boot**

<input type="checkbox"/> 0	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	<input type="checkbox"/> 6	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 15
<input type="checkbox"/> 16	<input type="checkbox"/> 17	<input type="checkbox"/> 18	<input type="checkbox"/> 19	<input type="checkbox"/> 20	<input type="checkbox"/> 21	<input type="checkbox"/> 22	<input type="checkbox"/> 23
<input type="checkbox"/> 24	<input type="checkbox"/> 25	<input type="checkbox"/> 26	<input type="checkbox"/> 27	<input type="checkbox"/> 28	<input type="checkbox"/> 29	<input type="checkbox"/> 30	<input type="checkbox"/> 31

**Sortie à activer lorsque ENA passe à 1**

<input type="checkbox"/> 0	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	<input type="checkbox"/> 6	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 15
<input type="checkbox"/> 16	<input type="checkbox"/> 17	<input type="checkbox"/> 18	<input type="checkbox"/> 19	<input type="checkbox"/> 20	<input type="checkbox"/> 21	<input type="checkbox"/> 22	<input type="checkbox"/> 23
<input type="checkbox"/> 24	<input type="checkbox"/> 25	<input type="checkbox"/> 26	<input type="checkbox"/> 27	<input type="checkbox"/> 28	<input type="checkbox"/> 29	<input type="checkbox"/> 30	<input type="checkbox"/> 31

**Sortie à désactiver lorsque ENA passe à 1**

<input type="checkbox"/> 0	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	<input type="checkbox"/> 6	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 15
<input type="checkbox"/> 16	<input type="checkbox"/> 17	<input type="checkbox"/> 18	<input type="checkbox"/> 19	<input type="checkbox"/> 20	<input type="checkbox"/> 21	<input type="checkbox"/> 22	<input type="checkbox"/> 23
<input type="checkbox"/> 24	<input type="checkbox"/> 25	<input type="checkbox"/> 26	<input type="checkbox"/> 27	<input type="checkbox"/> 28	<input type="checkbox"/> 29	<input type="checkbox"/> 30	<input type="checkbox"/> 31

**Sortie à activer lorsque ENA passe à 0**

<input type="checkbox"/> 0	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	<input type="checkbox"/> 6	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 15
<input type="checkbox"/> 16	<input type="checkbox"/> 17	<input type="checkbox"/> 18	<input type="checkbox"/> 19	<input type="checkbox"/> 20	<input type="checkbox"/> 21	<input type="checkbox"/> 22	<input type="checkbox"/> 23
<input type="checkbox"/> 24	<input type="checkbox"/> 25	<input type="checkbox"/> 26	<input type="checkbox"/> 27	<input type="checkbox"/> 28	<input type="checkbox"/> 29	<input type="checkbox"/> 30	<input type="checkbox"/> 31

**Sortie à désactiver lorsque ENA passe à 0**

<input type="checkbox"/> 0	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	<input type="checkbox"/> 6	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 15
<input type="checkbox"/> 16	<input type="checkbox"/> 17	<input type="checkbox"/> 18	<input type="checkbox"/> 19	<input type="checkbox"/> 20	<input type="checkbox"/> 21	<input type="checkbox"/> 22	<input type="checkbox"/> 23
<input type="checkbox"/> 24	<input type="checkbox"/> 25	<input type="checkbox"/> 26	<input type="checkbox"/> 27	<input type="checkbox"/> 28	<input type="checkbox"/> 29	<input type="checkbox"/> 30	<input type="checkbox"/> 31

**AOUT sur perte de ENA**

Etat de la sortie AOUT0 en mV (0 à 10000)

Etat de la sortie AOUT1 en mV (0 à 10000)

- **L'État des sorties analogiques (AOUT0 et AOUT1) sur perte de l'ENable** (Registres 330 et 331).  
Ces sorties seront forcées aux tensions correspondant à ces valeurs, lorsque la liaison de l' ENA est interrompue (cas d'arrêt d'urgence).

## Port\_serie

### ***Pour l'InterpCNC V3 seulement :***

The screenshot shows the configuration interface for two serial ports, COM1 and COM2. Each port configuration is contained within a separate panel.

**Port COM1 Configuration:**

- Mode:** Disable
- Communication settings:**
  - Baud rate: 115 200 bauds
  - Data bits: 8 bits
  - Parity: None (Aucune)
  - Stop bit: 1 bit
- Modbus Slave settings:** Slave ID: 1
- Modbus master settings:**
  - Inter frame delay (ms): 0
  - Max retry: 3
- DMX Device Settings:**
  - Base address: 1
  - Chanel used: 15

**Port COM2 Configuration:**

- Mode:** Disable
- Communication settings:**
  - Baud rate: 115 200 bauds
  - Data bits: 8 bits
  - Parity: None (Aucune)
  - Stop bit: 1 bit
- Modbus Slave settings:** Slave ID: 1
- Modbus master settings:**
  - Inter frame delay (ms): 0
  - Max retry: 3
- DMX Master settings:**
  - Chanel used (transmitted): 0

#### **La configuration de Communication (COM1 et COM2)**

- Ces paramètres (400 et 420) définissent le mode de fonctionnement des ports COM1 et 2 :

0=désactivé, 1=Esclave, 2=Maître, 3=Mode DMX (Slave pour le COM1, Master pour le COM2). Voir également le manuel de l'interpréteur Basic.

- Les paramètres **Communication settings** (401 à 404 et 421 à 424) sont de grands classiques de réglage des ports de communication série (Vitesse en bauds, nombre de bits de données, parité, bits de stop)

- Les paramètres **Slave ID** (405 et 425) définissent l'ID de la carte lorsqu'elle est configurée en tant qu'esclave (voir paramètres 400 et 420)

- Les paramètres **Inter frame delay** (408 et 426) définissent le délai minimum en millisecondes entre l'envoi de 2 trames Modbus (certains variateurs par exemple)

nécessitent au moins 5 ms entre 2 trames reçues)

- Les paramètres **Max retry** (409 et 427) définissent le nombre de tentatives infructueuses avant de retourner une erreur de transmission (visible dans « Toutes les Variables PLC » -> registres Modbus RTU master error (ou success)).

## Polling

### **Pour l'InterpCNC V3 seulement :**

La carte InterpCNC en tant que maître, peut communiquer via Modbus avec d'autres périphériques (variateurs, IHM, PLC, ...) en envoyant des requêtes de lecture ou écriture à un périphérique esclave.

**Le polling, appelé aussi mapping (ou mappage en français) consiste à créer une copie auto-actualisée d'une zone de données, vers une autre zone de données.**

Ainsi par exemple on pourra écrire des bits ou registres localement, et ces valeurs seront automatiquement envoyés tous les X ms pour mettre à jours les registres cibles du périphérique esclave.

De même en lecture: des bits et registres de l'esclave peuvent être copiés périodiquement tous les X ms vers des registres ou bits locaux du maître.

Automatic Polling #1		
Master COM port :	Slave modbus ID	Polling period (ms)
COM 1	0	50
Action		
Read slave coils and set local digital outputs or coils		
Data address on slave	Nombre de données	Local data address
0	1	0

Pour ce faire, spécifier le **port COM** auquel est connecté l'esclave, ainsi que l'**IDentifiant Modbus** de ce dernier et la fréquence souhaitée de rafraîchissement des trames (en ms).

L'action peut porter sur la lecture ou l'écriture de bits ou registres, situés chez l'esclave à l'adresse du champ **Data adress on slave**, pour le **Nombre de données** successives, et seront mappées localement chez le maître (la carte InterpCNC) à partir de l'adresse du champ **Local data address**

## Ethernet

### **Pour l'InterpCNC V3 seulement :**

**Reseau**

Attribution IP Configuration manuelle Discovery port (UDP)

Nom NETBIOS

Configuration IP

Adresse IP	Masque reseau	Passerelle
<input type="text" value="192.168.10.12"/>	<input type="text" value="255.255.255.0"/>	<input type="text" value="192.168.10.254"/>

Configuration Modbus

TCP server port	UDP server port
<input type="text" value="502"/>	<input type="text" value="500"/>

– **Les paramètres de Configuration Réseau** (Paramètres 520 à 544)

Ici vous pourrez donner un nom Netbios à la carte sur votre réseau, choisir le mode DHCP (à 1), ou bien définir l'adresse IP de votre carte, masque de sous réseau, passerelle, port, etc...

## Commande\_numerique

**Configuration générale axes CNC**

Durée des pulses (µs)

Ecart de jonction (mm)

Tolérance Arc (mm)

**Axe 1**

Inverser sens de rotation  Pulses actifs à 1

Résolution axe (pulses/mm)	<input type="text" value="250"/>	Vitesse maximale (mm/mn)	<input type="text" value="500"/>
Course maximale (mm)	<input type="text" value="200"/>	Accélération (mm/s <sup>2</sup> )	<input type="text" value="250"/>

Ces paramètres concernent la configuration avancée des signaux pulses et direction en mode commande numérique, pour chacun des axes.

Le logiciel **GALAAD** modifiera certains d'entre eux avec les réglages qui lui seront faits.

Il est déconseillé à l'utilisateur de modifier ces valeurs.

Les cartes IntercpCNC V2 et V3 sont compatibles avec **GALAAD** (<https://www.galaad.net>), en tant que logiciel de découpe plasma et d'usinage. La torche est normalement montée sur l'axe Z. **THC** signifie : Contrôle de la hauteur de la torche.

Le THC est un système géré par le firmware pour gérer le positionnement optimal de l'axe Z lors de la découpe plasma, en fonction de la vitesse et de la tension d'arc.

**NB:** La carte V2.4 ne prend pas en charge la gestion du THC.

## Pour la carte V3 :

**Paramètres généraux**

Mode fonctionnement THC : ARC OK sur entrée DIN. Régulation sur tension d'arc

Délai entre activation THC et début d'opération : 0 (ms)

**Mesure tension d'arc**

Source mesure tension : Entrée AIN 0

Registre personnalisé : 0

Tension ARC réelle pour entrée AINx à 0V (offset) : 0 (mV)

Tension ARC réelle pour entrée AINx à 10.0V : 0 (mV)

Note : Le réglage du filtre sur l'entrée analogique utilisée permet de régler la stabilité de la mesure.

**Limites de fonctionnement**

Montée axe Z maximale : 10 (mm)

Descente axe Z maximale : 3 (mm)

Tension Maxi. échantillonnage : 150000 (mV)

Tension Mini. échantillonnage : 50000 (mV)

Seuil vitesse engagement effectif THC : 90 (% de la vitesse de coupe)

Seuil désengagement THC : 50 (% de la vitesse de coupe)

**Régulation**

Gain proportionnel Vitesse : 25

Zone morte de régulation (fenêtre de tolérance) : 0 (mV)

« **Délai entre activation THC et début d'opération** » est une durée en ms pour retarder l'activation du THC.

Les générateurs plasma disposent généralement d'un diviseur interne afin que la tension de l'arc puisse être mesurée par une entrée analogique de la carte IntercpCNC. Ainsi certains générateurs ont donc une sortie qui varie de 0V à 10V, ou de 0V à 5V, ou de 1V à 4V, ou autre, selon les marques ou les modèles.

Bien entendu, la mesure de la tension d'arc est plus précise sur toute la plage de l'entrée: 0 V à 10 V.

En plus d'être un isolateur contre les parasites générés par le générateur et la torche, le module amplificateur **SOPROLEC THC** en option, permet d'adapter avec précision la plage de sortie de votre générateur, en une plage 0 à 10V.

Veuillez contacter **SOPROLEC** pour plus de renseignements si vous envisagez

l'achat de ce module.

"**Source mesure tension**" définit quelle entrée analogique de la carte sera utilisée pour la mesure de tension d'arc.

Ensuite, les deux paramètres "**Tension ARC réelle...**" informeront de la plage de sortie du générateur, ou bien ceux-ci devront être réglés sur 0 et 10000 mV si vous disposez du **module SOPROLEC THC**.

"**Montée axe Z maximale**" et "**Descente axe Z maximale**" définiront respectivement la montée maxi et la descente maxi en mm pour la torche, en fonction de l'épaisseur du matériau.

Le THC doit être désactivé lorsque la vitesse de coupe est trop lente (dans les courbes par exemple), et réactivé lorsque la vitesse revient.

« **Seuil vitesse engagement effectif THC** » définit le % de la vitesse de coupe auquel réactiver le THC.

« **Seuil désengagement THC** » définit le % de la vitesse de coupe auquel désactiver le THC.

La "**Zone morte de régulation**" est une zone morte en mV, configurée pour ignorer les fluctuations de tension indésirables, observées sur l'entrée analogique dédiée à la mesure de la tension d'arc.

## Pour la carte V2.2B :

THC			
Paramètres généraux			
Fonction THC	Inactif		
Mesure tension d'arc			
Source mesure tension	Entrée AIN 1	Filtre mesure Arc	3
Contrôle THC			
No sortie activation	Aucune	No entrée validation	Aucune
Retard activation	300	Consigne par	Logiciel
Consigne Mini	0	Consigne Maxi	1024
Régulation			
Période échantillonnage	3	Vitesse Maxi	1000
Gain Intégrateur	1000	Accel Maxi	1000
Correction Plus Maxi	1000	Gain Proportionnel	100000
<input checked="" type="checkbox"/> Limitation De Vitesse Par X Y		Correction Moin Maxi	1000
		% de vitesse X Y	100

**Remarque** : Ces paramètres sont écrits dans ces registres de la carte par le logiciel **GALAAD**.

Il n'est donc normalement pas nécessaire de les régler dans ICNCStudio, puisqu'ils seront "écrasés" par les valeurs envoyées par GALAAD.

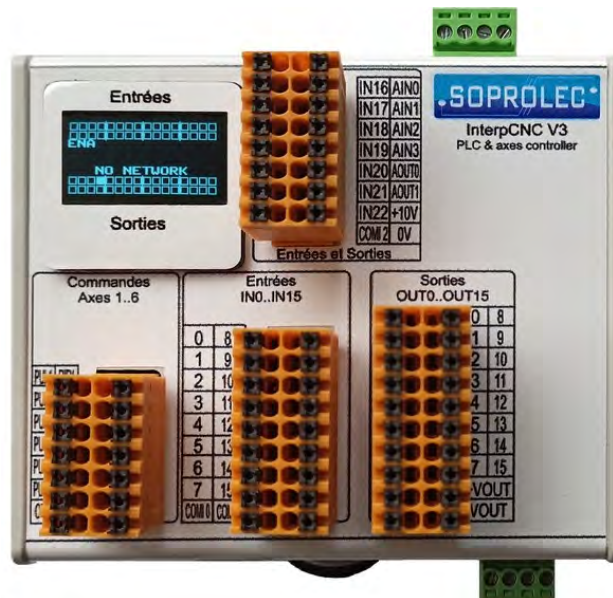
Veillez lire la section THC du manuel de GALAAD pour une information plus complète sur l'utilisation du THC.

## Notice de l'InterpCNC V3

**SOPROLEC**  
 ZAC DE L'EPINE  
 72460 SAVIGNE L'EVEQUE  
 Tél : +33 (0)2 4376 4476



## Carte d'axes SOPROLEC InterpCNC V3



### Notice d'installation

## Presentation

La carte InterpCNC V3 est une carte Automate destinée principalement au contrôle d'axes.

Elle dispose de 6 sorties de commandes d'axes pouvant être interpolés ou indépendants.

Développée sur la base d'un puissant processeur 32bits à 480 mHz, l'InterpCNC V3 offre aussi des performances idéales pour les applications de commande numérique (CNC) et également les applications d'automatisme nécessitant un contrôle/commande d'axe performant.

Elle a pour avantage de fournir un maximum de puissance, de connectivité, et de fonctionnalités, dans un boîtier très compact, et directement fixable sur un rail DIN.

De plus, l'InterpCNC dispose d'un puissant interpréteur de langage Basic permettant la gestion d'automatisme de manière autonome. (Voir notre documentation sur l'Interpréteur PLCBasic).

Votre projet se compose d'une part d'un programme Basic, mais aussi des registres et bits, entrées et sorties, que vous aurez déclarés et nommés dans chacun des tableaux.

Il se compose aussi des recettes que vous aurez pu définir dans l'éditeur de recettes.

L'ensemble sera sauvegardé dans un unique fichier portant l'extension « **.plc** »

L'interface de commande d'axe en mode Step/Direction est compatible avec toute la gamme de motorisations proposée par la société SOPROLEC (motorisation pas à pas, motorisation brushless).

Trois interfaces de communication sont disponibles :

USB : port Com virtuel (protocole Modbus RTU), pour une communication rapide dans les applications de commande numérique

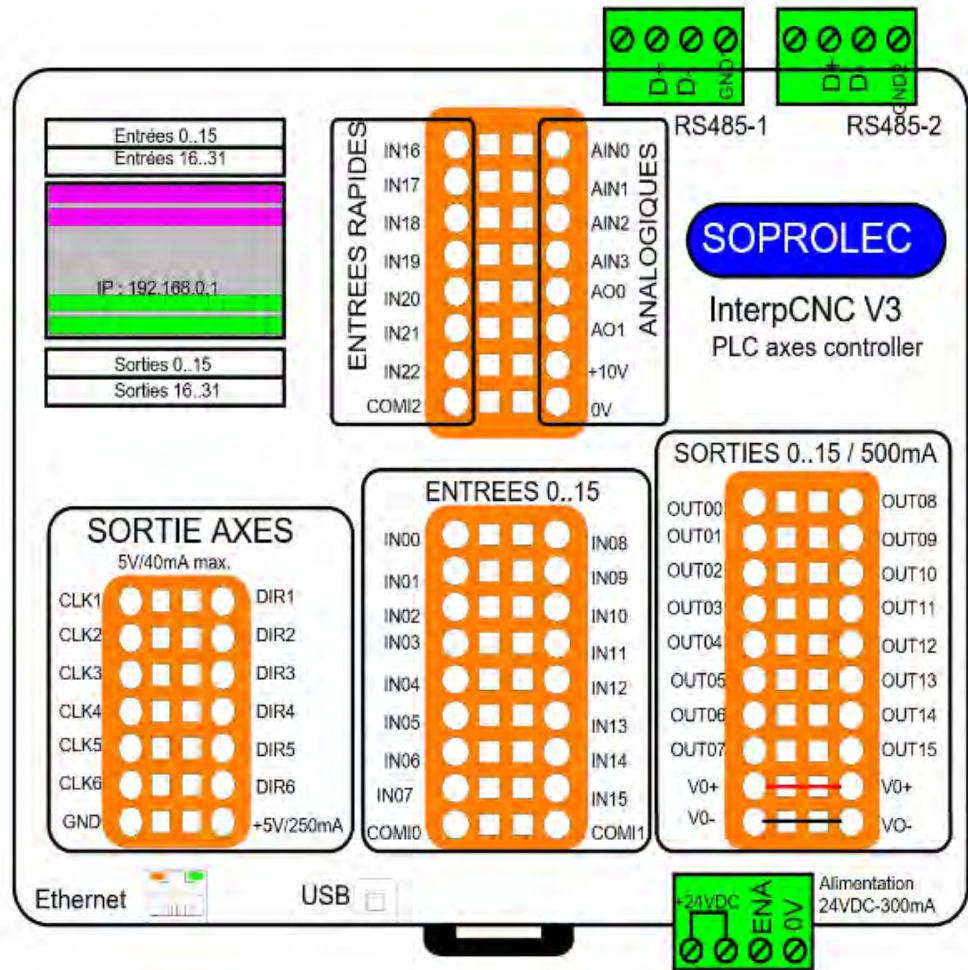
Serie : 2 ports RS485 (COM1 et COM2, protocole Modbus RTU) pour les applications industrielles

Ethernet : protocole Modbus TCP ou Modbus UDP.

Le port COM1 est également compatible avec le protocole DMX512, très utilisé dans le monde du spectacle pour le pilotage des matériels (voir Manuel de l'Interpréteur Basic).

Un mini écran OLED, vous permet de visualiser en temps réel l'état (0 ou 1) des 32 premières entrées et des 32 premières sorties de la carte, ainsi que diverses informations (version du Firmware à l'allumage, état de l'entrée ENable, adresse IP de la carte, etc...)

## Vue d'ensemble de la carte InterpCNC V3

**Alimentation :**

Alimentation de la carte en 24VDC/300mA

**Caractéristiques des sorties :**

Sorties 0 à 15 : Sorties Opto-Isolées, 500mA maxi par sortie.

**Attention** : ne pas consommer 500mA sur plusieurs sorties en même temps.

Ces sorties doivent être alimentées en externe sur VOut+ et Vout-, tension < 32V.

Sorties 16 à 96 : Sorties virtuelles, nécessitent un ou plusieurs boîtiers d'extension de sorties via communication Modbus (exemple : Kinco KS123). Voir mise en œuvre page 28.

Sorties PUL1 à PUL6, et DIR1 à DIR6 : Sortie TTL 5V/40mA maxi

**Caractéristiques des entrées :**

Les entrées numériques IN0 à IN15 sont Opto-Isolées. Elles répondent à la norme **IEC 61131-2** et sont de type 3, c'est à dire que le courant et la tension minimum nécessaires pour les faire passer à l'état haut se situent respectivement entre 2.27/2.45 mA, et 7.44/7.98 V pour la tension.

NB : Les entrées IN0 à IN15 peuvent être filtrées, c'est à dire que l'on peut leur appliquer un Filtre anti-rebond (4 délais possibles en ms, et 3 modes de filtrage): exemple : cas d'un contacteur sec (bouton ou relais) qui pourrait générer des impulsions rapides en n'établissant pas un contact franc et net.

Voir le tableau des Paramètres de la carte pour les réglages (Paramètres 210 à 213).

Entrées IN0 à IN15 : 0 à 24V

Les communs COMI0 ou COMI1 doivent être raccordés au 0V (Entrées PNP) ou au +24V (Entrées NPN).

Entrées IN16 à IN22 : entrées de comptage rapide de type TTL

Ces entrées sont dites rapides, car celles-ci correspondent des entrées physiques directement gérées par le micro-contrôleur (à la différence des entrées 0 à 15 opto-isolées qui sont gérées par un composant utilisant le bus SPI).

COMI3 doit être raccordé au 0V (PNP) ou au +24V (NPN).

L' **entrée IN21** peut être configurée en entrée codeur incrémental (2X ou 4X).

Entrées IN23 à IN255 : Entrées virtuelles, nécessitent un ou plusieurs boîtiers d'extension d'entrées via communication Modbus (exemple : Kinco KS123).

Entrée ENABLE : Fonction d'arrêt d'urgence. 0 à 32V maxi. Niveau haut à partir de 3,5V.

#### **Entrées/Sorties Analogiques :**

**4 Entrées analogiques AIN0 à AIN3** : -10V à +10V.

La résolution est de 16 bits (valeurs lues de -32767 à +32767).

-32768 : -10V

0 : 0V

32767 : +10V

NB : les paramètres 300 à 311 permettent d'étalonner les valeurs reçues sur ces 4 entrées analogiques. Voir tableau des paramètres.

**2 Sorties Analogiques AOUT0 et AOUT1** : 0 à 10V. Résolution 11 bits (de 0 à 2047).

0 : 0V

2047 : +10V

## **Installation**

L'installation sous Windows 7, 8, 10 et 11 ne nécessite pas de driver particulier.

En USB, la carte est reconnue automatiquement en Plug and Play en tant que « Périphérique Serie USB » (Port COM virtuel).

En Ethernet, la connexion est une connexion réseau classique sachant que la carte possède sa propre adresse IP (re-définissable), et que l'on utilise généralement le port 502 en TCP, ou 500 en UDP.

Il conviendra de s'assurer que le pare-feu de votre ordinateur ne bloque pas la connexion avec la carte, afin que le logiciel ICNCStudio puisse la détecter et s'y connecter.

Si vous connectez la carte InterpCNC sur votre réseau (via un routeur ou switch), assurez-vous que son adresse IP est bien sur le même réseau que le PC sur lequel sera utilisé ICNCStudio (=même début d'adresse IP, exemple: 192.168.10.xx)

Quelle que soit le type d'interface utilisé, tous les échanges de données restent basés sur le protocole Modbus. La carte peut alors être configurée soit en Maître ou Esclave (voir tableau des paramètres).

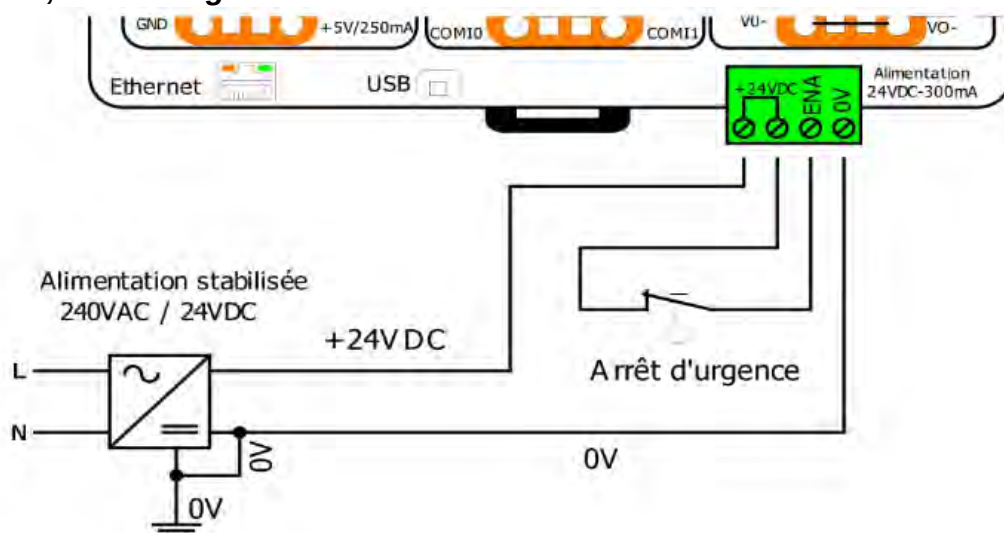
En règle générale l' IHM occupe toujours la position maître, et la carte InterpCNC la

position esclave.

Pour une mise en œuvre simplifiée : tous les connecteurs sont débrochables et pour les connecteurs orange, le raccordement des fils est de type rapide (utiliser un petit tournevis plat pour libérer chaque ressort central de pression, aussi bien à l'insertion qu'au retrait).

## Raccordement

### Alimentation, Arrêt d'urgence



Le raccordement entre +24V et ENA est indispensable. Utilisez donc un contact de type NF (normalement fermé) pour assurer cette liaison.

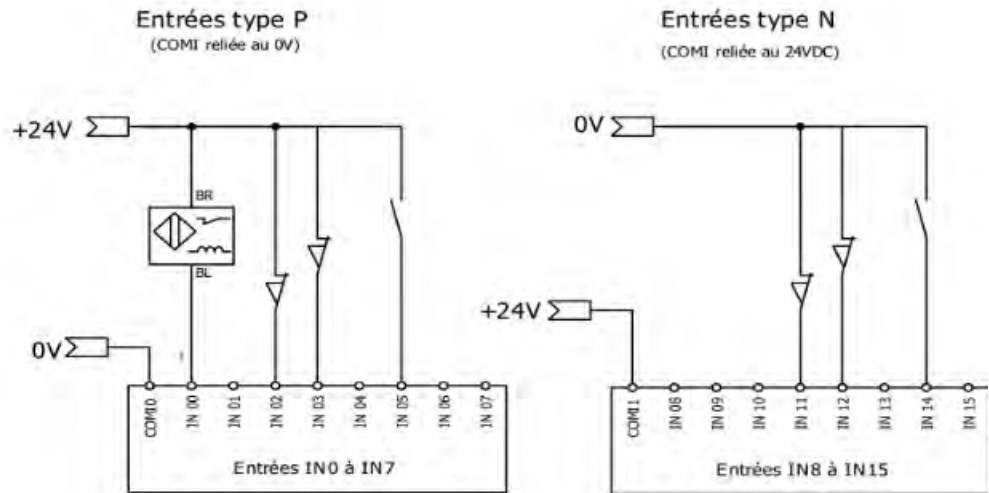
Par sécurité, un contact ouvert sur cette liaison désactivera les sorties de commandes d'axes, et laissera les sorties OUT0 à 15 en l'état, jusqu'au prochain appui sur le bouton d'arrêt d'urgence (acquiescement et reprise).

### Raccordement des entrées

Les entrées physiques de la carte peuvent être configurées soit en PNP, soit en NPN.

Une configuration différente est possible sur chaque groupe d'entrées associées à un COMI.

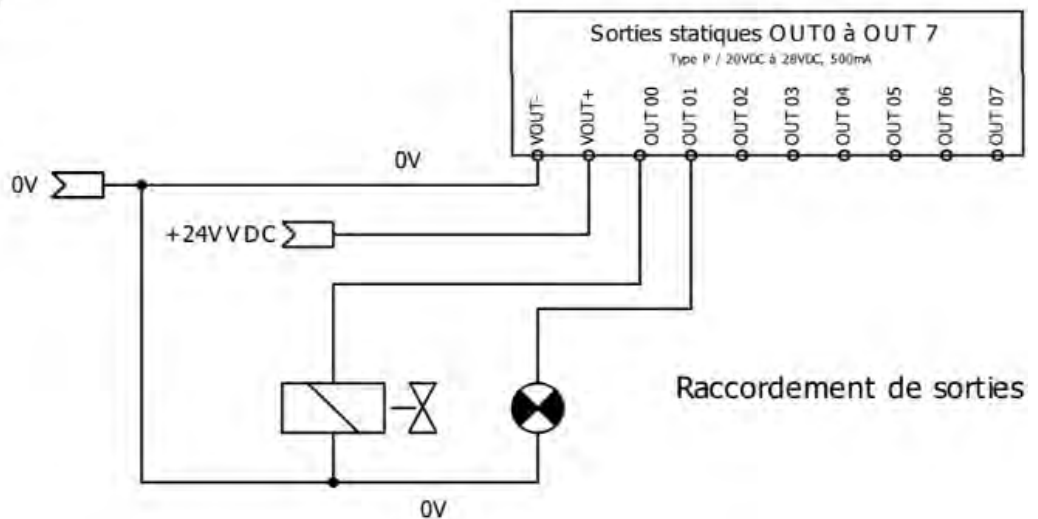
Exemples de raccordements :



### Raccordement des sorties

Les sorties statiques doivent être alimentées en externe (connecter le +24V à VOUT+, et le 0V à VOUT-). Elles sont de type PNP.

Exemple de raccordement :

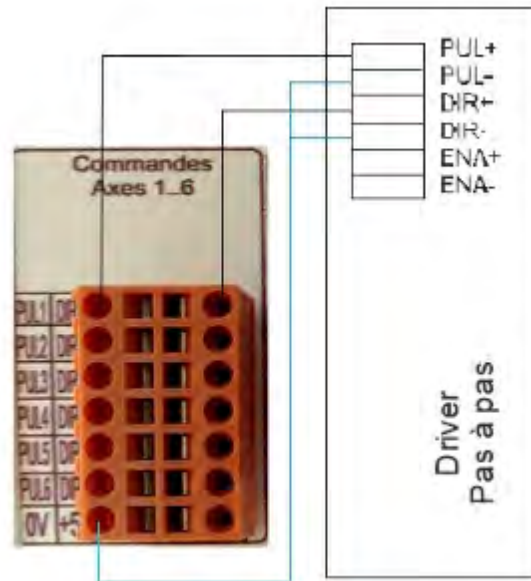


Idem pour les sorties 8 à 15.

NB : les +VOUT et -VOUT sont reliés en interne sur la carte. Vous pouvez donc n'alimenter qu'un seul côté (sauf si l'intensité cumulée des sorties s'avère importante, vous augmenterez ainsi le calibre).

### Commande d'un driver moteur

Les commandes pulses/direction de la carte sont à relier directement aux entrées correspondantes du driver.



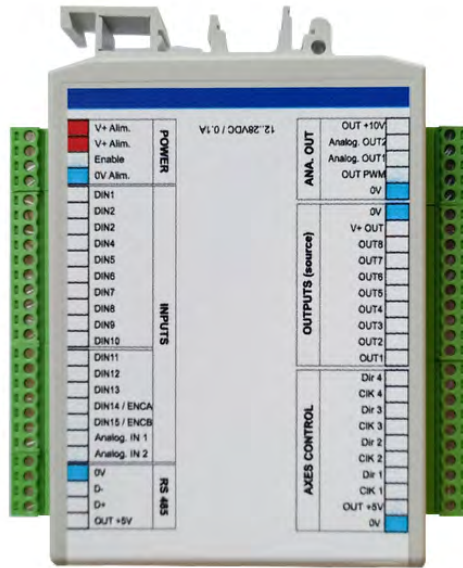
## Notice de l'InterpCNC V2

---

**SOPROLEC**  
ZAC DE L'EPINE  
72460 SAVIGNE L'EVEQUE  
Tél : +33 (0)2 4376 4476



**Cartes d'axes  
SOPROLEC  
InterpCNC V2**



## Notice d'installation

Les cartes **InterpCNC V2** sont des cartes de contrôle d'axes. Elle disposent de 5 sorties de commandes d'axes (V2.2) ou 4 sorties de commandes d'axes (V2.4) pouvant être interpolés ou indépendants.

Développées sur la base d'un puissant processeur 32bits, les InterpCNC V2 offrent des performances idéales pour les applications de commande numérique et également, les applications d'automatisme nécessitant un contrôle/commande d'axe économique.

De plus, l'InterpCNC dispose d'un interpréteur de langage Basic permettant la gestion d'automatismes de manière autonome.

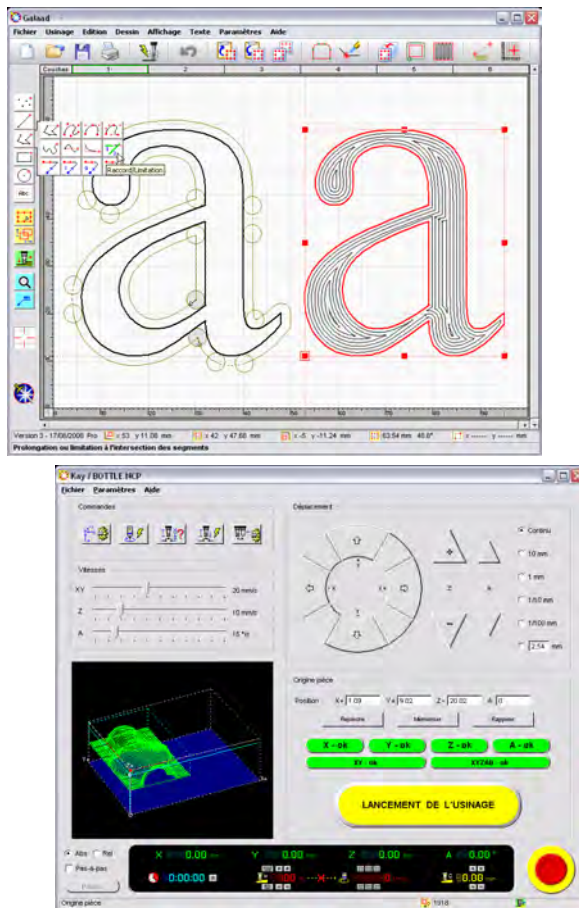
L'interface de commande d'axe en mode Step/Direction est compatible avec toute la gamme de motorisation proposée par la société SOPROLEC (motorisation pas à pas, motorisation brushless).

Deux interfaces de communication sont disponibles :

- USB pour une communication rapide dans les applications de commande numérique,
- RS485 MODBUS pour les applications industrielles.

Différentes bibliothèques de communication permettent l'utilisation de l'InterpCNC avec plusieurs logiciels de pilotage :

- GALAAD (Ensemble CAO/FAO Pilotage intégré)
- MULTI-CN solution de pilotage CNC avec automatisme professionnel
- Mach3 et Mach4 solutions de pilotage CNC



GALAAD CAO/FAO/Pilotage

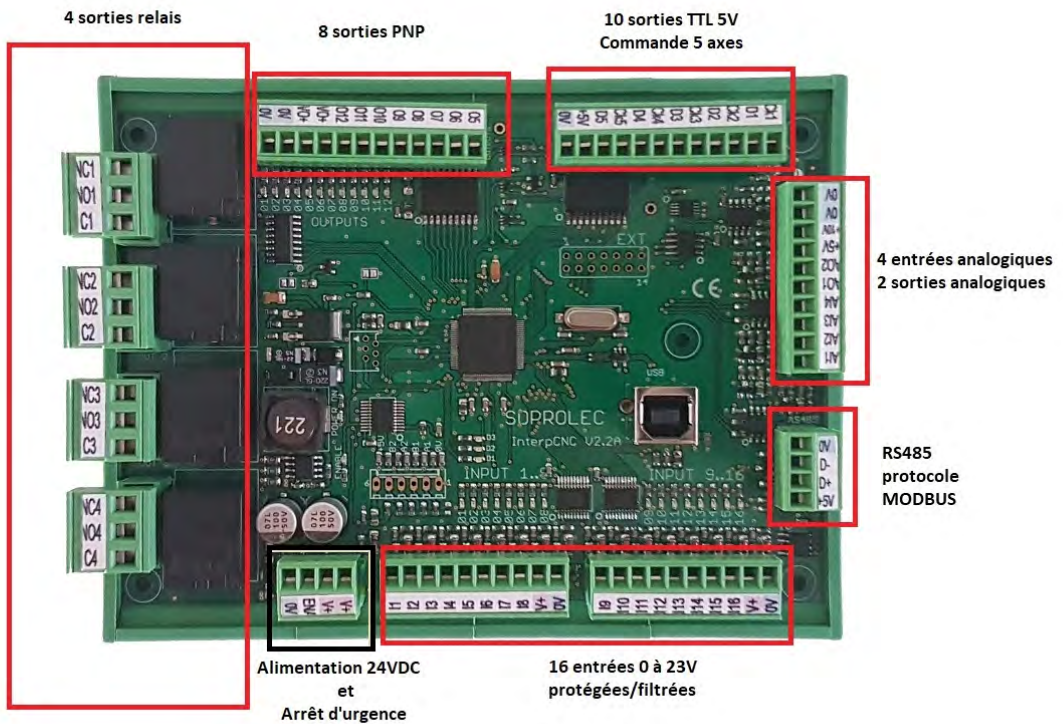
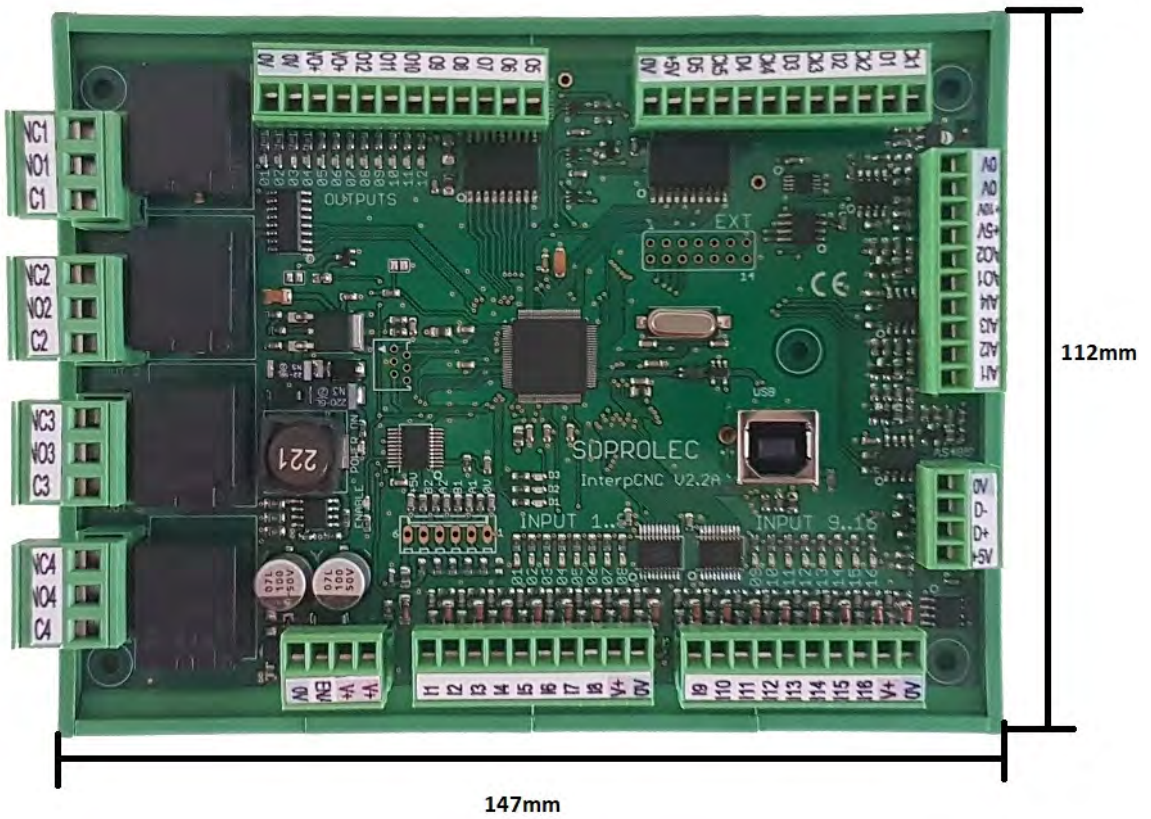


MULTI CN  
Interpréteur / Pilotage  
Gestion d'automatisme étendus



MACH4 pilotage CNC

**Vue d'ensemble de la carte InterpCNC V2.2**



Les bornes repérées V+ sont toutes reliées entre elles.  
 Les bornes 0V sont toutes reliées entre elles.

Les bornes +5V sont toutes reliées entre elles (sortie régulées +5V de la carte).

L'état des sorties est visualisé par les LEDs vertes.

L'état des entrées est visualisé par les LEDs rouges.

Le voyant ENABLE représente l'état de l'entrée ENA qui fait fonction d'arrêt d'urgence.

**Alimentation :**

Alimentation de la carte en 24VDC/250mA

**Caractéristiques des sorties :**

Sorties 1 à 4 : Relais à contact sec 250V/10A.

Sorties 5 à 12 : PNP 350mA (total de toutes les sorties)/24V

Sorties CK1 à CK5, et D1 à D5 : Sortie TTL 5V/20mA maxi

**Caractéristiques des entrées :**

Les entrées numériques I1 à I16 : 0 à 32V maxi. Niveau haut à partir de 3,5V. Filtre 1,5 KHz.

Entrée ENABLE : Fonction d'arrêt d'urgence. 0 à 32V maxi. niveau haut à partir de 3,5V.

1 entrée codeur incrémental ou 2 entrées de comptage rapide de type TTL.

**Entrées/Sorties Analogiques :**

**4 Entrées analogiques AI1 à AI4** : 0V à +10V. Impédance 1M.

La résolution est de 10 bits (de 0 à 1023).

0 : 0V

1023 : +10V

**2 Sorties Analogiques AO1 et AO2** : 0 à 10V. Résolution 10 bits (de 0 à 1023).

0 : 0V

1023 : +10V

**Duplication d'Axes :**

Afin de permettre l'utilisation de 2 moteurs pour un même axe, il est possible de coupler 2 axes.

Seul l'axe X peut être dupliqué, soit sur l'axe A, soit sur l'axe B.

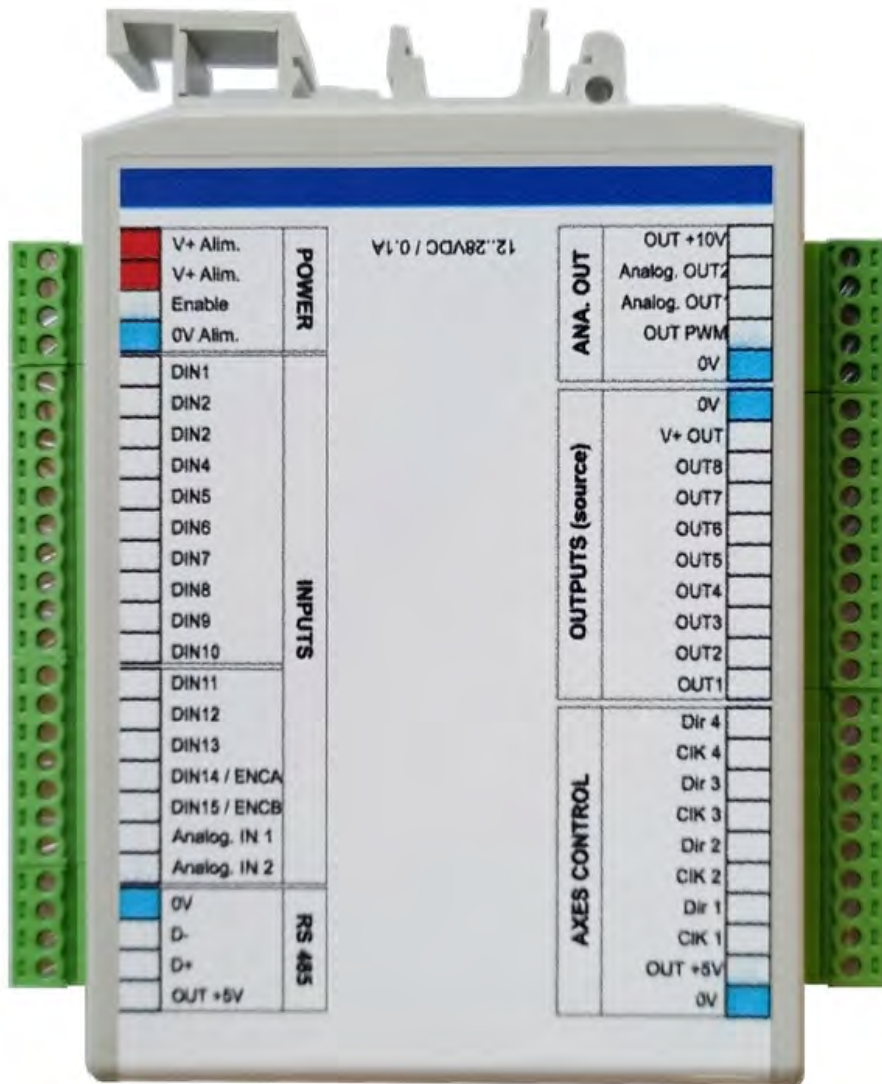
Ainsi la carte générera des signaux pulses identiques sur les sorties CK1 et CK4 (ou CK5), de même entre D1 et D4 (ou D5).

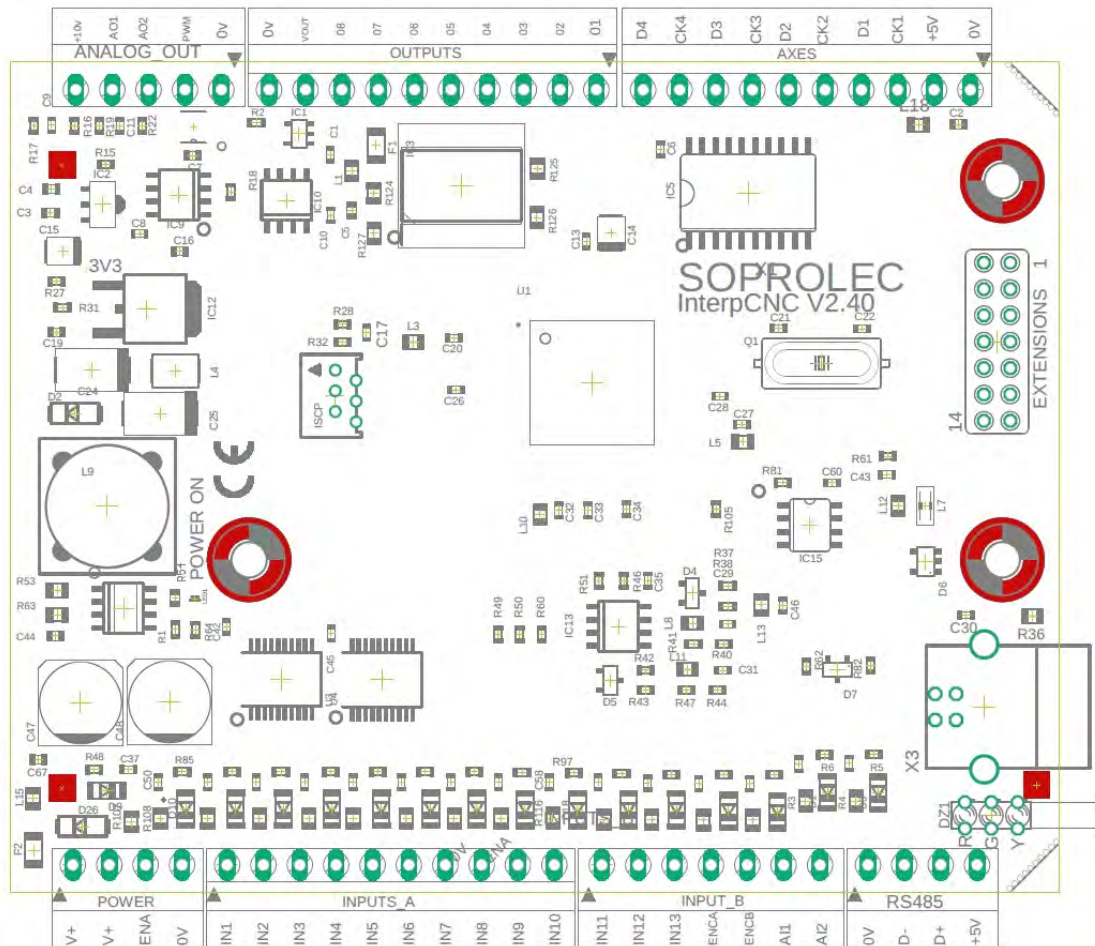
Dans les paramètres -> Axes: choisir "Duplication axes: X vers A" ou "X vers B", et cliquer sur "Send to PLC".

Tous les connecteurs sont débrochables pour une mise en oeuvre simplifiée.

---

**Vue d'ensemble de la carte InterpCNC V2.4**





Les bornes repérées V+ sont toutes reliées entre elles.  
 Les bornes 0V sont toutes reliées entre elles.  
 Les bornes +5V sont toutes reliées entre elles (sortie régulées +5V de la carte).

L'état des sorties est visualisé par les LEDs vertes.  
 L'état des entrées est visualisé par les LEDs rouges.

Le voyant ENABLE représente l'état de l'entrée ENA qui fait fonction d'arrêt d'urgence.

#### Alimentation :

Alimentation de la carte en 24VDC/250mA

#### Caractéristiques des sorties :

Sorties 1 à 8 : PNP 350mA (total de toutes les sorties)/24V  
 Sorties CK1 à CK4, et D1 à D4 : Sortie TTL 5V/20mA maxi

#### Caractéristiques des entrées :

Les entrées numériques DIN1 à DIN13 : 0 à 32V maxi. Niveau haut à partir de 3,5V. Filtre 1,5 KHz.

Entrée ENABLE : Fonction d'arrêt d'urgence. 0 à 32V maxi. niveau haut à partir de 3,5V.

1 entrée codeur incrémental ou 1 entrée de comptage rapide de type TTL.

#### Entrées/Sorties Analogiques :

**2 Entrées analogiques AIN1 à AIN2** : 0V à +10V. Impédance 1M.  
La résolution est de 10 bits (de 0 à 1023).  
0 : 0V  
1023 : +10V

**2 Sorties Analogiques AOUT1 et AOUT2** : 0 à 10V. Résolution 10 bits (de 0 à 1023).  
0 : 0V  
1023 : +10V  
**1 sortie analogique PWM (AO3)** : 0 à 5V TTL (fréquence 10 KHz)

**Duplication d'Axes :**

Afin de permettre l'utilisation de 2 moteurs pour un même axe, il est possible de coupler 2 axes.

Toutefois, seul l'axe X pourra être dupliqué sur l'axe A.

Ainsi la carte générera des signaux pulses identiques sur les sorties CK1 et CK4, de même entre D1 et D4.

Dans les paramètres -> Axes: choisir "Duplication axes: X vers A", et cliquer sur "Send to PLC".

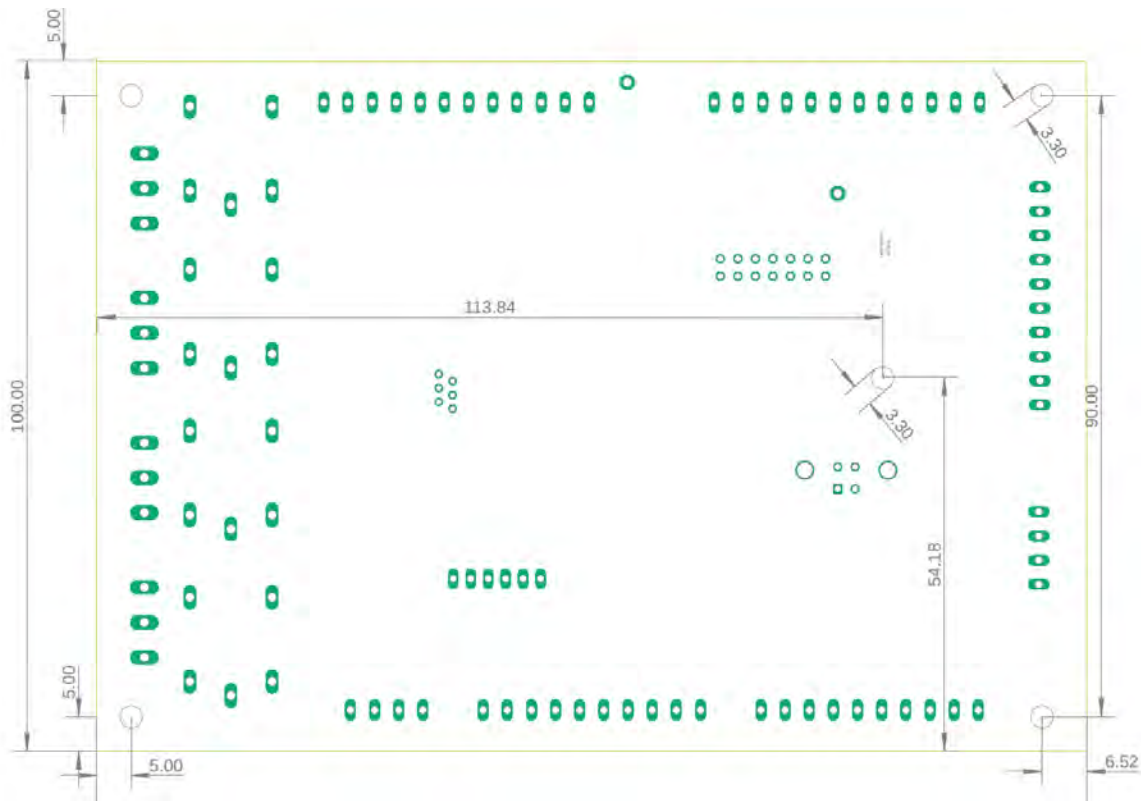
Tous les connecteurs sont débrochables pour une mise en oeuvre simplifiée.

## Installation

### Installation de la carte V2.2 :

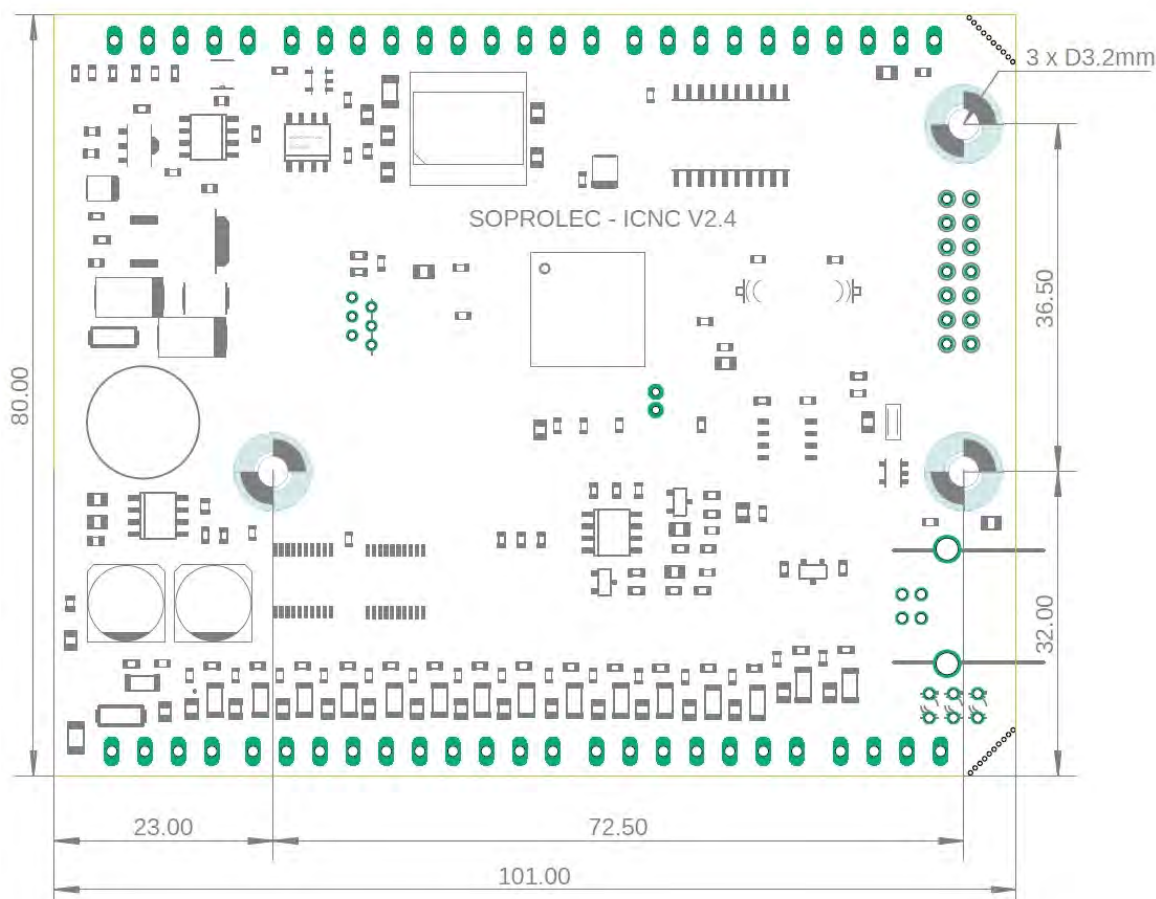
Le support pour fixation sur rail DIN est recommandé.

Dans le cas d'une fixation sur tout autre support, voici le plan de perçage à respecter:



## Installation de la carte V2.4 :

La carte est intégrée dans un boîtier de protection fixable sur rail DIN.  
 Dans le cas d'une fixation sur tout autre support (non recommandé sans boîtier, perte de la garantie si détérioration du scellé), voici le plan de perçage à respecter:



Lors de la première connexion d'une carte InterpCNC V2 au PC, Windows demande l'installation du driver de communication USB.  
 Pour Windows 8, Windows 10 et 11, l'installation du driver est automatique.  
 Pour Windows 7, ce dernier est disponible dans le dossier "**USBDriver**" (à télécharger depuis la rubrique "Product documents" de la fiche produit de la carte sur notre site [www.soprolec.com](http://www.soprolec.com)).

Utilisation avec GALAAD :

Copier les fichiers du dossier "GALAAD" dans le dossier d'installation de GALAAD.

Utilisation avec MULTICN :

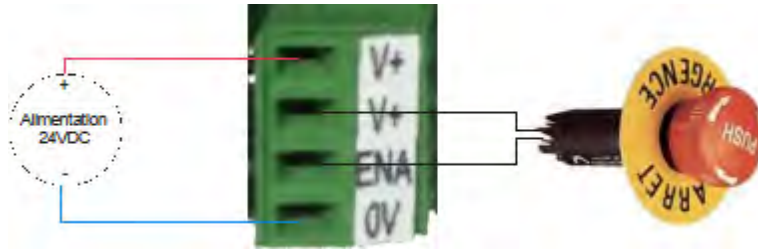
Copier le dossier "MultiCN" dans le dossier d'installation de MultiCN.

Utilisation avec Mach3 ou Mach4 :

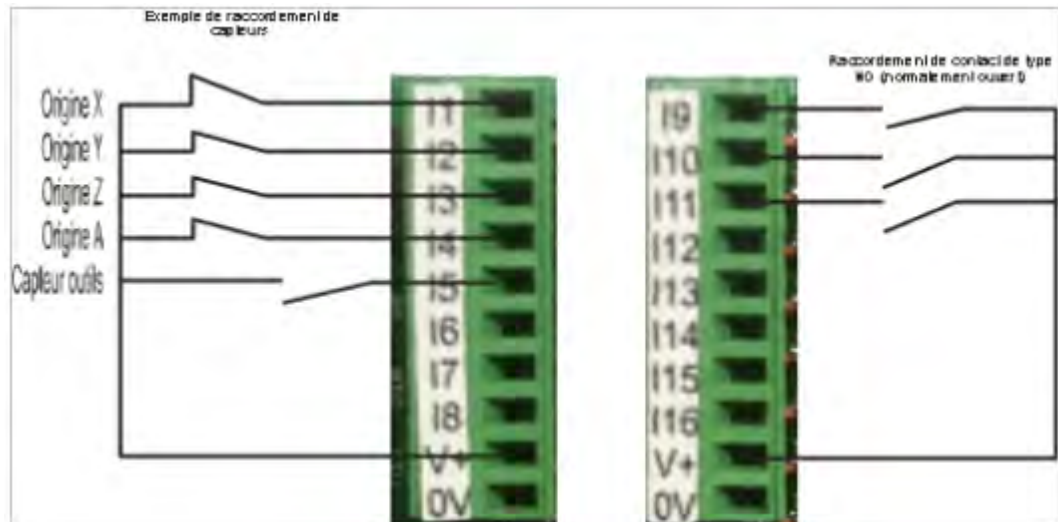
Copier les fichiers de notre PlugIn (à télécharger sur notre site [www.soprolec.com](http://www.soprolec.com)), dans le dossier "PLUGIN" de Mach3 ou Mach4.

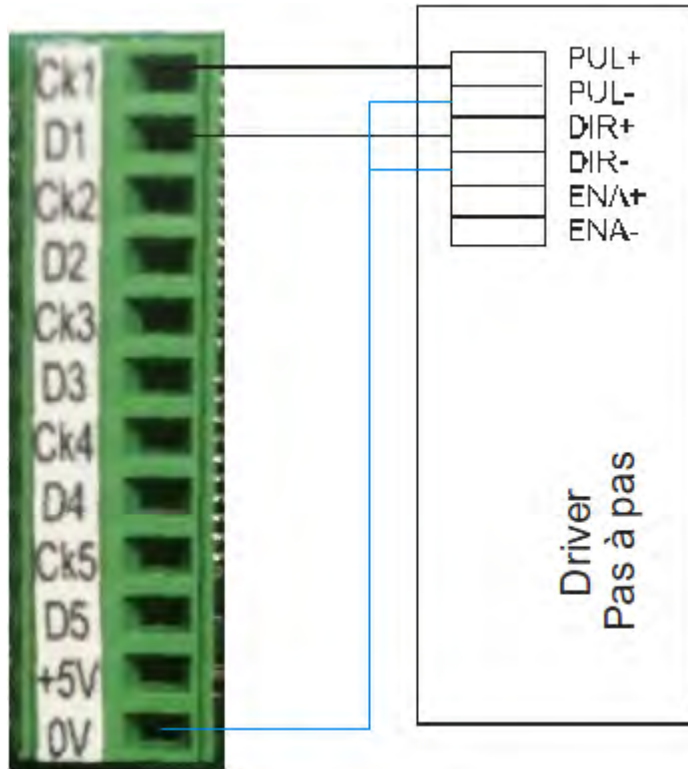
## Raccordement

### Raccordement de la carte V2.2 :

**Alimentation, Arrêt d'urgence :**

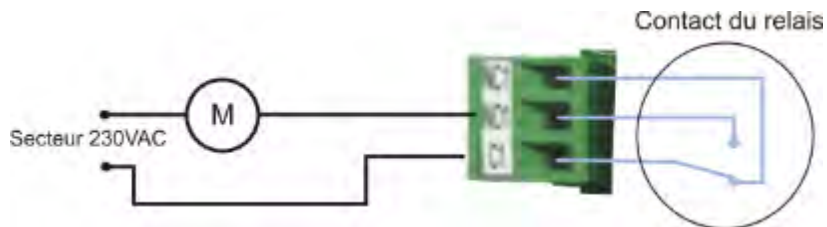
Le raccordement entre V+ et ENA est indispensable. Utilisez donc un contact de type NF (normalement fermé) pour assurer cette liaison. Par sécurité, cette entrée coupe physiquement l'alimentation des bobines des relais.

**Raccordement des entrées :****Commande d'un driver moteur :**

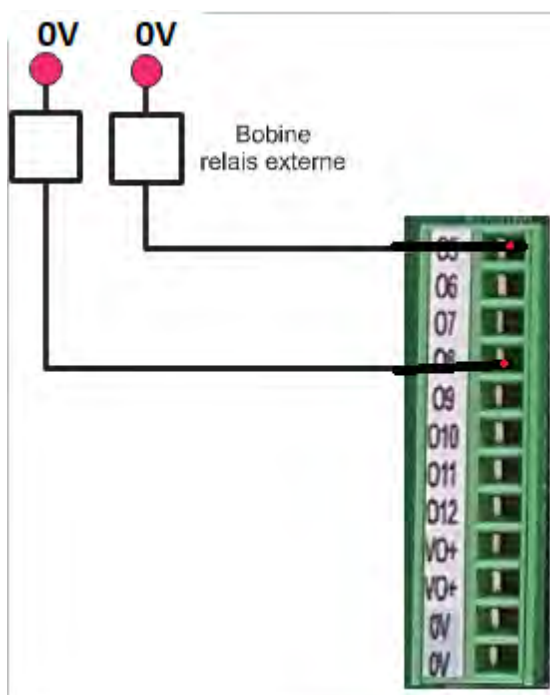


Raccordement du driver, moteur axe X

**Exemple d'utilisation d'une sortie relais pour la commande d'un moteur de broche:**



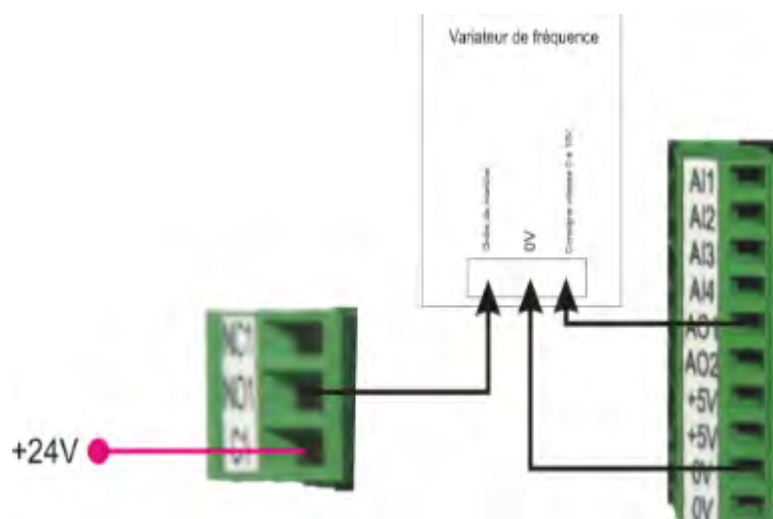
**Exemple utilisation des sorties OUT5 à OUT12 (PNP)**



Pilotage de 2 relais externes à l'aide des sorties OUT5 et OUT8

Les sorties étant de type PNP, il conviendra toujours d'alimenter VO+ (en +24V par exemple, le 0V étant commun avec celui de l'alimentation de la carte), pour obtenir cette même tension sur les sorties, à l'état 1.  
(NB: Les VO+ sont reliés en interne, il n'est donc pas utile d'alimenter les 2).

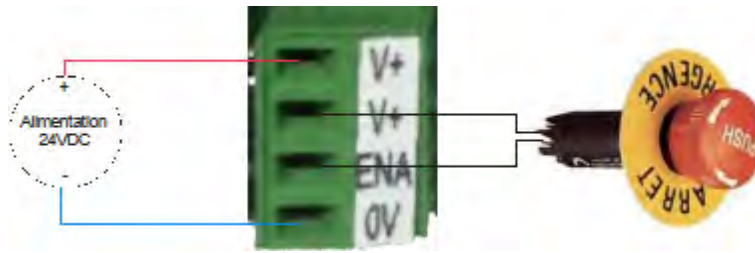
### Pilotage d'un variateur de vitesse de broche :



Sur cet exemple, la sortie OUT1 commande la mise en marche de la broche. La sortie analogique AO1 pilote la variation de vitesse.

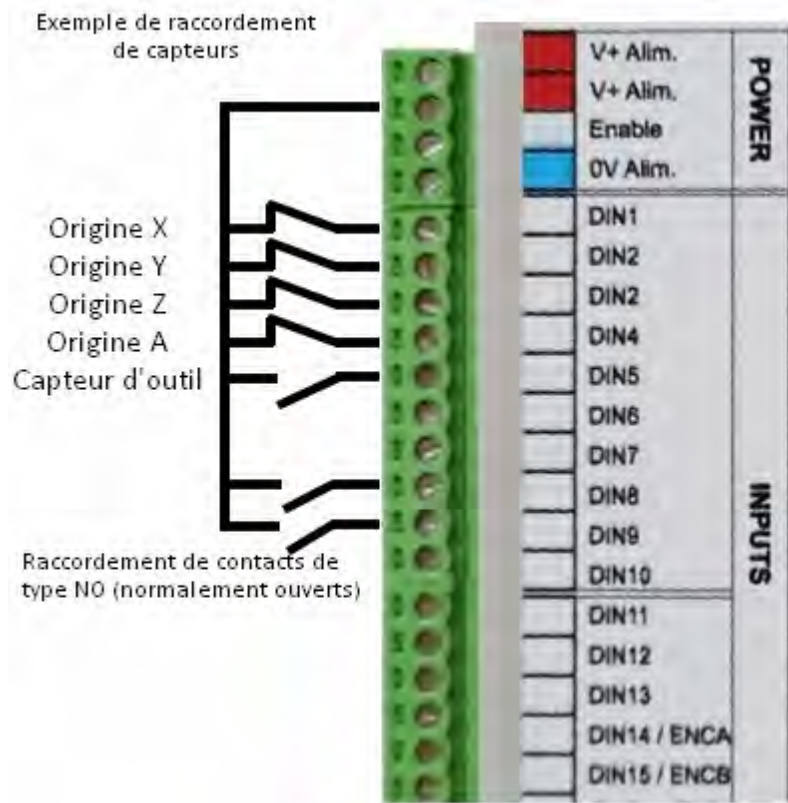
### Raccordement de la carte V2.4 :

#### Alimentation, Arrêt d'urgence :

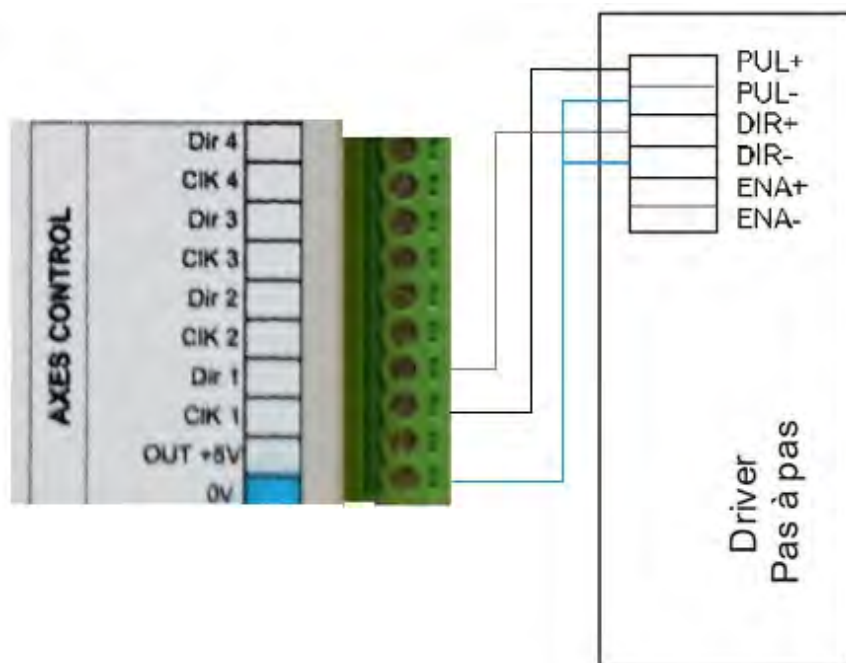


Le raccordement entre V+ et ENA est indispensable. Utilisez donc un contact de type NF (normalement fermé) pour assurer cette liaison. Par sécurité, cette entrée coupe physiquement l'alimentation des bobines des relais.

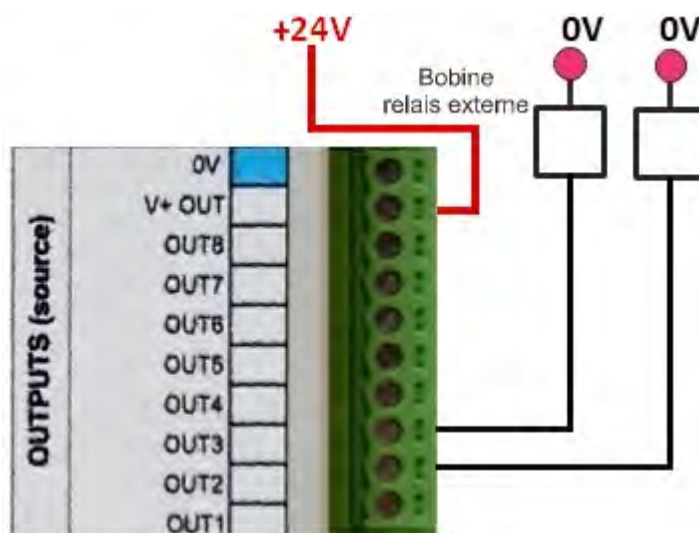
### Raccordement des entrées :



### Commande d'un driver moteur :



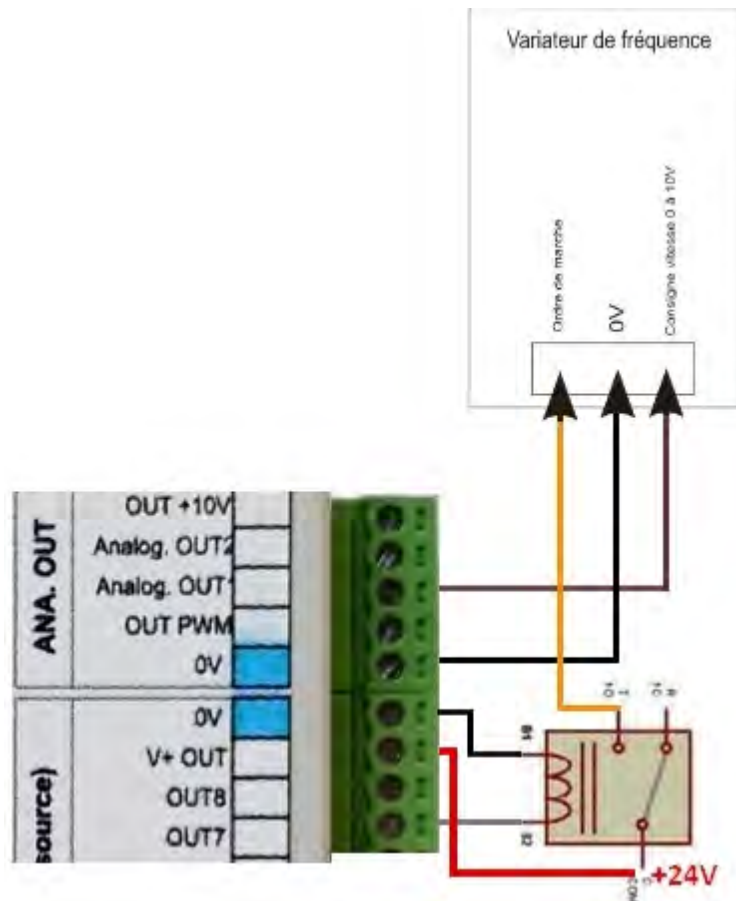
**Exemple d'utilisation des sorties OUT1 à OUT8 (PNP) :**



Pilotage de 2 relais externes à l'aide des sorties OUT2 et OUT3

NB: Les sorties étant de type PNP, il conviendra toujours d'alimenter V+OUT (en +24V par exemple, le 0V étant commun avec celui de l'alimentation de la carte), pour obtenir cette même tension sur les sorties, à l'état 1.

**Pilotage d'un variateur de vitesse de broche :**



Sur cet exemple, la sortie OUT7 commande la mise en marche de la broche.  
La sortie analogique AO1 pilote la variation de vitesse.

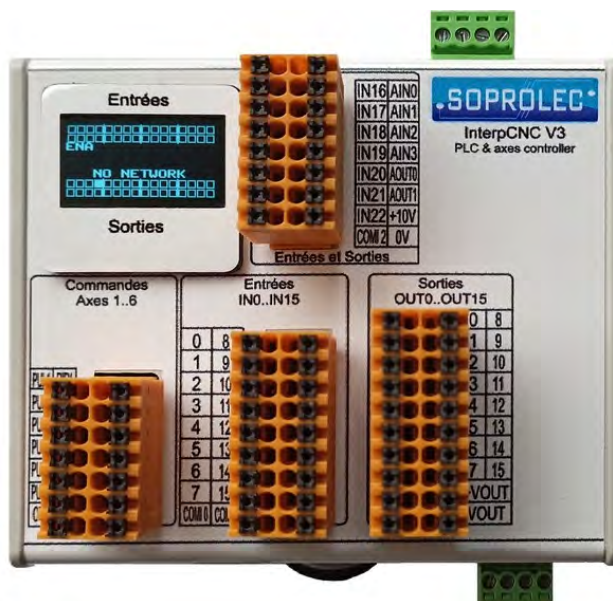
## Documentation\_MODBUS\_V3

**SOPROLEC**  
ZAC DE L'EPINE  
72460 SAVIGNE L'EVEQUE  
Tél : +33 (0)2 4376 4476



## Carte d'axes

# SOPROLEC InterpCNC V3



## Protocole de communication MODBUS

### Introduction

Cette documentation est destinée à la programmation avancée de la carte **InterpCNC V3**.

Elle porte à votre connaissance un certain nombre de registres et bits système, ainsi que leur type d'adressage.

Ainsi il devient possible en les lisant, en les testant, de réaliser par exemple vos propres fonctions permettant ce que les fonctions déjà existantes du PLC Basic n'auraient pas déjà prévu pour vous.

Les commandes Modbus décrites, permettent quant à elles de piloter la carte sans programme PLC Basic embarqué.

En effet, la carte **InterpCNC V3** est aussi capable d'exécuter les commandes Modbus en provenance par exemple d'un automate externe.

La carte reçoit dans son buffer les trames Modbus destinées à son ID, et les exécute.

Les liaisons RS485 COM1 et COM2 utilisent le protocole Modbus RTU. Ces 2 ports série doivent être configurés en mode Esclave ou maître pour la gestion d'extensions.

La liaison USB utilise également le protocole ModbusRTU avec une émulation port série (CDC).

La liaison Ethernet utilise également le protocole Modbus IP en TCP ou en UDP. En mode serveur, un maximum de 6 clients est autorisé.

## Identification des PLCs InterpCNC connectés au réseau Ethernet

L'automate répond à une requête de type broadcast UDP sur le port 58080. Ce port par défaut peut être changé via le paramètre 544.

La requête broadcast doit contenir la chaîne de 3 caractères suivants : « 0 1 »

La réponse des PLC sera une chaîne de caractère contenant l'information suivante :

Adresse IP, port TCP, Adresse MAC, nom Netbios,

Le format de la réponse est le suivant : "IP=%s :%d;MAC=%02X:%02X:%02X:%02X:%02X:%02X;NAME=%s"

Exemple de réponse :

«IP=192.168.10.147:502;MAC=2E:52:B9:1A:03:31;NAME=ICNC3-1A0331»

## Identification des PLCs InterpCNC connectés en USB

Les cartes reliées seront vues comme un port série standard. Vous pouvez identifier les InterpCNC grâce aux informations PID et VID qui sont les suivantes : ICNCVID = "0483" et ICNCPID = "5740"

L'exemple de code suivant en C# permet de lister les port série (code partiel):

```
// Search all CDC COM adapter in registry
private void GetCOMPortList(List<ConnexionInfo> USBCOMlist)
{
    try
    {
        ManagementObjectSearcher searcher =
            new ManagementObjectSearcher("root\\CIMV2",
                "SELECT * FROM Win32_PnPEntity WHERE Name LIKE '%(COM[0-9]%'");
        // "SELECT * FROM Win32_PnPEntity");

        try
        {
            foreach (ManagementObject queryObj in searcher.Get())
            {
                if (queryObj["Caption"].ToString().Contains("(COM"))
                {
                    //List<string> DevInfo = new List<string>();

                    string Caption = queryObj["Caption"].ToString();
                    int CaptionIndex = Caption.IndexOf("(COM");
                    string CaptionInfo = Caption.Substring(CaptionIndex +
1).TrimEnd(')');
                    string deviceId =
queryObj["deviceid"].ToString(); //"DeviceID"

                    int vidIndex = deviceId.IndexOf("VID_");
                    int pidIndex = deviceId.IndexOf("PID_");
                    string vid = string.Empty, pid = string.Empty;

                    if (vidIndex != -1 && pidIndex != -1)
```

```

    {
        string startingAtVid = deviceId.Substring(vidIndex + 4);
        characters long vid = startingAtVid.Substring(0, 4); // vid is four
        string startingAtPid = deviceId.Substring(pidIndex + 4);
        characters long pid = startingAtPid.Substring(0, 4); // pid is four
    }

    ConnexionInfo DevInfo = new ConnexionInfo();
    if ((vid == "0483") && (pid == "5740"))
        DevInfo.TypeConnexion = ePortType.TYPE_SERIAL_INTERPCNC;
    else
        DevInfo.TypeConnexion = ePortType.TYPE_SERIAL;

    DevInfo.COMPort = CaptionInfo; // Add(CaptionInfo);
    DevInfo.VID = vid;
    DevInfo.PID = pid;

    USBCOMlist.Add(DevInfo);
    mHandlerFormFunctionForAddItem?.Invoke(DevInfo);
}
}
}
catch (NullReferenceException ex)
{
}
}
catch (ManagementException ex)
{
}
}
}

```

## Lecture / écriture des paramètres

L'accès aux paramètres sauvegardés de la carte **InterpCNC V3** peut se faire à l'aide de l'utilitaire ICNCStudio, mais également en Modbus:

Le tableau des paramètres est accessible en Lecture/Écriture (Holding registers) à partir de l'adresse 32000.

Tous les paramètres sont au format 32 bits (2 registres par paramètre).

## Adresses des Bits en lecture seule (Input Bits)

### - Variables de type 1X -

Les registres représentent l'état des entrées TOR de la carte InterpCNC ainsi que les bits de status.

Toutes ces informations sont également disponibles sous forme de mots 16 bits dans les registres en lecture seule.

Dans le programme automate, ces variables sont accessibles avec les commandes :

- IN(0..255), DFM(0..255), DFD(0..255) pour les entrées
- STSBit(256..399) pour les bits de status

0 à 255	Entrées TOR Également accessible en Input register 1000 à 1015
256 à 263	Bits de mouvements en cours, 1 bit par axe, Également accessible en Input register 1016 / 8 bits de poids faible
264 à 271	Direction des mouvement (bit à 0 si mouvement négatifs, à 1 si positif) Également accessible en Input register 1016 / 8 bits de poids fort
272 à 279	Séquence de homing en cours Également accessible en Input register 1017 / 8 bits de poids faible
280 à 287	Erreur durant la séquence de homing Également accessible en Input register 1017 / 8 bits de poids fort
288 à 295	Fin de course sens négatif actifs Également accessible en Input register 1018 / 8 bits de poids faible
296 à 303	Fin de course sens positif actifs Également accessible en Input register 1018 / 8 bits de poids fort
304 à 311	Réserve Également accessible en Input register 1019 / 8 bits de poids faible
312 à 319	Réserve Également accessible en Input register 1019 / 8 bits de poids fort
320 à 327	COM Status Également accessible en Input register 1020 / 8 bits de poids faible
	320 Un ou plusieurs Registres sauvegardés (EEDATA) modifiés en dehors du programme PLC. Effacé lors de l'appel à IsEEdataChanged
	321 recipe Changed. Mis à 1 quand changement dans les recettes par voie de communication ou manipulation par les fonctions de tester avec commande PLC IsRCPChanged. Remis à 0 lors de l'appel à IsRCPChanged
	322 Paramètre InterpCNC modifié en dehors du PLC. Effacé lors de l'appel à IsPrmChanged
	323 Entrée USB en communication avec le protocole Grbl et plus Modbus RTU. Commutation automatique sur le protocole à la réception d'une trame de longueur <=4 caractères et commençant pas '\$' ou Ctrl-X (chr(0x18))
	324 : DMX Slave Trame reçue (passe automatiquement à 0 après lecture du bit par stsBit(324)
	325 : Le master DMX envoie des trames à l'automate
327 : Une trame Linky a été reçue. Passe à 0 après lecture de la commande PLC stsBit(STS LINKY RECEIVED)	
328 à 335	Status divers 1, Également accessible en Input register 1020 / 8 bits de poids fort
	328 État entrée Enable
	329 Carte verrouillée (identique à 328)
	333 Override vitesse usinage CNC <> 100 %
336 à 343	338 Carte SD détectée 342 Carte SD détectée et correctement formatée Également accessible en Input register 1021 / 8 bits de poids faible
344 à 351	Status PLC Basic Également accessible en Input register 1021 / 8 bits de poids fort

	346 PLCBasic running
352 à 359	Network Status Également accessible en Input register 1022 / 8 bits de poids faible
	352 Câble Ethernet connecté
	354 Adresse IP allouée
	355 Connexion IOT effective
	356 Horloge RTC initialisée par serveur SNTP
	357 Connexion au serveur SNTP ntp.pool.org active
	358 Configuration connexion SMTP terminée
360 à 367	Probe en cours axe 1 à 8 Également accessible en Input register 1022 / 8 bits de poids fort
368 à 375	Probe Error Axes 1 à 8 Également accessible en Input register 1023 / 8 bits de poids faible
376 à 383	Reserve 3
384 à 391	Reserve 4
392 à 399	Reserve 5

## Adresses des Registres en Lecture seule (Input registers)

### - Variables type 3X -

Ces registres représentent les variables internes de l'InterpCNC.

NB : Ils peuvent être lus en PLC Basic avec les mêmes commandes que pour les Holding registers, à condition d'ajouter 100000 à l'adresse du registre :

**GetMW(1xxxxx)**, **GetMDW(1xxxxx)**, **GetMI(1xxxxx)**, **GetMDI(1xxxxx)**, **GetMF(1xxxxx)**.

Exemples :

- GetMW(101000) pour lire l'état des entrées IN0 à IN15
- GetMDW(101330) pour lire le résultat d'un palpage sur l'axe 1

1000 à 1015	Mappage des entrées IN0 à IN255
1016 à 1025	Mappage des bits de status
	1016 Poids faible : Bits axes en cours de mouvement
	1016 Poids fort : Bit sens déplacement en cours (0 si négatif, 1 si positif)
	1017 Poids faible : Bits de homing en cours
	1017 Poids fort : Bits Erreurs Homing
	1018 Poids faible : Bits de fin de course négatif actifs
	1018 Poids fort : Bits de fin de course positif actifs.
	1019 : Réserve
	1020 : Status divers (voir détails sur bits 320 à 335)
	1021 : Status divers (voir détails sur bits 336 à 351)
1022 Poids faible : Status réseau Ethernet (voir détails sur bits 352 à 359)	
1022 Poids fort : Indicateurs d'axe en cours de séquence Pro	
1023 Poids faible : Indicateurs d'erreur séquence de palpage (p	

'1026	Nombre de caractères dans le buffer de Print PLC Basic (comm
'1027	Nombre de caractères dans le buffer de Trace PLC Basic (com
1030 à 1041	Compteurs de position des axes AXE1 à AXE 6 en pulses (reg signés également disponibles en Holding register 2400 à 2411)
1042 à 1053	Vitesses de déplacement actuelles des axes (32 bits signés)
1060 à 1071	Cible position actuelle (Dernière position cible demandée)
1072 à 1083	Cible vitesse actuelle (Dernière vitesse cible demandée)
1090 à 1097	Entrée analogique AIN0 à AIN7
1100	CPU Load
1101	CPU Load for PLC
1102 à 1107	CPU ID Processor (96 bits)
1108 à 1111	NOR FLASH UID (64 bits)
1112 à 1114	Ethernet MAC address
1115	Firmware version High
1116	Firmware Version Low
1117	Bootloader Version High
1118	Bootloader version Low
1120	NOR FLASH Page size
1121	NOR FLASH Sector Size size
1122	NOR FLASH Bloc Size
1123	NOR FLASH Bloc Count
1124	NOR FLASH Total size Kbyte
1130	Période en $\mu$ s entre deux événements sur entrée rapide IN16 LOW L'entrée doit être configurée en mode Interruption ou compteur. En mode Interruption, la mesure correspond à la période entre l fronts (montant ou descendant sans distinction). En mode compteur, ce sont uniquement les front montants qui s compte. Pour prendre en compte les front descendants, il faut in de l'entrée (voir paramètre 200).
1131	Période événements sur entrée rapide IN16 HIGH
1132	Période événements sur entrée rapide IN17 LOW
1133	Période événements sur entrée rapide IN17 HIGH
1134	Période événements sur entrée rapide IN18 LOW
1135	Période événements sur entrée rapide IN18 HIGH
1136	Période événements sur entrée rapide IN19 LOW
1137	Période événements sur entrée rapide IN19 HIGH
1138	Période événements sur entrée rapide IN20 LOW
1139	Période événements sur entrée rapide IN20 HIGH
1140	Période événements sur entrée rapide IN21 LOW
1141	Période événements sur entrée rapide IN21 HIGH
1142	Période événements sur entrée rapide IN22 LOW
1143	Période événements sur entrée rapide IN22 HIGH
1200	Nombre de connexions sur clients TCP (ICNC → client)
1201	Erreur transmission TCP Client LOW
1202	Erreur transmission TCP Client HIGH
1203	Compteur trame TCP Client LOW
1204	Compteur trame TCP Client HIGH

1220	Nombre de clients TCP connectés (Client → ICNC) (10 maxi)
1221	Erreur transmission TCP serveur LOW
1222	Erreur transmission TCP serveur HIGH
1223	Compteur trame TCP Serveur LOW
1224	Compteur trame TCP Client HIGH
1230	Compteur requête Modbus RTU master mapping 1 LOW
1231	Compteur requête Modbus RTU master mapping 1 HIGH
1232	Compteur requête Modbus RTU master mapping 2 LOW
1233	Compteur requête Modbus RTU master mapping 2 HIGH
1234	Compteur requête Modbus RTU master mapping 3 LOW
1235	Compteur requête Modbus RTU master mapping 3 HIGH
1236	Compteur requête Modbus RTU master mapping 4 LOW
1237	Compteur requête Modbus RTU master mapping 4 HIGH
1238	Compteur requête Modbus RTU master mapping 5 LOW
1239	Compteur requête Modbus RTU master mapping 5 HIGH
1240	Compteur requête Modbus RTU master mapping 6 LOW
1241	Compteur requête Modbus RTU master mapping 6 HIGH
1242	Compteur requête Modbus RTU master mapping 7 LOW
1243	Compteur requête Modbus RTU master mapping 7 HIGH
1244	Compteur requête Modbus RTU master mapping 8 LOW
1245	Compteur requête Modbus RTU master mapping 8 HIGH
1246	Compteur requête Modbus RTU master mapping 9 LOW
1247	Compteur requête Modbus RTU master mapping 9 HIGH
1248	Compteur requête Modbus RTU master mapping 10 LOW
1249	Compteur requête Modbus RTU master mapping 10 HIGH
1250	Compteur requête Modbus RTU master mapping 11 LOW
1251	Compteur requête Modbus RTU master mapping 11 HIGH
1252	Compteur requête Modbus RTU master mapping 12 LOW
1253	Compteur requête Modbus RTU master mapping 12 HIGH
1254	Compteur requête Modbus RTU master mapping 13 LOW
1255	Compteur requête Modbus RTU master mapping 13 HIGH
1256	Compteur requête Modbus RTU master mapping 14 LOW
1257	Compteur requête Modbus RTU master mapping 14 HIGH
1258	Compteur requête Modbus RTU master mapping 15 LOW
1259	Compteur requête Modbus RTU master mapping 15 HIGH
1260	Compteur requête Modbus RTU master mapping 16 LOW
1261	Compteur requête Modbus RTU master mapping 16 HIGH
1262	Compteur success Modbus RTU master mapping 1 LOW
1263	Compteur success Modbus RTU master mapping 1 HIGH
1264	Compteur success Modbus RTU master mapping 2 LOW
1265	Compteur success Modbus RTU master mapping 2 HIGH
1266	Compteur success Modbus RTU master mapping 3 LOW
1267	Compteur success Modbus RTU master mapping 3 HIGH
1268	Compteur success Modbus RTU master mapping 4 LOW
1269	Compteur success Modbus RTU master mapping 4 HIGH
1270	Compteur success Modbus RTU master mapping 5 LOW
1271	Compteur success Modbus RTU master mapping 5 HIGH
1272	Compteur success Modbus RTU master mapping 6 LOW
1273	Compteur success Modbus RTU master mapping 6 HIGH
1274	Compteur success Modbus RTU master mapping 7 LOW
1275	Compteur success Modbus RTU master mapping 7 HIGH

1276	Compteur success Modbus RTU master mapping 8 LOW
1277	Compteur success Modbus RTU master mapping 8 HIGH
1278	Compteur success Modbus RTU master mapping 9 LOW
1279	Compteur success Modbus RTU master mapping 9 HIGH
1280	Compteur success Modbus RTU master mapping 10 LOW
1281	Compteur success Modbus RTU master mapping 10 HIGH
1282	Compteur success Modbus RTU master mapping 11 LOW
1283	Compteur success Modbus RTU master mapping 11 HIGH
1284	Compteur success Modbus RTU master mapping 12 LOW
1285	Compteur success Modbus RTU master mapping 12 HIGH
1286	Compteur success Modbus RTU master mapping 13 LOW
1287	Compteur success Modbus RTU master mapping 13 HIGH
1288	Compteur success Modbus RTU master mapping 14 LOW
1289	Compteur success Modbus RTU master mapping 14 HIGH
1290	Compteur success Modbus RTU master mapping 15 LOW
1291	Compteur success Modbus RTU master mapping 15 HIGH
1292	Compteur success Modbus RTU master mapping 16 LOW
1293	Compteur success Modbus RTU master mapping 16 HIGH
1294	Compteur erreur Modbus RTU master mapping 1 LOW
1295	Compteur erreur Modbus RTU master mapping 1 HIGH
1296	Compteur erreur Modbus RTU master mapping 2 LOW
1297	Compteur erreur Modbus RTU master mapping 2 HIGH
1298	Compteur erreur Modbus RTU master mapping 3 LOW
1299	Compteur erreur Modbus RTU master mapping 3 HIGH
1300	Compteur erreur Modbus RTU master mapping 4 LOW
1301	Compteur erreur Modbus RTU master mapping 4 HIGH
1302	Compteur erreur Modbus RTU master mapping 5 LOW
1303	Compteur erreur Modbus RTU master mapping 5 HIGH
1304	Compteur erreur Modbus RTU master mapping 6 LOW
1305	Compteur erreur Modbus RTU master mapping 6 HIGH
1306	Compteur erreur Modbus RTU master mapping 7 LOW
1307	Compteur erreur Modbus RTU master mapping 7 HIGH
1308	Compteur erreur Modbus RTU master mapping 8 LOW
1309	Compteur erreur Modbus RTU master mapping 8 HIGH
1310	Compteur erreur Modbus RTU master mapping 9 LOW
1311	Compteur erreur Modbus RTU master mapping 9 HIGH
1312	Compteur erreur Modbus RTU master mapping 10 LOW
1313	Compteur erreur Modbus RTU master mapping 10 HIGH
1314	Compteur erreur Modbus RTU master mapping 11 LOW
1315	Compteur erreur Modbus RTU master mapping 11 HIGH
1316	Compteur erreur Modbus RTU master mapping 12 LOW
1317	Compteur erreur Modbus RTU master mapping 12 HIGH
1318	Compteur erreur Modbus RTU master mapping 13 LOW
1319	Compteur erreur Modbus RTU master mapping 13 HIGH
1320	Compteur erreur Modbus RTU master mapping 14 LOW
1321	Compteur erreur Modbus RTU master mapping 14 HIGH
1322	Compteur erreur Modbus RTU master mapping 15 LOW
1323	Compteur erreur Modbus RTU master mapping 15 HIGH
1324	Compteur erreur Modbus RTU master mapping 16 LOW
1325	Compteur erreur Modbus RTU master mapping 16 HIGH
1350	ProbePosition Axe 1 LOW

1351	ProbePosition Axe 1 HIGH
1352	ProbePosition Axe 2 LOW
1353	ProbePosition Axe 2 HIGH
1354	ProbePosition Axe 3 LOW
1355	ProbePosition Axe 3 HIGH
1356	ProbePosition Axe 4 LOW
1357	ProbePosition Axe 4 HIGH
1358	ProbePosition Axe 5 LOW
1359	ProbePosition Axe 5 HIGH
1360	ProbePosition Axe 6 LOW
1361	ProbePosition Axe 6 HIGH
1998	Compteur frame DMX reçues LOW
1999	Compteur frame DMX reçues HIGH
2000	Taille de la frame DMX reçue
2001 à 2512	Canaux DMX. Valeur comprise entre 0 et 255
2990	Taille du buffer CNC Modbus
2991	Taille du buffer de commande GRBL
2992	Taille du buffer de planification CNC
3000	Place disponible dans le buffer de communication CNC Modbus
3001	Place disponible dans le buffer de communication CNC Grbl (10)
3002	Place disponible dans le buffer de planificateur CNC (35 maxi)
3003	Status CNC Bit 0 : Alarme Bit 1 : Gcode en mode Test Bit 2 : Homing CNC en cours Bit 3 : Cycle en cours Bit 4 : Pause en cours (Feed hold) Bit 5 : Jog en cours Bit 6 : Sécurité porte active Bit 7 : Mode sleep actif Bit 8 Arrêt d'urgence actif Bit 9 : Changement d'outils manuel en cours
3004	Code alarme CNC : 0 : pas d'alarme 1 : Limite par fin de course 2 : Limite course par logiciel 3 : Cycle abandonné 4 : État initial contact palpeur erroné, 5 : Erreur détection contact palpeur 6 : Erreur liée à initialisation durant homing 7 : Erreur liée à l'ouverture de porte durant homing 8 : Contact de homing qui reste engagé malgré le dégagement 9 : Contact de homing non trouvé (course machine) 10 : Arrêt d'urgence enclenché 11 : Séquence de homing nécessaire 12 : Détection anormale du contact de palpeur 13 : Déclenchement du palpeur outils en cours de cycle ou en Jog 14 : Erreur timeout signal broche prête 15 : Tolérance écart max entre axes maître/esclave dépassée 16 : Erreur interne 17 : Defaut détectée sur moteur (entrée Erreur Moteur)

	18 : Timeout détection événement (commande modbus 1012)
3010..3011	Vitesse de déplacement CNC actuelle (UINT32, mm/mn)
3012	<p>THC Status :</p> <ul style="list-style-type: none"> <li>• b2 : THC autorisé. Activé par les commande THCO<sub>n</sub> ou THCO<sub>n</sub> Reset par commande THCStop.</li> <li>• b6 : THC Actif. Activé par THCO<sub>n</sub>, THCO<sub>n</sub>Auto ou THC Resum<sub>n</sub> par THCStop ou THCPause. Indique que le THC est actif.</li> <li>• b9 : THC verrouillé pour cause de sous vitesse par rapport à la de découpe demandée</li> <li>• b11 : Temporisation avant activation en cours (paramètre Setting_THC_Delay)</li> <li>• b12 : Échantillonnage pour réglage auto tension THC en cours échantillonnage demandé, ce bit est à 1 aussi durant le délai ini</li> <li>• b13 : Le résultat d'échantillonnage est en dehors de la toléran valeur utilisée est bornée à la plage autorisée. Le bit repasse à stop THC</li> </ul>
3013,,3014	(int32, mV) Mesure tension d'arc en mV. Mise à l'échelle automa fonction des paramètres d'offset et de tension max de mesure. I permanente même si le THC est inactif.
11000 à 11114	Buffer des commandes de Print (Utilisé par ICNCStudio)
11125 à 11189	Buffer des commandes Trace (64 octets, Utilisé par ICNCStudio)
12000 à 12999	Buffer pour lecture mémoire MCU (Utilisé par ICNCStudio)
12000 à 12200	Buffer pour lecture mémoire FLASH NOR (Utilisé par ICNCStudio)

## Registres en Lecture/Écriture (Holding registers)

### - Variables de type 4X -

Ces registres permettent d'agir sur la carte InterpCNC et de lancer les commande de gestion d'axes.

2000 à 2149	Buffer pour commande MODBUS
2150 à 2157	Sorties analogiques AOUT0 à AOUT7
2160 à 2165	Mappage des sorties OUT0 à OUT95
2166 à 2191	Mappage des bits utilisateurs
2360 à 2367	Compteur mode encodeur pour 4 canaux QEI
2380 à 2395	Compteurs rapides entrées TOR IN17 à IN24 (registres 32 bits L'entrée doit être configurée en mode Interruption ou compteur. En mode interruption, tous les changements d'état seront comptés. En mode compteur, les fronts montants seront comptabilisés. Pour les fronts descendants, il faut donc inverser la polarité de l'entrée (paramètres 200).
2400 à 2411	Compteurs de position des axes AXE1 à AXE 6 (registres 32 bits)
2412 à 2423	Vitesse actuelle (en pulses/s) des axes en cours de déplacement exploitable pour les commandes CNC)
2430	Override vitesse normale en mode CNC
2431	Override vitesse rapide en mode CNC
2444,,2445	(uint32_t, mV) consigne tension d'arc pour THC. Initialisée par le pilotage externe ou par échantillonnage de la mesure.
2446	

	(int16_t, mV) Offset appliqué à la consigne tension d'arc en temps d'ajustement manuel
2447	(uint16_t, %) : Override de vitesse sur déplacement THC (0% à 100%). Ce registre agit au même titre que le paramètre de gain en vitesse pour les paramètres THC. Il est initialisé à 100 à la mise sous tension. Une modification se traduira par un arrêt du THC. Il peut être modifié à tout moment avec prise en compte immédiate. Une modification peut également se faire à l'aide d'une action d'écriture différée/bufferisée.
2800 à 2899	Indexes pour lectures/écritures indexées. Soit 50 registres 32 bits pour l'indexation des différents types de données Modbus. 0 à 65535 pour des registres en lecture seule (Input registers) 100000 à 165535 pour les registres en lecture/écriture (Holding registers) 200000 à 265535 pour les bits en lecture seule (Input bits) 300000 à 365535 pour les bits en lecture/écriture (Coils)
3000 à 3999	Registres utilisateur en RAM (1000 registres 16 bits)
4000 à 4999	Registres utilisateurs en RAM sauvegardée (1000 registres 16 bits)
5000	Canal DMX Master pour transmission DMX Master
5001 à 5512	Valeurs des 512 canaux DMX pour la transmission DMX master
9995	Taille des recettes
9996	Index 0 recette
9997	Index 1 recette
9998	Index 2 recette
9999	Index 3 recette
10000 à 10999	Recette page 0
11000 à 11999	Recette page 1
12000 à 12999	Recette page 2
13000 à 13999	Recette page 3
32000 à 33999	Paramètres InterpCNC (index 0 à 1999, 2 registres par paramètre)
62000	MB_HOLD_ADDR_MCU_MEMORY_ReadPtr_low (Pointeur au bas de lecture contenu mémoire CPU)
62001	MB_HOLD_ADDR_MCU_MEMORY_ReadPtr_high
62002	MB_HOLD_ADDR_FLASH_NAND_ReadPtr_low
62003	MB_HOLD_ADDR_FLASH_NAND_ReadPtr_high

## Généralités sur l'envoi des commandes Modbus

L'envoi de commandes sur l'InterpCNC se fait à travers le buffer de commande situé entre les adresses 2000 et 2149 (variables de type Holding registers). Chaque commande est identifiée par un numéro de commande détaillé ci-après. Vous pouvez utiliser les fonctions Modbus 06 (write single register) ou 16 (write multiple registers) pour écrire dans ce buffer.

Si vous êtes limité à l'utilisation de la fonction Modbus 06, l'évènement qui lancera le traitement de la commande est l'écriture du code de commande situé à l'adresse 2000. Il convient donc au préalable d'avoir transféré les arguments de la commande.

Si vous utilisez la fonction Modbus 16, vous pouvez envoyer l'ensemble des arguments avec le code de commande sur une seule requête.

Si plusieurs maîtres ou client sont connectés à l'InterpCNC, nous vous recommandons vivement l'utilisation de la fonction Modbus 16 pour éviter les conflits d'accès au buffer de commande.

### Commande 100 : Arrêt d'un axe

Cette commande permet l' Arrêt d'un seul axe, à l'aide de son identifiant.  
Elle est équivalente à la commande de l'interpréteur Basic : **StopAxeID**

<b>Adresse</b>	2000	2001
<b>Paramètre</b>	Commande ID	Axe ID
<b>Valeur</b>	100	1,,6

### Commande 101 : Arrêt d'un ou plusieurs axes

Cette commande permet l' Arrêt d'un ou plusieurs axes, identifiés par leur bit respectif sur un mot de 16 bits.

Elle est équivalente à la commande de l'interpréteur Basic : **StopAxes**

La décélération utilisée est celle indiquée lors du lancement de la commande de déplacement.

Vous pouvez contrôler l'arrêt effectif des axes via les bits de statut 256 à 261 « Axe en mouvement »

<b>Adresse</b>	2000	2001
<b>Paramètre</b>	Commande ID	Axes bits
<b>Valeur</b>	101	0x01 à 0x3F

### Commande 102 : Déplacement d'un axe à une vitesse donnée

Cette commande permet le déplacement d'un axe jusqu'à ce qu'il atteigne une vitesse donnée.

Elle est équivalente à la commande de l'interpréteur Basic : **MoveSpeed**

Le mouvement peut être stoppé par une commande d'arrêt (commande 100 ou 101) ou en indiquant une vitesse de déplacement nulle.

L'arrêt effectif peut alors être contrôlé par les bits de statut « Axe en mouvement ».

Après le lancement d'une commande de déplacement en vitesse, vous gardez la possibilité de lancer une commande de déplacement en position (commande 103 ou 104).

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Axe ID	Accel/Decel	
<b>Valeur</b>	102	1,,6	LW Accel	HW Accel

## Commande 103 : Déplacement d'un axe vers une position cible

Le profil de vitesse est donné par l'accélération, la vitesse et la décélération. Cette commande permet le déplacement d'un axe jusqu'à une position cible. Elle est équivalente à la commande de l'interpréteur Basic : **MoveAxe**  
Chaque axe dispose de son propre générateur de profil. Il est donc possible de lancer différents mouvements simultanément sur plusieurs axes.  
Dès le lancement d'une commande de déplacement, le bit de statut « Axe en mouvement » associé à l'axe passe à 1. Il repasse à 0 lorsque la cible est atteinte.

Il est également possible de faire un changement de Cible/Vitesse à la volée. C'est-à-dire, durant un mouvement en cours. Si la nouvelle cible nécessite un retour en arrière, il sera exécuté automatiquement. Le bit de statut « axe en mouvement » ne passe pas à 0 durant l'inversion de sens de déplacement de l'axe.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Param	Commande ID	Axe ID	Accélération		Vitesse		Décélération		Position cible	
Value	103	1,,6	LW Accel	HW Accel	LW Fréquence	HW Fréquence	LW Fréquence	HW Fréquence	LW Target	HW Target

## Commande 104 : Déplacement d'un axe du nombre de pas indiqué par rapport à la position actuelle

Le profil de vitesse est donné par l'accélération, la vitesse et la décélération. Cette commande permet le déplacement d'un axe sur un nombre de pas défini. Elle est équivalente à la commande de l'interpréteur Basic : **MoveAxeRelatif**  
Si un axe est en mouvement en mode vitesse (commande 102), vous pouvez lancer une commande de déplacement relatif à la position actuelle. Cela permet par exemple de déclencher la rotation continue d'un moteur (fonction 102) jusqu'à la détection d'une cellule. A l'arrivée de cette information, de déplacer l'axe d'une distance donnée.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Param	Comman de ID	Axe ID	Accélération		Vitesse		Décélération		Nombre de pas	
Value	104	1,,6	LW Accel	HW Accel	LW Fréquence	HW Fréquence	LW Fréquence	HW Fréquence	LW Target	HW Target

## Commande 105 : Écriture du compteur de position actuelle (revient à écrire dans les registres de positions)

Cette commande est équivalente à la commande de l'interpréteur Basic : **SetPos(axe ID)**  
**Il est impératif de ne jamais changer les registres de position durant un mouvement.**

Adresse	2000	2001	2002	2003
---------	------	------	------	------

Paramètre	Commande ID	Axe ID	Position	
Valeur	105	1,,6	LW Position	HW Position

## Commande 106 : Lancement du Homing d'un axe

La commande de homing permet d'initialiser la position d'un axe après la mise sous tension à l'aide d'un fin de course.

La procédure de homing se déroule en 3 temps :

- Déplacement rapide jusqu'à la détection d'un fin de course
- Retour arrière lent jusqu'à la perte du signal fin de course, Le registre de position prend alors la valeur de l'argument « Home position to set »
- Mouvement complémentaire de dégagement relatif à la position de perte du signal Homing (suivant argument Clearance).

Si l'entrée utilisée pour la séquence de homing est déjà affectée à la fonction de gestion de fin de course, elle est provisoirement désactivée. Vous pouvez donc utiliser un capteur commun pour la fonction de capteur de prise d'origine ou de fin de course.

Cette commande est équivalente à la commande **Home** de l'interpréteur Basic :

2002 : **Homing Mode** est toujours à 0

2003 : **Input Number** est le numéro de l'entrée recevant le capteur.

2004 : **Expected Input state** : état de l'entrée déclenchant la fin de la procédure.

2014 et 2015 : **Max stroke** = Course maxi (en pas)

2016 : **Tempo Reverse Direction**, temps de pause (en ms), avant de revenir vers le capteur

2017 et 2018 : **Home Position to set**, valeur à laquelle est initialisé le compteur de position avant

le dégagement (le plus souvent mis à 0)

2019 et 2020 : **Clearance** = Dégagement relatif à la position d'origine (en pas)

Vous pouvez lancer des séquences de homing simultanément sur plusieurs axes. Au lancement de la procédure de homing, le bit de statut « Homing en cours » passe à 1.

Il repasse automatiquement à 0 lorsque le homing est terminé (avec ou sans erreur).

Si le homing ne s'est pas déroulé normalement ou a été interrompu par une commande de Stop axe, le bit de statut « Homing Error » sera positionné à 1. Les erreurs peuvent être liées à des paramètres erronés dans la commande ou au fait que l'entrée n'a pas été activée dans la course maxi de déplacement autorisée dans la commande.

Le bit d'erreur repasse automatiquement à 0 lors du lancement d'une nouvelle commande de homing.

Dans un programme utilisant la fonction de homing, il est donc nécessaire de tester le bit de statut « Homing en cours » (bits 272 à 277) puis, de vérifier son bon déroulement à l'aide du bit « Homing Error » (bits 280 à 285)

L'argument « Home position to set » permet d'indiquer la valeur que prend le registre de position au moment de la perte du capteur homing durant le mouvement lent inverse. Cet argument peut être positif ou négatif.

Prenons l'exemple où vous souhaitez une position de homing à 0 mais dégagée de 500 pas du capteur. Il vous faut alors indiquer une position de Homing à -500 et un

dégagement de 500.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Paramètre	Command ID	Axe ID	Homing Mode	Input Number	Expected Input State	Direction	High Speed		Low Speed	
Valeur	106	1,,6	0	0,,255	0 or 1	0 : Neg 1 :Positive	LW High speed	HW High speed	LW Low speed	LW Low speed

2010	2011	2012	2013	2014	2015	2016	2017	2018	
Accélération		Décélération		Max Stroke		Tempo Reverse Direction	Home Position to set		
LW Acceleration	HW Acceleration	LW Deceleration	HW Deceleration	LW Max Stroke	HW Max Stroke	Time in ms	LW Home Position	HW Home Position	LW Home Position

### Commande 107 : Lancement d'un Palpage sur une entrée

Cette commande est équivalente à la commande **Probe** de l'interpréteur Basic :

2002 : **Input Number** est le numéro de l'entrée recevant le capteur.

2003 : **Expected Input state** est l'état attendu sur l'entrée pour la détection.

2010 et 2011 : **Max stroke** correspond à la course Maxi (en pas). C'est une valeur signée dont le signe ± indique la direction.

Le déroulement de la séquence est matérialisé par les bits de statut « Probe in progress » (bits de statut 360 à 365) et « Probe Error ». (bits de statut 368 à 373) Lors du lancement de la commande, le bit « Probe in progress » passe à 1 et le bit « Probe Error » passe à 0.

A la fin de la séquence, le bit « Probe en cours » passe à 0. Si une erreur s'est produite durant la séquence ou que cette séquence a été arrêtée par une commande de stop axe, le bit « Probe error » sera activé. Il convient donc de tester le bit « Probe Error » après le passage à 0 du bit « Probe en cours ».

Le résultat de cette commande est la position de l'axe au moment de enclenchement de l'entrée indiquée. Ces positions sont disponibles dans les Inputs registers 1330 à 1341.

NB : Plusieurs séquences de palpation peuvent être lancées simultanément sur différents axes.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Param	Command ID	Axe ID	Input Number	Expected Input state	Accélération		Vitesse		Décélération	
Valeur	107	1,,6	0,,255	0 or 1	LW Acceleration	HW Acceleration	LW Velocity	HW Velocity	LW Deceleration	HW Deceleration

Fonction C associée :

```
int ICNC3_ProbeAxe(modbus_t *ctx,
    uint8_t AxeID,
    uint16_t InputNumber,
    uint8_t ExceptedInputState, /* 0 if DIN=0 when switch is pressed (NC contact type) */
    uint32_t Acceleration,
```

```
uint32_t Speed,
uint32_t Deceleration,
int32_t StrokeLimitStep)
```

## Commande 108 : Lancement d'un Palpage sur plusieurs entrées

Cette commande permet de déplacer un axe jusqu'à atteindre une condition de fin de mouvement sur les entrées DIN0 à DIN31

La séquence se termine par l'acquisition de la position de capture lorsque la condition suivante est remplie :

```
((ActualInputStates0_31 AND ANDMask) XOR XORMask) <> 0)
```

2012 et 2013 : **Max stroke** correspond à la course Maxi (en pas). C'est une valeur signée dont le signe ± indique la direction.

Le déroulement de la séquence est matérialisé par les bits de statut « Probe in progress » (bits de statut 360 à 365) et « Probe Error ». (bits de statut 368 à 373) Lors du lancement de la commande, le bit « Probe in progress » passe à 1 et le bit « Probe Error » passe à 0.

A la fin de la séquence, le bit « Probe en cours » passe à 0. Si une erreur s'est produite durant la séquence ou que cette séquence a été arrêtée par une commande de stop axe, le bit « Probe error » sera activé. Il convient donc de tester le bit « Probe Error » après le passage à 0 du bit « Probe en cours ».

Le résultat de cette commande est la position de l'axe au moment où le résultat du test logique est <> 0, Ces positions sont disponibles dans les Inputs registers 1330 à 1341.

NB : Plusieurs séquences de palpation peuvent être lancées simultanément sur différents axes.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
<b>Param</b>	Command ID	Axe ID	AND mask		XOR mask		Accélération		Vitesse		Décélération		Max stroke (signed value → Direction)	
<b>Valeur</b>	108	1,,6	LW AND mask	HW AND mask	LW XOR mask	HW XOR mask	LW Acceleration	HW Acceleration	LW Velocity	HW Velocity	LW Deceleration	HW Deceleration	LW Max Stroke	HW Max Stroke

Fonction C associée :

```
int ICNC3_ProbeAxewithMask(modbus_t *ctx,
uint8_t AxeID,
uint32_t ANDMask,
uint32_t XORMask,
uint32_t Acceleration,
uint32_t Speed,
uint32_t Deceleration,
int32_t StrokeLimitStep) ;
```

## Commande 109 : Lancement d'un Palpage sur seuil d'entrée analogique

Cette commande permet de déplacer un axe jusqu'à atteindre un seuil sur une entrée analogique

La séquence se termine par l'acquisition de la position de capture lorsque le seuil est atteint en appliquant le test suivant:

Si opérateur = 0 : palpation tant que l'entrée analogique est  $\leq$  au seuil indiqué

Si opérateur = 1 : palpation tant que l'entrée analogique est  $\geq$  au seuil indiqué

2012 et 2013 : **Max stroke** correspond à la course Maxi (en pas). C'est une valeur signée dont le signe  $\pm$  indique la direction.

Le déroulement de la séquence est matérialisé par les bits de statut « Probe in progress » (bits de statut 360 à 365) et « Probe Error ». (bits de statut 368 à 373) Lors du lancement de la commande, le bit « Probe in progress » passe à 1 et le bit « Probe Error » passe à 0.

A la fin de la séquence, le bit « Probe en cours » passe à 0.

Si une erreur s'est produite durant la séquence ou que cette séquence a été arrêtée par une commande de stop axe, le bit « Probe error » sera activé et le registre de position capturée ne sera pas actualisé. Il convient donc de tester le bit « Probe Error » après le passage à 0 du bit « Probe en cours ».

Le résultat de cette commande est la position de l'axe au moment où le seuil de déclenchement est atteint, Ces positions sont disponibles dans les Inputs registers 1330 à 1341.

NB : Plusieurs séquences de palpation peuvent être lancées simultanément sur différents axes.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	
Param	Command ID	Axe ID	AIN number	Operator	Threshold mV	Accélération		Vitesse		Décélération		Max st (signed v)	Directi
Valeur	109	1,6	0,7	0,1	0,10000	LW Acceleration	HW Acceleration	LW Velocity	HW Velocity	LW Deceleration	HW Deceleration	LW Max H	Stroke

Fonction C associée :

```
int ICNC3_ProbeAxeOnAnalogInput(modbus_t* ctx,
    uint8_t AxeID,
    uint32_t AnalogInputNumber,
    uint32_t Operator,
    int32_t Threshold_mV,
    uint32_t Acceleration,
    uint32_t Speed,
    uint32_t Deceleration,
    int32_t StrokeLimitStep);
```

Exemple :

```
#define PROBE_UNTIL_AIN_LOWER_THAN_THRESHOLD 0
#define PROBE_UNTIL_AIN_GREATER_THAN_THRESHOLD 1
#define STATUS_BIT_Z_PROBING_IN_PROGRESS 362 // Input bit #362
#define STATUS_BIT_Z_PROBING_ERROR 370 // Input bit #370
#define Z_PROBE_POSITION 354 // Input register #354, INT32

// Probe Z axis
// Analog input channel 0
// Move until AIN0 greater than 5000mV
// Accel = 10000Hz/s, Velocity = 1000Hz, Decel = 10000Hz/s
// Negative direction and maximum stroke of 2500 steps
//
```

```

// Return -1 in case of communication error
// Return 0 in case of probe error (ie, AIN0 level is greather than
// thresold when probe start or thresold can't be reach befor 2500 steps
// Return 1 with Probe position in case of success
//
int ProbeWithAnalogChanel(modbus_t* ctx, int* ProbePosition)
{
    int res;
    uint8_t inStatusBit;
    int ProbePositionResult;

    res = ICNC3_ProbeAxeOnAnalogInput(ctx, // Modbus handler
context
        (uint8_t)3, // Axe Z ID
        0, // Analog input #0
        PROBE_UNTIL_AIN_GREATHER_THAN_THRESHOLD, // Operator
        5000, // 5000mV threshold
        10000, // Acceleration (Hz/s)
        1000, // Velocity (Hz)
        10000, // Deceleration Hz/s
        -2500); // 2500 steps in negative direction as limited
stroke
    if (res <= 0)
        return -1; // Communication error

    // Wait for en of probe or communication Error
    do {
        Sleep(100);

        // Read Z probe in progress status bit
        res = modbus_read_input_bits(ctx,
STATUS_BIT_Z_PROBING_IN_PROGRESS, 1, &inStatusBit);

        } while ((inStatusBit == 1) || (res<=0));
    if (res <= 0)
        return -1; // Communication error

    // Check for probe error
    res = modbus_read_input_bits(ctx, STATUS_BIT_Z_PROBING_ERROR, 1,
&inStatusBit);
    if (res <= 0)
        return -1; // Communication error

    if (inStatusBit != 0)
        return 0; // Probe error

    // Read probe position result
    res = ICNC3_Get32bitsInputRegister(ctx, Z_PROBE_POSITION,
&ProbePositionResult);
    if (res <= 0)
        return -1; // Communication error

    *ProbePosition = ProbePositionResult;
    return 1; // Succes
}

```

## Commande 110 : Forçage des entrées

Cette commande permet de définir le forçage de l'état d'une entrée.

Elle est équivalente à la commande de l'interpréteur Basic : **SetIn** InputNumber, State

Lorsqu'un forçage est actif, toutes les fonctions de l'automate utilisant les entrées ne verront que l'état forcé de cette entrée.

Adresse	2000	2001	2002
Paramètre	Commande ID	Numéro Input	Forçage
Value	110	0,,255	-1, 0 ou 1

-1 => Pas de forçage

0 => Forçage à 0

1 => Forçage à 1

## Commande 200 : PLCBasic command

Cette commande permet le Lancement direct d'une commande PLCBasic (exactement comme si elle était envoyée depuis le champ 'Commande :' depuis ICNCStudio). Celle-ci est donc exécutée immédiatement.

Adresse	2000	2001	2002	2003	---	2001+n/2				
Paramètre	Commande ID	Longueur commande	Cmd	Cmd	Cmd	Cmd				
Value	200	0,,255	Cmd[1]	Cmd[0]	Cmd[3]	Cmd[2]	---	---	10	Cmd[n-1]

Une commande doit se terminer par un saut de ligne(CHR(10)).

La longueur de la commande est exprimée en nombre de caractères de la commande (y compris le saut de ligne).

Si la commande compte un nombre impair de caractères, le dernier registre de commande doit être le saut de ligne CHR(10).

Une commande peut être lancée même si le programme automate est en cours de fonctionnement.

Si un programme PLC est en mode RUN, l'envoi d'une commande erronée provoquera l'arrêt du programme PLC.

## Utilisation des lectures/écritures indexées

Vous disposez pour cela de 2 plages d'adresses.

- la zone des index située aux adresses 2800 à 2899 soit 50 registres 32 bits.
- la zone de valeurs indexées aux adresses 2900 à 2950 soit 50 registres 16 bits.

La zone des index doit être initialisée au lancement de votre application pour pointer sur les informations Modbus que vous souhaitez regrouper. Ces index peuvent pointer sur des variables Modbus de différents types (Holding registers, Input registers, Input bits ou Coils).

Après initialisation de cette zone suivant vos besoins, vous pouvez lire/écrire vos données dans la zone de valeurs indexées. Si des indexes correspondent à des variables en lecture seule, les opérations d'écriture seront ignorées.

Prenons l'exemple d'une application où vous souhaitez lire régulièrement les informations suivantes :

#### Input registers

1000 : Etat des entrées IN0 à IN15

1016 : Bits indicateurs d'axes en mouvement

1090 : Etat entrée analogique 0

3000 : Place disponible dans le buffer de commande CNC

3003 : Bits de status CNC

#### Holding registers

2160 : Etat des sorties OUT0 à OUT15

2400-2401 : Position Axe 1

2402-2403 : Position Axe 2

Ces informations étant de types divers (Holding et Input registers), il est intéressant d'utiliser l'indexation. Dans le cas contraire, la lecture de l'ensemble de ces informations requiert une multitude de requêtes.

**NB:** Pour accéder aux **Holding registers** par le tableau des Index, il convient d'**ajouter 100000** aux adresses Modbus requises, ajouter **200000** pour accéder aux adresses des **Input bits**, et ajouter **300000** aux adresses des **Coils**.

Dans le cas présent, il faut initialiser la table d'index avec les valeurs suivantes (codées en 32 bits):

[1000, 1016, 1090, 3000, 3003, 102160, 102400, 102401, 102402, 102403]

Soit le tableau Modbus des index suivants avec la correspondance dans le tableau de valeurs indexées :

Tableau des index			
Adresse Modbus	Registres 16 bits	Valeur 32 bits De l'index	Valeur pointée
2800	0x03E8	1000	Etat des entrée IN0 à IN15
2801	0x0000	0x000003E8	
2802	0x03F8		
2803	0x0000	1016	Bits indicateurs d'axes en mouvement
2804	0x0442	1090	
2805	0x0000	0x00000442	Etat entrée analogique 0
2806	0x0BBB	3000	Place disponible dans le buffer de commande CNC
2807	0x0000	0x00000BBB	
2808	0x0BBB	3003	Bits de status CNC
2809	0x0000	0x00000BBB	
2810	0x8F10	102160	État des sorties OUT0 à OUT15
2811	0x0001	0x00018F10	
2812	0x9000	102400	Position Axe 1 (poids faible)
2813	0x0001	0x00019000	
2814	0x9001	102401	Position Axe 1 (poids fort)
2815	0x0001	0x00019001	
2816	0x9002	102402	Position Axe 2 (poids faible)
2817	0x0001	0x00019002	
2818	0x9003	102403	Position Axe 2 (poids fort)
2819	0x0001	0x00019003	

Tableau des valeurs indexées	
Adresse Modbus	Valeur lue
2900	Etat des entrée IN0 à IN15
2901	Bits indicateurs d'axes en mouvement
2902	Etat entrée analogique 0
2903	Place disponible dans le buffer de commande CNC
2904	Bits de status CNC
2905	État des sorties OUT0 à OUT15
2906	Position Axe 1 (poids faible)
2907	Position Axe 1 (poids fort)
2908	Position Axe 2 (poids faible)
2909	Position Axe 2 (poids fort)

La lecture des registres 2900 à 2909 en une requête nous permet donc d'obtenir toutes les informations requises.

## Fonctions dédiées au pilotage CNC

L'interpCNC dispose d'un mode de fonctionnement bufferisé dédié à l'enchaînement de commandes indépendamment des temps de communication entre le PC et la CNC.

Quelque soit le canal de communication (USB, Ethernet, RS485), le protocole utilisé est le protocole Modbus.

On distinguera 2 types de commandes CNC :

- Les commandes bufferisées
- Les commande immédiates

Les commandes CNC bufferisées sont numérotées entre 1000 et 1999.  
Les commandes CNC à effet immédiat sont numérotée entre 2000 et 2999

Il est également possible de basculer le protocole de communication de la liaison USB pour exploiter un mode compatible Grbl. Ce basculement se produit automatiquement lors de la réception d'une trame de longueur  $\leq 4$  caractères et commençant pas '\$' ou Ctrl-X (chr(0x18)).  
Il reste actif jusqu'à la mise hors tension.

### 1° partie : commandes bufferisées

On distingue 3 buffers exploités pour la gestion des fonctions CNC:

1 buffer pour la communication avec un protocole compatible Grbl sur la liaison USB.

1 buffer pour la communication via des commandes Modbus.

1 buffer lié aux commandes traitées par le planificateur de mouvement.

Le buffer Grbl permet d'exploiter l'interpréteur Gcode embarqué à l'aide de nombreuses applications disponibles en open source. Nous allons dans un premier temps détailler le pilotage de la CNC via le protocole Modbus tout en sachant que

via ce protocole et la commande Modbus 1000 ci-dessous détaillée, il est également possible de travailler avec des commandes Gcode.

Il faut également noter que le programme automate embarqué peut fonctionner en parallèle du traitement des commande CNC. Vous pouvez donc avoir des fonctions autonomes d'automatisme qui n'interfèrent pas avec la gestion réalisée par le PC de pilotage.

Les commandes bufferisées vont être placées dans le buffer de communication dédié CNC via Modbus. Elle seront alors traitées par le planificateur pour assurer la fluidité des déplacements et éventuellement, les actions synchronisées avec ces mêmes déplacements.

Les commandes non bufferisées qui ont un effet immédiat. Il s'agit principalement de fonctions de modification globale du fonctionnement comme les changement de vitesses, les demandes de pause...

La place disponible dans le buffer de commande CNC Modbus peut être lue dans le registre (Input register 3000). Sa taille peut être obtenue par la lecture du registre 2990 (registre 16 bits).

La place disponible dans le buffer de commandes traitées par le planificateur peut être lue dans le registre de niveau de remplissage des commandes planifiées (Taille disponible dans Input register 2992).

Vous trouverez ci-dessous, le détail des commandes nécessaires à l'exploitation des fonctions CNC.

#### Commande 1000 : Exécution d'une instruction Gcode

Envoi d'une instruction de type Gcode. La chaîne de caractères est décomposée et placée dans le buffer d'envoi de commandes Modbus. Attention de prendre en compte l'inversion par octet dans les registres 16 bits.

Longueur est le nombre de caractère ASCII de la commande Gcode.

Le buffer utilisé pour cette commande est le buffer de commandes Grbl et non le buffer de commande Modbus.

Il convient donc de vérifier la place disponible dans ce buffer avant l'envoi d'une nouvelle commande.

La place disponible dans ce buffer est accessible à travers le Input register 3001.

Adresse	2000	2001	2002	2003	---	2001+
Paramètre	Commande ID	Longueur commande	Cmd	Cmd	Cmd	Cmd
Value	1000	0,,255	Cmd[1]   Cmd[0]	Cmd[3]   Cmd[2]	---	Cmd[n]   C

Exemple : pour envoyer une commande Gcode : G01 X12.2 F3000

Occupation dans le buffer Grbl : 2 + Longueur commande / 2

## Commande 1001 : Définition de la vitesse d'usinage en mm/mn

Permet d'indiquer la vitesse d'usinage pour les commandes de déplacements à suivre. La valeur de vitesse est exprimée en mm/mn et de type entier non signé 16 bits.

La valeur initiale de la vitesse d'usinage à la mise sous tension est de 1mm/mn.

<b>Adresse</b>	2000	2001	2002
<b>Paramètre</b>	Commande ID	float	
<b>Value</b>	1001	Vitesse mm/mn	

Occupation dans le buffer Modbus : 3 registres

## Commande 1002 : Déplacement linéaire interpolé des axes vers des positions cibles (positions absolues)

Cette commande permet de définir les trajectoires d'usinage.

Le premier argument permet de définir la vitesse à prendre en compte (vitesse d'usinage ou vitesse rapide) et également à indiquer les axes concernés par la commande. La longueur de la trame dépendra donc du nombre d'axes à déplacer. La vitesse d'usinage doit au préalable être définie à l'aide de la commande 1001.

Les positions sont données en mm et de type float. Les résolutions des axes doivent donc être correctement paramétrées (voir paramètre 1100 à 1105).

Seules les positions cibles des axes à déplacer et indiquées dans le premier argument, doivent être envoyées dans la trame.

<b>Adresse</b>	2000	2001	2002	2003	2004	2005	...
<b>Paramètre</b>	Commande ID	Indicateur	Float		Float		...
<b>Value</b>	1002	0x01 à 0x7F	Position 1 (mm)		Position 2 (mm)		...

Détail du paramètre Indicateur :

- Bit 0 => La cible de l'axe X est indiquée dans la trame
- Bit 1 => La cible de l'axe Y est indiquée dans la trame
- Bit 2 => La cible de l'axe Z est indiquée dans la trame
- Bit 3 => La cible de l'axe A est indiquée dans la trame
- Bit 4 => La cible de l'axe B est indiquée dans la trame
- Bit 5 => La cible de l'axe C est indiquée dans la trame
- Bit 6 => La vitesse de déplacement est la vitesse maxi si le bit est à 1. Sinon, la vitesse d'usinage est utilisée

Occupation dans le buffer Modbus : 2 + 2 \* Nombre d'axes à déplacer

## Commande 1003 : Interpolation circulaire

Permet de réaliser une trajectoire de type interpolation circulaire (Cercle ou arc de cercle).

Le déplacement se fait à la vitesse indiquée par la dernière commande ICNC3\_PushSetFeedRate.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Paramètre	Commande ID	Direction	X		Y		I		J	
Value	1003	0 : CW 1 : CCW	Position X (mm)		Position Y (mm)		Position I (mm)		Position J (mm)	

## Commande 1010: Action synchronisée (Sortie TOR, Sortie analogique, Registre)

Exécution d'une commande synchronisée avec les mouvements.

Les commandes peuvent être de différentes natures :

- Écriture de l'état d'une sortie TOR
- Écriture d'une sortie analogique
- Écriture d'un registre Modbus

Vous pouvez placer plusieurs actions synchronisées dans le buffer. Elles seront toutes associées à la prochaine commande de déplacement et exécutées avant le mouvement concerné.

Si aucune commande n'est actuellement présente dans le buffer, l'action est traitée immédiatement.

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Adresse	Valeur
Value	1010	1, 2 ou 3	Suivant Type	Suivant type

Action type 1 : Définition de l'état d'une sortie tout ou rien,

Action type 2 : Définition de l'état d'une sortie analogique

Action type 3 : Écriture dans un Holding registre r.

Détails des différents types de commandes synchronisées :

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	N° sortie TOR	Valeur
Value	1010	1	0 à 255	0 ou 1

Action de type 1 : écriture d'une sortie TOR

Adresse	2000	2001	2002	2003
Paramètre	Commande	Type	N° sortie	Valeur

	ID	d'action	Analog.	
<b>Value</b>	1010	2	0 à 7	0 à 10000mV

Action de type 2 : écriture d'une sortie analogique

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Type d'action	Adresse registre	Valeur
<b>Value</b>	1010	3	2000 à 65535	0 à 65535

Action de type 3 : écriture d'un registre Modbus (Holding register)

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Type d'action	Not used	Not used
<b>Value</b>	1010	4	0 à 65535	0 à 65535

Action de type 4 : Activation THC. Consigne de tension fixée par logiciel de pilotage dans holding register 2444,,2445

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Type d'action	Durée mesure (ms)	Not used
<b>Value</b>	1010	5	0 à 65535	0 à 65535

Action de type 5 : Activation . Consigne de tension fixée automatiquement par mesure de la tension d'arc en début de coupe. Le résultat de la mesure est stocké dans les holding registers 2444,,2445

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Type d'action	Not used	Not used
<b>Value</b>	1010	6	0 à 65535	0 à 65535

Action de type 6 : Arrêt de la fonction THC, L'axe Z est arrêté à sa position actuelle. Le traitement de cette action diffère des autres. En particulier, pour permettre une resynchronisation avec les prochaines commande de déplacement des axes, cette commande va figer le traitement des commandes durant la décélération de Z si il est en mouvement. Par conséquent, cette action ne doit pas être utilisée en plein milieu d'une trajectoire.

Si vous devez figer la THC dans une découpe, utilisez l'action THCPause (Type 7)

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Type d'action	Not used	Not used
<b>Value</b>	1010	7	0 à 65535	0 à 65535

Action de type 7 : Pause THC pour mettre en suspend la régulation de hauteur de torche. Si il y a un mouvement en cours, il est arrêté.

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Type d'action	Not used	Not used
<b>Value</b>	1010	8	0 à 65535	0 à 65535

Action de type 8 : Reprise THC pour remettre en fonction le THC qui a été suspendu par une action THCPause (type 7).

Toutes les actions synchronisées occupent 4 registres dans le buffer Modbus.

#### Commande 1011 : Temporisation bufferisée

Exécution d'une temporisation dans la succession de commandes placées dans le buffer.

<b>Adresse</b>	2000	2001
<b>Paramètre</b>	Commande ID	Durée temporisation en ms
<b>Value</b>	1011	0,,65535

Occupation dans le buffer Modbus : 2 registres

#### Commande 1012 : Attente d'un état ou d'un événement

Mettre la machine en attente d'un état (DIN, AIN ou registre) ou d'un événement. Cette commande permet de stopper l'exécution des commandes bufferisées jusqu'à ce qu'une condition soit remplie.

Cette commande prend en charge un timeout qui peut avoir pour effet de stopper la machine et de la mettre en état d'alarme ou de laisser le process se dérouler. Le code d'alarme est 18,

Vous pouvez par exemple utiliser cette commande pour contrôler la sortie de variateur de broche qui indique que la broche a atteint sa vitesse (attendre l'état d'une entrée).

Autre utilisation possible sur une machine de découpe plasma pour la détection du contact tôle (attente d'un front montant du palpeur de tôle).

#### Attente de l'état d'une entrée DIN

Adresse	2000	2001	2002	2003	2004	2005
Paramètre	Commande ID	Type d'action	Timeout (ms)	Action Timeout	Type attente	Numéro DIN
Value	1012	1	0..65535	0 ou 1	1 à 4	0 à 255

Type d'action = 1 pour l'attente d'état d'une entrée DIN

Timeout : Temps d'attente maximum de l'évènement (en milliseconde)

Action Timeout :

= 0 => continuer normalement en cas de timeout,

= 1 => Stop et alarme en cas de timeout, Le code d'alarme timeout est 18

Type d'attente :

1 : Attente d'un front montant sur l'entrée

2 : Attente d'un front descendant sur l'entrée

3 : Attente de l'état haut

4 : Attente de l'état bas

Numéro DIN : Numéro d'entrée concerné par l'attente (0 à 255)

#### Attente de l'état d'une entrée analogique

Adresse	2000	2001	2002	2003	2004	2005	2006
Paramètre	Commande ID	Type d'action	Timeout (ms)	Action Timeout	Type attente	Seuil tension	Numéro AIN
Value	1012	2	0..65535	0 ou 1	1 à 3	0,,10000	0,,7

Type Action = 2 pour l'attente de niveau d'une entrée analogique (AIN0 à AIN7)

Timeout : Temps d'attente maximum de l'évènement (en milliseconde)

Action Timeout :

= 0 => continuer normalement en case de timeout,

= 1 => Stop et alarme en cas de timeout, Le code d'alarme timeout est 18

Type d'attente :

1 : Attendre que l'entrée analogique soit inférieure à un seuil

2 : Attendre que l'entrée analogique soit supérieure à un seuil

3 : Attente que l'entrée analogique soit égale à une valeur

Seuil : Seuil ou valeur attendu exprimé en mv

Numéro AIN : Numéro d'entrée analogique concerné par l'attente

#### Attente de la valeur d'une information modbus (registre ou bit)

Adresse	2000	2001	2002	2003	2004	2005	2006 (LW)	2007 (H)
---------	------	------	------	------	------	------	-----------	----------

Paramètre	Commande ID	Type d'action	Timeout (ms)	Action Timeout	Type attente	Seuil	Adresse modbus étendu
<b>Value</b>	1012	3	0..65535	0 ou 1	1 à 4	0,,65535	0,,365535

Type Action = 3 pour l'attente de valeur d'une information modbus

Timeout : Temps d'attente maximum de l'évènement (en milliseconde)

Action Timeout :

= 0 => continuer normalement en case de timeout,

= 1 => Stop et alarme en cas de timeout, Le code d'alarme timeout est 18

Type d'attente :

1 : Attendre que la valeur modbus soit inférieure à un seuil

2 : Attendre que la valeur modbus soit supérieure à un seuil

3 : Attente que la valeur modbus soit égale à une valeur

Adresse modbus à surveiller :

L'adresse est indiquée dans un format 32 bits pour permettre l'accès aux différent type de variables modbus.

Pour un registre de type Input registre : Adresse comprise entre 0 et 65535

Pour un registre de type Holding register : Adresse comprise entre 100000 et 165535

Pour une bit de type Input bit : Adresse comprise entre 200000 et 265535

Pour une bit de type Coil bit : Adresse comprise entre 300000 et 265535

Seuil : Seuil ou valeur attendu. Pour les valeur de type Bit, valeur de 0 ou 1

## 2° partie : Commandes non bufferisées

### Commande 1100 : Modification override Usinage et déplacement rapide

Action immédiate de modification des vitesse d'usinage (feed rate) et hors matière (Rapide Move).

L'override de la vitesse d'usinage dépend de la vitesse d'usinage en cours.

Vous pouvez également lire/écrire ces valeurs directement dans les Holding registers 2430 et 2431.

Adresse	2000	2001	2002
<b>Paramètre</b>	Commande ID	Override usinage en %	Override Hors matiere en %
<b>Value</b>	1005	0..65535	0..65535

### Commande 1101 : Pause de l'usinage en cours

Utilisez de préférence la commande 1200 avec code de fonction 130

Arrêt immédiat des mouvements en cours. Les buffers ne sont pas vidés.

Il est également possible d'affecter une entrée TOR pour la mise en pause d'un usinage via le paramètre 951. Il faut alors indiquer un numéro d'entrée de 0 à 255 affecté à cette fonction. Pour l'inhiber, indiquer un numéro de -1.

<b>Adresse</b>	2000
<b>Paramètre</b>	Commande ID
<b>Value</b>	1101

#### Commande 1102 : Reprise d'un usinage interrompu

Utilisez de préférence la commande 1200 avec code de fonction 129

Reprise de l'usinage en cours

Il est également possible d'affecter une entrée TOR pour la reprise d'un usinage via le paramètre 952. Il faut alors indiquer un numéro d'entrée de 0 à 255 affecté à cette fonction. Pour l'inhiber, indiquer un numéro de -1.

<b>Adresse</b>	2000
<b>Paramètre</b>	Commande ID
<b>Value</b>	1102

#### Commande 1110 : Exécution d'une séquence de homing machine

Lancement d'une commande de homing suivant les paramètres internes pré-configurés.

<b>Adresse</b>	2000	2001
<b>Paramètre</b>	Commande ID	Axes concernés
<b>Value</b>	1110	0..31

Axes concernés :

0 : lancer intégralité de la séquence de homing indiquée dans les paramètres

1,,6 : Spécifier l'axe à initialiser

#### Commande 1111 : Exécution d'un déplacement manuel (Jog)

Cette commande permet de déplacer les axes en mode relatif ou absolue. Elle ne sera prise en compte que si le status CNC actuel est « Idle » ou « Jog », Les déplacements en cours peuvent être interrompus par la commande direct « Arrêt Jog » (commande 120 0, fonction 133).

Vous pouvez combiner les déplacements simultanés de plusieurs axes. La vitesse indiquée sera la vitesse combinée des différents mouvements.

Comme pour les commandes de déplacements linéaires, la longueur de la trame dépend du nombre d'axes à déplacer.

Les positions des axes doivent être données dans l'ordre croissant des indexes d'axe.

Exemple, pour déplacer les axes X et Z, l'indicateur vaudra 5, les positions seront données avec Position X puis Position Z.

Si le bit 6 de l'indicateur des axes à déplacer est à 0, les positions seront des positions absolues. Si le bit vaut 1, il s'agira d'une position relative.

Pour les déplacements manuels, il est recommandé d'utiliser cette commande de Jog en mode incrémentiel. En cas de rupture de communication, les mouvements seront ainsi automatiquement limités.

<b>Adresse</b>	2000	2001	2002	2003	2004	2005	2006	2007	...
<b>Paramètre</b>	Commande ID	Indicateur	Float		Float		Float		
<b>Value</b>	1111	0x01 à 0x7F	Vitesse (m/mn)		Position 1 (mm)		Position 2 (mm)		

Détail du paramètre Indicateur :

- Bit 0 => La cible de l'axe X est indiquée dans la trame
- Bit 1 => La cible de l'axe Y est indiquée dans la trame
- Bit 2 => La cible de l'axe Z est indiquée dans la trame
- Bit 3 => La cible de l'axe A est indiquée dans la trame
- Bit 4 => La cible de l'axe B est indiquée dans la trame
- Bit 5 => La cible de l'axe C est indiquée dans la trame
- Bit 6 => 0 → Positions absolues ; 1->Positions relatives

#### Commande 1200 : Exécution directe d'une commande

Cette commande permet d'agir immédiatement sur le fonctionnement de la CNC. Elle est utilisée pour un contrôle en temps réel sur différents paramètres et sur l'état de fonctionnement de la machine.

<b>Adresse</b>	2000	2001
<b>Paramètre</b>	Commande ID	Code fonction
<b>Value</b>	1200	0..255

Valeurs possible de la sous-commande :

Code fonction	Action	Détails
88	Acquittement alarme	

129	Reprise usinage	Reprise d'un usinage après une mise en pause
130	Pause usinage	Mise en pause de l'usinage, Arrêt de la broche Les buffers ne sont pas vidés.
133	Arrêt Jog	Arrêt d'un mouvement de jog en cours
138	Annuler survitesse usinage	La survitesse usinage repasse à 100 %
143	Annuler survitesse rapide	La survitesse hors matière repasse à 100 %
153	Annuler survitesse broche	La survitesse broche repasse à 100 %
255	Arrêt immédiat	Arrêt des mouvements CNC avec rampe décélération. Si le THC est actif, il est arrêté. Les buffers de commandes et de planification sont également purgés.

## Utilisation de l'horloge interne RTC

L'interpCNC dispose d'une horloge interne permettant de gérer la date et l'heure. Cette horloge n'est cependant pas sauvegardée lors de la mise hors tension. Il convient donc de l'initialiser avant d'en exploiter les fonctions.

L'initialisation peut se faire par :

- Les commandes modbus 112, 113 et 114,

- Le programme automate à l'aide de la commande RTC,

- Automatiquement via un serveur SNTP si l'interpCNC dispose d'un accès internet.

Pour la mise à jour automatique par STNP, les paramètres 546 et 547 doivent être correctement réglés. Le serveur SNTP utilisé est « sntp.pool.org ».

L'horloge sera alors initialisée en tenant compte de fuseau horaire indiqué dans le paramètre 547 et de l'heure d'été/hiver si le bit b1 du paramètre 547 est actif.

Vous disposez de 2 bits de status qui permettent de déterminer l'état de la synchronisation :

stsBit(STS\_RTC\_SYNCHRONIZED) qui indique que l'horloge a été réglée,

stsBit(STS\_SNTP\_CONNECTED) qui indique qu'une connexion SNTP est établie.

La synchronisation par serveur SNTP, si elle est activée, est renouvelée automatiquement toutes les heures.

Il est possible de lire les différents registres de l'horloge (heure, date) dans les Input Registres 1987 à 1995.

Dans le programme automate, s'agissant de registres en lecture seule, il convient d'ajouter 100000 à l'adresse pour une lecture avec la commande GetMW. Pour lire ces registres, les commandes seront donc GetMW(101987) à GetMW(101995)

Note pour la lecture à travers les registres modbus :

Pour obtenir des données consistantes, la lecture de l'Input Register 1987 (Heure RTC) provoque la création d'un tampon mémorisant la date et l'heure actuelles. Ce tampon reste valide pendant au minimum 100ms. L'idéal est donc de lire l'ensemble des informations requises dans ce laps de temps.

De nombreuses fonctions PLCBasic sont également disponibles pour exploiter l'horloge RTC. Elles sont détaillées dans la documentation spécifique PLCBasic.

### Commande 112 : Réglage de la date sur l'horloge RTC

Cette commande permet de régler la date de l'horloge interne à l'automate.

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	Jour	Mois	Année
<b>Value</b>	112	1,,31	1,,12	0,,99

### Commande 113 : Réglage de l'heure sur l'horloge RTC

Cette commande permet de régler l'heure de l'horloge interne à l'automate.

<b>Adresse</b>	2000	2001	2002	2003
<b>Paramètre</b>	Commande ID	heure	minutes	seconde
<b>Value</b>	113	0,,23	0,,59	0,,59

## Commande 114 : Réglage simultané date et heure sur l'horloge RTC

Cette commande permet de régler simultanément la date et l'heure de l'horloge interne à l'automate.

Adresse	2000	2001	2002	2003	2004	2005	2006
Paramètre	Commande ID	Jour	Mois	Année	heure	minutes	seconde
Value	114	1,,31	1,,12	0,,99	0,,23	0,,59	0,,59

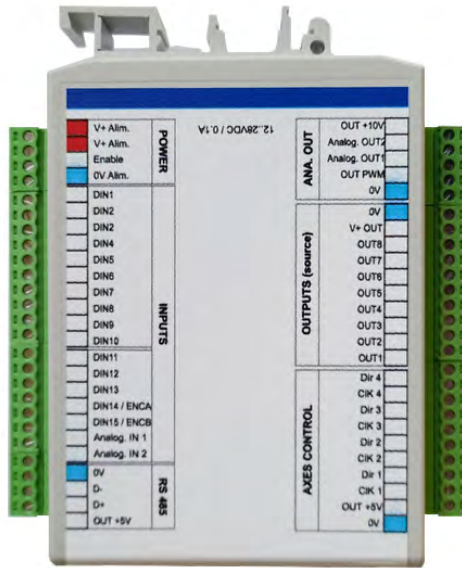
## Documentation\_MODBUS\_V2

---

**SOPROLEC**  
 ZAC DE L'EPINE  
 72460 SAVIGNE L'EVEQUE  
 Tél : +33 (0)2 4376 4476



**Cartes d'axes  
 SOPROLEC  
 InterpCNC V2**



## Protocole de communication MODBUS

### Introduction

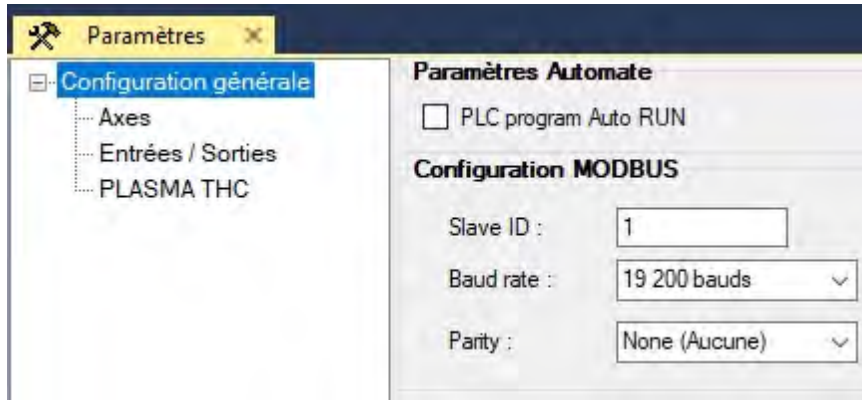
L'**InterpCNC V2** dispose d'une interface RS485 half-duplex utilisant le protocole de communication MODBUS RTU.

Le présent document indique les différentes commandes Modbus utilisables pour le contrôle de la carte d'axes.

L'utilisation de la communication Modbus peut se faire simultanément avec la communication USB de la carte.

La configuration MODBUS se fait depuis la fenêtre des Paramètres ->

## Configuration générale



EE\_MODBUS\_BASE\_ADDRESS (paramètre N°103) : Adresse Modbus de la carte

EE\_MODBUS\_SPEED (Paramètre N°104) : Vitesse de communication

NB: Le paramètre "vitesse de communication Modbus" est aussi disponible aux adresses: 4554 – 4555 (Poids Fort - poids faible)

EE\_MODBUS\_POLARITY (Paramètre N°105) : Parité

Les autres paramètres de communications fixes sont :

Bits de données : 8 bits

Bits de stop : 1 bit

**Note :** Le système d'adressage des automates peut varier suivant les constructeurs.

Les adresses Modbus indiquées ci-dessous peuvent donc être correctes pour certains automates et décalées de 1 case pour d'autres automates.

## Lecture / écriture des paramètres

L'accès aux paramètres sauvegardés de la carte InterpCNC peut se faire à l'aide de l'utilitaire ICNCStudio, mais également en Modbus.

Le tableau des paramètres est accessible en Lecture/Écriture (Holding registers) à partir de l'adresse 4352.

Tous les paramètres sont au format 32 bits (2 registres par paramètre).

## Adresses des Bits en lecture seule (Input Bits)

### Adresses des bits en lecture seule

Les registres représentent l'état des entrées TOR de la carte InterpCNC ainsi que les bits de status.

Toutes ces informations sont également disponible sous forme de mots 16 bits

dans les registre en lecture seule.

L'accès à ces bits se fait par la commande Modbus 2 (Variables de type 1X)

Adresse	Fonction
2048 à 2063	État entrée IN1 à IN16
2064 à 2079	Réservé
2080	Axe X en cours de mouvement interpolé
2081	Axe Y en cours de mouvement interpolé
2082	Axe Z en cours de mouvement interpolé
2083	Axe A en cours de mouvement interpolé
2084	Axe B en cours de mouvement interpolé
2085	Buffer de communication USB Vide
2086	Buffer de communication USB gelé
2087	État de l'entrée /Enable
2088	Carte verrouillée
2089	Fin de course (non utilisé)
2090	Prise d'origine (homing) en cours
2091	Erreur durant la prise d'origine
2092	Palpage en cours
2093	Erreur durant palpation
2094	Écriture EEPROM interne en cours
2095	Erreur d'écriture sur EEPROM
2096	Réserve
2097	Sur-vitesse active (override)
2098	Sur-vitesse autorisée(override)
2099	Fonction d'attente de l'état d'une entrée en cours
2100	Erreur d'attente de l'état d'une entrée (Timeout)
2101	Fonction THC active (spécifique machine plasma)
2102	Axe X en cours de mouvement indépendant
2103	Axe Y en cours de mouvement indépendant
2104	Axe Z en cours de mouvement indépendant
2105	Axe A en cours de mouvement indépendant
2106	Axe B en cours de mouvement indépendant
2107	Fin de course (Positif ou négatif) de l'axe X enclenché
2108	Fin de course (Positif ou négatif) de l'axe Y enclenché
2109	Fin de course (Positif ou négatif) de l'axe Z enclenché
2110	Fin de course (Positif ou négatif) de l'axe A enclenché
2111	Fin de course (Positif ou négatif) de l'axe B enclenché
2112	Sens de déplacement actuel de l'axe X
2113	Sens de déplacement actuel de l'axe Y
2114	Sens de déplacement actuel de l'axe Z
2115	Sens de déplacement actuel de l'axe A
2116	Sens de déplacement actuel de l'axe B
2117	État entrée codeur A
2118	État entrée codeur B
2119	État entrée codeur C
2120 à 2304	Réservé

## Adresses des bits en lecture / Écriture

### Adresses des bits en lecture / Écriture

Ces registres Permettent l'accès aux sorties (OUT1 à OUT12) de l'InterpCNC bit à bit.

L'accès à ces bits se fait par la commande Modbus 1, 5 et 15 (Variables de type 0X),

L'accès à ces sorties est également possible à l'aide de mot 16 bits (voir adresses 1038 et 1039)

Adresse	Fonction
256 à 267	Lecture / Écriture des sorties physiques OUT1 à OUT12
268 à 287	Lecture / Écriture des sorties virtuelles OUT13 à OUT32
288	Bit verrouillage/déverrouillage de la carte
289 à 319	Réservé
320 à 415	96 Bits de communication (disponibles utilisateur)
416 à 512	Réservé

## Adresses des Registres en Lecture seule (Input registers)

### Adresses des Registres en lecture seule

Les registres en lecture seule sont accessibles avec les fonctions Modbus 4 (variables type 3X),

Ces registres représentent les variables internes de l'InterpCNC.

Adresse	Fonction
1024	Status POIDS FORT (détail ci-dessous)
1025	Status Poids faible (détail ci-dessous)
1026	Position X POIDS FORT
1027	Position X Poids faible
1028	Position Y POIDS FORT
1029	Position Y Poids faible
1030	Position Z POIDS FORT
1031	Position Z Poids faible
1032	Position A POIDS FORT
1033	Position A Poids faible
1034	Position B POIDS FORT
1035	Position B Poids faible
1036	État actuel des entrées (IN17 à IN32)
1037	État actuel des entrées (IN1 à IN16)
1038	État actuel des sorties (OUT17 à OUT32)
1039	État actuel des sorties (OUT1 à OUT16)
1040	Entrée analogique 1
1041	Entrée analogique 2
1042	Entrée analogique 3
1043	Entrée analogique 4
1044	Position résultat du dernier « palpage » réalisé Poids Fort
1045	Position résultat du dernier « palpage » réalisé Poids Faible

1046 à 1053	Réserve
1054	Status étendu Poids faible (détails ci-dessous)
1055	Status étendu poids FORT (détails ci-dessous)
1100	Entrée analogique 1 (idem 1040)
1101	Entrée analogique 2 (idem 1041)
1102	Entrée analogique 3 (idem 1042)
1103	Entrée analogique 4 (idem 1043)
1104	État actuel des sorties OUT 1 à OUT 32 pf/PF (idem 1038 mais 4X)
1106	État actuel des entrées IN1 à IN32 pf/PF (idem 1036 mais 4X)
1108	Status B0 à B31 (idem 1024 mais pf/PF)
1110	Status Etendu B32 à B63 (idem 1054 mais pf/PF)
1112	Position X – Axe 1 (Idem 1026 mais pf/PF)
1114	Position X – Axe 2 (Idem 1028 mais pf/PF)
1116	Position X – Axe 3 (Idem 1030 mais pf/PF)
1118	Position X – Axe 4 (Idem 1032 mais pf/PF)
1120	Position X – Axe 5 (Idem 1034 mais pf/PF)
1122	Position résultat Probe (Idem 1044 mais pf/PF)
1124	Velocity axe 1 pf/PF
1126	Velocity axe 2 pf/PF
1128	Velocity axe 3 pf/PF
1130	Velocity axe 4 pf/PF
1132	Velocity axe 5 pf/PF

#### Détails du registre de status :

B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21
<b>B Limit Switch</b>	<b>A Limit Switch</b>	<b>Z Limit Switch</b>	<b>Y Limit Switch</b>	<b>X Limit Switch</b>	<b>Axe B Async Moving</b>	<b>Axe A Async Moving</b>	<b>Axe Z Async Moving</b>	<b>Axe Y A sync Moving</b>	<b>Axe X Async Moving</b>	THCAc vated

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5
EEWrite Error	EEWrite In Progress	<b>Probe Error</b>	<b>Probe In Progress</b>	<b>Homing Error</b>	<b>Homing In Progress</b>	Stroke Limit	<b>Board Locked</b>	<b>Emergency Stop</b>	Buffer Freezed	B

B0..B4 : Axes actuellement en cours de déplacement sur une interpolation linéaire.

B7 : Emergency Stop : Etat de l'entrée ENABLE/.

1 => L'entrée ENABLE n'est pas active. Carte verrouillée.

0 => L'entrée ENABLE est active (24V). La carte peut alors être utilisée.

B8 : Board Locked : Fonctionnement des sorties et des commandes d'axes verrouillées.

Tant que ce bit est actif, les mouvements d'axes et l'action sur les sorties est verrouillée.

(Voir détail sur la commande N°66 : Reset et Ré-armement

B10 : Homing In Progress : La commande Homing est en cours d'exécution.

B11 : Homing Error : La fonction Homming a échoué. Ce bit repasse automatiquement à 0 lors du

lancement d'une nouvelle commande de Homing.

B12 : Probe In Progress : Fonction de palpage en cours.

B13 : Probe Error : La fonction de palpage a échoué. Ce bit repasse automatiquement à 0 lors du lancement d'une nouvelle commande de Palpage.

B22 à B26 : Un mouvement indépendant est en cours sur l'axe en question. Repasse à 0 lorsque le mouvement programmé est terminé ou interrompu par l'une des commandes de Stop.

B27 à B31 : Indicateur d'état des fins de courses

### Détails du registre de status étendu :

B31	B30	B29	B28	B27	B26	B25	B24	B23
Reserved	Reserved	Reserved	Reserved	Homing B Error	Homing B	Homing A Error	Homing A	Homing Z Error
B15	B14	B13	B12	B11	B10	B9	B8	B7
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Recipe Data Changed	Reserved	Basic Running

B0..B4 : Direction actuelle des déplacements (0 si négatif, 1 si positif). Ces bits ne représentent pas la valeurs physique des sorties de direction (qui dépend des paramètres de sens de rotation) mais bien les sens de déplacement.

B5 : Perte de caractères durant la transmission par l'interpréteur Basic

B6 : Perte de caractères durant la réception par le programme Basic

B7 : Programme Basic en cours d'exécution

B18 à B26 : Status de prise d'origine des axes. Ces bit sont à exploiter lors de l'utilisation de la commande PLCBasic Home(...) ou de la commande de homing Modbus N°78

### Registres en Lecture/Écriture (Holding registers)

Les registres en lecture seule sont accessibles avec les fonctions Modbus 3, 6 et 16 (variables de type 4X).

Ces registres permettent d'agir sur la carte InterpCNC et de lancer les commande de gestion d'axes.

Adresse	Fonction	Format
---------	----------	--------

4096 à 4145	Registres utilisés pour l'envoi de commandes	
4146	Sortie analogique 1 (0 à 1023)	
4147	Sortie analogique 2 (0 à 1023)	
4148	Sorties tout ou rien OUT 17 à OUT 32	4X INV
4149	Sorties tout ou rien OUT 1 à OUT 16	
4150 à 4195	46 Registres (RAM) disponibles pour utilisateur	
4196	Simulation clavier matriciel Poids faible	4X INV
4197	Simulation clavier matriciel Poids fort	
4198 à 4217	Tableau utilisateur de 10 valeurs 32 bits (non sauvegardé)	4X
4218	Lecture position codeur PF (à partir de V2.1D)	4X INV
4219	Lecture position codeur pf (à partir de V2.1D)	
4220	Lecture compteur PF entrée A	4X INV
4221	Lecture compteur Pf entrée A	
4222	Lecture compteur PF entrée B	4X INV
4223	Lecture compteur Pf entrée B	
4224 à 4351	128 Registres (RAM) disponibles pour utilisateur	
4352 à 5119	Paramètres sauvegardés de l'InterpCNC	4X INV
5120 à 5631	Zone mémoire utilisateur sauvegardée ( 1Ko EEPROM)	
5632	Taille des recettes (EEPROM)	
5633	Index 0 recette (EEPROM)	
5634	Index 1 recette (EEPROM)	
5635	Index 2 recette (EEPROM)	
5636	Index 3 recette (EEPROM)	
5637 à 5886	Recette Page 0 (Voir détail ci dessous)	
5887 à 6136	Recette Page 1 (Voir détail ci dessous)	
6137 à 6386	Recette Page 2 (Voir détail ci dessous)	
6387 à 6636	Recette Page 3 (Voir détail ci dessous)	
	Sur ICNC22 et ICNC24, depuis le firmware 5.40 et tous accessibles avec les fonctions Get/Set MW/MI/MDW/MDI/MF :	
6700	Position X – Axe 1 pf/PF	4X
6702	Position Y – Axe 2 pf/PF	4X
6704	Position Z – Axe 3 pf/PF	4X
6706	Position A – Axe 4 pf/PF	4X
6708	Position B – Axe 5 pf/PF	4X
6710	Lecture position codeur pf/PF (Idem 4218 mais registres inversés 4X)	4X INV
6712	Lecture compteur entrée A pf/PF (idem 4220 mais registres inversés 4X)	4X INV
6714	Lecture compteur entrée B pf/PF (idem 4222 mais registres inversés 4X)	4X INV
7000 à 7031	Registres utilisateurs accessibles avec les commandes SetUserMem/GetUserMem 0,,15 Soit 16 registres 32 bits dont les 10 premiers sont accessibles en modbus adresses 4198 à 4217	4X
7032 à 7077	Registres utilisateurs également accessibles aux adresses modbus 4150 à 4195	4X

7078 à 7333	Registres utilisateurs également accessibles aux adresses modbus 4224 à 4351	4X
-------------	--	----

## Généralités sur l'envoi des commandes Modbus

Le contrôle de la carte **InterpCNC V2** se fait par l'envoi de commande à l'adresse 4096 à l'aide de la commande Modbus n°16 (Écriture multiple de registres). Les arguments qui suivent l'adresse 4096 sont utilisés en paramètres de la commande concernée.

Chaque écriture dans le registre 4096 est considérée comme l'envoi d'une nouvelle commande.

Si votre automate ne dispose pas de la fonction 16 mais uniquement de la fonction 6 (Écriture d'un registre), commencez par envoyer les paramètres de commandes avant l'envoi de la commande à l'adresse 4096.

### Commande 66 : Reset et Ré-armement

Cette commande permet de déverrouiller le fonctionnement de la carte InterpCNC lorsque le bit de status « Board Locked » est actif.

Il est également possible de réarmer les bit d'erreur liés aux fonction de palpé et de prise d'origine.

Adresse	Valeur
4096	Commande = 66
4097	Bits de ré-armement B0 : Réarmement de « Board Locked », B1 : Réarmement du bit d'erreur de prise d'origine « Homing Error » B2 : Réarmement du bit d'erreur de palpé « Probe Error »

Avec la configuration par défaut de la carte, à la mise sous tension, le bit « Board Locked » est actif. Toutes les actions sur les sorties et les commandes de déplacement d'axes sont ainsi verrouillées.

Au lancement de l'application de commande, il est donc nécessaire d'envoyer une commande N°66 avec la valeur 1 en bit de ré-armement.

Si l'entrée ENABLE de la carte est bien actif, sera aura pour effet de libérer le fonctionnement de la carte.

Si en cours de fonctionnement, l'entrée Enable est désactivé, le bit « Board Locked » sera à nouveau actif et la carte verrouillée.

Par sécurité, même si l'entrée ENABLE repasse à 1, le fonctionnement reste verrouillé jusqu'à un déverrouillage volontaire par l'application de commande ou l'action sur une entrée configurée à cette effet (entrée de réarmement configurée dans le paramètre N°88 : EE\_INPUT\_UNLOCK

Note : Il est possible de configurer la carte en mode réarmement Automatique

.....	ID	0	--	0	0	--
EE_AU	Emergency Stop input	0		0	32	Emergency stop input number (Virtually connected to Enable in
EE_POLARITY_FDC	Switch Polarity	0		0	255	Special input switch polarity
EE_OUTPUT_SET_NC_READY	Output Set Board Ready	0		0	-1	Output seted when board is ready (unlocked)
EE_OUTPUT_RESET_NC_READY	Output Reset Board Ready	0		0	-1	Output reseted when board is ready (unlocked)
EE_OUTPUT_SET_NC_NOT_READY	Output Set Board Not Ready	0		0	-1	Output seted when board is not ready (locked)
EE_OUTPUT_RESET_NC_NOT_READY	Output Reset Board Not Ready	0		0	-1	Output reseted when board is not ready (locked)
EE_INPUT_UNLOCK	Unlock Input Number	0		0	32	Input number used to reset the board (unlock)
EE_INPUT_UNLOCK_POLARITY	Unlock Input Number Polarity	1		0	1	Unlock input polarity (0=NO; 1=NC)
EE_OUTPUT_START	Output Boot state	0		0	-1	Initial Output state after boot

Dans l'exemple ci-dessus, l'entrée de réarmement EE\_INPUT\_UNLOCK est à 0 et la polarité de cette entrée est de type NC.

Cette condition étant toujours vrai, le réarmement se fera automatiquement dès que l'entrée ENABLE de la carte sera à un niveau haut.

## Commande 80 : Déplacement indépendant d'un axe

(ICNC\_CMD\_MOVE\_PROFILE\_ABS\_ASYNC).

La commande 80 permet de lancer le déplacement indépendant (sans interpolation) d'un axe.

L'interpCNC dispose de générateurs de profils indépendants pour chaque axe. Vous pouvez par conséquent lancer des déplacements indépendants et simultanément sur les différents axes.

A chaque axe est associé un bit de status indiquant la progression des mouvements indépendants.

Ces bits de status sont accessibles dans le registre de status général et également, en lecture de bit (AxeXAsyncMoving à AxeBAsyncMoving),

Il est possible d'interrompre l'ensemble des déplacements à l'aide des commandes 42 (ICNC\_CMD\_BREAKE\_AXES\_AND\_CLEAR) et 43 (ICNC\_CMD\_STOP).

Il est également possible d'interrompre un à un les déplacements à l'aide de la commande 41 (ICNC\_CMD\_BREAKE\_AXES).

Si l'axe est déjà en cours de mouvement, la cible sera changée à la volée et le profile de vitesse sera recalculé automatiquement.

Adresse	Valeur
4096	Commande = 80
4097	Axe à déplacer (1=X ; 2=Y ; 4=Z ; 8=A ; 16=B)
4098	Fréquence start stop (Hz)
4099	Accélération (Khz/s)
4100	Vitesse (Hz) Poids fort
4101	Vitesse (Hz) Poids faible
4102	Décélération (Khz/s)
4103	Position Poids fort
4104	Position Poids faible

## Commande 6 : Déplacement interpolés des axes avec profile de vitesse

(ICNC\_CMD\_PROFILE\_ABS\_BUF).

La commande 6 permet de lancer un déplacement interpolé sur les axes.  
La longueur de la trame transmise dépend du nombre d'axe à déplacer.

Le champs « Axes » permet de sélectionner les axes à prendre en compte pour le déplacement interpolé.

Le champs Vitesse indique la vitesse en Hertz de déplacement de l'axe majeur (celui qui a le plus grand déplacement à faire).

Par exemple :

Si axe vaut 3, les axes X et Y seront positionnés et les positions de X et de Y seront indiquées respectivement dans les champs Position 1 et Position 2.

Si axe vaut 5, les axes X et Z seront positionnés et les positions de X et de Z seront indiquées respectivement dans les champs Position 1 et Position 2.

Note : Vous pouvez lancer un déplacement interpolé même si certains axes sont déjà en cours de mouvements indépendants. En revanche, vous devez prendre garde de ne pas lancer un mouvement interpolé sur des axes déjà en cours de mouvement indépendant.

Les commandes de déplacement sont bufferisées par l'InterpCNC. Vous pouvez donc envoyer plusieurs commandes qui seront traitées les unes à la suite des autres.

Adresse	Valeur
4096	Commande = 6
4097	Axes à déplacer (b0:X ; b1:Y ; b2:Z ; b3:A ; b4:B)
4098	Vitesse (Hz) de l'axe majeur Poids fort
4099	Vitesse (Hz) de l'axe majeur Poids faible
4100	Position 1 Poids fort
4101	Position 1 Poids faible
4102	Position 2 Poids fort
4103	Position 2 Poids faible
4104	Position 3 Poids fort
4105	Position 3 Poids faible
4106	Position 4 Poids fort
4107	Position 4 Poids faible
4108	Position 5 Poids fort
4019	Position 5 Poids faible

## Commande 71 : Prise d'origine des axes (ICNC\_CMD\_MACHINE\_HOME)

La commande 71 permet de lancer la séquence de prise d'origine sur un axe ou plusieurs axes.

Les paramètres de la séquence de prise d'origine sont à renseigner dans les paramètres de la carte.

Seules les courses maximales sont à indiquer lors du lancement de la commande 71.

Durant la prise d'origine le bit de status « HomingInProgress » est à 1.  
En fin de prise d'origine, il repasse à 0.

Vous pouvez alors tester le bit « HomingError » pour vous assurer le bon déroulement de la procédure de recherche d'origine.

Adresse	Valeur
4096	Commande = 71
4097	Axes à déplacer (b0:X ; b1:Y ; b2:Z ; b3:A ; b4:B)
4098	Course maximum Axe 1 Poids fort
4099	Course maximum Axe 1 Poids faible
4100	Course maximum Axe 2 Poids fort
4101	Course maximum Axe 2 Poids faible
4102	Course maximum Axe 3 Poids fort
4103	Course maximum Axe 3 Poids faible
4104	Course maximum Axe 4 Poids fort
4105	Course maximum Axe 4 Poids faible
4106	Course maximum Axe 5 Poids fort
4107	Course maximum Axe 5 Poids faible

Exemple 1 - Prise d'origine des axes X et Y, course maximum de 10000 pas sur X, course maximum de 20000pas sur Y.

Adresse	Valeur	Fonction
4096	71	Commande Homing
4097	3	Axe X(b0) et Y(b1)
4098	0	Maxi X poids fort
4099	10000	Maxi X poids faible
4100	0	Maxi Y poids fort
4101	20000	Maxi Y poids faible

Exemple 2 - Prise d'origine des axes X et Z, course maximum de 10000 pas sur X, course maximum de 20000 pas sur Z.

Adresse	Valeur	Fonction
4096	71	Commande Homing
4097	5	Axe X(b0) et Z(b2)
4098	0	Maxi X poids fort
4099	10000	Maxi X poids faible
4100	0	Maxi Z poids fort
4101	20000	Maxi Z poids faible

### Commande 78 : Prise d'origine d'un axe (ICNC\_CMD\_HOME\_AXE)

La commande 78 permet de lancer des séquences de prise d'origine individuelles sur les différents axes.

Contrairement à la commande 71 ( ICNC\_CMD\_MACHINE\_HOME), vous

disposez alors de bits de status individuels pour chacun des axes (voir bits de status étendus B18 à B27).

Les séquence de homing peuvent être lancées de manière asynchrone et indépendante sur chaque axes.

Adresse	Valeur
4096	Commande = 78
4097	Numéro de l'axe (1 à 5)
4098	Numéro de l'entrée
4099	Type de contact utilisé (NC=0, NO=1)
4100	Accélération (Khz/s)
4101	Vitesse rapide de déplacement vers le capteur – Poids fort
4102	Vitesse rapide de déplacement vers le capteur – Poids faible
4103	Décélération (Khz/s)
4104	Course maximum recherche capteur - Poids fort
4105	Course maximum recherche capteur - Poids faible
4106	Vitesse lente de dégagement du capteur
4107	Position initiale en fin de homing – Poids Fort
4108	Position initiale en fin de homing – Poids Faible

#### Commande 40 : Déplacement jusqu'à un capteur (fonction de palpage ICNC\_CMD\_PROBE)

Cette commande n° 40 permet de déplacer un axe jusqu'à la détection d'un capteur.

Lors de la détection de l'état attendu sur l'entrée indiqué, l'axe en mouvement s'arrête (avec la décélération indiquée dans les paramètres de la commande). La position de l'axe à l'instant du changement d'état (donc, avant la décélération) est alors placée dans les registres aux adresses 1044 et 1045.

Durant le palpage, le bit de status « ProbeInProgress » est à 1. Ce bit passe à 0 en fin de palpage ou en cas d'erreur.

Le bit « ProbeError » passe à 1 si l'entrée attendue n'a pas été détectée dans la cours indiquée.

Adresse	Valeur
4096	Commande = 40
4097	Axes à déplacer (b0:X ; b1:Y ; b2:Z ; b3:A ; b4:B)
4098	Direction (1 ou -1)
4099	Numéro de l'entrée concernée (1 à 32)
4100	État de l'entrée attendu (NC=0, NO=1)
4101	Course maximum Poids fort
4102	Course maximum Poids faible
4013	Vitesse Poids fort
4014	Vitesse Poids faible
4015	Fréquence start/stop
4016	Accélération (Khz/s)
4017	Décélération (Khz/s)

Lancement du palpage de l'axe Z

Adresse	Valeur	Fonction
4096	40	Commande Probe
4097	4	Axe Z(b3) sur l'axe Z
4098	-1	Sens Z-
4099	3	Entrée N°3
4100	1	Détecter le passage à 1 de l'entrée
4101	0	Course maximum
4102	10000	Course maximum de 10000 pas
4103	0	Vitesse Poids fort
4104	1000	Vitesse poids faible 1000Hz
4105	100	Fréquence start/stop = 100Hz
4106	50	Accélération 50KHz/s
4107	100	Décélération 100KHz/s

Tester le bit « ProbeInProgress » (Bit 12) du status et attendre le passage à 0

Après passage à 0 du bit B12, vérifier si le bit « ProbeError » est à 0

Puis lire le résultat du palpement dans les registres 1044 et 1045

### Commande 67 : Écriture de la position des axes (ICNC\_CMD\_WRITE\_POSITION)

Cette commande n°67 permet de modifier le compteur de position d'un ou de plusieurs axes.

Adresse	Valeur
4096	Commande = 67
4097	Compteur d'axe à initialiser (b0:X ; b1:Y ; b2:Z ; b3:A ; b4:B)
4098	Compteur de position 1 Poids Fort
4099	Compteur de position 1 Poids faible
4100	Compteur de position 2 Poids Fort
4101	Compteur de position 2 Poids faible
4102	Compteur de position 3 Poids Fort
4103	Compteur de position 3 Poids faible
4104	Compteur de position 4 Poids Fort
4105	Compteur de position 4 Poids faible
4106	Compteur de position 5 Poids Fort
4107	Compteur de position 5 Poids faible

Exemple 1 : Initialisation du compteur de position de l'axe X :

Forçage compteur de position axe X à 1000 :

Adresse	Valeur	Fonction
4096	67	Commande d'écriture de compteur de position
4097	1	Axe X(b0)
4098	0	Position X poids Fort
4099	1000	Position X Poids faible

Exemple 2 : Initialisation des compteurs de position des axes X et Z :

Forçage compteur de position axe X à 1000 et Z à 0:

Adresse	Valeur	Fonction
4096	67	Commande d'écriture de compteur de position
4097	5	Axe X(b0) et Axe Z(b2)
4098	0	Position poids Fort X
4099	1000	Position Poids faible X
4100	0	Position Poids Fort Z
4101	0	Position Poids Faible Z

Cette commande n°96 permet de modifier la vitesse de déplacement d'un axe en cours de déplacement.

La modification de vitesse d'un axe ne peut se faire que si celui-ci est en cours de mouvement « asynchrone ».

Si il y a un couplage X/A ou X/B, la vitesse de l'axe couplé sera également modifiée.

Si l'axe est à l'arrêt, la commande sera ignorée.

Les paramètres d'accélération, de décélération et de fréquence start/stop seront respectés durant le changement de vitesse. Les paramètres utilisés sont ceux indiqués lors du lancement de la commande de déplacement asynchrone avec la commande N°80 : ICNC\_CMD\_MOVE\_PROFILE\_ABS\_ASYNC

La cible du déplacement en cours n'est pas modifiée par cette commande. Si l'axe est en cours de décélération finale (arrivée à l'objectif), la commande n'aura donc aucun effet.

Adresse	Valeur
4096	Commande = 96
4097	Axe concerné par le changement de vitesse (b0:X ; b1:Y ; b2:Z ; b3:A ; b4:B)
4098	Nouvelle vitesse (fréquence en Hz) Poids Fort
4099	Nouvelle vitesse (fréquence en Hz) Poids faible

Exemple 1 : Changement de vitesse de l'axe X :

Faire passer l'axe X à 10000Hz en cours de déplacement asynchrone :

Adresse	Valeur	Fonction
4096	67	Commande d'écriture de compteur de position
4097	1	Axe X(b0)
4098	0	Nouvelle vitesse X poids Fort
4099	10000	Nouvelle vitesse X Poids faible

### Commande 42 : Arrêt avec rampe des axes (ICNC\_CMD\_BREAKE\_AXES\_AND\_CLEAR)

La commande n°42 provoque un arrêt de tous les axes en cours de mouvement.  
L'arrêt se fait avec une rampe de décélération.

Adresse	Valeur
4096	Commande = 42

### Commande 43: Arrêt immédiat sans rampe (ICNC\_CMD\_STOP)

La commande n°43 provoque un arrêt immédiat de tous les axes en cours de mouvement.  
L'arrêt se fait donc sans rampe de décélération.  
Cette commande provoque également le verrouillage de la carte.

Adresse	Valeur
4096	Commande = 43

### Commande 41: Arrêt avec rampe d'un axe (ICNC\_CMD\_BREAKE\_AXES)

La commande n°41 provoque un arrêt avec rampe des axes indiqués. Les mouvements interpolés ne sont pas concernés par cet arrêt. Cette commande est donc utilisée pour l'arrêt des axes en cours de déplacements indépendants.

Adresse	Valeur
4096	Commande = 41
4097	Axes à arrêter (b0:X ; b1:Y ; b2:Z ; b3:A ; b4:B)

### Commande 13: Definition état sortie OUT bufferisée (ICNC\_CMD\_SET\_OUTPUT\_BUF)

La commande n°13 permet de définir l'état d'une sortie de manière synchronisée avec le traitement des commande du buffer.

Adresse	Valeur
4096	Commande = 13
4097	Numéro de la sortie DOUT
4098	Etat de la sortie

### Commande 14: Definition de l'état de toutes les sorties OUT bufferisée (ICNC\_CMD\_SET\_OUTPUT\_ALL\_BUF)

La commande n°14 permet de définir l'état simultanément sur toutes les sorties de manière synchronisée avec le traitement des commande du buffer.

Adresse	Valeur
4096	Commande = 14
4097	Etat des sorties 17 à 32
4098	Etat des sorties 1 à 16

### Commande 15 : Definition de l'état d'une sortie analogique bufferisée (ICNC\_CMD\_SET\_ANALOG\_BUF)

La commande n°15 permet de définir le niveau d'une sortie analogique de manière synchronisée avec le traitement des commande du buffer.

Adresse	Valeur
4096	Commande = 15
4097	Numéro de la sortie analogique
4098	Valeur de la sortie analogique (en points DAC)

### Commande 20 : Temporisation traitement buffer (ICNC\_CMD\_TEMPO\_BUF)

La commande n°20 permet temporiser la suite du traitement du buffer de commande de manière synchronisée avec le traitement des commande du buffer.

Adresse	Valeur
4096	Commande = 20
4097	Valeur tempo en *100ms (poids fort)
4098	Valeur tempo en *100ms (poids faible)

### Traitement des erreurs et diagnostic

Si la carte InterpCNC reçoit une trame éronnée (qui ne peut être décodée), elle est simplement ignorée et la carte n'envoie pas de réponse.

En revanche, si l'InterpCNC reçoit une trame sans erreur mais qui ne peut être interprétée (par exemple, écriture d'un registre inexistant), elle retournera une réponse d'erreur sous la forme suivante :

N° Octet réponse	Signification
1	Adresse de la carte
2	Code de fonction de la requête + 128
3	Code d'erreur
4	LO(CRC)

5

HIGH(CRC)

Les codes d'erreur sont :

- 1 : code de fonction invalide
- 2 : Adresse incorrecte
- 3 : Nombre d'adresses ou valeur incorrecte,
- 4 : Erreur non déterminée,

Sur la carte InterpCNC, les LEDs D1 et D2 peuvent servir au diagnostic des problèmes de communication.

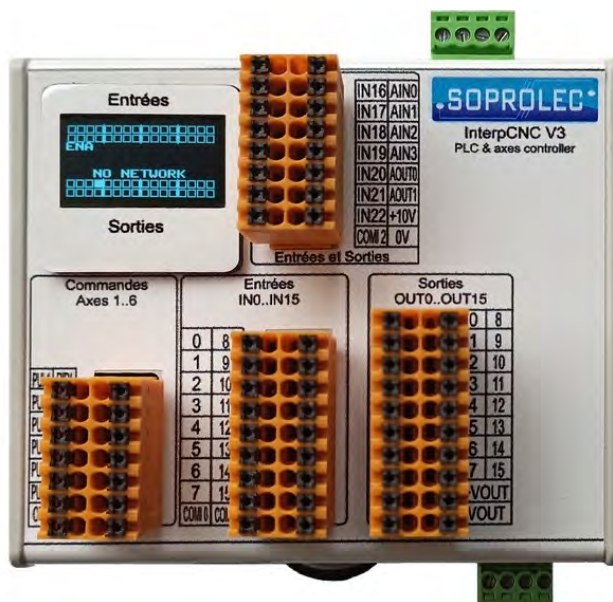
La LED D1 change d'état à chaque fois qu'une trame est traitée par l'InterpCNC. Il faut donc que la trame reçue soit complète, sans erreur de communication et destinée à la carte (ID de l'esclave).

La LED D2 change d'état à chaque trame complète reçue par la carte (qu'elle lui soit destinée ou non). Elle représente donc l'activité globale sur le bus RS485.

Note : D'autres fonctions de diagnostic utilise ces même LEDs et en particulier, pour l'utilisation de la communication USB. Les indications ci-dessus sont donc valables uniquement si la connexion USB n'est pas utilisée simultanément.

## Configurations\_speciales\_V3

Ce chapitre traite de fonctionnalités avancées, offertes à la fois par le hardware et le firmware de la carte **InterpCNC V3**, et configurables facilement depuis ICNCStudio.



## Protection par mot de passe

Dans certains cas, et notamment pour les application industrielles, une protection

par mot de passe est nécessaire.

Celle-ci permet de protéger votre programme PLC Basic contre la lecture, contre l'écriture, ou les deux.

Cela permet par exemple d'empêcher l'utilisation d'une machine avec un programme autre que celui avec lequel elle a été livrée, ou encore de récupérer le programme embarqué pour l'utiliser sur une autre machine dont il aurait été fait un clone non autorisé.

## Verrouillage de la carte

- Choisir et cocher l'option **Lecture protégée** et/ou **Écriture protégée**
- Choisir un mot de passe, et cliquer sur **Valider**
- > une nouvelle fenêtre vous demande alors de confirmer le mot de passe choisi. Cliquer ensuite sur **Send to PLC**. Le symbole du cadenas fermé apparaît. Pour plus de détails, voir la [Protection par mot de passe](#), au chapitre des Configurations.

## Deverrouillage de la carte

- Cliquer sur le cadenas, et saisir votre mot de passe.
- Décocher les options de verrouillage, et cliquer sur **Send to PLC**
- NB: le mot de passe choisi précédemment reste en saisie automatique non lisible dans son champ de saisie, même si la carte est déverrouillée.

**Attention: Si votre mot de passe est définitivement oublié, il faudra dans ce cas contacter la société SOPROLEC pour obtenir un code de déverrouillage qui sera généré spécifiquement pour votre carte.**

## Astuce

Le bouton **Load from PLC** (ou encore taper **reboot** dans la ligne de commande de l'éditeur Basic) peut vous permettre de rafraîchir l'état si vous ne savez plus où vous en êtes.

## Module d'extension I/O

### Configuration de la carte pour utilisation d'un module d'extension I/O Kinco :

Le système de mappings (aussi appelé polling) permet de transmettre avec fréquence de rafraîchissement réglable, des trames **Modbus** pour piloter et échanger des données avec tous types de périphériques (Variateurs, drivers, IHM, PLC, etc...)

Exemple avec un module d'extension I/O Kinco:  
le **KS123-14DR** -> 6 sorties/relais et 8 entrées, branché sur COM2:



#### Raccordement Modbus RS485:

- D+ sur A
- D- sur B

1) Dans le tableau des paramètres de la carte, configurer COM2:

- 420 -> 2 pour Master (la carte InterpCNC V3 est Maître)
- 421 -> Baud Rate (ex: 38400)
- 422 -> 8 (bits de données)
- 423 -> 0 (Parité)
- 424 -> 1 (Bits de Stop)
- 425 -> 1 (Modbus Esclave ID)

Configuration COM 2		
420	COM2 Mode : 0=None, 1=Slave, 2=Master	2
421	COM2 baud rate	38400
422	COM2 Bits de données	8
423	COM2 Parité	0
424	COM2 Bits de stop	1
425	COM2 Modbus Esclave ID	1

## 2) Redéfinir un mappage pour les entrées, et un mappage pour les sorties :

Mappage modbus Maître -> Esclave		
Mappage #1		
600	Port (0=disable, 1=COM1, 2=COM2)	2
601	Adresse Maître	110
602	Nombre de données	6
603	ID Esclave	10
604	Adresse Esclave	0
605	R/Coils(0),R/Inp.(1)R/Holding(2)R/Inp.reg(3),W/Coils(4),W/Holdings(5)	4
606	Période rafraîchissement	50
Mappage #2		
610	Port (0=disable, 1=COM1, 2=COM2)	2
611	Adresse Maître	32
612	Nombre de données	8
613	ID Esclave	10
614	Adresse Esclave	0
615	R/Coils(0),R/Inp.(1)R/Holding(2)R/Inp.reg(3),W/Coils(4),W/Holdings(5)	1
616	Période rafraîchissement	50

Ici, l'état des 6 sorties du module sera commandé par l'état des 6 Coils (bits Modbus) n° 110 à 115.

Les 8 entrées sont mappées sur les entrées virtuelles 32 à 39 (DIN).

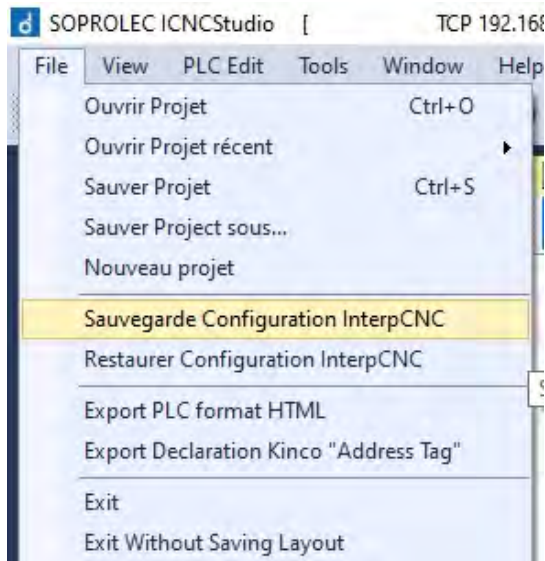
Cet état sera rafraîchi par une lecture tous les 50ms.

NB : l'ID Esclave est l'adresse Modbus du module (elle peut-être redéfinie à l'aide d'un petit utilitaire Kinco).

Bien sûr, ce type de module ne permet pas de bénéficier d'entrées et sorties aussi rapides que celles de l' InterpCNC V3 (d'autant plus que les sorties de ces modules sont sur relais), mais dans certaines applications cette solution peut-être intéressante car vous n'êtes plus limité à 16 entrées et 16 sorties numériques.

## Les exportations et importations (V2 et V3)

A l'onglet **File** de la barre de Menu :



→ l'option « **Sauvegarde Configuration InterpCNC** ».  
Celle-ci permet de sauvegarder sous la forme d'un fichier .ini les données suivantes :

- Exportation des Paramètres de la carte.
- Exportation de la Mémoire Sauvegardée (EEPROM)
- Exportation des Recettes

L'option « Restaurer Configuration InterpCNC » permet comme son nom l'indique de recharger à partir du fichier .ini, les paramètres et registres d'une sauvegarde précédente.

Tout ou partie de ces 3 rubriques sont à la fois sauvegardables ou restaurables (options à cocher).

→ L' **Export Declaration Kinco Address Tag**, est une fonction très utile qui génère un fichier .csv associant le nom (Tag) attribué à chaque Bit ou Registre utilisé dans notre programme PLCBasic, avec son adresse Modbus et son type d'accès (lecture, lecture/écriture : 0X, 1X, 3X, 4X, etc...).

Ce fichier est ensuite extrêmement pratique pour la conception des écrans d'IHM associés à votre programme PLCBasic, avec les logiciels DTools ou HMIWare de la marque **Kinco**.

Il suffit alors d'importer votre fichier .csv d' Address Tag avec la commande « Import Address Tag ».

Vous disposez alors dans votre projet de conception des pages de votre IHM, du même nommage des bits et registres, et des adresses Modbus utilisés par la carte.

Le développement des écrans d'IHM Kinco s'en trouve vraiment facilité.

A l'onglet **PLC Edit** :

→ L' **Export Programme Analysé**. Cette fonction permet de sauvegarder au format .txt, votre programme PLCBasic au format brut, c'est à dire sans les commentaires et où chaque nom de constante est remplacé par sa valeur numérique.

Ce fichier .txt peut ensuite être ouvert, et un 'Copier/Coller' de son contenu vers la fenêtre du programme PLC donne un aperçu du programme tel qu'il est exécuté par l'interpréteur Basic de la carte.